IBM System/36

Programming with RPG II

When You Are:	You Can Find Information In:
Planning to Install Your Computer	What to Do Before Your Computer Arrives or Converting from System/34 to System/36
Getting Your Computer Ready to Use	Setting Up Your Computer Performing the First System Configuration For Your System System Security Guide
Operating Your Systems	Learning About Your Computer Operating Your Systems
Operating and Using the Utilities	Development Support Utility Guide Source Entry Utility Guide Data File Utility Guide Creating Displays Work Station Utility Guide Utilities Messages
Programming Your Computer	Concepts and Programmer's Guide System Reference Sort Guide Work Station Utility Guide **Programming with RPG II** **RPG II Messages**
Communicating with Another Computer or Remote Device	(communication manuals) (communication message manuals)
Determining the Cause of a Problem	System Messages (message manuals) System Problem Determination manual for your System Unit

IBM System/36

Programming with RPG II

Program Number 5727-RG1

Program Number 5727-RG6

File Number
S36-28

Order Number
SC21-9006-4

Fifth Edition (October 1986)

This major revision obsoletes SC21-9006-3.

Changes are periodically made to the information herein; any such changes will be reported in subsequent revisions. Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change or addition.

See *About This Manual* for a summary of major changes to this edition.

This edition applies to Release 5, Modification Level 0, of IBM System/36 RPG II Program Product (Program 5727-RG1 and Program 5727-RG6), and to all subsequent releases and modifications until otherwise indicated in new editions.

Publications are not stocked at the address given below. Requests for IBM publications should be made to your IBM representative or to your IBM-approved remarketer.

A form for the reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Canada Ltd. Information Development, Department 849, 895 Don Mills Road, North York, Ontario, Canada, M3C 1W3. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Contents

Part 2. Reference . . . .

About This Manual

Who should use this manual . . .

This manual is a guide and reference for the RPG II programming language on System/36. It is intended for people who have a basic understanding of data processing concepts and of the RPG programming language. For convenience, RPG II is referred to as RPG.

Using this manual, you can:

- Design RPG programs
- Code RPG programs
- Enter and compile RPG programs
- Test and debug RPG programs
- Follow coded RPG examples and sample programs

How this manual is arranged . . .

Part 1. Guide

Part 1 (Chapters 1 through 18) is a programmer's guide. It is organized in the sequence of tasks that a programmer must perform to use an RPG program: design, code, enter, compile, test, and put into production. Chapters 1 through 4 discuss designing, coding, entering, compiling, and testing:

- Chapter 1 discusses designing an RPG program.
- Chapter 2 discusses coding an RPG program.
- Chapter 3 discusses entering and compiling an RPG program.
- Chapter 4 discusses testing an RPG program.

Chapters 5 through 11 discuss using various kinds of files. Each file that your program uses must be assigned to an input/output device. You code that device name in columns 40 through 46 of the file description specifications. For example, if a file uses a disk as an input/output device, DISK is coded as the device name. The coded device name is a convenient way to refer to each kind of file. Thus, a file assigned to a disk is called a DISK file, a file assigned to a display station is called a WORKSTN file, and so on.

- Chapter 5 discusses DISK files.
- Chapter 6 discusses WORKSTN files.
- Chapter 7 discusses PRINTER files.
- Chapter 8 discusses SPECIAL files.
- Chapter 9 discusses CONSOLE, KEYBORD, and CRT files.
- Chapter 10 discusses BSCA files.
- Chapter 11 discusses primary and secondary files. Primary and secondary are not device names. Instead, they indicate how the files are processed by the RPG program cycle.

  Some files (CONSOLE and KEYBORD) can be used only for input, some (CRT and PRINTER) can be used only for output, and some (DISK, WORKSTN, SPECIAL, and BSCA) can be used for both input and output. Therefore, when you use an RPG program, you must know how you can use each file.

  Typical RPG programs, and the kinds of files you might use for input and for output, are listed below:

What the RPG Program Does	Files You Can Use for Input	Files You Can Use for Output
Inquire into a file	DISK WORKSTN SPECIAL	PRINTER WORKSTN SPECIAL
Process a file	DISK WORKSTN CONSOLE KEYBORD SPECIAL	DISK WORKSTN CRT PRINTER SPECIAL
Enter data	DISK WORKSTN CONSOLE KEYBORD SPECIAL BSCA	DISK WORKSTN CRT PRINTER SPECIAL BSCA
Print a report	DISK WORKSTN CONSOLE KEYBORD SPECIAL BSCA	PRINTER

If you wanted to create a program to display accounts receivable information about your customers, you would probably use a DISK file for input, a WORKSTN file for both input and output, and possibly a PRINTER file for output as well. The DISK input file would be a customer master file containing all the accounts receivable records for your customers. As an input file, the WORKSTN file would select the desired records from the DISK file. As an output file, the WORKSTN file would display the selected records from the DISK file. The program might do no processing other than reading records from the DISK file and writing records to the WORKSTN file. Sample program AR230R in Chapter 6 shows this example. If you used a PRINTER file also, you could write the output records to the printer.

When you are ready to code your program, read the chapters that explain how to code a program that uses the kinds of files your program uses. For example, for information about coding a program that uses a DISK file, a WORKSTN file, and a PRINTER file, you should read Chapters 5, 6, and 7.

Chapters 12 through 17 discuss various things that your RPG program can do:

- Chapter 12 discusses using indicators.
- Chapter 13 discusses using arrays and tables.

- Chapter 14 discusses using data structures.
- Chapter 15 discusses using auto report.
- Chapter 16 discusses editing numeric output fields.
- Chapter 17 discusses changing the hexadecimal value of characters.

Chapter 18 presents some tips for coding efficient programs.

Part 2. Reference

Part 2 (Chapters 19 through 32) is a reference.

- Chapter 19 presents both a general and a detailed explanation of the RPG program cycle.
- Chapters 20 through 27 explain all possible entries (column by column) for each RPG specification sheet.
- Chapter 28 explains each RPG operation code in alphabetical order.
- Chapter 29 shows a storage dump of an RPG program.
- Chapter 30 explains the differences between the way RPG programs work on System/36 and the way they work on System/34.
- Chapter 31 discusses using ideographic data.
- Chapter 32 discusses problem determination.

What you should know . . .

Before you use this manual, you should be familiar with certain information:

- You should know how to use the controls and indicators on your display screen and how to use the keys on your keyboard, such as:
  - Cursor movement keys
  - Command keys
  - Field exit keys
  - Insert and delete keys
  - Error Reset key

  This information is contained in:

  - *IBM 5291 Display Station Operator's Guide*, GA21-9409
  - *IBM 5292 Color Display Station Operator's Guide*, GA21-9416
  - *IBM 5251 Models 1 and 11 Display Station, IBM 5252 Dual Display Station Operator's Guide*, GA21-9248

- *IBM 5251 Models 2 and 12 Display Station Operator's Guide*, GA21-9323

- You should know how to operate your display station to use the System/36 System Support Program (SSP) to do such things as:

  - Signing on and signing off the display station
  - Interacting with displays
  - Using help
  - Entering control commands and procedure commands
  - Responding to messages

  This information is contained in the manual *Operating Your System 5360, 5362*, SC21-9452 if you use a System/36 System Unit 5360 or a System/36 System Unit 5362, and in the manual *Operating Your System 5364*, SC21-9353 if you use a System/36 System Unit 5364. Refer also to the manual *Using Your Display Station*, SC21-9455.

- You should know how to design and code displays by using:

  - Screen design aid (SDA) utility
  - BLDMENU and FORMAT procedures

  This information is contained in the manual *Creating Displays: Screen Design Aid and System Support Program*, SC21-7902.

- You should know how to communicate with the SSP by using:

  - Operation control language (OCL) statements
  - Utility programs and utility control statements
  - Procedures
  - Commands

  This information is contained in the *System Reference* manual, SC21-9020.

- You should know how to design and code efficient programs. This information is contained in the *Concepts and Programmer's Guide*, SC21-9019.

- You should know how the RPG program cycle works, how indicators affect the program cycle, and how to code entries on the RPG specification sheets. This information is contained in the *IBM Introduction to RPG II and RPG III: Batch Processing with Program Described Files*, GC21-7514. It is also available from an IBM RPG II coding class.

- You should know how to use the development support utility (DSU) or the source entry utility (SEU). This information is contained in the *Development Support Utility Guide*, SC09-1085, which explains how to enter and update your source and procedure members using a full screen editor, and the *Source Entry Utility (SEU) Guide*, SC21-7901.

- You should know how to interpret displayed and printed messages. This information is contained in the *System Messages* manual, SC21-7938, in the *RPG II Messages* manual, SC21-7940, and in the *Utilities Messages* manual, SC21-7939.

- If you communicate with an IBM 3741 Data Entry Work Station you should know that 3741 Status messages are treated as data to be handled by the user. Information on possible status messages and data format are contained in the *IBM 3741 Data Station Reference Manual*, GA-9183.

Naming conventions

In this manual, the following conventions are used for program, display, and menu names:

- Program names use the format aannnR, where:

  - aa identifies the type of application:

    AR means accounts receivable

    IM means inventory management

    OE means order entry

  - nnn is a number that identifies the type of program:

    100-199 for data entry

    200-299 for inquiry

    300-349 for file maintenance

    350-399 for sort

    400-499 for file update

    900-949 for printing reports and program listings

  - R identifies the programming language as RPG

- Display names are formed by adding a D to the end of the name of the program that uses the display. For example, AR230RD is the name of a display used by an RPG accounts receivable inquiry program. If the program uses more than one display, a sequence number is added to the display name. For example, if program AR230R uses two displays, the displays are named AR230RD1 and AR230RD2.

- Menu names use the format aannnM, where:

  - aa identifies the type of application
  - nnn is a number assigned to the menu
  - M identifies the name as a menu name

Coding conventions

Where specification sheets show which columns to code, shading indicates that no entry is allowed in the column, no coding in an unshaded column indicates that more than one entry is allowed in the column, and a character in an unshaded column indicates that that character is the only entry allowed in the column.

If you need more information . . .

For an index of major topics discussed in all System/36 manuals, see the *Guide to Publications*, GC21-9015.

For information about converting your programs from IBM System/34 to System/36, see the manual *Converting from System/34 to System/36*, SC21-9053.

For information about data communications, see the *Interactive Communications Feature: Reference*, SC21-7910, and the *Interactive Communications Feature: Guide and Examples*, SC21-7911.

For information about designing structured programs, see the *Structured Programming Textbook*, SR20-7149, and the *Structured Programming Workbook*, SR20-7150.

For information about protecting the security of your system, see the *System Security Guide*, SC21-9042.

For information about determining whether a problem is in an IBM product or in your own program, see the *System Problem Determination - 5360* manual, SC21-7919 if you use a System/36 System Unit 5360, the *System Problem Determination - 5362* manual, SC21-9063 if you use a System/36 System Unit 5362, and the *System Problem Determination - 5364* manual, SC21-9375 if you use a System/36 System Unit 5364.

RPG coding and debugging material

- RPG Control and File Description Specifications, GX09-1035
- RPG Calculation Specifications, GX09-1035
- RPG Extension and Line Counter Specifications, GX09-1033
- RPG Input Specifications, GX09-1033
- RPG Telecommunications Specifications, GX09-1034
- RPG Output Specifications, GX09-1034
- RPG Auto Report Specifications, GX09-1032
- RPG Indicator Summary, GX09-1032

- RPG Debugging Template, GX21-9129

System coding sheets

- Display Format Specifications, GX21-9800
- IBM 5250 Display Station Keyboard Template Assignment Sheet and Display Screen Layout Sheet, GX21-9271
- IBM Printer/Display Layout Sheet, GX21-9174
- Translation Table and Alternate Collating Sequence Coding Sheet, GX21-9096

Summary of Changes

The main enhancement made to System/36 RPG for release 5 is adding the DO structured programming operation code.

The DO operation allows an operation or a series of operations to be performed a fixed number of times. Programmer indicates how many times the operation(s) will be performed when coding the DO statement and the associated END statement. See detailed descriptions of the DO operation in Chapter 18 and Chapter 28.

Note: *This manual may refer to products that are announced, but are not yet available. Such information is for planning purposes only and is subject to change before general availability.*

Part 1. GUIDE

Chapter 1. Designing an RPG Program

Chapter 1. Designing an RPG Program

This chapter begins with an overview of the six steps in creating and using an RPG program. These steps are related to the sequence of chapters in Part 1 of this manual, the programmer's guide.

After that overview, this chapter discusses step 1, designing an RPG program. The design step includes designing the output, designing the processing, and designing the input.

Steps in Using RPG

There are six steps in creating and using an RPG program (see Figure 1-1):

1. Design
2. Code
3. Enter
4. Compile
5. Test
6. Put into production

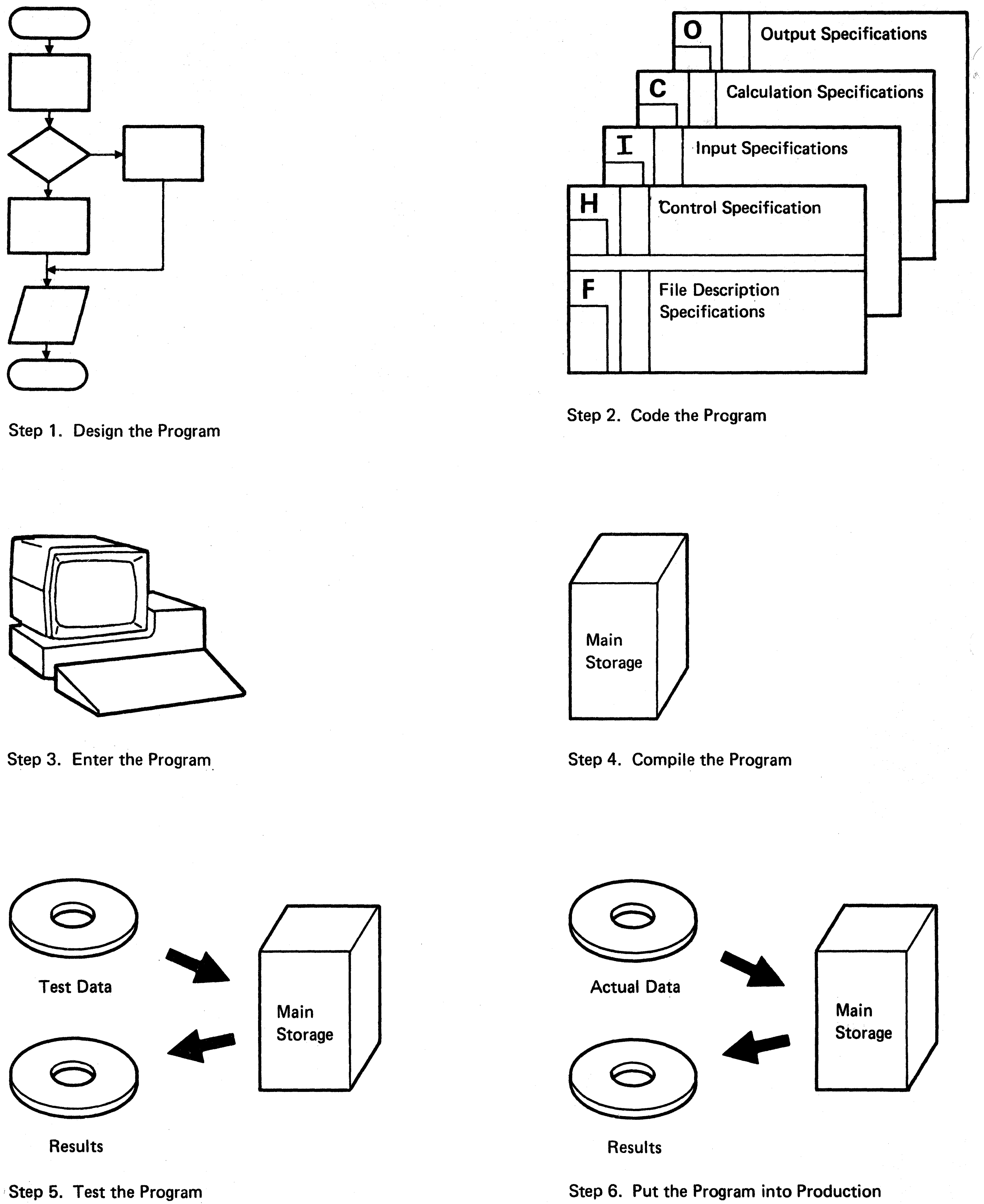


Figure 1-1. Steps in Using an RPG Program

Step 1. Design

Designing means planning what you want the program to do. Designing is probably the most important step in programming, so it is worth the time to design your programs carefully before you start to code them. A well-designed program is the best way to ensure good performance (fast processing and efficient use of resources) from your computer. Later sections of this chapter discuss the design step in more detail. For a complete explanation of the principles of program design, see the *Concepts and Programmer's Guide.*

Step 2. Code

Coding a program means writing the instructions that tell the computer what data to use, how to process it, and what to do with the results. You write these instructions on RPG specification sheets. Chapter 2 discusses coding in general terms.

Step 3. Enter

Entering a program means getting your written instructions (your coding) into the computer. You use the RPGONL procedure, the RPGSEU procedure, or the source entry utility to enter your instructions. Chapter 3 explains how to enter your program.

Step 4. Compile

Compiling a program means translating your entered instructions (called a *source program* into a form that the computer can use (called a *load module*). You use the RPGONL or RPGC procedure to compile your source program. Both of these procedures allow you to display your compiled program at your display station. The RPGONL procedure also allows you to correct errors in your source program and recompile the program at your display station. Chapter 3 explains how to compile your program.

Step 5. Test

Testing a program means running a load module with some sample data to be sure that it produces the proper results. Testing helps you debug your load module before you run it with your actual data. Chapter 4 discusses testing your program.

Step 6. Put into Production

Putting a program into production means using it to process your actual data. Chapters 5 through 18 discuss using your load module to process data.

Designing Your Program

A program usually begins with an idea or a request to produce a certain kind of result. Therefore, you know what the program should do before you begin to code it. The planning you do to decide how to code a source program that will produce that result is called *designing* the program.

For example, suppose you want to create a program that allows people in your company to display information about your customers. An example of such a program is sample program AR230R in Chapter 6. That program displays accounts receivable information about a company's customers. To design that program, the programmer had to consider such questions as:

- What information will the people in the company need to know about the customers? Name? Address? Phone number? Balance due? Credit limit?

- How many people will need this information? Will more than one person need it at the same time?

- How will a person request information about a customer? By entering the customer's number? The customer's name?

- What file or files contain the information to be displayed? How are those files organized?

- Can a person change the displayed information or only look at it?

- Will a person want a printed copy of the displayed information?

The answers to questions like these determine how to code the program.

Designing a program includes:

- Deciding what output you need from your program

- Deciding what processing will produce the output you need

- Deciding what input is required by and available to your program

This sequence may seem backwards because it starts at the results (the output) and ends at the beginning (the input). The reason for designing the output first is to make sure that you start with a clear understanding of what your program will do. If you know what output you need, you can decide what calculations are necessary to produce that output. Designing the output first is like knowing where you are going before you set out on a trip: it helps you decide the best way to get there.

Designing the Output

Your program will produce output records. You must decide what you will do with those records. In general, you have three choices (or any combination of the three choices):

- You can display them.
- You can print them.
- You can store them.

If you want to display the output records at your display station, you have to decide what information you want displayed and how you want it laid out. To define how you want your displays laid out, you use the display layout sheet. Then you use the screen design aid (SDA) utility to create your display screens. For more information about SDA, see the manual *Creating Displays.*

If you want to print the output records, you also have to decide what information you want printed (which fields from which records) and how you want it laid out on the printed report. To indicate how you want the printed report laid out, use the printer layout sheet.

If you want to keep the output records in storage, you still have to decide what information you want to keep and how you want to organize the fields in the output records.

After you design all your output records, you describe those records on the RPG file description specifications and output specifications.

Designing the Processing

Designing the processing means planning the calculations that will produce the necessary output. When you design the processing, you must be aware of how the RPG program cycle works. The RPG program cycle controls certain operations performed on each record, so the program cycle partly determines how you can process your data.

The phrase *program cycle* refers to the series of operations that an RPG program automatically performs on each record that it reads. Each RPG program goes through the same general cycle of operations.

The program cycle has three basic logic steps:

- Reading information (input)
- Doing calculations (processing)
- Writing results (output)

These basic logic steps can be divided into several substeps in which you can assign indicators to control when calculation and output operations occur. These substeps and indicators are then coded on the RPG calculation specifications.

For a detailed explanation of the RPG program cycle, see Chapter 19. For more information about how indicators affect the RPG program cycle, see Chapter 12.

Designing the Input

After you decide what output you need and what calculations will produce that output, the next step is to find out where the input data for your program will come from. It might come from one or more files already on the computer, from one or more display stations on your computer, from one or more other computers, or from a combination of these sources. You have to know the names used for input files, the location of fields in the input records, the sequence of record types, the formats of numeric data, and the indicators used. When you know this information, you can describe your input records on the RPG input specifications.

Chapter 2. Coding an RPG Program

Chapter 2. Coding an RPG Program

After you have designed your program, you must write the individual statements that make up your source program. These statements should be coded on RPG specification sheets. Each line coded on a specification sheet represents a statement in the source program. Each specification sheet contains 80 columns. Column headings indicate the kind of information to code in particular columns.

There are eight kinds of specifications:

- Control specification
- File description specifications
- Extension specifications
- Line counter specifications
- Telecommunications specifications
- Input specifications
- Calculation specifications
- Output specifications

Each of these specifications is described briefly in this chapter.

Most RPG programs do not use all eight kinds of specifications. In fact, none of them is required in every program, and a typical program probably uses only file description, input, calculation, and output specifications.

When the computer compiles your source program, the specifications you use must be in the order shown in Figure 2-1. The specifications can be coded in any order, but normally you code them in the same order in which you design the program: first output, then calculations, then input.

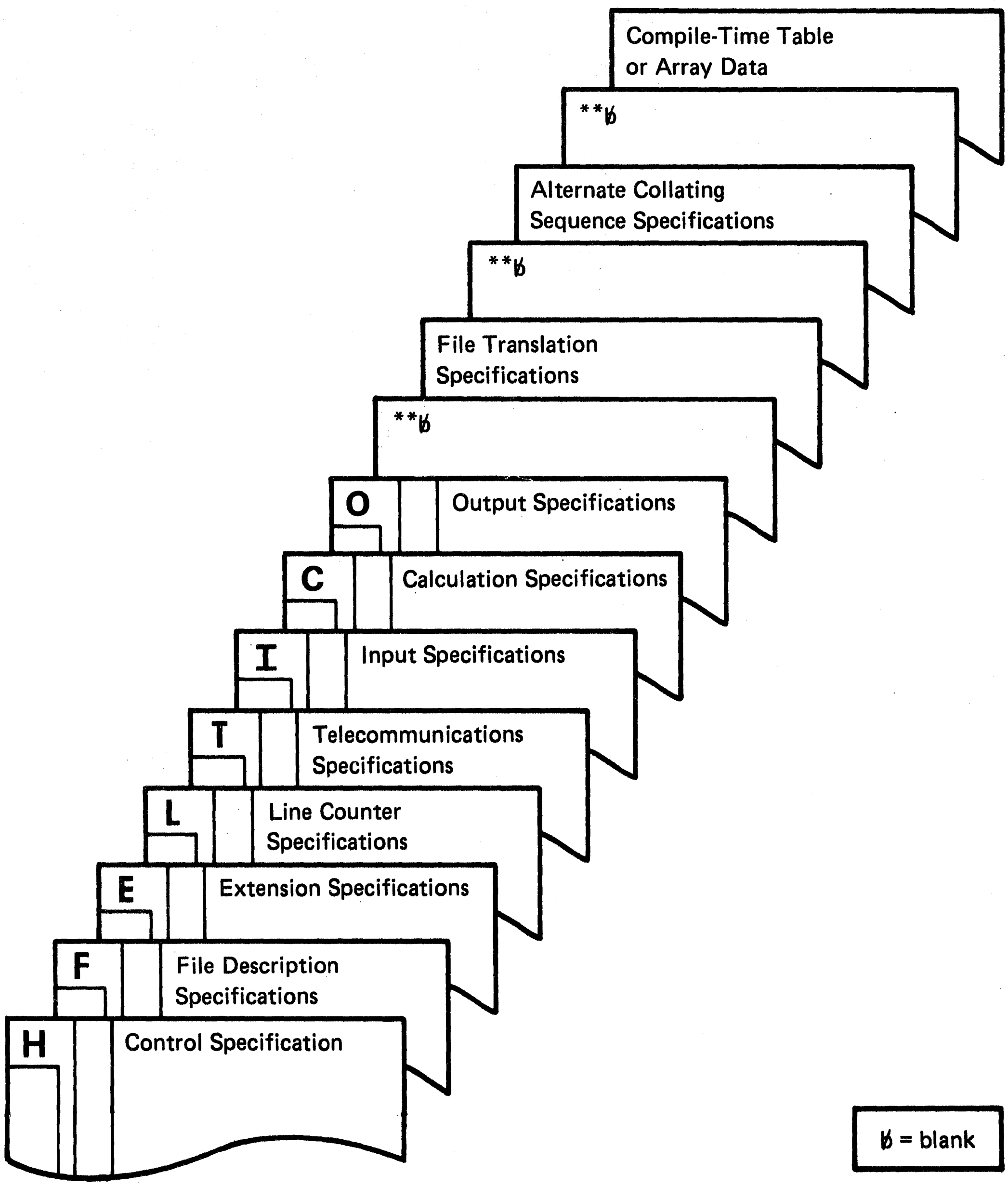


Figure 2-1. Required Order of Specifications for an RPG Source Program

Control Specification

The control specification provides the RPG compiler with information such as the following about your program and your computer:

- Name of the program
- Storage size needed for the program to run
- Date format for the program
- Whether an alternative collating sequence or file translation is used

For a detailed description of the control specification, see Chapter 20.

File Description Specifications

File description specifications describe all the files that your program uses. The information for each file includes:

- Name of the file
- How the file is used
- Size of records in the file
- Input or output device used for the file
- Whether the file is conditioned by an external indicator

For a detailed description of the file description specifications, see Chapter 21.

Extension Specifications

Extension specifications describe all record address files, table files, and array files used in the program. The information includes:

- Name of the file, array, or table
- Number of entries in a table or array input record
- Number of entries in a table or array
- Length of the table or array entry

For a detailed description of the extension specifications, see Chapter 22.

Line Counter Specifications

Line counter specifications describe the page or form on which output is printed. The information includes:

- Number of lines per page
- Line of the page at which overflow occurs

For a detailed description of the line counter specifications, see Chapter 23.

Telecommunications Specifications

Telecommunications specifications describe each BSCA file in the program. The information includes:

- Name of the file
- Description of the communication network used
- Type of station
- Type of control
- Type of code used
- Station identification

For a detailed description of the telecommunications specifications, see Chapter 24.

Input Specifications

Input specifications describe the records and fields in the input files used by the program. The information for each record includes:

- Name of the file
- Sequence of record types
- Whether record-identifying indicators, control-level indicators, field-record-relation indicators, or field indicators are used
- Whether data structures, look-ahead fields, record identification codes, or match fields are used
- Type of each field (alphameric or numeric; packed-decimal, zoned-decimal, or binary format)
- Location of each field in the record

- Name of each field in the record

For a detailed description of the input specifications, see Chapter 25.

Calculation Specifications

Calculation specifications describe the calculations to be performed on the data and the order in which the calculations are to be performed. Calculation specifications can also be used to control certain input and output operations. The information includes:

- Control-level and conditioning indicators for the operation specified
- Fields or constants to be used in the operation
- The operation to be performed
- Whether resulting indicators are set after the operation is performed

For a detailed description of the calculation specifications, see Chapter 26.

Output Specifications

Output specifications describe the records and fields in the output files and the conditions under which output operations are performed. The information includes:

- Name of the file
- Type of record to be written
- Spacing and skipping instructions for PRINTER and CRT files
- Output indicators that condition when the record is to be written
- Name of each field in the output record
- Location of each field in the output record
- Edit codes and edit words
- Constants to be written
- Format name for a WORKSTN file

For a detailed description of the output specifications, see Chapter 27.

Chapter 3. Entering and Compiling an RPG Program

Chapter 3. Entering and Compiling an RPG Program

Using the RPG Procedures

To enter and compile an RPG program, use one or more of the following RPG procedures:

- RPGONL, which lets you create a source program and then alternately compile and correct errors in your source program online (at your display station)

- RPGSEU, which lets you create or change a procedure or an RPG or auto report source member

- RPGC, which lets you compile an RPG source program

- RPGX, which lets you print a cross-reference listing for an RPG program

- AUTOC, which lets you compile an RPG source program that contains auto report specifications

- RPGSDA, which lets you create or change a display format

Each of these procedures is explained in this chapter.

To begin one of these procedures, use any one of the following methods:

- Choose an option from the RPGP programming menu.

- Include any of the RPG procedures in one of your own procedures and run your own procedure.

- Call the procedure directly. To call one of the procedures directly, use one of the following methods:

  - Type HELP, a space, and the procedure name (with or without parameters) on the command line, and press the Enter key.
  - Type the procedure name on the command line of the display screen, and press the Enter key.
  - Type the procedure name (with or without parameters) on the command line, and press the Help key.

– Type the procedure name, a space, and the source member name on the command line, and press the Enter key. (If you use this method, prompts are displayed only for the RPGONL procedure.)

RPGP Programming Menu

There are three ways to display the RPGP programming menu:

- Type HELP RPGP on the command line of the display screen, and press the Enter key.
- Type RPGP on the command line, and press the Help key.
- Use the Help menus.

The programming menu lists six options from which you can choose to perform tasks related to RPG programming:

```
                         RPGP                                W4
                RPG II programming procedures

Select one of the following:

1.  Develop an RPG II program
2.  Create or change an RPG II or auto report program
3.  Compile an RPG II program
4.  Print an RPG II cross reference
5.  Compile an auto report program
6.  Create or change display formats



Cmd3-Previous menu   Cmd7-End   Cmd5-Main help menu   Home-Sign on menu

Ready for option number or command

                                              © 1983 IBM Corp.
```

To choose one of the six procedures, enter a number from 1 through 6:

- Enter 1 to use the RPGONL procedure.
- Enter 2 to use the RPGSEU procedure.
- Enter 3 to use the RPGC procedure.
- Enter 4 to use the RPGX procedure.
- Enter 5 to use the AUTOC procedure.
- Enter 6 to use the RPGSDA procedure.

When chosen from the RPG programming menu, all six of these procedures remember the parameters you specify when you use a procedure. Then, the next time you choose any of the six procedures from the programming menu, the parameters you specified previously are automatically entered for you in response to the new prompts. For example, if you select the RPGC procedure on the programming menu, all 18 of the parameters you specified for the RPGC procedure are stored. When the RPGC procedure is complete, the programming menu is displayed again. Then, if you select the RPGSEU procedure on the programming menu, the display for the RPGSEU procedure shows the source member name and the library name that you entered for the RPGC procedure. If you sign off, the parameters are reset to the default values.

You can use the following keys from the RPGP programming menu:

- Command key 3 to return to the previous help menu
- Command key 7 to end help
- Command key 12 for information about how to use help
- The Home key to return to the sign-on menu
- The Help key for additional information about the procedures
- Command key 24 to make the RPGP Help menu your default help menu

RPG Interactive Program-Development Procedure (RPGONL Procedure)

The RPGONL procedure allows you to enter and compile your source program. The procedure identifies any errors in your source program by displaying your source program online (that is, at your display station) with error messages inserted immediately before the specification containing the error. You can correct these errors online and recompile your program as many times as necessary until it compiles successfully. The RPGONL procedure has three displays.

Using the First RPGONL Display

The first display for the RPGONL procedure looks like this:

```
                         RPGONL PROCEDURE                Optional-*
                 RPG II interactive program-development procedure

Name of source program to be compiled  . . . . . . . . . . TEST
Name of library containing program to be compiled  . . . . YOURLIB
Name of library to contain compiled program  . . . . . . .          *
Name of data dictionary to be used . . . . . . . . . . . .          *
Create or change source before first compilation?  Y,N . . Y




Cmd3-Previous menu  Cmd7-End  Cmd14-Compiler options   © 1985 IBM Corp.
```

Respond to each prompt by entering the appropriate information.

Name of source program to be compiled: Enter the name of the your source program.

Name of library containing program to be compiled: Enter the name of the library that contains the source member to be compiled. If no library name is specified, the library name is determined in one of the following ways:

- If RPGONL is the first procedure to be run during this session, the current library name is used.
- If the last procedure run during this session was RPGONL, RPGC, RPGX, RPGSEU, or AUTOC, the library name used in that last procedure becomes the default used for RPGONL. For RPGC, RPGX, RPGSEU, and AUTOC, this applies only when these procedures are not run from the job queue or are not evoked. If these procedures are run from the job queue, or are evoked, or if RPGR or RPGSDA is run, the

library name for RPGONL is not changed from the current library name.

- If the last procedure run during this session was another language procedure (any COBOL, FORTRAN, or ASSEMBLER procedure), the current library name is used.

Name of library to contain compiled program: Enter the name of the library that is to contain the compiled program. If no library name is specified, the library name is determined in one of the following ways:

- If RPGONL is the first procedure to be run during this session, the current library name is used.

- If the last procedure run during this session was RPGONL, RPGC, RPGX, RPGSEU, or AUTOC, the library name used in that last procedure becomes the default used for RPGONL. For RPGC, RPGX, RPGSEU, and AUTOC, this applies only when these procedures are not run from the job queue or are not evoked. If these procedures are run from the job queue, or are evoked, or if RPGR or RPGSDA is run, the library name for RPGONL is not changed from the current library name.

- If the last procedure run during this session was another language procedure (any COBOL, FORTRAN, or ASSEMBLER procedure), the current library name is used.

Name of data dictionary to be used: Enter the name of the current data dictionary to be used during the compilation, if you are using communication formats defined through the interactive data definition utility (IDDU). There is no default value for the data dictionary. The dictionary must exist.

Create or change source before first compilation?: Enter Y or N.

Y (yes) means that you want the procedure to call the development support utility (DSU), if you have it installed, or the source entry utility (SEU), to allow you to create or change your source program before the first compilation.

N (no) means that you do not want to create or change your source program before the first compilation.

If no option is chosen, Y is assumed.

You can use the following keys from the first RPGONL display:

- Command key 3 to return to the previous Help menu

- Command key 7 to end the RPGONL procedure

- The Help key for additional explanation of the parameters

- Command key 14 to see the second RPGONL display, which allows you to specify more parameters.

Using the Second RPGONL Display

The second RPGONL display looks like this:

Procedure Name and Parameters Specified on the First Display

```
                           RPGONL PROCEDURE                    Optional-*
                        RPG II compiler options

RPGONL TEST,YOURLIB,,,Y

Override print option in source  . . SOURCE,NOSOURCE,PSOURCE            *
Override debug option in source  . . . . . . . DEBUG,NODEBUG            *
Override size-to-execute option in source  . . . . . .  2-64            *
Create cross-reference listing . . . . . . . . . NOXREF,XREF  NOXREF
Create program that can be run . . . . . . . . . LINK,NOLINK  LINK
Create program that must be
  link-edited  . . . . . . . . . . . . . . . NOOBJECT,OBJECT  NOOBJECT
Name of subroutine input library . . . . . . . . . . . . . .            *
Never-ending program . . . . . . . . . . . . . . . NONEP,NEP  NONEP
Maximum number of
  requesting display stations. . . . . . . . . . . . . . 0-99  0
Generate CONSOLE file display formats  . . . . . . GEN,NOGEN  GEN
Size of work files in blocks . . . . . . . . . . . .  1-9999  40
Create program with
  memory resident overlays . . . . . . . . . . . . NOMRO,MRO  NOMRO

Cmd2-Page back  Cmd7-End                             (c) 1985 IBM Corp.
```

Respond to each prompt by entering the appropriate information.

Override print option in source: Enter SOURCE, PSOURCE, or NOSOURCE. Use this option to override an entry in column 11 of the control specification of your RPG program.

SOURCE means that you want the RPG compiler to print a full compiler listing. A full compiler listing includes the source program, information about tables and arrays, the relative location of fields and their attributes, field names that are not referred to, diagnostics, and a map of main storage. The map of main storage lists the identification, the starting address, and the size of each separately identifiable segment of code in the program, tells how much main storage is required to run the program, and lists the number of library sectors required for the program.

NOSOURCE means that you do not want the RPG compiler to print a compiler listing. If you use this option, a prolog is printed along with the following information from the overlay linkage editor: the amount of main storage required to run the program, the starting address of the program, and the number of library sectors required for the program. Use this option when you already have a listing of the program.

PSOURCE means that you want the RPG compiler to print a partial compiler listing. A partial listing includes the source program, information about indicators used in the program, diagnostics, and the following information from the overlay linkage editor: the amount of main storage required to run the program, the starting address of the program, and the number of library sectors required for the program. It does not include information about tables and arrays, information about fields, or a map of main storage.

If no option is specified, the entry coded in column 11 of the control specification is used.

Override debug option in source: Enter DEBUG or NODEBUG. Use this option to override the entry in column 15 of the control specification.

DEBUG means that the DEBUG operation is to be used in the calculation specifications.

NODEBUG means that the DEBUG operation is not to be used.

If no option is specified, the entry coded in column 15 of the control specification of your RPG program is used.

Override size-to-execute option in source: Enter the program size in K bytes (one K byte equals 1024 bytes). The program size must be an even number from 2 through 64. Use this option to override the entry coded in columns 12 through 14 of the control specification of your RPG program. If no size is specified, the entry coded in columns 12 through 14 of the control specification is used.

Create cross-reference listing: Enter NOXREF or XREF.

NOXREF means that you do not want the RPG compiler to create a cross-reference listing for the program.

XREF means that you do want a cross-reference listing. The cross-reference listing is created only if the program contains no terminal errors. A prompt on the third display for this procedure allows you to choose whether to display or print the listing.

If no option is specified, NOXREF is assumed.

Create program that can be run: Enter LINK or NOLINK.

LINK means that you want to create a load module, that is, a program that you can run without first having to use the overlay linkage editor procedure OLINK to link-edit the program.

NOLINK means that you do not want to create a load module.

If no option is entered, LINK is assumed.

Create program that must be link-edited: Enter NOOBJECT or OBJECT.

NOOBJECT means that you do not want to create an object module.

OBJECT means that you want to create an object module, that is, a compiled *subroutine* (not a load module). You must use the OLINK procedure to link-edit this module (by itself or with one or more other assembler programs or subroutines) before you can run the module.

If no option is specified, NOOBJECT is assumed.

Create memory-resident overlays: Enter NOMRO or MRO.

NOMRO means that you do not want overlays to remain in memory.

MRO means that you want to retain overlays in memory. This enables you to keep more than one overlay in memory.

The default is NOMRO.

Name of subroutine input library: Enter the name of the library that contains one or more assembler subroutines to be combined with the program being compiled. If no library name is specified, the name of the source input library is assumed.

Never-ending program: Enter NONEP or NEP.

NONEP means that the program is not a never-ending program.

NEP means that the program is a never-ending program. A never-ending program is one that uses system resources (such as disk storage, display stations, or printers) that are not shared with other programs. Use this option if your program will be requested frequently.

If no option is specified, NONEP is assumed.

Maximum number of requesting display stations: Enter the number (0 through 99) of display stations that can use a single copy of the program at the same time. If no number is entered, a value of 0 is assumed (the program is not a MRT program).

Generate CONSOLE file display formats: Enter GEN or NOGEN.

GEN means that you want the procedure to create and compile source specifications for 24-line, 1920-character display formats for a CONSOLE file. Specifications are created and compiled only if the program contains no terminal errors.

NOGEN means that you do not want the procedure to create or compile the source specifications for the display formats for a CONSOLE file.

If no option is chosen, GEN is assumed. However, the procedure ignores this option if your program does not use a CONSOLE file.

Size of work files in blocks: Enter the number (1 through 9999) of blocks for the compiler work files. If no size is specified, 40 blocks is assumed. However, if the work files become full, they are automatically extended by the compiler.

You can use the following keys from the second RPGONL Procedure display:

- Command key 2 to return to the first display for the RPGONL procedure

- Command key 7 to end the RPGONL procedure

- The Help key for additional explanation of the parameters

What happens next depends on how you responded to the last prompt on the first display for this procedure. That prompt is *Create or change source before first compilation*?

If you responded Y to that prompt and now press the Enter key, the RPGONL procedure calls DSU or, if DSU is not installed on the system, SEU, and allows you to create or change your program before the first compilation. After you finish using DSU or SEU, your program is compiled.

If you responded N to that option and now press the Enter key, neither DSU nor SEU is called and you cannot create or change your source program before the first compilation.

During the compilation, a diagnosed source member is created. A *diagnosed source member* is a source member in which informational, warning, and terminal error messages diagnosed by the RPG compiler are inserted immediately before the specifications containing errors. The inserted messages have the following format:

- Columns 1 and 2 contain question marks.

- Column 3 contains I, W, or T to indicate that the message is an informational, warning, or terminal error message.

- Columns 4 and 5 are blank.

- Column 6 contains H, F, E, L, T, I, C, or O to indicate the specification type.

- Column 7 contains an asterisk to indicate that this line is a comment.

- Columns 8 through 80 contain the message.

When the compilation is complete, the editor (DSU or SEU) displays the diagnosed source member on your display screen. You can see the results of the compilation and can correct any errors diagnosed by the compiler.

When you finish correcting these errors, press command key 7 to end editing. If you chose to replace the source program, you will next see the DSU end-of-job display.

```
                    EXIT OPTIONS FOR SOURCE MEMBER


Type choices, press Enter.

ITEM                CHOICE          POSSIBLE CHOICES

Save member . . . . 1              1=Yes    2=No
Member name . . . . EXAMPLE1
Library name. . . . DSULIB
Reference number. . 000001         1-999999
Serialize member. . 2              1=Yes    2=No
Beginning column. . 001            1-93
Remove diagnostics..2              1=Yes    2=No
Subtype. . . . . . .RPG            UNS,ARP,ARS,ASM,BAP,BAS,BGC,BGD,BGF,
                                   COB,DFU,DTA,FMT,FOR,MNU,MSG,PHL,RPG,
                                   SRT,WSU

Print member . . . .2              1=Yes    2=No


Cmd3=Return to editing                   Cmd4=Display member list
Cmd6=Return to editing with processing   CMD7=Exit DSU
```

If DSU is not installed on your system, you will see the SEU end-of-job display:

```
                          END OF JOB




Member name       . . . . . . . . . . . . . . . . . .  TEST

Library name      . . . . . . . . . . . . . . . . . .  YOURLIB

Reference number . . . . . . . . . . . . . . . . . . .  000006

Library member subtype . . . . . . . . . . . . . . . .  35

  2 DTA   13 BAP   16 MNU   19 SRT   33 COB   36 WSU
 11 ARS   14 DFU   17 MSG   31 ASM   34 FOR   40 UNS
 12 ARP   15 FMT   18 PHL   32 BAS   35 RPG

Remove diagnostics from diagnosed source member? Y,N  Y
```

On both DSU and SEU end-of-job displays you can choose to have all of the RPG error messages removed from the source program before it is placed back into the source program library. You should consider the number of error messages, the size of the source program, and the amount of free space in the source program library before you replace the source program without removing the error messages. To remove the error messages, select the Y option for the *Remove diagnostics* prompt on the DSU end-of-job display, or for the *Remove diagnostics from diagnosed source member?* prompt on the SEU end-of-job display.

When you finish using the diagnosed source member, the third display for the RPGONL procedure is shown.

Using the Third RPGONL Display

The third RPGONL display looks like this:

```
                          RPGONL PROCEDURE

                        Continuation options

 Would you like to view the compiler listing? . . . Y,N         N

 Would you like to recompile the source program? . . Y,N        Y
 --------------------------------------------------------------------

 Source program  . . . . . . . . .  TEST
 Input library . . . . . . . . . .  YOURLIB
 Output library  . . . . . . . . .  YOURLIB
 Data dictionary . . . . . . . . .



 Cmd1-Work with different member  Cmd7-End
 Cmd14-Compiler options                               © 1985 IBM Corp.
```

Respond to each prompt by entering the appropriate information.

Would you like to view the compiler listing?: Enter Y or N.

Y (yes) means that you want the compiler listing to be displayed at your display station.

N (no) means that you do not want the compiler listing to be displayed at your display station.

If no option is specified, N is assumed.

Would you like to recompile the source program?: Enter Y or N.

Y (yes) means that you want to recompile your program.

N (no) means that you do not want to recompile your program.

If no option is specified, Y is assumed.

Command keys 1 and 14 override this option. If you press command key 1, the program is not recompiled, even if you entered Y in response to this prompt. Instead, the first RPGONL display for the current program appears, so you can enter a new program name and use the procedure with the new program. If you press command key 14, the second RPGONL display for the current program appears, so you can change any compiler options for the current program and recompile the program, even if you entered N in response to this prompt.

Press command key 7 to end the RPGONL procedure.

Press the Help key for additional information about the third RPGONL display.

If you respond Y to *Would you like to view the compiler listing?*, the display for the COPYPRT procedure appears. The COPYPRT procedure allows you to display and optionally print the compiler listing. The display for the COPYPRT procedure looks like this:

```
                              **COMPLETE**
NO. I/D    PROC    JOBNAME     USER    PRINTER    ID   FORM  PAGES   RECS
001 SP0061 RPGONL  W3145108    MSM     $PRINTDM   P2   0001  4       44







    SELECTED HEADER - 1            PRINT Y/N -  N              COPIES - 01
    FROM PAGE          TO PAGE             ENTER/HELP KEY  CMD 7 - END
```

To display the compiler listing, type the listed job number in response to the *Selected header* prompt at the bottom of the display and press the Enter key. To print the compiler listing, type the listed job number in response to the *Selected header* prompt, change the default N to Y on the *Print* prompt, and press the Enter key. For more information about the COPYPRT procedure, see the *System Reference* manual.

Press command key 7 to end the COPYPRT procedure.

You can continue using the RPGONL procedure until you press command key 7 while any of the three displays is shown or until you type N and press the Enter key in response to the second prompt on the third display, *Would you like to recompile the source program?* Each time you recompile your program, the parameters you specified the previous time you used the procedure are automatically entered for you in response to the new prompts.

Creating or Changing an RPG or Auto Report Program (RPGSEU Procedure)

The RPGSEU procedure allows you to use the source entry utility (SEU) to create or change an RPG program. For a complete explanation of SEU, see the *Source Entry Utility (SEU) Guide.*

The display for the RPGSEU procedure looks like this:

```
                         RPGSEU PROCEDURE                  Optional-*

                 Creates or updates an RPG II or auto report
                 procedure or source member with SEU.

Name of member to be created or updated. . . . . . . . .  TEST

Type of member. . . . . . . . . . . . . . . . . . R,A,P   R

Name of member containing SEU formats. . . . . . . . . .  #SE@XTRA

Length of statement. . . . . . . . . . . . . . . .40-120          *

Name of library containing member. . . . . . . . . . . .  YOURLIB




CMD3-Previous menu
                                                  COPR IBM Corp. 1983
```

Respond to each prompt by entering the appropriate information.

Name of member to be created or updated: Enter the name of the library member to be created or changed.

Type of member: Enter R, A, or P.

R means an RPG source member.

A means an RPG source member containing auto report specifications.

P means a procedure member.

If no option is specified, R is assumed.

Name of member containing SEU formats: Enter the name of the load member that contains SEU formats. If no option is specified, the default for SEU is assumed.

Length of statement: Enter the maximum length allowed for each source or procedure statement. If the member exists, the statement length of the member is assumed. If the member is being created and if no statement length is specified, the values that are allowed and assumed are as follows:

Member Type	**Allowed Statement Length**	**Assumed Statement Length**
R	80 through 96	96
A	80 through 96	96
P	40 through 120	120

Name of library containing member: Enter the name of the library that contains or is to contain the member being created or changed. If no library name is specified, the name of the current library is assumed.

You can use the following keys from the RPGSEU procedure display:

- Command key 3 to return to the previous Help menu
- Command key 7 to end the RPGSEU procedure
- The Help key for additional information about the parameters

Compiling an RPG Program (RPGC Procedure)

The RPGC procedure compiles an RPG source program. The RPGC procedure has two displays.

Using the First RPGC Display

The first display for the RPGC procedure looks like this:

```
                          RPGC PROCEDURE                   Optional-*

                 Compiles an RPG II source program.

Name of source program to be compiled. . . . . . . . . . . .   TEST

Name of library containing source program. . . . . . . . . .   YOURLIB

Create diagnosed source member. . . . . . . . . . . NODSM,DSM   NODSM

Output option for compiler listings. . . .PRINT,NOPRINT,CRT   PRINT

Create cross-reference listing. . . . . . . . . . NOXREF,XREF   NOXREF

Maximum number of requesting display stations. . . . . 0-99   00

Never-ending program. . . . . . . . . . . . . . . . NONEP,NEP   NONEP

Name of library to contain compiled program. . . . . . . . .          *

Cmd3-Previous menu  Cmd4-Put on job queue  Cmd14-More options
                                                  COPR IBM Corp. 1986
```

Respond to each prompt by entering the appropriate information.

Name of source program to be compiled: Enter the name of your source program.

Name of library containing source program: Enter the name of the library that contains the source member to be compiled. If no library name is specified, the current library is assumed.

Create diagnosed source member: Enter NODSM or DSM.

NODSM means that you do not want the RPG compiler to create a diagnosed source member.

DSM means that you do want a diagnosed source member. A diagnosed source member is a source member in which informational, warning, and terminal error messages diagnosed by the RPG compiler are inserted immediately before the specifications containing errors. The inserted messages have the following format:

Columns 1 and 2 contain question marks.

Column 3 contains I, W, or T to indicate that the message is an informational, warning, or terminal error message.

Columns 4 and 5 are blank.

Column 6 contains H, F, E, L, T, I, C, or O to indicate the specification type.

Column 7 contains an asterisk to indicate that this line is a comment.

Columns 8 through 80 contain the message.

When the RPGC procedure is complete, you can use SEU to correct the specifications containing errors or to remove the error messages from the source member.

If no option is specified, NODSM is assumed.

Output option for compiler listings: Enter PRINT, NOPRINT, or CRT.

PRINT means that you want the compiler listing created by the RPGC procedure to be printed.

NOPRINT means that you do not want the compiler listing to be printed or displayed.

CRT means that you want the compiler listing to be displayed at the display station that requested the RPGC procedure.

If no option is specified, PRINT is assumed.

Create cross-reference listing: Enter NOXREF or XREF.

NOXREF means that you do not want the RPGC procedure to create a cross-reference listing for the program.

XREF means that you do want a cross-reference listing to be created. The cross-reference listing is created only if the program contains no terminal errors. The cross-reference listing is part of the compiler listing, so whether the cross-reference listing is displayed or printed depends on your response to the preceding prompt.

If no option is specified, NOXREF is assumed.

Maximum number of requesting display stations: Enter the number (0 through 99) of display stations that can use a single copy of the program at the same time. If no number is entered, a value of 0 is assumed (the program is not a MRT program).

Never-ending program: Enter NONEP or NEP.

NONEP means that the program is not to be a never-ending program.

NEP means that the program is to be a never-ending program. A never-ending program is one that uses system resources (such as disk storage, display stations, or printers) that are not shared with other programs. Use this option if your program will be requested frequently.

If no option is specified, NONEP is assumed.

Name of library to contain compiled program: Enter the name of the library that is to contain the compiled program. If this option is not specified, the source input library is assumed.

You can use the following keys from the first RPGC Procedure display:

- Command key 3 to return to the previous Help menu
- Command key 4 to place the program on the input job queue
- Command key 7 to end the RPGC procedure
- The Help key for additional information about the parameters
- Command key 14 to see the second display for the RPGC procedure

Using the Second RPGC Display

The second RPGC display looks like this:

Procedure Name and Parameters Specified on the First Display

```
                              RPGC PROCEDURE                          Optional-*

RPGC TEST,YOURLIB,NODSM,PRINT,NOXREF,0,NONEP,,

Override print option in source . . SOURCE,PSOURCE,NOSOURCE                     *
Override debug option in source . . . . . . . DEBUG,NODEBUG                     *
Override size-to-execute option in source . . . . . .  2-64                     *
Halt on serious program error . . . . . . . . . NOHALT,HALT  NOHALT
Replace duplicate members . . . . . . . .  REPLACE,NOREPLAC  REPLACE
Create program that can be run  . . . . . . . . LINK,NOLINK  LINK
Create program that must be
  link-edited . . . . . . . . . . . . . . . NOOBJECT,OBJECT  NOOBJECT
Name of subroutine input library  . . . . . . . . . . . . .                     *
Generate CONSOLE file display formats . . . . . . . GEN,NOGEN  GEN
Size of work files in blocks  . . . . . . . . . . .  1-9999  40
Name of data dictionary to be used  . . . . . . . . . . . .                     *
Create program with
  memory resident overlays . . . . . . . . . . . NOMRO,MRO  NOMRO


Cmd2-Page back      Cmd4-Put on job queue
                                                     COPR IBM Corp. 1986
```

Respond to each prompt by entering the appropriate information.

Override print option in source: Enter SOURCE, PSOURCE, or NOSOURCE. Use this option to override an entry in column 11 of the control specification of your RPG program.

SOURCE means that you want the RPG compiler to print a full compiler listing. A full compiler listing includes the source program, information about tables and arrays, the relative location of fields and their attributes, field names that are not referred to, diagnostics, and a map of main storage. The map of main storage lists the identification, the starting address, and the size of each separately identifiable segment of code in the program, tells how much main storage is required to run the program, and lists the number of library sectors required for the program.

PSOURCE means that you want the RPG compiler to print a partial compiler listing. A partial listing includes the source program, information about indicators used in the program, diagnostics, and the following information from the overlay linkage editor: the amount of main storage required to run the program, the starting address of the program, and the number of library sectors required for the program. It does not include information about tables and arrays, information about fields, or a map of main storage.

NOSOURCE means that you do not want the RPG compiler to print a compiler listing. If you use this option, a prolog is printed along with the following information from the overlay linkage editor: the amount of main storage required to run the program, the starting address of the program, and the number of library sectors required for the program. Use this option when you already have a listing of the program.

If no option is specified, the entry coded in column 11 of the control specification of your RPG program is used.

Override debug option in source: Enter DEBUG or NODEBUG. Use this option to override the entry in column 15 of the control specification of your RPG program.

DEBUG means that the DEBUG operation is to be used in the calculation specifications.

NODEBUG means that the DEBUG operation is not to be used.

If no option is specified, the entry coded in column 15 of the control specification of your RPG program is used.

Override size-to-execute option in source: Enter the program size in K bytes (one K byte equals 1024 bytes). The program size must be an even number from 2 through 64. Use this option to override the entry coded in columns 12 through 14 of the control specification. If no size is specified, the entry coded in columns 12 through 14 of the control specification is used.

Halt on serious program error: Enter NOHALT or HALT.

NOHALT means that you do not want the compiler to stop and display an error message if a warning or terminal error is found in the program.

HALT means that you want the compiler to stop and display an error message if a warning or terminal error is found in the program.

If no option is specified, NOHALT is assumed.

Replace duplicate members: Enter REPLACE or NOREPLAC.

REPLACE means that, if a load or subroutine member is being created and if a load or subroutine member with the same name already exists in the output library, you want the newly compiled program to replace the existing load or subroutine member.

NOREPLAC means that, if a load or subroutine member is being created and if a load or subroutine member with the same name already exists in the output library, you want an error message to be displayed.

If no option is specified, REPLACE is assumed.

Create program that can be run: Enter LINK or NOLINK.

LINK means that you want to create a load module, that is, a program that you can run without first having to use the Overlay Linkage Editor procedure OLINK to link-edit the program.

NOLINK means that you do not want to create a load module.

If no option is entered, LINK is assumed.

Create program that must be link-edited: Enter NOOBJECT or OBJECT.

NOOBJECT means that you do not want to create an object module.

OBJECT means that you want to create an object module, that is, a compiled *subroutine* (not a load module). You must use the OLINK procedure to link-edit this module (by itself or with one or more other assembler programs or subroutines) before you can run the module.

If no option is specified, NOOBJECT is assumed.

Create memory-resident overlays: Enter NOMRO or MRO.

NOMRO means that you do not want overlays to remain in memory.

MRO means that you want to retain overlays in memory. This enables you to keep more than one overlay in memory.

The default is NOMRO.

Name of subroutine input library: Enter the name of the library that contains one or more assembler subroutines to be combined with the program being compiled. If no library name is specified, the name of the source member library is assumed.

Generate CONSOLE file display formats: Enter GEN or NOGEN.

GEN means that you want the procedure to create and compile source specifications for 24-line, 1920-character display formats for a CONSOLE file. The specifications are created and compiled only if the program contains no terminal errors.

NOGEN means that you do not want the procedure to create or compile the source specifications for the display formats for a CONSOLE file.

If no option is specified, GEN is assumed. However, the procedure ignores this option if your program does not use a CONSOLE file.

Size of work files in blocks: Enter the number (1 through 9999) of blocks for the compiler work files. If no size is specified, 40 blocks is assumed. However, if the work files become full, they are extended automatically by the compiler.

You can use the following keys from the second RPGC Procedure display:

- Command key 2 to return to the first display for the RPGC procedure
- Command key 4 to place the program on the input job queue
- Command key 7 to end the RPGC procedure
- The Help key for additional information about the parameters

Name of the data dictionary to be used: Enter the name of the current data dictionary to be used during the compilation, if you are using communication formats defined through the interactive data definition utility (IDDU). There is no default value for the data dictionary.

Printing an RPG Cross-Reference Listing (RPGX Procedure)

The RPGX procedure prints a cross-reference listing for an RPG program. No diagnostic checking is provided with the RPGX procedure. Therefore, you should use this command only for RPG source programs that have been successfully compiled and for which object programs have been produced. Unpredictable or confusing results may occur if auto report source statements or RPG source statements containing errors are used as input to the RPGX procedure. The display for the RPGX procedure looks like this:

```
                        RPGX PROCEDURE
           Requests cross-reference for RPG II source program.

Name of source program . . . . . . . . . . . . . . . . . . . . . . TEST
Size of $SOURCE file in blocks . . . . . . . . . . . . .  1-9999   40
Name of library containing source program  . . . . . . . . . . . YOURLIB




Cmd3-Previous menu    Cmd4-Put on job queue
                                                 COPR IBM Corp. 1983
```

Respond to the prompts by entering the appropriate information.

Name of source program: Enter the name of your source program.

Size of $SOURCE file in blocks: Enter the number (1 through 9999) of blocks for the work files. If no number is entered, 40 blocks is assumed. However, if the work files become full, they are extended automatically by the compiler.

Name of library containing source program: Enter the name of the library that contains the source program to be listed. If no library name is specified, the name of the current library is assumed.

You can use the following keys from the RPGX Procedure display:

- Command key 3 to return to the previous Help menu
- Command key 4 to place the program on the input job queue
- Command key 7 to end the RPGX procedure
- The Help key for additional information about the parameters

Cross-Reference Listing

The RPGX procedure or the XREF option in the RPGC and AUTOC procedures provide a cross-reference listing of the symbols defined and referenced in the respective RPG and autoreport source programs. The cross-reference listing can be very helpful when you are modifying or expanding your program. The execution of the cross-reference listing step in the RPGC or AUTOC procedure depends upon the following:

- The listing is provided only when XREF is specified for the RPGC or AUTOC procedure. The default is no cross-reference listing (NOXREF).
- The listing is not provided if terminal errors occur in the RPG or auto report compilation.
- The Sort utility is required to sort the symbol entries and provide a cross-reference listing.

The symbols used in an RPG or auto report program are sorted and placed in the following categories in the cross-reference listing:

- Filenames
- Indicators
- Tables and arrays
- Fields and data structures
- Labels

Listing Format

The format of the cross-reference listing is as follows:

```
SYMBOL    LNG   TYPE     DEC  DEFN  REFERENCES
x------x  xxxx  xx----x   x   xxxx  xxxx xxxx xxxx*
```

where:

SYMBOL is from 1 to 8 characters in length and defines the filenames, indicators, tables/arrays, data structures, fields, and labels used in the RPG or auto report program. Alphameric and numeric literals are not processed by the cross-reference listing option. When you are using a continuation line option on a file description specification, anything that is not a field name will not show up in the SYMBOL column. However the keyword of the continuation line option will appear (FMTS, for example).

LNG is four positions long and defines the length of the field or data structure, the length of an element in a table or array, or the record length for the file named. LNG is not used for indicators or labels. LNG is also not used when a field has been defined by the *LIKE DEFN operation code. In this case, the length of the field shows as

****. For a data structure, the length of the data structure shows as *DS*.

TYPE is 2 to 7 positions in length and defines the type of file named (by using columns 15 and 16 from the file description specifications) or the type of label being defined and referenced. TYPE is used only for filenames or labels.

DEC is one position long and defines the number of decimal positions in a numeric field. DEC is not used for filenames, alphameric fields, indicators, data structures, or labels.

DEFN is four positions long and defines the statement number in which the symbol is defined. If the symbol is defined multiple times in the program, the first definition is assumed. The use of a field in a data structure is considered to be the definition of that field; all other uses of that field are considered to be references. The definition of an array is considered to be in the extension specification specifying the array even if the array is also specified in a data structure.

REFERENCES are four positions in length and define the statement number in which the symbol is referenced. The number of entries under REFERENCES depends on the number of times the symbol is used in the program. If the symbol is unreferenced, there are no entries. If the symbol is referenced multiple times, multiple lines of references could be printed for the related symbol. An asterisk (*) printed beside a reference indicates that the contents of the symbol are, or could be, altered in this statement. An asterisk indicates that a field is used as a calculation result field and the operation code is not DEFN, or that an indicator is specified in positions 59 through 70 of the input specifications or in positions 54 through 59 of the calculation specifications.

Sample Cross-Reference Listing

The information that is printed in the cross-reference listing for each symbol type looks like this:

```
   ** FILENAME LEGEND **

SYMBOL   LNG   TYPE DEFN  REFERENCES


CONTROL  0030   UC  0001 0005   0021   0045
WORKSTN  0030   CP  0002 0009   0043


   ** INDICATOR LEGEND **

SYMBOL              DEFN  REFERENCES


01                  0005
02                  0009
03                  0011 0020
20                  0032 0035*  0045
21                  0038 0041*  0047
99                  0021 0028   0029


   ** TABLE AND ARRAY LEGEND **

SYMBOL   LNG    DEC DEFN  REFERENCES


AO8      0005    0  0003 0023*  0023   0024  0027*  0027   0027
ARY      0015       0004 0024*  0026*


   ** FIELD AND DATA STRUCTURE LEGEND **

SYMBOL   LNG    DEC DEFN  REFERENCES


CENTS    0002    0  0015
COST     0007    2  0018 0007   0039*  0039
DESC     0018       0019 0008   0033*
DOLLAR   0005    0  0014
INVDTA   *DS*       0012 0010   0046   0048
IX       0001    0  0022 0023   0023   0024  0024   0025*  0025  0026  0027
NAME     0008       0016
PARTNO   0005    0  0013 0006   0021
STOCK    0010       0017


   ** LABEL LEGEND **

SYMBOL      TYPE    DEFN  REFERENCES


ADDRCD     BEGSR    0031 0028
FND        TAG      0030 0020
UPDRCD     BEGSR    0037 0029
```

Compiling an Auto Report Program (AUTOC Procedure)

The AUTOC procedure compiles an RPG program that contains auto report specifications. The AUTOC procedure has two displays.

Using the First AUTOC Display

The first AUTOC display looks like this:

```
                        AUTOC PROCEDURE                         Optional-*
  Compiles an RPG II program that contains auto report specifications.

Name of source program to be compiled . . . . . . . . . . . . . TEST
Name of library containing source program . . . . . . . . . .   YOURLIB
Call RPG II compiler . . . . . . . . . . . . . . . . COMP,NOCOMP  COMP
Output option for compiler listings . . . . .PRINT,NOPRINT,CRT  PRINT
Create cross-reference listing . . . . . . . . . . . NOXREF,XREF  NOXREF
Maximum number of requesting display stations . . . . . . 0-99  00
Never-ending program . . . . . . . . . . . . . . . . . NONEP,NEP  NONEP
Name of library to contain compiled program . . . . . . . . .         *

Cmd3-Previous menu   Cmd4-Put on job queue   Cmd14-More options
                                                  COPR IBM Corp. 1986
```

Respond to each prompt by entering the appropriate information.

Name of source program to be compiled: Enter the name of your source program.

Name of library containing source program: Enter the name of the library that contains the source member to be compiled. If no library name is specified, the name of the current library is assumed.

Call RPG II compiler: Enter COMP or NOCOMP.

COMP means that you want the RPG compiler to be run as part of the auto report function.

NOCOMP means that you do not want the RPG compiler to be run as part of the auto report function.

If no option is specified, COMP is assumed.

Output option for compiler listings: Enter PRINT, NOPRINT, or CRT.

PRINT means that you want the listings created by the AUTOC procedure to be printed.

NOPRINT means that you do not want the listings to be printed or displayed.

CRT means that you want the listings created by the AUTOC procedure to be displayed at the display station that requested the AUTOC procedure.

If no option is specified, PRINT is assumed.

Create cross-reference listing: Enter NOXREF or XREF.

NOXREF means that you do not want the AUTOC procedure to create a cross-reference listing for the RPG program.

XREF means that you do want a cross-reference listing. The cross-reference listing is created only if the program contains no terminal errors. The cross-reference listing is part of the compiler listing, so whether the cross-reference listing is displayed or printed depends on your response to the previous prompt.

If no option is entered, NOXREF is assumed.

Maximum number of requesting display stations: Enter the number (0 through 99) of display stations that can use a single copy of the program at the same time. If no number is specified, a value of 0 is assumed (the program is not a MRT program).

Never-ending program: Enter NONEP or NEP.

NONEP means that the program is not to be a never-ending program.

NEP means that the program is to be a never-ending program. A never-ending program is one that uses system resources (such as disk storage, display stations, or printers) that are not shared with other programs. Use this option if your program will be requested frequently.

If no option is entered, NONEP is assumed.

Name of library to contain compiled program: Enter the name of the library that is to contain the compiled program. If no library name is specified, the name of the source input library is assumed.

You can use the following keys from the first AUTOC Procedure display:

- Command key 3 to return to the previous Help menu
- Command key 4 to place the program on the input job queue
- Command key 7 to end the AUTOC procedure
- The Help key for additional information about the parameters
- Command key 14 to see the second display for the AUTOC procedure

Using the Second AUTOC Display

The second AUTOC display looks like this:

Procedure Name and Parameters Specified on the First Display

```
                              AUTOC PROCEDURE                    Optional-*

AUTOC TEST,YOURLIB,COMP,PRINT,NOXREF,0,NONEP,,

Override print option in source . . SOURCE,PSOURCE,NOSOURCE                 *
Override debug option in source . . . . . . . DEBUG,NODEBUG                 *
Override size-to-execute option in source . . . . . .  2-64                 *
Halt on serious program error . . . . . . . . . NOHALT,HALT  NOHALT
Replace duplicate members . . . . . . . .  REPLACE,NOREPLAC  REPLACE
Create program that can be run  . . . . . . . . LINK,NOLINK  LINK
Create program that must be
  link-edited . . . . . . . . . . . . . . . NOOBJECT,OBJECT  NOOBJECT
Name of subroutine input library  . . . . . . . . . . . . .                 *
Generate CONSOLE file display formats . . . . . . GEN,NOGEN  GEN
Size of work files in blocks  . . . . . . . . . . .  1-9999  40
Name of data dictionary to be used  . . . . . . . . . . . .                 *
Create program with
  memory resident overlays . . . . . . . . . . . NOMRO,MRO   NOMRO


Cmd2-Page back     Cmd4-Put on job queue
                                                 COPR IBM Corp. 1986
```

Respond to each prompt by entering the appropriate information.

Override print option in source: Enter SOURCE, PSOURCE, or NOSOURCE. Use this option to override an entry in column 11 of the control specification of your RPG program.

SOURCE means that you want the RPG compiler to print a full compiler listing. A full compiler listing includes the source program, information about tables and arrays, the relative location of fields and their attributes, field names that are not referred to, diagnostics, and a map of main storage. The map of main storage lists the identification, the starting address, and the size of each separately identifiable segment of code in the program, tells how much main storage is required to run the program, and lists the number of library sectors required for the program.

PSOURCE means that you want the RPG compiler to print a partial compiler listing. A partial listing includes the source program, information about indicators used in the program, diagnostics, and the following information from the overlay linkage editor: the amount of main storage required to run the program, the starting address of the program, and the number of library sectors required for the program. It does not include information about tables and arrays, information about fields, or a map of main storage.

NOSOURCE means that you do not want the RPG compiler to print a compiler listing. If you use this option, a prolog is printed along with the following information from the overlay linkage editor: the amount of main storage required to run the program, the starting address of the program, and the number of library sectors required for the program. Use this option when you already have a listing of the program.

If no option is specified, the entry coded in column 11 of the control specification of your RPG program is used.

Override debug option in source: Enter DEBUG or NODEBUG. Use this option to override the entry in column 15 of the control specification of your RPG program.

DEBUG means that the DEBUG operation is to be used in the calculation specifications.

NODEBUG means that the DEBUG operation is not to be used.

If no option is specified, the entry coded in column 15 of the control specification of your RPG program is used.

Override size-to-execute option in source: Enter the program size in K bytes (one K byte equals 1024 bytes). The program size must be an even number from 2 through 64. Use this option to override the entry coded in columns 12 through 14 of the control specification of your RPG program. If no size is specified, the entry coded in columns 12 through 14 of the control specification of your RPG program is used.

Halt on serious programming error: Enter NOHALT or HALT.

NOHALT means that you do not want the compiler to stop and display an error message if a warning or terminal error is found in the program.

HALT means that you want the compiler to stop and display an error message if a warning or terminal error is found in the program.

If no option is specified, NOHALT is assumed.

Replace duplicate members: Enter REPLACE or NOREPLAC.

REPLACE means that, if a load or subroutine member is being created and if a load or subroutine member with the same name already exists in the output library, you want the newly compiled program to replace the existing load or subroutine member.

NOREPLAC means that, if a load or subroutine member is being created and if a load or subroutine member with the same name already exists in the output library, you want an error message to be displayed.

If no option is specified, REPLACE is assumed.

Create program that can be run: Enter LINK or NOLINK.

LINK means that you want to create a load module, that is, a program that you can run without first having to use the Overlay Linkage Editor procedure OLINK to link-edit the program.

NOLINK means that you do not want to create a load module.

If no option is entered, LINK is assumed.

Create program that must be link-edited: Enter NOOBJECT or OBJECT.

NOOBJECT means that you do not want to create an object module.

OBJECT means that you want to create an object module, that is, a compiled *subroutine* (not a load module). You must use the OLINK procedure to link-edit this module (by itself or with one or more other assembler programs or subroutines) before you can run the module.

If no option is specified, NOOBJECT is assumed.

Create memory-resident overlays: Enter NOMRO or MRO.

NOMRO means that you do not want overlays to remain in memory.

MRO means that you want to retain overlays in memory. This enables you to keep more than one overlay in storage.

The default is NOMRO.

Name of subroutine input library: Enter the name of the library that contains one or more subroutines to be combined with the program being compiled. If no library name is specified, the name of the source input library is assumed.

Generate CONSOLE file display formats: Enter GEN or NOGEN.

GEN means that you want the procedure to create and compile source specifications for 24-line, 1920-character display formats for CONSOLE files. The specifications are created and compiled only if the program contains no terminal errors.

NOGEN means that you do not want the procedure to create or compile the source specifications for the display formats.

If no option is specified, GEN is assumed. However, the procedure ignores this option if your program does not use a CONSOLE file.

Size of work files in blocks: Enter the number (1 through 9999) of blocks for the compiler work files. If no size is specified, 40 blocks is assumed.

You can use the following keys from the second AUTOC Procedure display:

- Command key 2 to return to the first display for the AUTOC procedure
- Command key 4 to place the program on the input job queue
- Command key 7 to end the AUTOC procedure
- The Help key for additional information about the parameters

Name of the data dictionary to be used: Enter the name of the current data dictionary to be used during the compilation, if you are using communication formats defined through the interactive data definition utility (IDDU). There is no default value for the data dictionary.

Creating or Changing Display Formats (RPGSDA Procedure)

The RPGSDA procedure allows you to create or change display formats. There are no displays for the RPGSDA procedure. Instead, the procedure calls the screen design aid (SDA) utility, and the SDA menu is displayed:

```
                           SDA MAIN OPTIONS                  W4
Select one of the following:


  MENUS            1.  Design menus

  DISPLAYS         2.  Design display formats
                   3.  Design display formats for WSU

  PROGRAMS         4.  Build RPG II WORKSTN file specifications
                   5.  Build WSU programs

  SERVICES         6.  EDIT      Source and procedure members
                   7.  VIEW      Display formats in $SFGR load members
                   8.  PRINT     Display formats in source members
                   9.  COMPILE   Source format members with $SFGR

Option . . . . .


HELP-Cursor selected help  CMD3-Previous menu  CMD7-End SDA session
```

For a complete explanation of SDA, see the manual *Creating Displays.*

Solving Problems That Occur At Compilation Time

The following charts describe some problems that may occur when you compile your program and some possible ways to solve these problems.

No Compiler Listing Is Produced

Items to Check	How to Check the Item	Recommended Recovery Action
You coded B in column 11 of the control (H) specification.	Use DSU/SEU or RPGSEU procedure to look at the source program specifications.	Either use DSU/SEU to change the B to blank, or use the RPGONL, RPGC, or AUTOC procedure to recompile your program, and choose SOURCE or PSOURCE as the *Override print option in source* parameter.
You chose NOSOURCE as the *Override print option in source* parameter on the RPGONL, RPGC, or AUTOC procedure.	Look at the prolog on the compiler listing to see which option you chose.	Use the RPGONL, RPGC, or AUTOC procedure to recompile your program, and change NOSOURCE to PSOURCE, SOURCE, or blank as the *Override print option in source* parameter.
You chose N in response to the prompt *Would you like to view the compiler listing*? on the RPGONL procedure.	This item cannot be checked.	Change N to Y as the response to the prompt *Would you like to view the compiler listing*? on the RPGONL procedure.

No Compiler Listing Is Produced (continued)

Items to Check	How to Check the Item	Recommended Recovery Action
You chose NOPRINT as the *Output options for compiler listing* parameter on the RPGC or AUTOC procedure.	This item cannot be checked.	Use the RPGC or AUTOC procedure to recompile your program, and change NOPRINT to PRINT or CRT as the *Output option for compiler listings* parameter.
Your display station is configured to the wrong printer.	Use the SSP STATUS command (D W) to see how the system is configured.	Reconfigure the display station to the correct printer.
The spool writer is stopped.	Use the SSP STATUS command (D P) at the system console to see the status of the spool file.	Use the SSP START command (S P,printer-id) at the system console.

No Load Module Is Produced

Items to Check	How to Check the Item	Recommended Recovery Action
You chose NOLINK as the *Create program that can be run* parameter on the RPGONL, RPGC, or AUTOC procedure.	Look at the prolog on the compiler listing to see what option you chose.	Use the RPGONL, RPGC, or AUTOC procedure to recompile your program, and change NOLINK to LINK as the *Create program that can be run* parameter.
You used the wrong name as the subroutine input library name on the RPGONL, RPGC, or AUTOC procedure.	Look at the prolog on the compiler listing to see what subroutine input library name you used.	Use the RPGONL, RPGC, or AUTOC procedure to recompile your program, and change the name of the subroutine input library.

A Load Module Is Produced but Cannot Be Found

Items to Check	How to Check the Item	Recommended Recovery Action
You did not code a program identification name in columns 75 through 80 of the control (H) specification.	Look at the compiler listing.	Do one of the following: 1. Use DSU/SEU to code a program identification name in columns 75 through 80 of the control specification, and then use the RPGONL, RPGC, or AUTOC procedure to recompile your program. 2. Use the SSP LIBRLIBR procedure to change the default name of the load module (RPGOBJ) to the required name.

Items to Check	How to Check the Item	Recommended Recovery Action
The name of the library in which you are looking is not the same as the name you used as the output library name on the RPGONL, RPGC, or AUTOC procedure.	Look at the prolog on the compiler listing to see what name you used as the output library name.	Do any one of the following: 1. Look in the library you used as the output library name on the RPGONL, RPGC, or AUTOC procedure. 2. Use the RPGONL, RPGC, or AUTOC procedure to recompile your program, and change the name of the output library. 3. Use the SSP LIBRLIBR procedure to copy the load module from the library you used as the output library name on the RPGONL, RPGC, or AUTOC procedure to the required library.

No Subroutine Module Is Produced

Items to Check	How to Check the Item	Recommended Recovery Action
You chose NOOBJECT as the *Create program that must be link-edited* parameter on the RPGONL, RPGC, or AUTOC procedure.	Look at the prolog on the compiler listing to see what option you chose.	Use the RPGONL, RPGC, or AUTOC procedure to recompile your program, and change NOOBJECT to OBJECT as the *Create program that must be link-edited* parameter.
You used the wrong name as the subroutine input library name on the RPGONL, RPGC, or AUTOC procedure.	Look at the prolog on the compiler listing to see what subroutine input library name you used.	Use the RPGONL, RPGC, or AUTOC procedure to recompile your program, and change the name of the subroutine input library.

A Subroutine Module Is Produced but Cannot Be Found

Items to Check	How to Check the Item	Recommended Recovery Action
You did not code a program identification name in columns 75 through 80 of the control (H) specification.	Look at the compiler listing.	Do one of the following: 1. Use SEU to code a program identification name in columns 75 through 80 of the control specification, and then use the RPGONL, RPGC, or AUTOC procedure to recompile your program. 2. Use the SSP LIBRLIBR procedure to change the default name of the subroutine module (RPGOBJ) to the required name.

A Subroutine Module Is Produced but Cannot Be Found (continued)

Items to Check	How to Check the Item	Recommended Recovery Action
The name of the library in which you are looking is not the same as the name you used as the output library name on the RPGONL, RPGC, or AUTOC procedure.	Look at the prolog on the compiler listing to see what name you used as the output library name.	Do any one of the following: 1. Look in the library you used as the output library name on the RPGONL, RPGC, or AUTOC procedure. 2. Use the RPGONL, RPGC, or AUTOC procedure to recompile your program, and change the name of the output library. 3. Use the SSP LIBRLIBR procedure to copy the load module from the library you used as the output library name on the RPGONL, RPGC, or AUTOC procedure.

No Diagnosed Source Member Is Produced

Items to Check	How to Check the Item	Recommended Recovery Action
You chose NODSM as the *Create diagnosed source member* parameter on the RPG procedure.	Look at the prolog on the compiler listing to see what option you chose.	Use the RPGC procedure to recompile your program, and change NODSM to DSM as the *Create diagnosed source member* parameter.

If your problem is still not solved, please refer to Chapter 32 (*Problem Determination*) in this manual.

Chapter 4. Testing an RPG Program

Chapter 4. Testing an RPG Program

Testing is the final step in creating an RPG program. Testing means running your load module with sample data, not actual data. You should test all possible combinations of variables in your program to be sure that the program processes the data correctly in every case. Testing usually reveals some program errors that must be corrected before you can use the load module to process your actual data. Finally, after this debugging (finding and correcting all errors), the program is ready to be put into production by running it with actual data.

This chapter explains how to run a load module and briefly suggests how to debug a load module.

Running an RPG Load Module

There are three ways to run an RPG load module:

- Enter control language LOAD and RUN statements from the display station keyboard. If the program uses DISK files, you must also include a control language FILE statement for each DISK file. You can use a control language SWITCH statement to set any external indicators (U1 through U8) used by the program. To attach a display station to a program that uses a WORKSTN file, you can use a control language WORKSTN statement.

- Enter the name of a procedure that contains the required control language statements.

- Select a menu option.

For a complete explanation of the control language statements and of how to write a procedure, see the *System Reference* manual.

For information on how to place the program on the input job queue, see the manual *Operating Your Computer*.

Example of Control Language Statements To Run a Program

The following control language statements load and run an RPG load module named PROG1 that uses an input DISK file named INPUT and an output DISK file named OUTPUT:

```
// LOAD PROG1
// FILE NAME-INPUT
// FILE NAME-OUTPUT,BLOCKS-10
// RUN
```

RPG Halt Messages

Errors in an RPG program can cause the program to halt while it is being compiled or run. When the program halts, a halt message is displayed. If the program is run from the input job queue, the halt messages are displayed at the system console. If the program is run from a display station (and not placed on the input job queue), the halt messages are displayed at the display station. If the program is a multiple requester terminal (MRT) program, the messages go to the system console.

When a halt message is displayed, the person at the system console or at the display station must respond by entering one of the following options:

- *0-Continue*: Control is returned to the program, and processing continues.

- *1-Bypass*: The remainder of the program cycle is bypassed, and the next record is read. For some messages, option 1 means that you should try the operation again. This information is in the second-level text for these messages.

- *2-Controlled Cancel*: End-of-job operations specified by the program are done, tables are dumped, and file labels are cataloged.

- *3-Immediate Cancel*: The job is canceled, but control is not returned to the RPG program. New data entered for this job is not preserved.

For a complete explanation of the halt messages and of the necessary responses, see the *RPG II Messages* manual.

Debugging an RPG Load Module

You can use the DEBUG operation to debug any RPG program. In addition, you can use other techniques to debug a program that uses a WORKSTN file.

Using the DEBUG Operation

The DEBUG operation is an RPG function that helps you find errors in a program. This operation causes one or, optionally, two records to be written to an output file. The first record contains a list of all indicators that are on at the time the DEBUG operation occurs in the calculation specifications. If you code the name of a field or array in the result field of the DEBUG operation, a second record is also written to the output file. The second record shows the contents of the field or array specified in the result field.

The DEBUG operation can be coded at any point or at several points in the calculation specifications. The output records are written whenever the DEBUG operation occurs.

Factor 1 of the DEBUG operation can contain a literal or the name of a field. The literal or the contents of the specified field are written in the first record. If factor 1 is left blank, the statement number of the DEBUG operation is written in the first record.

Factor 2 must contain the name of the output file to which the records are written. The file cannot be a WORKSTN file. The same filename must be used as factor 2 for all DEBUG statements in a program.

The result field can contain the name of a field or array whose contents are written in the second output record. If the result field is left blank, only one record is written when that DEBUG operation occurs.

To use the DEBUG operation, you must also code a 1 in column 15 of the control specification. If you leave that column blank, the DEBUG operation is treated as a comment. You can override the entry in column 15 by specifying DEBUG or NODEBUG in the RPGONL, RPGC, or AUTOC procedure (see Chapter 3, *Entering and Compiling an RPG Program*).

Records Written by the DEBUG Operation

The DEBUG operation always causes at least one record to be written. That record has the following format:

Output Positions	Information
1-8	DEBUG-
9-16	Literal or contents of field coded in factor 1 (optional), or the statement number of the DEBUG operation code in the program.
17	Blank
18-32	INDICATORS ON-
33-any position (depending on length of output record)	The names of all indicators that are on, each separated by a blank. More than one record may be needed.

The second record is written only when an entry is coded in the result field. The second record has the following format:

Output Positions	Information
1-14	FIELD VALUE-
15-any position (depending on length of field)	The contents of the result field (up to 256 characters). If the result field is an array, more than one output record may be needed to contain the array.

Debugging a Program That Uses a WORKSTN File

Because the logic for WORKSTN file processing is supplied by both the RPG program and the display format specifications, it may be more difficult to find coding errors for the WORKSTN file than for other files. The following techniques may help you debug a WORKSTN program:

- Always compare the $SFGR listing to the RPG input and output specifications. The From, To, and End positions used on the RPG specifications should normally match the From, To, and End positions listed for the $SFGR input and output buffers.

- If the wrong format is displayed, check the status of the indicators to be certain the status is as you expected. If the status of the indicators is incorrect, the wrong format may be displayed or a correct format may be followed by an additional format that overlays and thereby hides the correct format. The specification of erase input (columns 31 and 32 of the S specifications) or override fields (columns 33 and 34 of the S specifications) may also cause a partial format to be displayed that overlays the correct format.

- Always include a record type for blank records. Blank records can occur in one of two ways:

  - If the record is the first input record for a display station (in most programs the first input record for a display station is blank)
  - If N (no) is specified in column 22 (return input) of the display format S specification and no data keys were pressed

- If the program goes to end of job prematurely, check whether all display stations have been released or whether Y (yes) was specified in column 35 (suppress input) of the S specification. Either situation can result in no display stations being allowed to enter input, which causes end of file on the WORKSTN file. If either of the preceding conditions is true for a NEP program and if the person at the display station enters a STOP SYSTEM command, the WORKSTN file goes to end of file.

- If the command display unexpectedly follows a program display, the program may have gone to end of job *before* any data was entered for the display (see the RESTORE parameter of the control language WORKSTN statement in the *System Reference* manual). If RESTORE-NO is specified, a display from the program may be on the screen after the program has gone to end of job, so it appears as if the program is still running. If RESTORE-YES is specified, the command display appears on the screen immediately when the program goes to end of job.

- Avoid using multiple formats on the same section of the screen until the program logic is debugged.

- During the debugging operations, display a constant on the screen for every format. This should help you analyze the screen contents.

- Use the DEBUG operation code in selected locations to trace the program flow. Suggested locations and the resulting debug information are as follows:

Location	**Debug Information**
As first calculation	Shows the contents of the specified input record and the indicator status for a primary file
After any READ operation	Shows the contents of the specified input record and the indicator status for a demand file
Before every EXCPT operation	Shows the status of the indicators that control which records (formats) are to be produced as output

Location	Debug Information
As last detail calculation	Shows the indicator status before heading and detail output
After an ACQ operation	Shows the work station ID and the indicator off if the operation was successful
After a REL operation	Shows the display stations that are released from the program
After every TAG operation	Shows the program flow
As first statement in each subroutine	Shows the program flow
Conditioned by LR	Shows when the program ends

- After each WORKSTN output record, define a record with the same conditioning indicators and write that record to the DEBUG file (see Figure 4-1). The record should contain:

  - The format name
  - The work station ID, if used in the program
  - The release status, if the display station is released in the output specifications
  - SLN (starting line number), if used in the program
  - Data fields as needed

If the following types of error messages occur, check the probable causes listed:

- Error messages involving program checks to the WORKSTN device are probably caused by:

  - Invalid use of erase input fields (columns 31 and 32 of the S specification)
  - Clearing all or a portion of the screen containing the input fields.

- Error messages involving invalid WORKSTN identifiers are probably caused by an earlier release of the display station in either calculation or output operations.

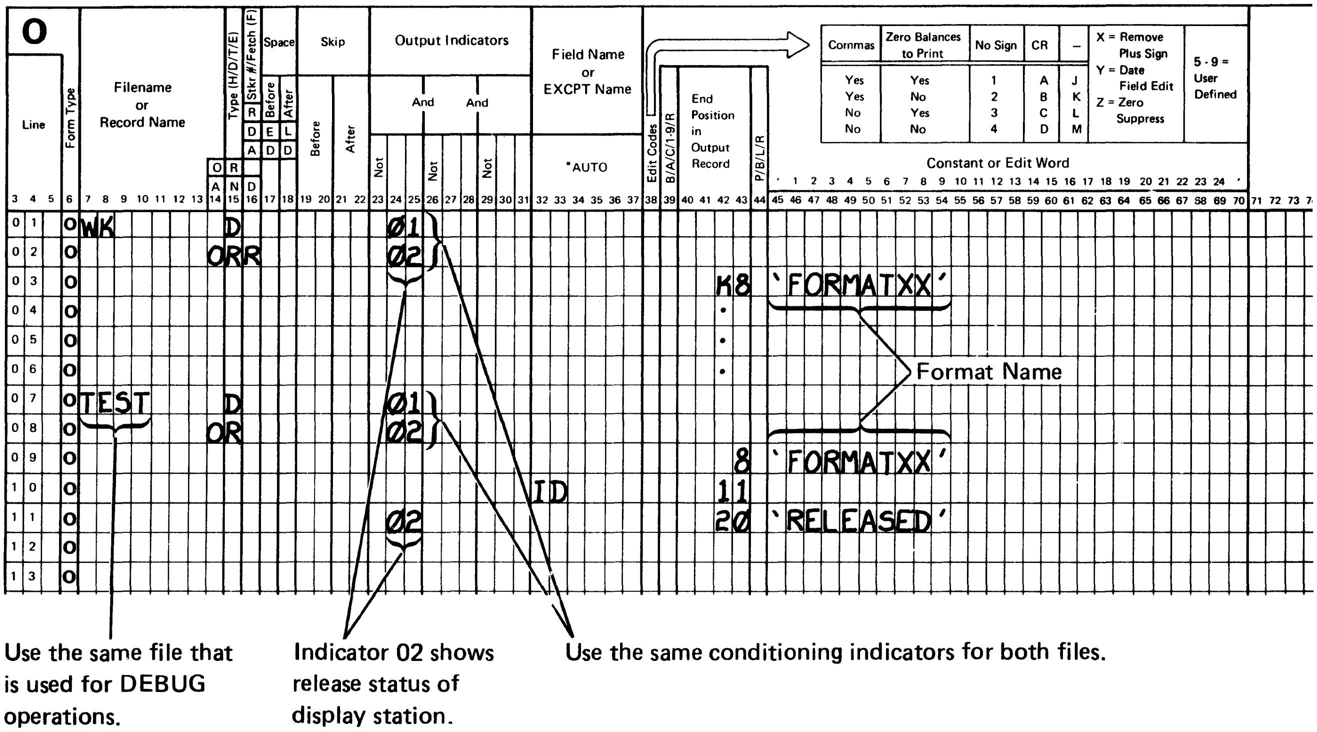


Figure 4-1. Writing the WORKSTN Output Record to the DEBUG File

Chapter 5. Using a DISK File

Chapter 5. Using a DISK File

A DISK file is a file that contains data read from a disk or written to a disk. A DISK file can be organized in one of three ways:

- Sequential
- Direct
- Indexed

This chapter explains how to code RPG specifications so that you can create, read, update, and add records to each of these kinds of DISK files.

SEQUENTIAL FILES

In a sequential file, the position of a record depends on the order in which records are placed in the file. The first record placed in the file occupies the first record position in the file, the second record placed in the file occupies the second record position, and so on. Figure 5-1 shows how a sequential file is organized.

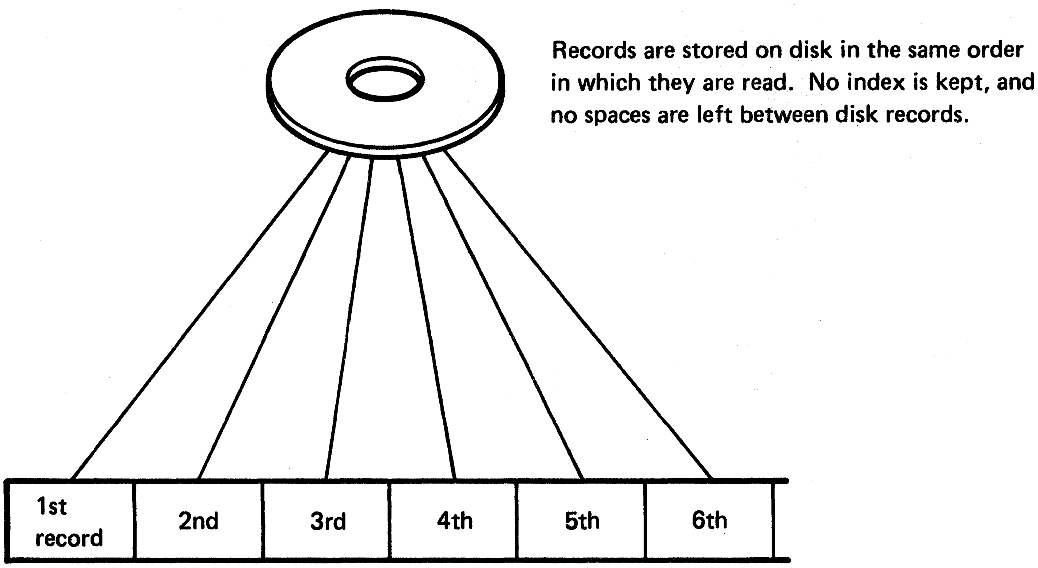
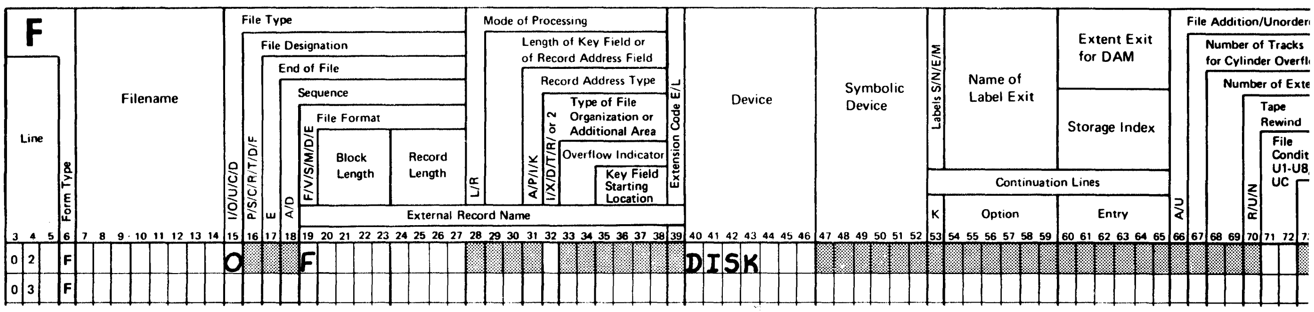


Figure 5-1. Organization of a Sequential File

Creating a Sequential File

To create a sequential file, you define the DISK file as an output file and write records to the file. The control language statements for the program must include a FILE statement. That FILE statement must use either the RECORDS parameter or the BLOCKS parameter to specify the size of the file, and it must use the DFILE-YES parameter if you want to allow records to be deleted from the file. For information about the FILE statement, see the *System Reference* manual.

Define the file by using the unshaded portions of the file description specifications shown below:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain O to indicate that the file is an output file.

Column 19 must contain F or blank to indicate that all records in the file must be the same length.

Columns 20 through 23 must contain blanks or the block length. The block length must equal the record length or be a multiple of the record length. The maximum block length is 9999. If you leave these columns blank, the block length equals the record length.

Columns 24 through 27 must contain the length of the record you are creating. The record length can be any number from 1 to 4096.

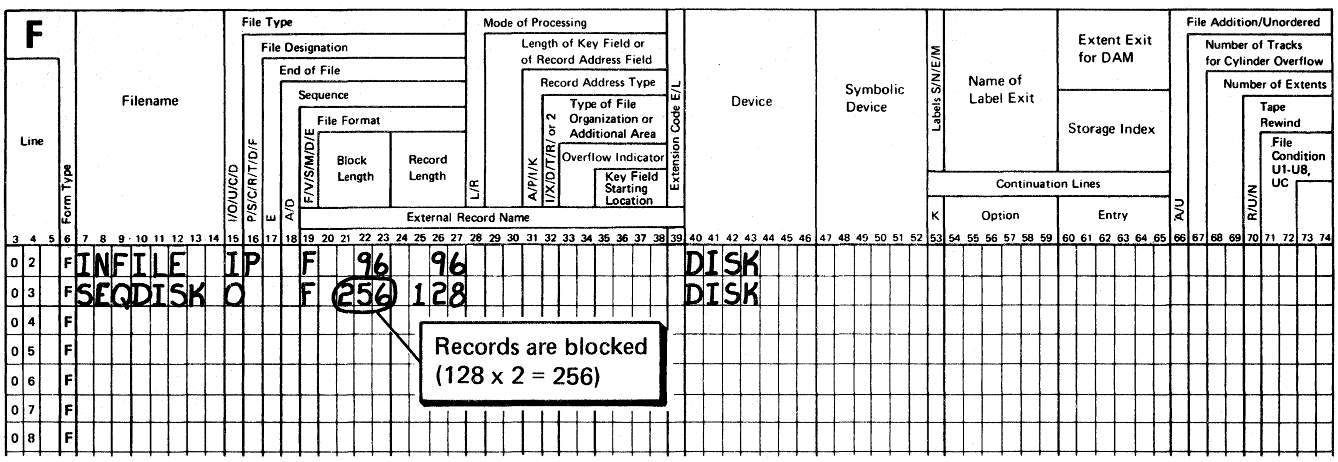
Column 32 can contain a number from 1 through 9 to indicate that the program uses two input/output areas for the file, or a blank to indicate that the program uses only one input/output area.

Columns 40 through 43 must contain DISK.

Columns 71 and 72 can contain an external indicator, U1 through U8, to condition the use of this file.

Example of Creating a Sequential File

Suppose you want to create a customer file on disk. Customer numbers are sequential; that is, you assign each new customer the next higher number. Figure 5-2 shows how to code the file description, input, and output specifications to create this sequential file.



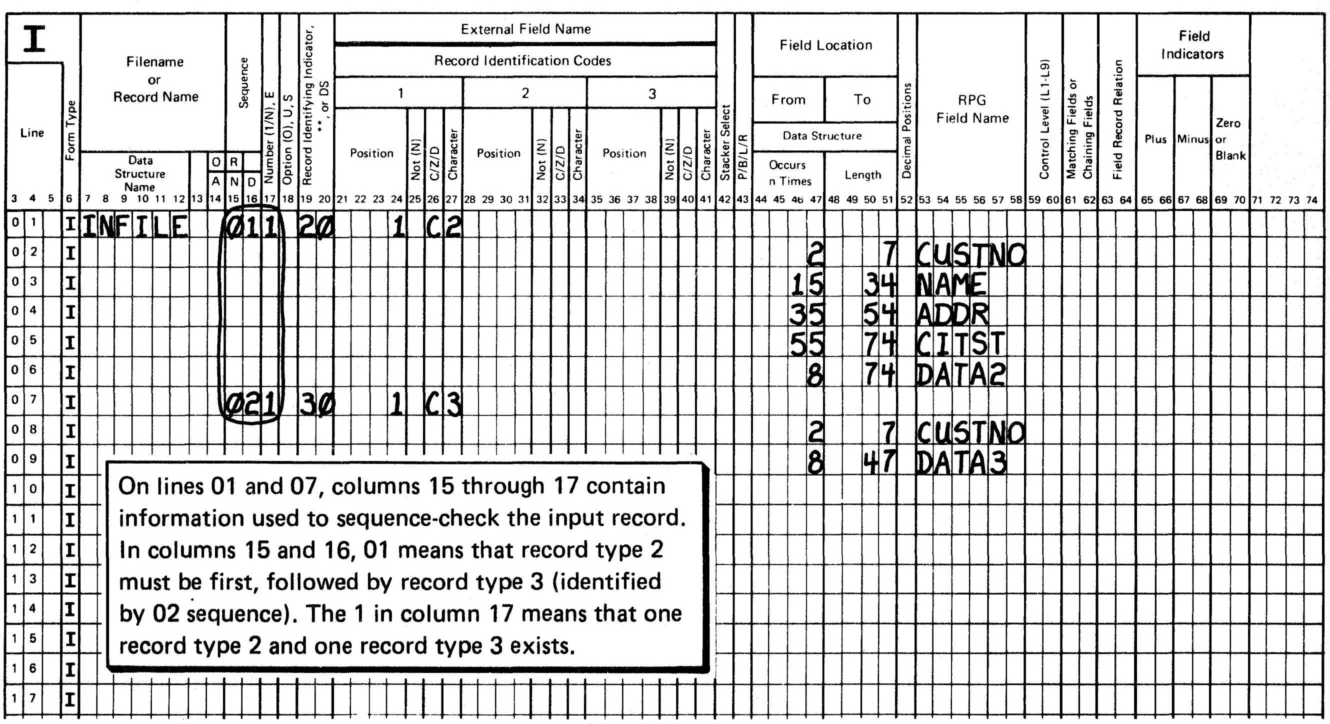


Figure 5-2 (Part 1 of 2). Creating a Sequential Customer File

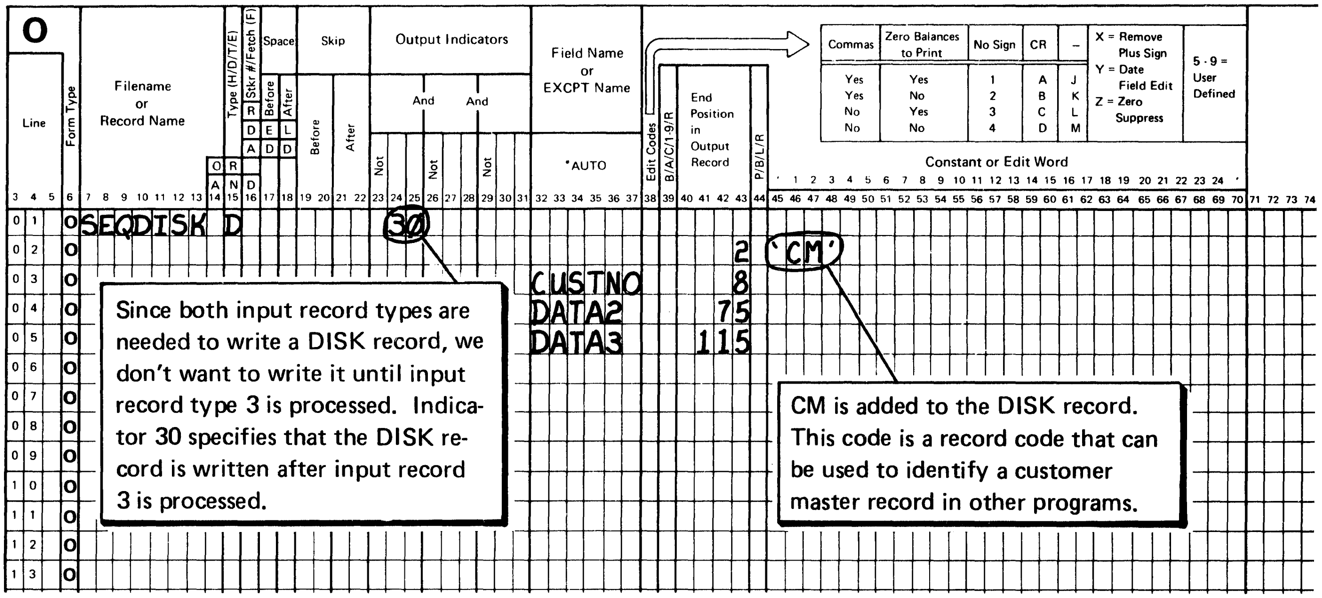


Figure 5-2 (Part 2 of 2). Creating a Sequential Customer File

Reading a Sequential File

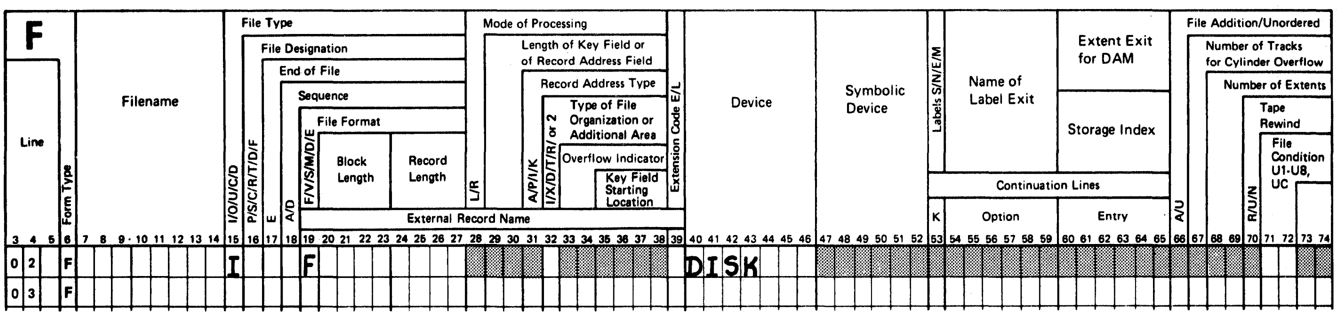
Sequential files can be read in any of the following ways:

- Consecutively
- Randomly by relative record number
- Randomly by relative record number and/or consecutively
- Randomly by address output (addrout) file

Note: An alternative index file can also be created for a sequential file to provide another method of reading the records in the file. For information about creating an alternative index file, see *Creating an Alternative Index File for an Indexed File* later in this chapter.

Reading Consecutively

Reading consecutively means reading records in the order in which they occur in the file. If you want to read all the records in the file, code the file for consecutive processing as shown in the file description specifications below:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain I to indicate that the file is an input file.

Column 16 must contain P, S, T, or D:

- If you code P (primary) or S (secondary), the file is read as part of the RPG program cycle. For an explanation of how primary and secondary files are read, see Chapter 11.
- If you code T (table), column 39 must contain E. Your program must also include an extension specification for a preexecution-time table or array. The file name on that extension specification must be the same as the file name on this file description specification. For information about extension specifications, see Chapters 13 and 22.

- If you code D (demand), you must code a READ operation code in the calculation specifications in order to read the file. For information about the READ operation, see Chapter 28.

Column 17 can contain E or blank if column 16 contains P or S. E indicates that the program must process all records from the file before the program can end. Blank indicates that the program can end before it processes all records from the file.

Column 18 can contain A, D, or blank if column 16 contains P or S:

- A indicates that the program checks that the records in the file are in ascending sequence.

- D indicates that the program checks that the records in the file are in descending sequence.

- Blank indicates that the program does not check the sequence of records in the file.

Column 19 must contain F or blank.

Columns 20 through 23 must contain the block length or blanks.

Columns 24 through 27 must contain the record length.

Column 32 can contain a number from 1 through 9 or a blank. A number indicates that the program uses two input/output areas. Blank indicates that the program uses only one input/output area.

Column 39 must contain E if column 16 contains T.

Columns 40 through 43 must contain DISK.

Columns 71 and 72 can contain an external indicator, U1 through U8.

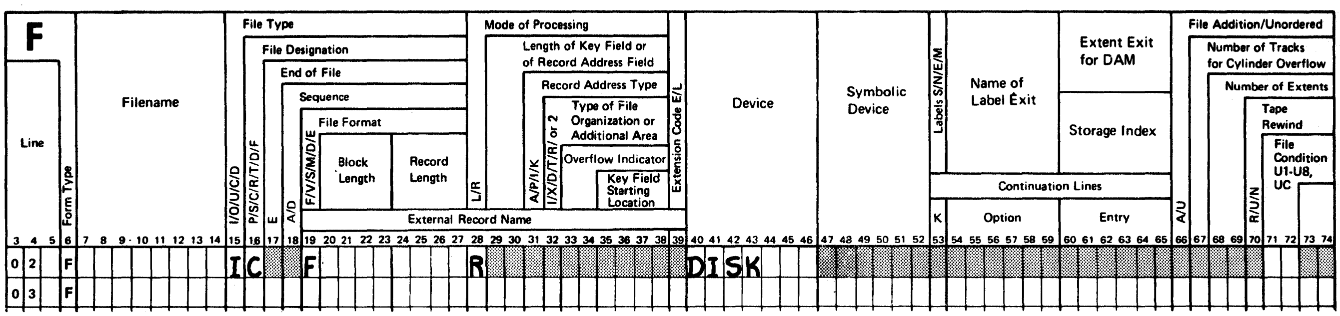
Reading Randomly by Relative Record Number

Sometimes you want to read only some of the records in the file. Reading consecutively can be slow in this case, because reading consecutively means reading every record in a file. It would be faster to read only the records you specifically identify. Reading only specific records is called *random processing.*

One way to identify which records to read is to identify the position of each record in the file. A number that identifies the position of each record relative to the beginning of the file is called a *relative record number.* For example, the relative record number of the first record in a file is 1, the relative record number of the second record in the file is 2, and so on.

You can process files randomly by relative record number if the files are chained files (that is, if there is a C in column 16 of the file description specifications). Chained files are not read at input time of the RPG program cycle. Instead, they are read only when the CHAIN operation occurs during the calculation part of the cycle. For information about the CHAIN operation code, see Chapter 28. Chained records can be read during total calculations or during detail calculations.

If you want to read records randomly by relative record number, code the unshaded columns of the file description specifications shown below:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain I to indicate that the file is an input file.

Column 16 must contain C to indicate that the file is a chained file. You must code a CHAIN operation in the calculation specifications in order to read a chained file.

Column 19 must contain F or blank.

Columns 20 through 23 must contain the block length or blanks.

Columns 24 through 27 must contain the record length.

Column 28 must contain R to indicate that the file is to be processed randomly by relative record number.

Columns 40 through 43 must contain DISK.

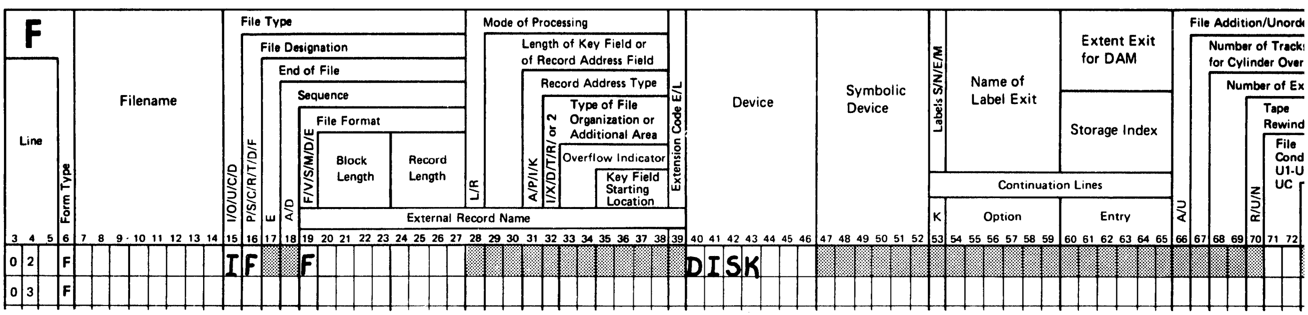
Columns 71 and 72 can contain an external indicator, U1 through U8.

Reading Randomly by Relative Record Number and/or Consecutively

If you want to read a file both randomly and consecutively, use a full-procedural file. You can read a full-procedural file randomly like a chained file and/or consecutively like a demand file. That is, you can chain to a specific relative record number in the file and then read records consecutively from that point. To read the file randomly, you use a CHAIN operation in the calculation specifications; to read it consecutively, you use a READ or READP operation. You cannot use a READE operation to read the file consecutively, because the READE operation cannot read by relative record number. For example, if you code a CHAIN operation to relative record number 10 and then code a READ operation, the program chains to relative record number 10 and then reads the following record.

It is not necessary to code both a CHAIN and a READ or READP operation, but you must code at least one CHAIN, one READ, or one READP operation in order to read a full-procedural file. For information about the CHAIN, READ and READP operation codes, see Chapter 28.

Code a full-procedural file as an input file. Code entries in the unshaded columns of the file description specifications shown below:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain I to indicate that the file is an input file.

Column 16 must contain F to indicate that the file is a full-procedural file. You must code a CHAIN, READ or READP operation in the calculation specifications in order to read a full-procedural file.

Column 19 must contain F or blank.

Columns 20 through 23 must contain the block length or blanks.

Columns 24 through 27 must contain the record length.

Columns 40 through 43 must contain DISK.

Columns 71 and 72 can contain an external indicator, U1 through U8.

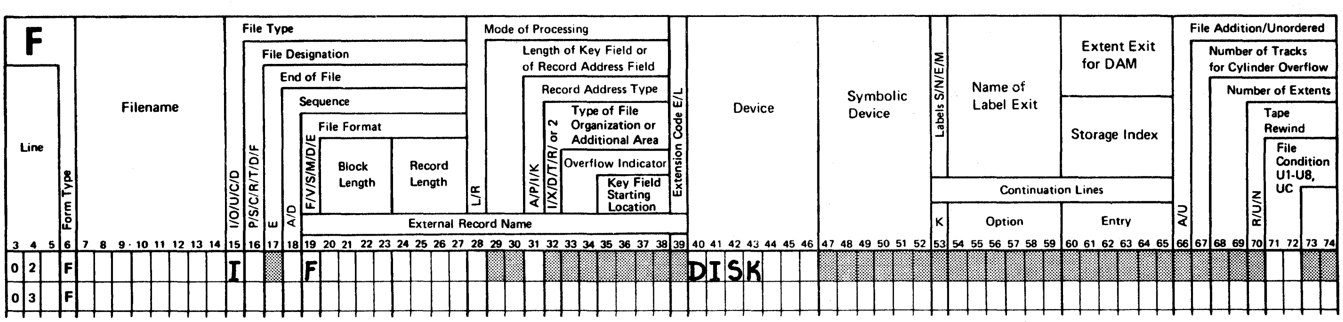
Reading Randomly by Address Output (Addrout) File

An *address output (addrout) file* is a record address file produced by a sort program. (A *record address file* is an input file that tells the program which records to read from a DISK file and the order in which to read them.) An address output file contains the relative record numbers of the records in a DISK file. The advantages of an address output file are that:

- The space required for the address output file is much less than the space required for a sorted sequential file.
- The sort runs much faster.
- The original file is unchanged.

You can have only one address output file in a program. When an RPG program uses an address output file, it automatically reads the relative record numbers consecutively from the address output file. You do not have to code a READ operation for the address output file. Then, using the relative record number, the program randomly reads the DISK file to process the corresponding record. In this way, the program can process a sequential DISK file in a new sequence without actually sorting the records and creating a new file. Also, once the file description and extension specifications are coded for the DISK file and for its associated address output file, you can code the DISK file as an ordinary sequential file. If the DISK file is a full-procedural file, you must code a READ operation in the calculation specifications; you cannot use a READE, READP, or CHAIN operation to read a full-procedural file randomly by an address output file. No input specifications are required for the address output file.

If you want to read records randomly by an address output file, code the file description specifications as shown below:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain I to indicate that the file is an input file.

Column 16 must contain P, S, or F:

- If you code P (primary) or S (secondary), the record is read as part of the RPG program cycle. For an explanation of how primary and secondary records are read, see Chapter 11.

- If you code F (full-procedural), you must code a READ operation in the calculation specifications. CHAIN, READE, and READP operations are not allowed with the address output files.

Column 18 can contain A, D, or blank if column 16 contains P or S:

- A indicates that the program checks that the records in the file are in ascending sequence.
- D indicates that the program checks that the records in the file are in descending sequence.
- Blank indicates that the program does not check the sequence of records in the file.

Column 19 must contain F or blank.

Columns 20 through 23 must contain the block length or blanks.

Columns 24 through 27 must contain the record length.

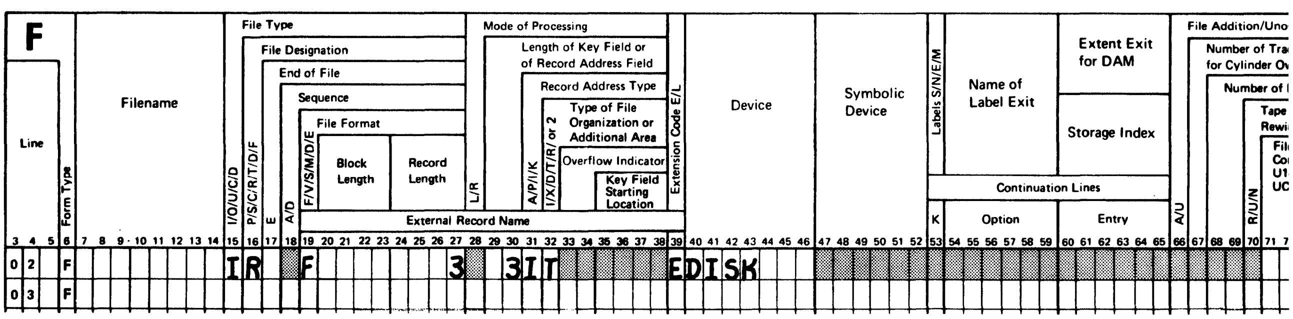
Column 28 must contain R if column 16 contains P or S. The R indicates that the file is processed randomly by an address output file.

Column 31 must contain I if column 16 contains P or S. The I indicates that relative record numbers from the address output file are used to process the file.

Columns 40 through 43 must contain DISK.

Columns 71 and 72 can contain an external indicator, U1 through U8.

For the address output file, code entries in the unshaded columns of the file description specifications shown below:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain I to indicate that the file is an input file.

Column 16 must contain R to indicate that the file is a record address file.

Column 17 must contain E or blank. E indicates that all records from the file must be processed before the program can end. Blank indicates that the program can end whether or not all records from the file are processed.

Column 19 must contain F or blank.

Columns 20 through 23 must contain the block length or blanks.

Column 27 must contain 3 because each record in an address output file is a relative record number, which is always three positions long.

Column 30 must contain 3 because relative record numbers in address output files are always three positions long.

Column 31 must contain I to indicate that binary relative record numbers are used in processing.

Column 32 must contain T to indicate that the file is an address output file.

Column 39 must contain E to indicate that the file is further described on extension specifications.

Columns 40 through 43 must contain DISK.

Columns 71 and 72 can contain an external indicator, U1 through U8.

Two entries are required on the extension specifications:

E Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
3 4 5	6	7 8	9 10	11 12 13 14 15 16 17 18	19 20 21 22 23 24 25 26	27 28 29 30 31 32	33 34 35	36 37 38 39	40 41 42	43	44	45	46 47 48 49 50 51	52 53 54	55	56	57	58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
0 1	E																	
0 2	E																	

Columns 11 through 18 must contain the name of the address output file. This name must be the same one coded on the file description specifications for the address output file.

Columns 19 through 26 must contain the name of the sequential file to be processed by the address output file. The name must be the same one coded on the file description specifications for the sequential file.

Updating A Sequential File

Updating records in a sequential file involves reading a record, changing some data in the record, and writing the record back to its original location in the file. If you try to update a record that was not the last record read, error message RPG-9043, TRIED RECORD UPDATE BEFORE INPUT FOR FILE, is displayed. The fields to be updated must be described on both the input and the output specifications.

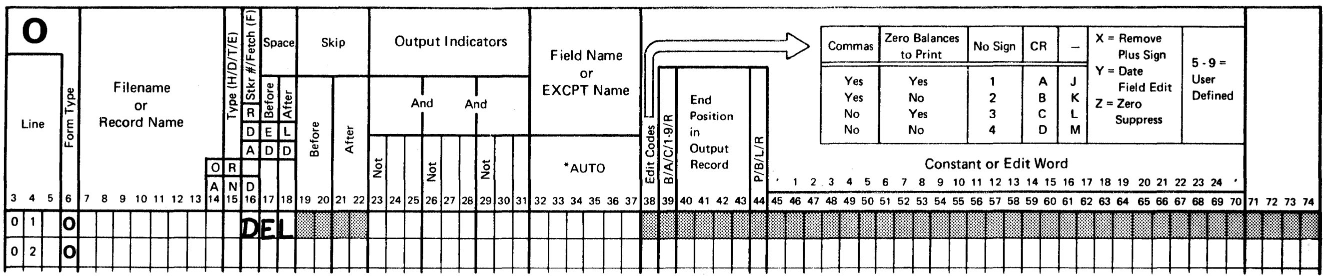
When you update records in a sequential file, the file can be processed in any of the following ways:

- Consecutively
- Randomly by relative record number
- Randomly by relative record number and/or consecutively
- Randomly by address output (addrout) file

Deleting Records from a Sequential File

Updating a file can include deleting records from the file. To allow records to be deleted from the file, the DFILE-YES parameter must be specified on the control language FILE statement when the file is created. For information about the FILE statement, see the *System Reference* manual. If you try to delete a record from a file that does not allow deletions, error message RPG-9067, INVALID OPERATION ATTEMPTED, is displayed.

To delete a record, you first read the record (either randomly or consecutively) and then, with DEL coded in the output specifications, write the record back to the same file. Code entries in the unshaded columns of the output specifications shown below:



Columns 7 through 14 must contain the name of the file if this is the first record on the output specifications or if the previous record on the output specifications is for a different file.

Column 15 must contain D, T, or E to indicate whether the record is to be written at detail, total, or exception output time of the RPG program cycle.

Columns 16 through 18 must contain DEL to indicate that the record is to be deleted.

Columns 23 through 31 can contain output indicators.

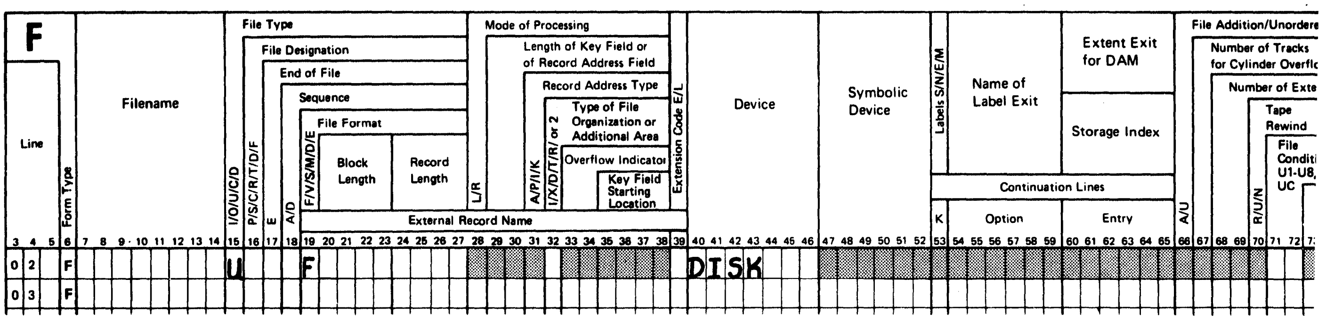
Columns 32 through 37 can contain an EXCPT name if column 15 contains E.

Records are not physically removed from a file when they are deleted. Instead, deleted records are filled with hexadecimal FFs. That is, all the bits for every character in the deleted record are set on.

Updating Consecutively

You can update records in a sequential file consecutively. If the file is a primary or secondary file (P or S in column 16 of the file description specifications), the program reads a record from the file at input time in the RPG program cycle, and the program writes a record to update the file during detail output or exception output time in the program cycle. If the file is a demand file (D in column 16), the program reads a record when a READ operation occurs in the calculation specifications, and it writes a record to update the file at detail output, total output, or exception output time in the program cycle.

Code the file description specifications as shown below:



Column 15 must contain U to indicate that the file is an update file.

Entries in the other columns are the same as those for reading consecutively.

Example of Updating and Deleting Records

Sometimes you want to update the records in your customer master file. The transaction file contains two input record types. One type (those with D in column 1) identifies records to be deleted from the master file. The other type (those with 3 in column 1) contains information needed to update the master file. Figure 5-3 shows how to code the RPG specifications to update records and delete records from the master file.

Line	Form Type	Filename	File Type	File Designation	End of File	Sequence	File Format	Block Length	Record Length	Device
02	F	TRANS	I	P		A	F	96	96	DISK
03	F	MASTER	U	S		A	F	256	128	DISK
04	F									

Line	Form Type	Filename or Record Name	And/Or	Sequence	Record Identifying Indicator	Position (1)	Not (1)	C/Z/D (1)	Character (1)	Position (2)	Not (2)	C/Z/D (2)	Character (2)	Position (3)	Not (3)	C/Z/D (3)	Character (3)	From	To	Decimal Positions	RPG Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields
01	I	TRANS		AA	Ø1	1		C	D														
02	I																	2	7		CUSTNO	L1	M1
03	I			AB	Ø2	1		C	3														
04	I																	2	7		CUSTNO	L1	M1
05	I																	12	17	2	CHG		
06	I																	18	23	2	PAYMT		
07	I																	24	29	2	CRDT		
08	I	MASTER		NS	Ø3	1		C	C	2		C	M	128	N	C	D						
09	I		OR		Ø4	1		C	C	2		C	M	128		C	D						
10	I																	3	8		CUSTNO	L1	M1
11	I																	16	35		NAME		
12	I																	76	76		CRLIM		
13	I																	8Ø	85	2	CHARGE		
14	I																	86	91	2	PAY		
15	I																	92	97	2	CREDIT		
16	I																	98	103	2	BAL		
17	I																						

Figure 5-3 (Part 1 of 2). Updating and Deleting Records in a Sequential File

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust	Resulting Indicators (High / Low / Equal)	Comments
01	C		Ø1 MR		SETON						21	
02	C		Ø2 MR		ADD	CHG	TOTCHG	6	2			
03	C		Ø2 MR		ADD	PAYMT	TOTPAY	6	2			
04	C		Ø2 MR		ADD	CRDT	TOTCRD	6	2			
05	C		Ø3 MR		ADD	TOTCHG	CHARGE					
06	C		Ø3 MR		ADD	TOTPAY	PAY					
07	C		Ø3 MR		ADD	TOTCRD	CREDIT					
08	C		Ø3 MR		ADD	TOTCHG	BAL					
09	C		Ø3 MR		SUB	TOTPAY	BAL					
10	C		Ø3 MR		SUB	TOTCRD	BAL					
11	C	L1			SETOF						21	
12	C	L1			SUB	TOTCHG	TOTCHG					
13	C	L1			SUB	TOTPAY	TOTPAY					
14	C	L1			SUB	TOTCRD	TOTCRD					
15	C											
16	C											
17	C											
18	C											
19	C											
20	C											

Indicator 01 is on for an input record which deletes a disk record. Indicator 21 is set for use during update of the MASTER file.

Set off indicator for next cycle.

The values in the MASTER file are updated by totals accumulated from transaction records.

Clear accumulator fields for next cycle.

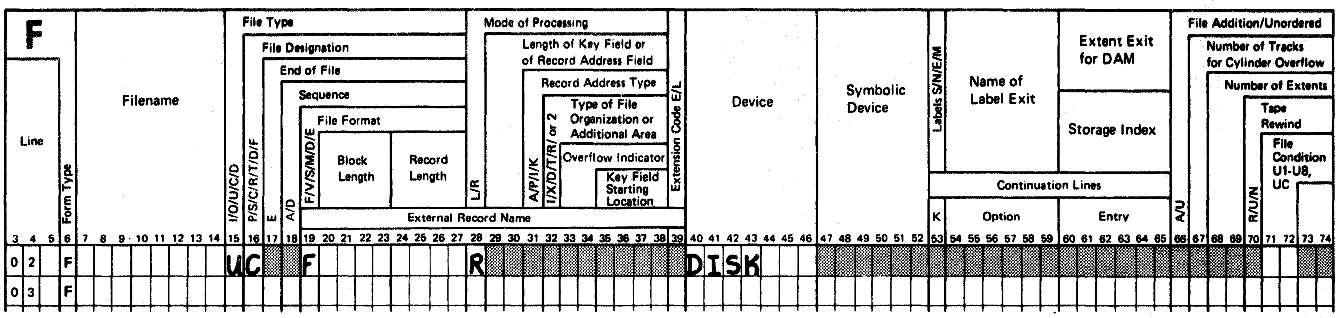
Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Space / Skip	Output Indicators	Field Name or EXCPT Name	Edit Codes	End Position in Output Record	Constant or Edit Word
01	O	MASTER	D		Ø3 MR N21				
02	O					CHARGE		85	
03	O					PAY		91	
04	O					CREDIT		97	
05	O					BAL		103	
06	O		DDEL		Ø4 MR 21				
07	O								

Figure 5-3 (Part 2 of 2). Updating and Deleting Records in a Sequential File

Updating Randomly by Relative Record Number

You can update records in a sequential file randomly by relative record number. The file is defined as a chained file (C in column 16 of the file description specifications). This means that the program reads a record from the file when a CHAIN operation occurs in the calculation specifications, and it writes a record to update the file during detail output, total output, or exception output time in the RPG program cycle.

Code the file description specifications as shown below:



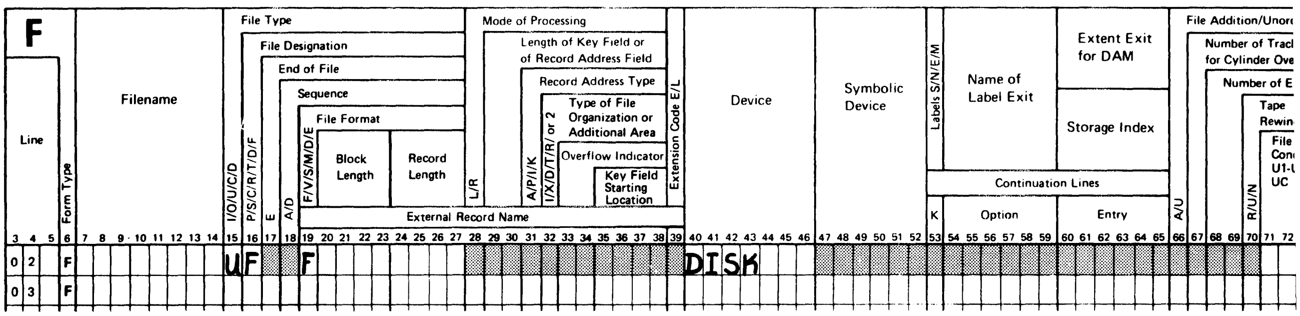
Column 15 must contain U to indicate that the file is an update file.

Entries in the other columns are the same as those for reading randomly by relative record number.

Updating Randomly by Relative Record Number and/or Consecutively

You can update records in a sequential file randomly by relative record number and/or consecutively. That is, the file is defined as a full-procedural file (F in column 16 of the file description specifications). The record to be updated can be read either randomly by relative record number with a CHAIN operation or consecutively with a READ or READP operation. The record cannot be read consecutively with a READE operation, because the READE operation cannot read by relative record number. The output operation to update the record can occur during detail output, total output, or exception output time of the RPG program cycle.

Code the file description specifications as shown below:



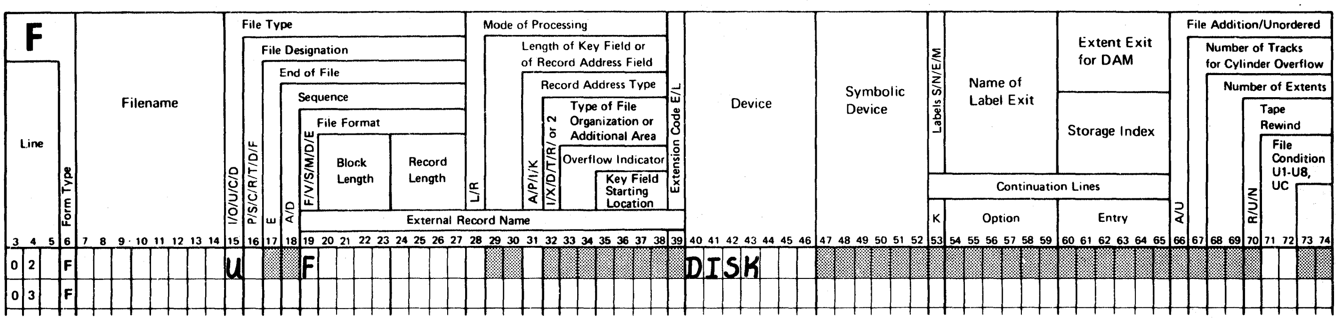
Column 15 must contain U to indicate that the file is an update file.

Entries in the other columns are the same as those for reading randomly by relative record number and/or consecutively.

Updating Randomly by Address Output (Addrout) File

You can update records in a sequential file processed by an address output (addrout) file. The sequential file can be a primary, secondary, or full-procedural file (P, S, or F in column 16 of the file description specifications). If the file is a primary or secondary file, the program reads a record at input time of the RPG program cycle, and it writes a record to update the file at detail output or exception output time of the program cycle. If the file is a full-procedural file, the program reads a record when the READ operation occurs in the calculation specifications, and it writes a record to update the file at detail output, total output, or exception output time of the program cycle.

Code the file description specifications as shown below:



Column 15 must contain U to indicate that the file is an update file.

Entries in the other columns are the same as those for reading randomly by address output file.

Entries on the file description specifications for the address output file and on the extension specifications are the same as those for reading randomly by address output file.

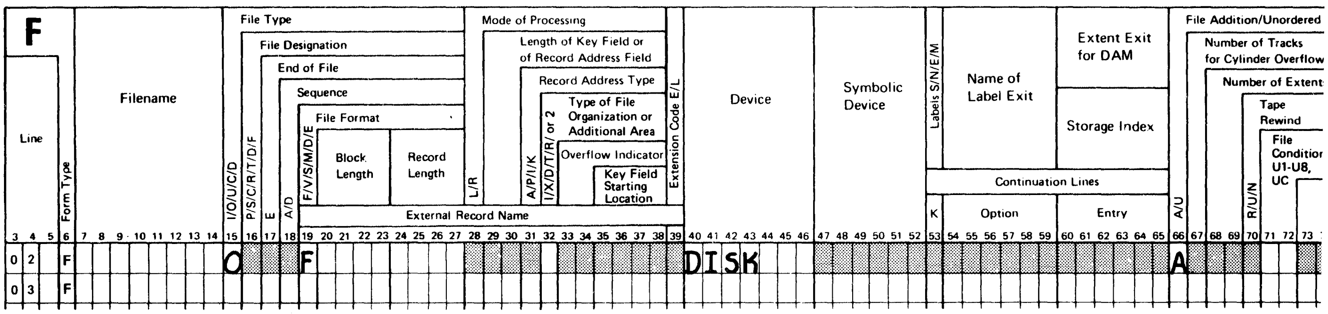
Adding Records to a Sequential File

After a file is created, you can add records to it in either of two ways:

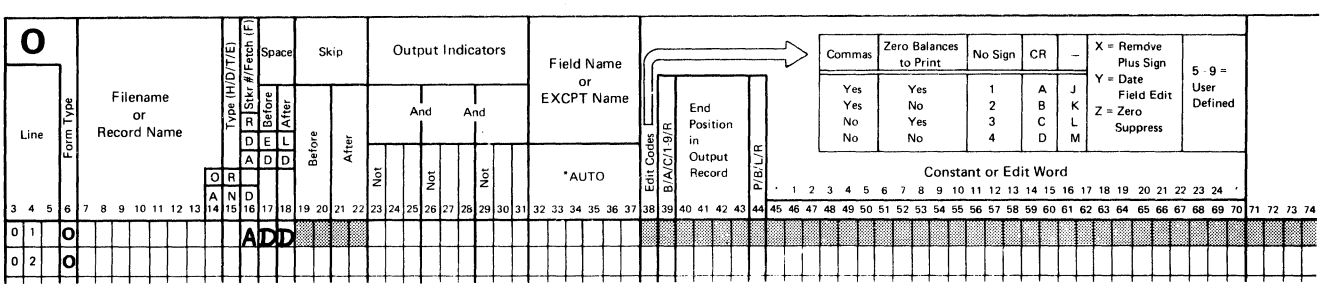
- At the end of records in the file
- Between records in the file

Adding Records at the End of a File

To add records at the end of a sequential file, code the file description and output specifications as shown below:



On the file description specifications, all entries except column 66 are the same as those for creating a sequential file. The A in column 66 indicates that you will add records to the file described on this line.



On the output specifications, columns 7 through 14 must contain the name of the file if this is the first record on the output specifications or if the previous record on the output specifications is for a different file.

Column 15 must contain H (heading), D (detail), T (total), or E (exception) to indicate the type of record to be written.

Columns 16 through 18 must contain ADD to indicate that the fields defined on the following lines form the record to be added to the file.

Columns 23 through 31 can contain conditioning indicators.

Columns 32 through 37 can contain an EXCPT name if column 15 contains E.

Example of Adding Records at the End of a File

As you get new customers, you want to add them to the sequential customer file you created in Figure 5-2, *Example of Creating a Sequential File.* Because you assign customer numbers sequentially, you can add each new customer record at the end of the records already in the file.

Figure 5-4 shows how to code the RPG specifications to add records at the end of that sequential customer file.

Line	Form Type	Filename	File Type	File Designation	End of File	Sequence	File Format	Block Length	Record Length	Device	File Addition/Unordered
02	F	INFILE	I	P			F	96	96	DISK	
03	F	SEQDISK	O				F	256	128	DISK	A
04	F										

Line	Form Type	Filename or Record Name	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, **, or DS	Position	Not (N)	C/Z/D	Character	From	To	RPG Field Name
01	I	INFILE	Ø1	1		2Ø	1		D	2			
02	I										2	7	CUSTNO
03	I										8	9	STATE
04	I										1Ø	11	CO
05	I										12	14	CITY
06	I										15	34	NAME
07	I										35	54	ADDR
08	I										55	74	CITST
09	I		Ø2	1		3Ø	1		D	3			
10	I										2	7	CUSTNO
11	I										8	8	CRLIM
12	I										9	47	REST
13	I												

Figure 5-4 (Part 1 of 2). Adding Records at the End of a Sequential File

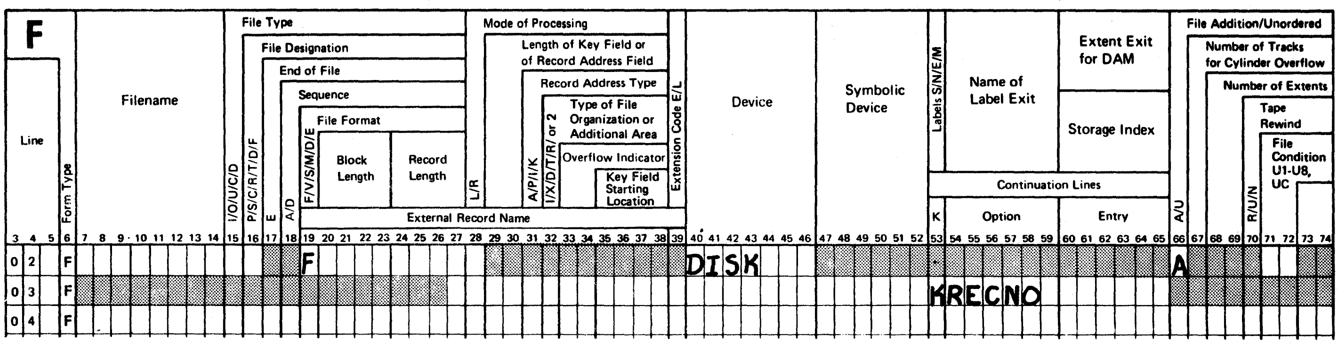
Line	Form Type	Filename or Record Name	Type (H/D/T/E)	ADD	Output Indicators	Field Name or EXCPT Name	End Position in Output Record	Constant or Edit Word
01	O	SEQDISK	D	ADD	30			
02	O						2	'CM'
03	O					CUSTNO	8	
04	O					STATE	10	
05	O					CO	12	
06	O					CITY	15	
07	O					NAME	35	
08	O					ADDR	55	
09	O					CITST	75	
10	O					CRLIM	76	
11	O					REST	115	
12	O							

Figure 5-4 (Part 2 of 2). Adding Records at the End of a Sequential File

Adding Records between Records in a File

You can also add records between records in a sequential file that is processed randomly by relative record number. For example, you may have to add new records between existing records in order to keep the file in a particular order when the control fields of the new records are not higher in sequence than the control fields of records already in the file. Such a file must be one that allows records to be deleted. That is, when the file was created, the DFILE-YES parameter must have been specified on the control language FILE statement. For information about the FILE statement, see the *System Reference* manual.

To add records between records in a sequential file, code the unshaded columns of the file description specifications shown below:



In the first line of the file description specifications:

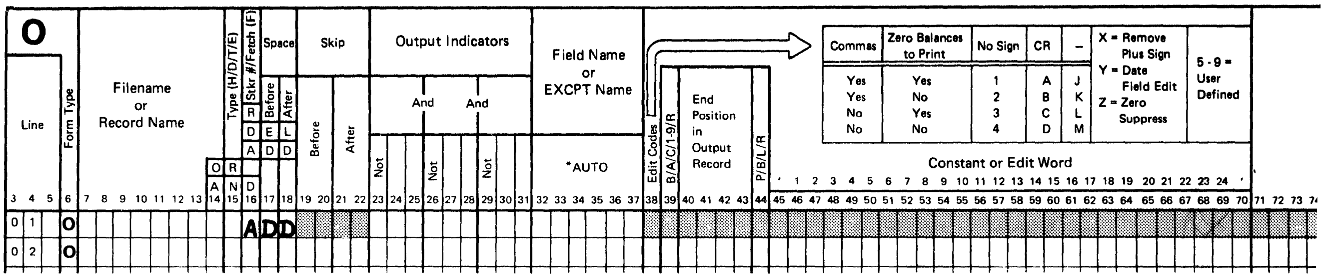
- Columns 7 through 14 must contain the name of the file.
- Column 15 must contain I or U to identify the file as an input file or an update file.
- Column 16 must contain C or F to identify the file as a chained or full-procedural file.
- Column 19 must contain F or blank to indicate that all records in the file have the same length.
- Columns 20 through 23 must contain the block length or blanks.
- Columns 24 through 27 must contain the record length or blanks.
- Column 28 must contain R if column 16 contains C.
- Columns 40 through 43 must contain DISK.
- Column 66 must contain A to indicate that you will add records to the file.
- Columns 71 and 72 can contain an external indicator, U1 through U8.

In the second line:

- Column 53 must contain K to indicate that this line is a continuation line that provides additional information about the file.

- Columns 54 through 58 must contain RECNO, which stands for relative record number.

- Columns 60 through 65 must contain the name of the field into which the relative record is placed. The field must be defined on either the input specifications or the calculation specifications as a 7-position numeric field with zero decimal positions. That field is called the RECNO field.



On the output specifications for the record or records to be added, columns 7 through 14 must contain the name of the output file.

Column 15 can contain D, T, or E, to indicate whether the record is to be written at detail, total, or exception output time of the RPG program cycle.

Columns 16 through 18 must contain ADD to indicate that the fields defined in the following lines form the record or records to be added to the file.

Columns 23 through 31 can contain output indicators.

Columns 32 through 37 can contain an EXCPT name if column 15 of the output specifications contains E.

The RECNO field identifies the position in the file where the output record is to be added. (That record is the one described on the output specification that contains ADD in columns 16 through 18.) You must place into the RECNO field the relative record number of the record to be added to the file. It must be the relative record number of a deleted record. One way to place the relative record number into the RECNO field is to code the following sequence of operations in the calculation specifications:

1. Code a CHAIN operation with the relative record number in factor 1, the name of the chained file in factor 2, and a resulting indicator in columns 54 and 55 that turns on when a record is not found.

2. Code a Z-ADD operation with the same indicator that you used for the CHAIN operation coded as a conditioning indicator (in columns 10 and 11, 13 and 14, or 16 and 17), the relative record number in factor 2, and the RECNO field in the result field.

When a CHAIN operation (for a chained or full-procedural file) or a READ, READE, or READP operation (for a full-procedural file) reads a nondeleted record, data management places into the RECNO field the relative record number of the record read.

When the program tries to add a record to a file, if the relative record number is not the number of a deleted record, the program stops and error message RPG-9070, OUTPUT TO A NONDELETED RECORD, is displayed. If you respond to the message by choosing option 1, the program continues running but it does not add the record to the file.

You cannot use the RECNO field to add records at the end of a sequential file. For example, if a file contains relative record numbers 1 through 5 and 7 through 10, you can add a record at relative record number 6 but not at relative record number 11. If you try to add a record at the end of a sequential file by using a RECNO field, error message RPG-9068, RELATIVE RECORD NUMBER BEYOND EXTENT FOR FILE, is displayed.

Example of Adding Records between Records in a File

Figure 5-5 shows how to code the RPG specifications to add records between records in a sequential file.

Line	Form Type	Filename	File Type	File Designation	File Format	Block Length	Record Length	Device	K	Option	Entry	A/U
02	F	INPUT	I	P	F	96	96	DISK				
03	F	MASTER	I	F	F	96	96	DISK				A
04	F								K	RECNO	RRN	
05	F											

Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator	Position	Not (N)	C/Z/D	Character	From	To	Decimal Positions	RPG Field Name
01	I	INPUT	NS	Ø1	1		C	I				
02	I								2	8	Ø	ITEMNO
03	I								2	96		RECORD
04	I		NS	Ø2	1	N	C	I				
05	I	MASTER	NS									
06	I											

Line	Form Type	Indicators	Indicators (And)	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Resulting Indicators (High)
01	C	Ø1		ITEMNO	CHAIN	MASTER				1Ø
02	C	Ø1	1Ø		Z-ADD	ITEMNO	RRN	7	Ø	
03	C	Ø1	1Ø		EXCPT	ADDREC				
04	C									

Line	Form Type	Filename or Record Name	Type (H/D/T/E)		Field Name or EXCPT Name	End Position in Output Record	Constant or Edit Word
01	O	MASTER	E	ADD	ADDREC		
02	O					1	'M'
03	O				RECORD	96	
04	O						

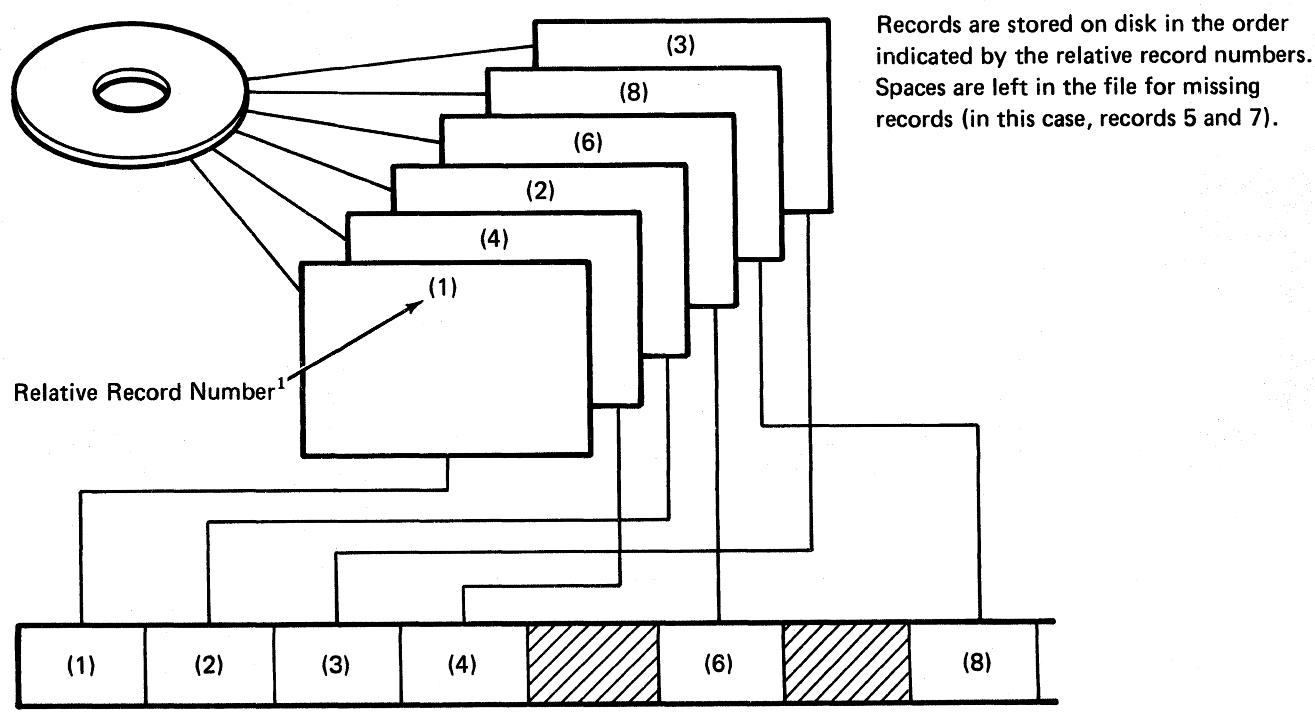
Figure 5-5. Adding Records between Records in a Sequential File

DIRECT FILES

A direct file is one in which records are assigned specific record locations on disk. Figure 5-6 shows how direct files are organized. Each record is assigned a specific location in the file, regardless of the order in which it is put in the file. If the file allows records to be deleted (that is, if the control language FILE statement used the DFILE-YES parameter when the file was created), unused records in the file contain hexadecimal FFs. If the file does not allow deletions, unused records contain blanks.

Direct file organization allows your program to find and read any record in the file *directly* without first checking other records or searching an index. Therefore, direct file organization has advantages over sequential organization.

The location assigned to a record is called the *relative record number*. The relative record number is not a disk address; rather, it is a number that states the position of a record in a file. For example, the fifth record in a file has relative record number 5.



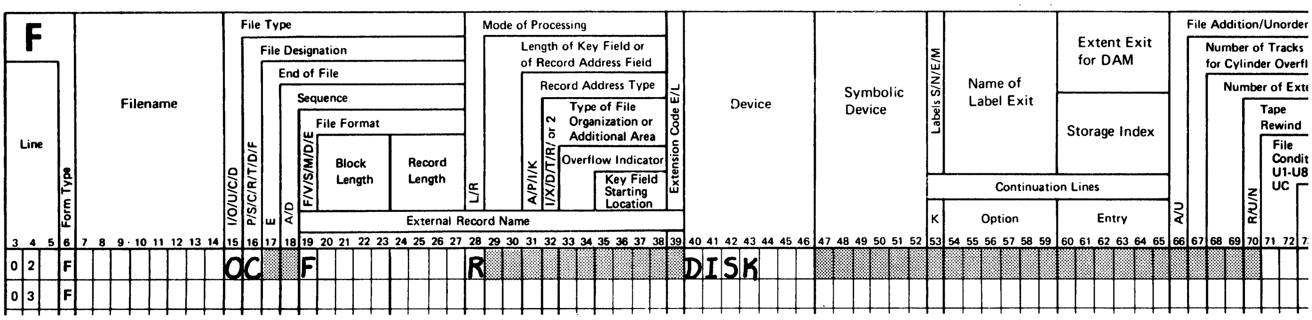
[1]The programmer usually derives relative record numbers from information in the records.

Figure 5-6. Organization of a Direct File

Creating a Direct File That Does Not Allow Deletions

To create a direct file that does not allow records to be deleted, you must define a DISK file as a chained output file and then write records to the file. Before any output is written to the file, the disk space required for the file is automatically filled with blanks. To write a record to the file, you must first determine what the relative record number of that record will be in the file to be created. Then use that relative record number as factor 1 in a CHAIN operation; as factor 2, use the name of the file to be created. When the CHAIN operation occurs, it reads the blank record at the specified relative record number. When the output operation occurs, the record is written to the same relative record number. The output operation can occur during detail output, total output, or exception output time of the RPG program cycle. If the CHAIN operation tried to read a record that was past the end of the file but you write a record to the file anyway, the record overlays the record written during the previous output operation to this file. If no record is written to the relative record number in the direct file, the space reserved in the direct file for that record remains blank (see Figure 5-6).

To create a direct file that does not allow deletions, code entries in the unshaded columns of the file description specifications shown below:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain O to indicate that the file is an output file, because the file is created when it is first written.

Column 16 must contain C to indicate that the file is a chained file. You must use the CHAIN operation in the calculation specifications to identify the relative record number of each record to be written to the file.

Column 19 must contain F or blank to indicate that all records in the file have the same length.

Columns 20 through 23 must contain the block length or blanks. For files processed randomly, the block length should be the same as the record length. If you leave columns 20 through 23 blank, the block length is the same as the record length.

Columns 24 through 27 must contain the length of each record. The maximum length of a record is 4096.

Column 28 must contain R to indicate that the file is to be processed randomly.

Columns 40 through 43 must contain DISK.

Columns 71 and 72 can contain an external indicator, U1 through U8, to condition the use of this file.

Example of Creating a Direct File That Does Not Allow Deletions

In Figure 5-7, the direct file being created, CUSTFILE, is defined on the file description specifications as a chained output file (O and C in columns 15 and 16). The CHAIN operation on the calculation specifications reads the relative record number from the CUSNO field of the input file RECIN, and writes a record from RECIN to the corresponding relative record number in the output file CUSTFILE. Indicator 04 turns on if the record is not found.

Line	Form Type	Filename	File Type	File Designation	End of File	Sequence	File Format	Block Length	Record Length	Mode of Processing	Device
02	F	RECIN	I	P			F	60	60		DISK
03	F	CUSTFILE	O	C			F	60	60	R	DISK
04	F										

Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator	Position	Not (N)	C/Z/D	Character	From	To	Decimal Positions	RPG Field Name
01	I	RECIN	NS	01	1		C	1				
02	I								2	5	0	CUSTNO
03	I								6	23		CUSTNM
04	I								24	39		ADDR
05	I								40	55		CTYST
06	I								56	60		ZIP
07	I								2	60		RECORD
08	I											

Line	Form Type	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Resulting Indicators High (54-55)
01	C	01	CUSTNO	CHAIN	CUSTFILE			04
02	C							

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Output Indicators	Field Name or EXCPT Name	End Position in Output Record	Constant or Edit Word
01	O	CUSTFILE	D	01 N04			
02	O					1	'A'
03	O				RECORD	60	
04	O						

Figure 5-7. Creating a Direct File That Does Not Allow Deletions

Creating a Direct File That Allows Deletions

To create a direct file that allows records to be deleted, you must define a DISK file on the file description specifications as an output file to be processed randomly, and you must use the RECNO continuation line. Also, you must specify the DFILE-YES parameter on the control language FILE statement for the file.

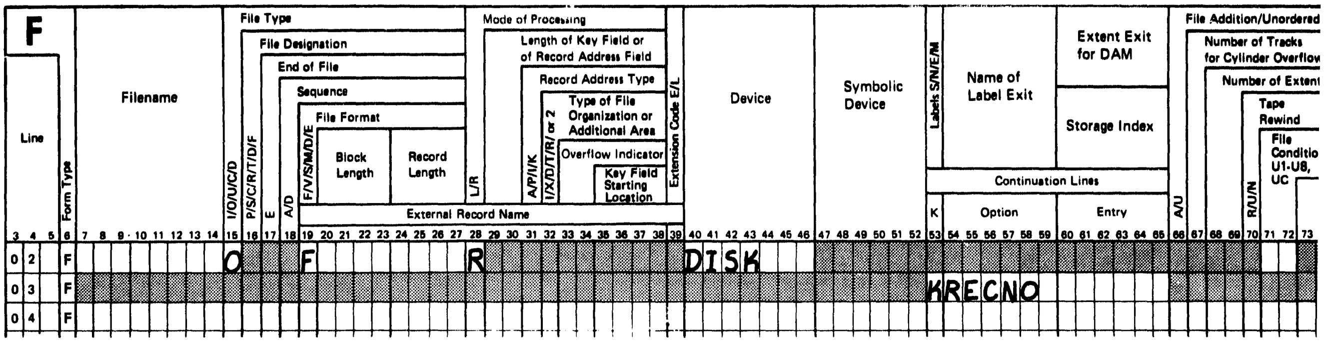
To write records to this output file, place into the RECNO field the relative record number of the record you want to write, and write data to that record during detail time, total time, or exception time of the RPG program cycle. This method of creating a direct file does not use a CHAIN operation to indicate the relative record number of the record to be written.

Before any output is written to the direct file that allows deletions, the disk space required for the file is automatically filled with deleted records (hexadecimal FFs). The relative record number that you place in the RECNO field indicates where the output record is to be written to the direct file. The information in the output record is written over the deleted record, replacing the hexadecimal FFs with data.

If a deleted record is not replaced with data, it remains in the file. A record can be added later at this relative record number (see *Adding Records to a Direct File* later in this chapter). A deleted record cannot be read; if a CHAIN operation chains to a deleted record, the indicator coded in columns 54 and 55 of the calculation specifications turns on to indicate that a record was not found at that relative record number.

If the direct file contains a record with the same relative number as the record you are writing, error message RPG-9070, OUTPUT TO A NONDELETED RECORD, is displayed. If the person using the display station responds to the message with option 1, the program bypasses the duplicate record and continues processing.

To create a direct file that allows deletions, code entries in the unshaded columns of the file description specifications shown below:



On the first line:

- Columns 7 through 14 must contain the name of the file.
- Column 15 must contain O to indicate that the file is an output file.
- Column 19 must contain F or blank.
- Columns 20 through 23 must contain the block length or blanks.
- Columns 24 through 27 must contain the record length.
- Column 28 must contain R to indicate that the file is to be processed randomly.
- Columns 40 through 43 must contain DISK.
- Columns 71 and 72 can contain an external indicator, U1 through U8.

On the second line:

- Column 53 must contain K to indicate that this is a continuation line that provides additional information about the file being described.
- Columns 54 through 58 must contain RECNO, which stands for relative record number.
- Columns 60 through 65 must contain the name of the field that contains the relative record number. The field must be defined on either the input specifications or the calculation specifications as a 7-position numeric field with zero decimal positions. That field is called the RECNO field.

Example of Creating a Direct File That Allows Deletions

In Figure 5-8, the direct file being created, CUSTFILE, is defined on the file description specifications as an output file that is processed randomly (O and R in columns 15 and 28). The file description continuation line indicates that CUSTNO, which is a field in the input file RECIN, contains the relative record number of the record to be written to the output file CUSTFILE. An output record is written for each record read from RECIN. No calculation specifications are required.

```
     F (File Description Specifications)
0 2  FRECIN   IP  F  60  60            DISK
0 3  FCUSTFILEO   F  60  60R           DISK
0 4  F                                                 KRECNO CUSTNO
0 5  F
```

```
     I (Input Specifications)
0 1  IRECIN   NS  01   1 C1
0 2  I                                        2   80CUSTNO
0 3  I                                        9  23 CUSTNM
0 4  I                                       24  39 ADDR
0 5  I                                       40  55 CTYST
0 6  I                                       56  60 ZIP
0 7  I                                        2  60 RECORD
0 8  I
```

```
     O (Output Specifications)
0 1  OCUSTFILED       01
0 2  O                                 1 'A'
0 3  O                       RECORD   60
0 4  O
```

Figure 5-8. Creating a Direct File That Allows Deletions

Reading a Direct File

After the direct file is created, you can read records from it when you want to display the information, create or update other files, or print a report. You can read records from a direct file in the following ways:

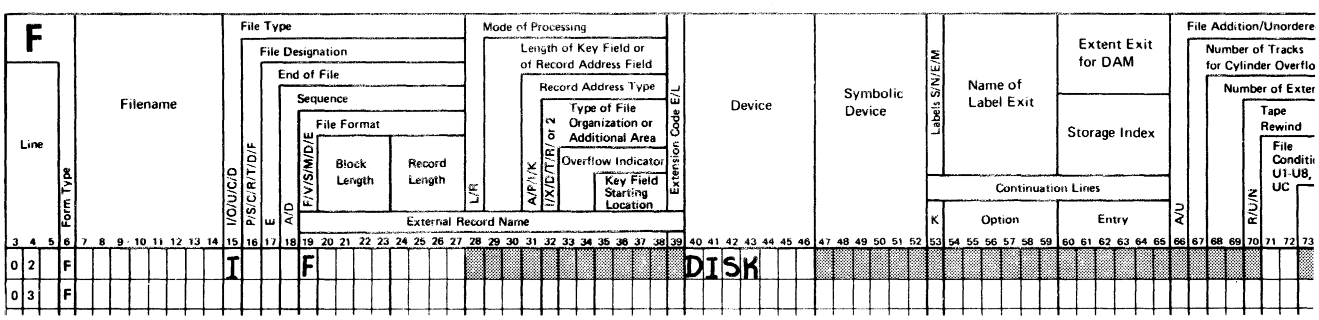
- Consecutively
- Randomly by relative record number
- Randomly by relative record number and/or consecutively
- Randomly by address output (addrout) file

Note: An alternative index file can also be created for a direct file to provide another method of reading the records in the file. For information about creating an alternative index file see *Creating an Alternative Index File for an Indexed File* later in this chapter.

Reading Consecutively

Reading a direct file consecutively means reading the records in the order in which they occur in the file. That is, the first record in the file is read first, the second record is read second, and so on. If the file allows records to be deleted, the program does not read deleted records; it skips them and reads the next record present. You read a direct file consecutively if you want to look at most or all of the records in the file. In this case, reading consecutively is much more efficient than reading randomly.

To read a direct file consecutively, code entries in the unshaded columns of the file description specification shown below:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain I to indicate that the file is an input file.

Column 16 must contain P, S, or D:

- If you code P (primary) or S (secondary), the program reads a record at input time of the RPG program cycle. For an explanation of how primary and secondary files are read, see Chapter 11.

- If you code D (demand), the program reads a record when a READ operation occurs in the calculation specifications. For information about the READ operation, see Chapter 28.

Column 17 must contain E or blank if column 16 contains P or S. E indicates that all records from the file must be processed before the program can end. Blank indicates that the program can end before it processes all records from the file.

Column 18 must contain A, D, or blank if column 16 contains P or S:

- A indicates that the program checks that the records in the file are in ascending sequence.

- D indicates that the program checks that the records are in descending sequence.

- Blank means that the program does not check the record sequence.

Column 19 must contain F or blank.

Columns 20 through 23 must contain the block length or blanks.

Columns 24 through 27 must contain the record length.

Columns 40 through 43 must contain DISK.

Columns 71 and 72 can contain an external indicator, U1 through U8.

Example of Reading Consecutively

Suppose you want to process a direct customer file, named CUSTFILE, to produce a monthly report. This report lists all customers that have placed no orders during the month. Sales personnel can use this report to plan follow-up calls. The file is in sequence by customer number, and the program checks every record. Therefore, the file is processed consecutively.

Figure 5-9 shows how to code the specifications to read records consecutively from CUSTFILE to produce REPORT1, a list of recently inactive customers. The OR line on the input specifications causes the program to skip blank record locations, because record-identifying indicator 03 on the OR line is not used elsewhere in the program.

Line	Form Type	Filename	File Type	File Designation	End of File	Sequence	File Format	Block Length	Record Length	Overflow Indicator	Device
02	F	CUSTFILE	I	P			F	256	128		DISK
03	F	REPORT1	O				F	100	100	OF	PRINTER
04	F										

Line	Form Type	Filename or Record Name	AND/OR	Sequence	Record Identifying Indicator, or **	Position (1)	Not (N)	C/Z/D	Character	Position (2)	Not (N)	C/Z/D	Character	Field Location From	Field Location To	Decimal Positions	RPG Field Name
01	I	CUSTFILE		NS	02	1		C	2	2		C	A				
02	I		OR		03												
03	I													3	6		CUSTNO
04	I													10	12		SLSMN#
05	I													13	30		CUSNAM
06	I													47	62		CTYSTA
07	I													68	69		CREDIT
08	I													70	75	0	LSTORD
09	I													82	88	2	CURPER
10	I													89	95	2	LSTPER
11	I													96	101	2	ARLT30
12	I													102	107	2	AR3060
13	I													108	113	2	AR6090
14	I													114	119	2	AROV90
15	I																

An OR line with a record-identifying indicator not used elsewhere in the program causes unwanted records to be bypassed, including blank records.

Figure 5-9 (Part 1 of 2). Reading Records Consecutively from a Direct Customer File

Line	Form Type	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Resulting Indicators (Equal)	Comments
01	C	Ø2	CURPER	COMP	Ø				Ø4	
02	C	Ø2 Ø4		Z-ADD	ARLT3Ø	TOTAR	7	2		
03	C	Ø2 Ø4		ADD	AR3Ø6Ø	TOTAR				
04	C	Ø2 Ø4		ADD	AR6Ø9Ø	TOTAR				
05	C	Ø2 Ø4		ADD	AROV9Ø	TOTAR				
06	C									

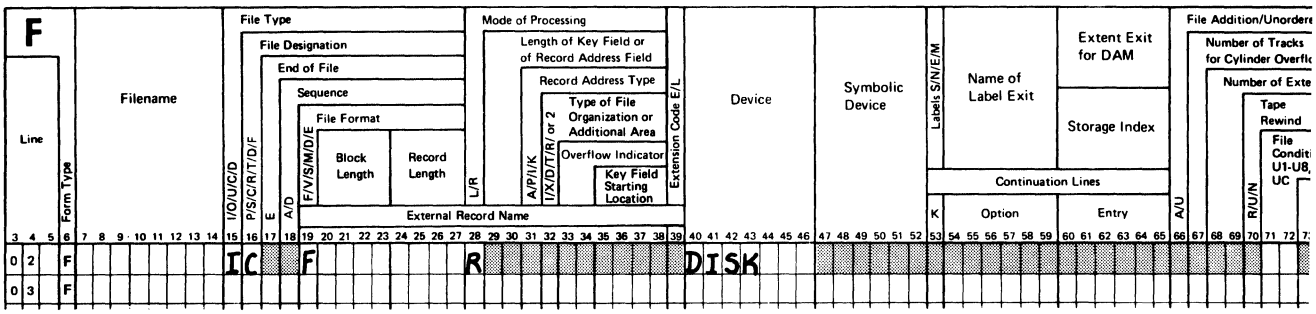
Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Space After	Skip Before	Output Indicators	Field Name or EXCPT Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	Constant or Edit Word
01	O	REPORT1	H	2	Ø6	1P					
02	O		OR			OF					
03	O									8	'CUSTOMER'
04	O									21	'NAME'
05	O									42	'CITY, STATE'
06	O									56	'SALESMAN'
07	O									69	'LAST ORDER'
08	O									84	'SLS PREV PER'
09	O									9Ø	'CRDT'
10	O									99	'TOT A/R'
11	O		D	1		Ø2 Ø4					
12	O						CUSTNO			6	
13	O						CUSNAM			28	
14	O						CTYSTA			46	
15	O						SLSMN#			53	
16	O						LSTORDY			68	
17	O						LSTPERT			82	
18	O						CREDIT			89	
19	O						TOTAR	J		1ØØ	
20	O										

Figure 5-9 (Part 2 of 2). Reading Records Consecutively from a Direct Customer File

Reading Randomly by Relative Record Number

Reading a direct file randomly by relative record number means reading only those records that you specifically identify by their position relative to the beginning of the file. To read those records, you must use a CHAIN operation in the calculation specifications. Factor 1 of the CHAIN operation must contain the relative record number itself or the name of the field, table, or indexed array that contains the relative record number. For more information about the CHAIN operation, see Chapter 28.

To read a direct file randomly by relative record number, code entries in the unshaded columns of the file description specifications shown below:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain I to indicate that the file is an input file.

Column 16 must contain C to indicate that the file is a chained file. To read this file, you must code a CHAIN operation in the calculation specifications.

Column 19 must contain F or blank.

Columns 20 through 23 must contain the block length or blanks.

Columns 24 through 27 must contain the record length.

Column 28 must contain R to indicate that the file is processed randomly.

Columns 40 through 43 must contain DISK.

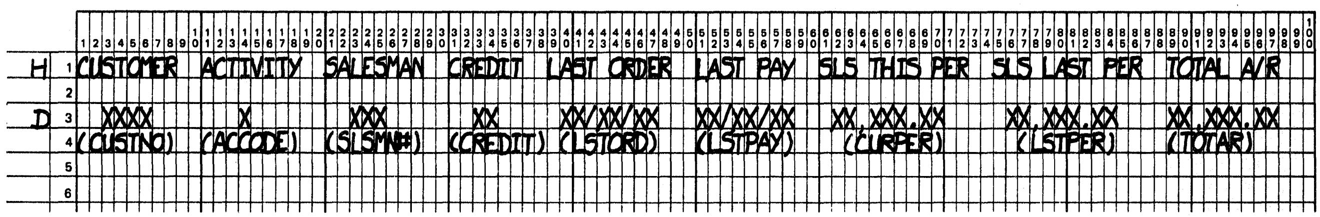
Columns 71 and 72 can contain an external indicator, U1 through U8.

Example of Reading Randomly by Relative Record Number

In the *Example of Reading Consecutively*, we showed how to process the direct customer file CUSTFILE consecutively. Now suppose that you want to read records from that file randomly. You want to make inquiries each day about customer accounts whose records have record identification code I in position 1, followed by the customer account number (CSTMER).

The program reads an input record (the customer account number) from the primary input file, INFILE. The program uses the customer account number as the relative record number to chain to CUSTFILE. If the program finds a record in CUSTFILE that has the same customer account number as the record in INFILE, the computer prints sales and accounts receivable information for that customer. If the program does not find a record in CUSTFILE that matches the customer account number, the message RECORD NOT FOUND--INVALID RECORD NUMBER is printed at run time.

Figure 5-10 shows the printer output for this example. Figure 5-11 shows how to code the specifications for this example.




```
CUSTOMER   ACTIVITY   SALESMAN   CREDIT   LAST ORDER   LAST PAY   SLS THIS PER   SLS LAST PER   TOTAL A/R

  3119        A          105       01       4/17/83     4/01/83        360.00         239.50       360.00
  6678 RECORD NOT FOUND--INVALID RECORD NUMBER
  1703        I           35       03      11/19/82    12/01/82           .00            .00          .00
```

Figure 5-10. Printer Output from Random Inquiries into a Direct Customer File

Line	Form Type	Filename	File Type	File Designation	File Format	Block Length	Record Length	Mode of Processing	Overflow Indicator	Device
02	F	INFILE	I	P	F	96	96			DISK
03	F	CUSTFILE	I	C	F	256	128	R		DISK
04	F	PRINTOUT	O		F	1ØØ	1ØØ		OF	PRINTER
05	F									
06	F									

The direct file is defined as a chained input file to be retrieved randomly.

Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator	Position	C/Z/D	Character	Field Location From	Field Location To	Decimal Positions	RPG Field Name	Field Indicators Zero or Blank
01	I	INFILE	NS	Ø7	1	C	I					
02	I							2	5		CSTMER	
03	I	CUSTFILE	NS									
04	I							2	2		ACCODE	
05	I							3	6	Ø	CUSTNO	Ø8
06	I							1Ø	12		SLSMN#	
07	I							68	69		CREDIT	
08	I							7Ø	75	Ø	LSTORD	
09	I							76	81	Ø	LSTPAY	
10	I							82	88	2	CURPER	
11	I							89	95	2	LSTPER	
12	I							96	1Ø1	2	ARLT3Ø	
13	I							1Ø2	1Ø7	2	AR3Ø6Ø	
14	I							1Ø8	113	2	AR6Ø9Ø	
15	I							114	119	2	AROV9Ø	
16	I											

A chained file must contain an alphabetic sequence entry.

Figure 5-11 (Part 1 of 2). Reading a Direct Customer File Randomly by Relative Record Number

```
C SETOF                                   08
C 07       CSTMER    CHAINCUSTFILE         13
C 07N13N08          Z-ADDARLT30    TOTAR   72
C 07N13N08          ADD  AR3060    TOTAR
C 07N13N08          ADD  AR6090    TOTAR
C 07N13N08          ADD  AROV90    TOTAR
```

The customer number from the input record is used as the relative record number to chain to the direct file. Indicator 13 will turn on if a record is not found in the direct file.

Indicator 13 is used to condition subsequent operations.

```
OPRINTOUTH  204    1P
O       OR         OF
O                                8 'CUSTOMER'
O                               18 'ACTIVITY'
O                               28 'SALESMAN'
O        D  1      07N13N08
O                        CUSTNO  6
O                        ACCODE 14
O                        TOTAR J97
O        D  1      07 13N08
O                        CSTMER  6
O                               31 'RECORD NOT FOUND--INVALI'
O                               46 'D RECORD NUMBER'
O        D  1      07 08N13
O                        CSTMER  6
O                               24 'NO RECORD IN FILE'
```

Other headings (see Figure 5-10).

N13 means that this line will be printed if a record is found in the direct file.

Other fields (see Figure 5-10).

When a record is not found in the direct file, this line is printed.

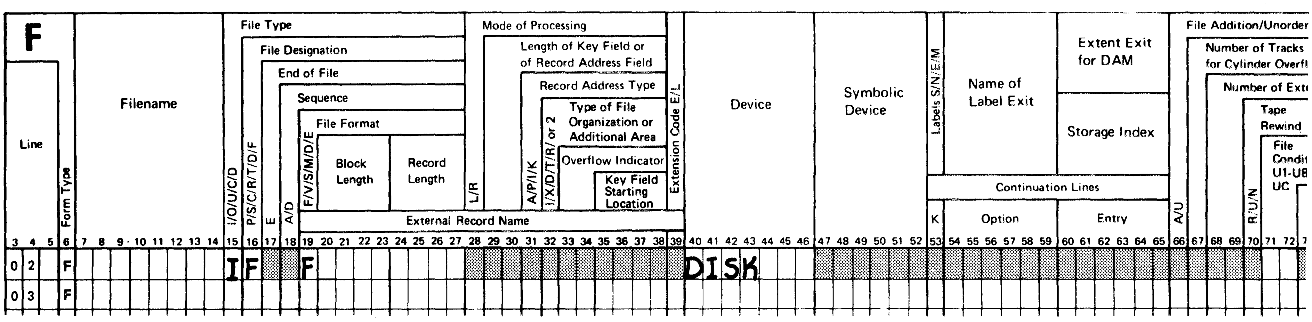
Figure 5-11 (Part 2 of 2). Reading a Direct Customer File Randomly by Relative Record Number

Reading Randomly by Relative Record Number and/or Consecutively

If you want to read a file both randomly and consecutively, use a full-procedural file. You can read a full-procedural file randomly like a chained file and/or consecutively like a demand file. That is, you can chain to a specific relative record number in the file and then read records consecutively from that point. To read the file randomly, you use a CHAIN operation in the calculation specifications; to read it consecutively, you use a READ or READP operation. You cannot use a READE operation to read the file consecutively, because the READE operation cannot read by relative record number. For example, if you code a CHAIN operation to relative record number 10 and then code a READ operation, the program chains to relative record number 10 and then reads the following record.

It is not necessary to code both a CHAIN and a READ or a READP operation, but you must code at least one CHAIN, one READ, or one READP operation in order to read a full-procedural file. For information about the CHAIN READ, and READP operation codes, see Chapter 28.

Code a full-procedural file as an input file. Code entries in the unshaded columns of the file description specifications shown below:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain I to indicate that the file is an input file.

Column 16 must contain F to indicate that the file is a full-procedural file. You must code a CHAIN, READ, or a READP operation in the calculation specifications in order to read a full-procedural file.

Column 19 must contain F or blank.

Columns 20 through 23 must contain the block length or blanks.

Columns 24 through 27 must contain the record length.

Columns 40 through 43 must contain DISK.

Columns 71 and 72 can contain an external indicator, U1 through U8.

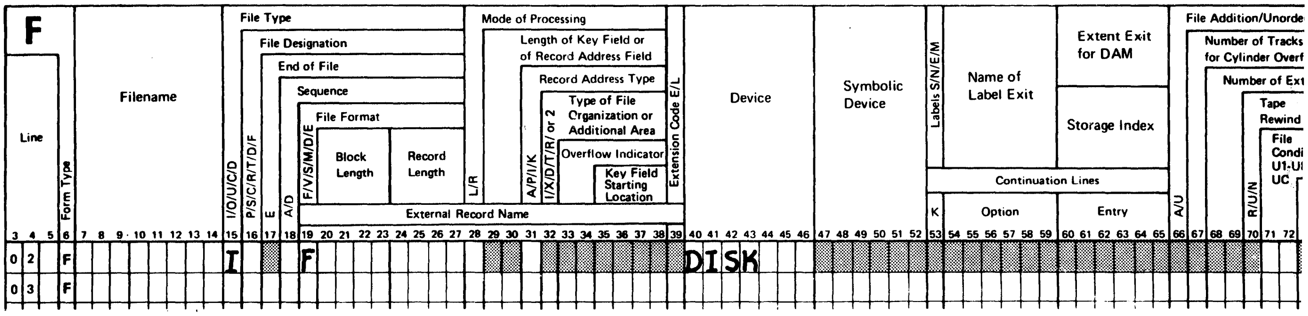
Reading Randomly by Address Output (Addrout) File

An *address output (addrout) file* is a record address file produced by a sort program. (A *record address file* is an input file that tells the program which records to read from a DISK file and the order in which to read them.) An address output file contains the relative record numbers of the records in a DISK file. The advantages of an address output file are that:

- The space required for the address output file is much less than the space required for a sorted sequential file.
- The sort runs much faster.
- The original file is unchanged.

You can have only one address output file in a program. When an RPG program uses an address output file, it automatically reads the relative record numbers consecutively from the address output file. You do not have to code a READ operation for the address output file. Then, using the relative record number, the program randomly reads the DISK file to process the corresponding record. In this way, the program can process a direct DISK file in a new sequence without actually sorting the records and creating a new file. Also, once the file description and extension specifications are coded for the DISK file and for its associated address output file, you can code the DISK file as though you were reading the direct file sequentially. If the DISK file is a full-procedural file, you must code a READ operation in the calculation specifications; you cannot use a CHAIN, READE or READP operation to read a full-procedural file randomly by an address output file. No input specifications are required for the address output file.

If you want to read records randomly by an address output file, code the file description specifications as shown below:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain I to indicate that the file is an input file.

Column 16 must contain P, S, or F:

- If you code P (primary) or S (secondary), the record is read as part of the RPG program cycle. For an explanation of how primary and secondary records are read, see Chapter 11.

- If you code F (full-procedural), you must code a READ operation in the calculation specifications.

Column 18 can contain A, D, or blank if column 16 contains P or S:

- A indicates that the program checks that the records in the file are in ascending sequence.

- D indicates that the program checks that the records in the file are in descending sequence.

- Blank indicates that the program does not check the sequence of records in the file.

Column 19 must contain F or blank.

Columns 20 through 23 must contain the block length or blanks.

Columns 24 through 27 must contain the record length.

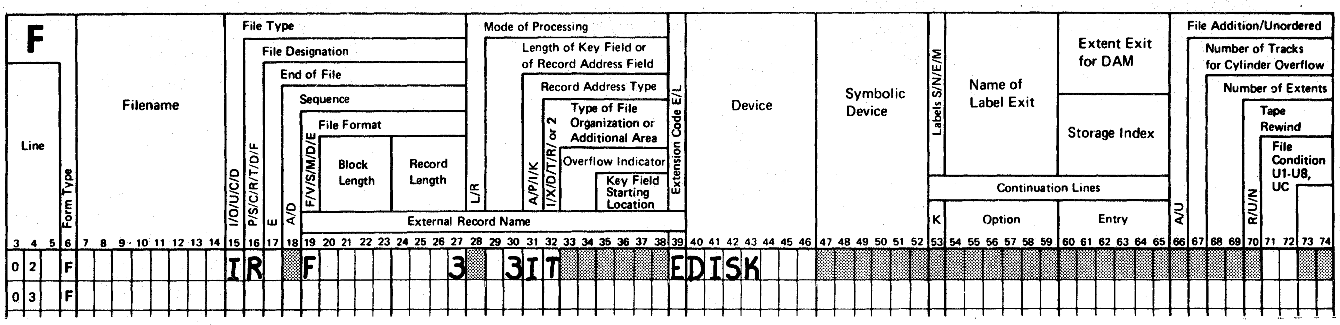
Column 28 must contain R if column 16 contains P or S. The R indicates that the file is processed randomly by an address output file.

Column 31 must contain I if column 16 contains P or S. The I indicates that relative record numbers from the address output file are used to process the file.

Columns 40 through 43 must contain DISK.

Columns 71 and 72 can contain an external indicator, U1 through U8.

For the address output file, code entries in the unshaded columns of the file description specifications shown below:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain I to indicate that the file is an input file.

Column 16 must contain R to indicate that the file is a record address file.

Column 17 must contain E or blank. E indicates that all records from the file must be processed before the program can end. A blank indicates that the program can end whether or not all records from the file are processed.

Column 19 must contain F or blank.

Columns 20 through 23 must contain the block length or blanks.

Column 27 must contain 3 because each record in an address output file is a relative record number, which is always three positions long.

Column 30 must contain 3 because relative record numbers in address output files are always three positions long.

Column 31 must contain I to indicate that the file is an address output file.

Column 32 must contain T to indicate that the file is an address output file.

Column 39 must contain E to indicate that the file is further described on extension specifications.

Columns 40 through 43 must contain DISK.

Columns 71 and 72 can contain an external indicator, U1 through U8.

Two entries are required on the extension specifications:

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
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
0 1	E																	
0 2	E																	

Columns 11 through 18 must contain the name of the address output file. This name must be the same one coded on the file description specifications for the address output file.

Columns 19 through 26 must contain the name of the direct file to be processed by the address output file. This name must be the same one coded on the file description specifications for the direct file.

Updating a Direct File

Updating records in a direct file involves reading a record, changing some data in the record, and writing the record back to its original location in the file. If you try to update a record that was not the last record read, error message RPG-9043, FILE TRIED RECORD UPDATE BEFORE INPUT, is displayed. The fields to be updated must be described on both the input and the output specifications.

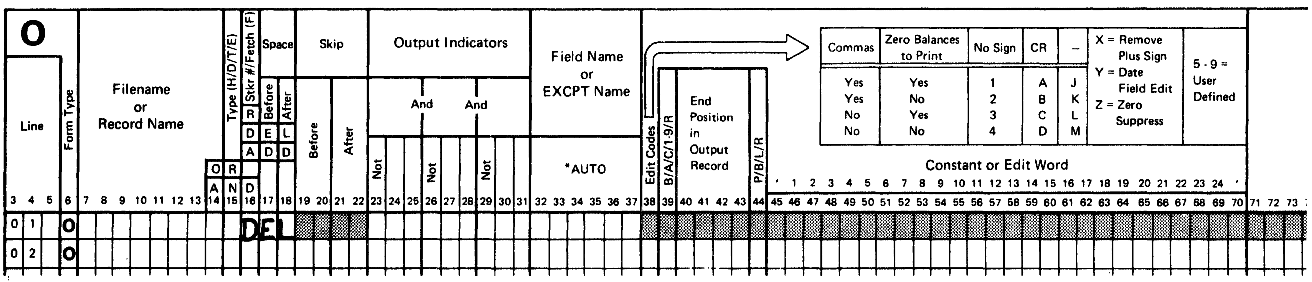
When you update records in a direct file, the file can be processed in any of the following ways:

- Consecutively
- Randomly by relative record number
- Randomly by relative record number and/or consecutively
- Randomly by address output (addrout) file

Deleting Records from a Direct File

Updating a file can include deleting records from the file. To allow records to be deleted from the file, the control language FILE statement coded when the file was created must use the DFILE-YES parameter. For information about the FILE statement, see the *System Reference* manual. If you try to delete a record from a file that does not allow deletions, error message RPG-9067, INVALID OPERATION ATTEMPTED, is displayed.

To delete a record, you first read the record (either randomly or consecutively) and then, with DEL coded in the output specification, write the record back to the same file. Code entries in the unshaded columns of the output specifications shown below:



Columns 7 through 14 must contain the name of the output file.

Column 15 must contain D, T, or E, to indicate that the record is to be deleted at detail, total, or exception output time of the program cycle.

Columns 16 through 18 must contain DEL to indicate that the record is to be deleted.

Columns 23 through 31 can contain output indicators.

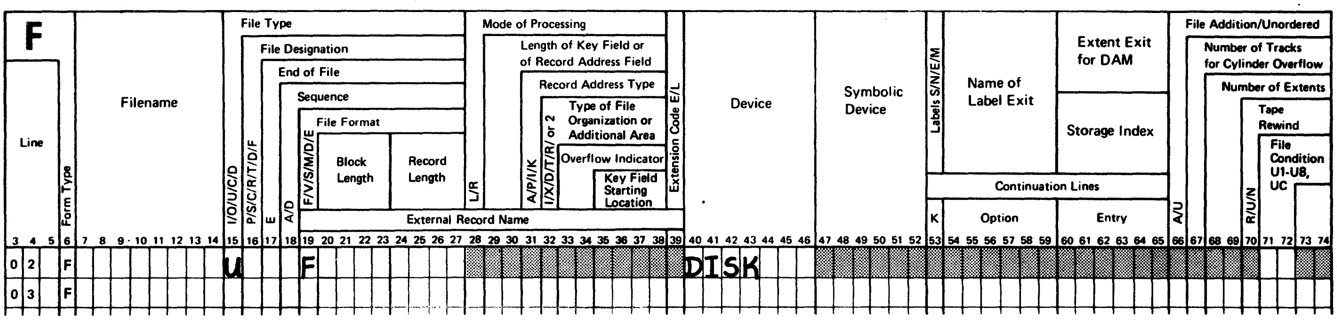
Columns 32 through 37 can contain an EXCPT name if column 15 contains E.

Records are not physically removed from a file when they are deleted. Instead, deleted records are filled with hexadecimal FFs. That is, all the bits for every character in the deleted record are set on.

Updating Consecutively

You can update records in a direct file consecutively. If the file is a primary or secondary file (P or S in column 16 of the file description specifications), the program reads a record from the file at input time in the RPG program cycle, and the program writes a record to update the file during detail output or exception output time in the program cycle. If the file is a demand file (D in column 16), the program reads a record when a READ operation occurs in the calculation specifications, and the program writes a record to update the file at detail output, total output, or exception output time in the program cycle.

Code the file description specifications as shown below:

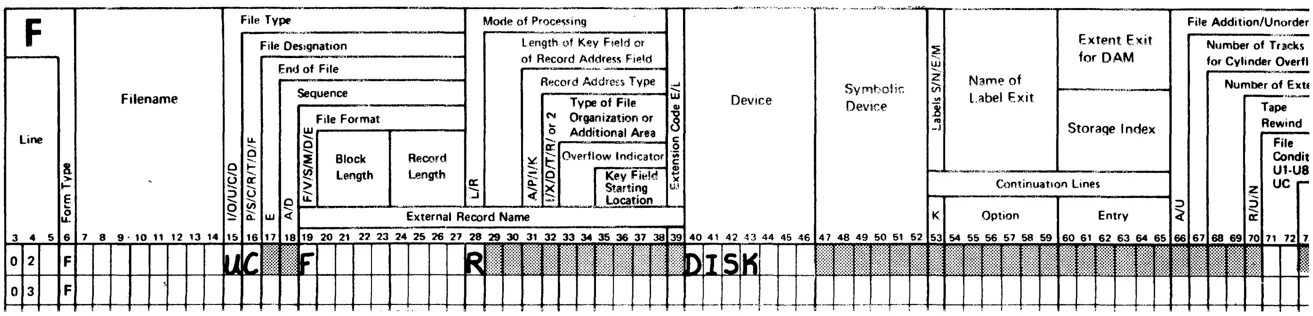


Column 15 must contain U to indicate that the file is an update file.

Entries in the other columns are the same as those for reading consecutively.

Updating Randomly by Relative Record Number

You can update records in a direct file processed randomly by relative record number. The file is defined as a chained file (C in column 16 of the file description specifications). This means that the program reads a record from this file when a CHAIN operation occurs in the calculation specifications, and that the program can write a record to update the file during detail output, total output, or exception output time in the RPG program cycle. Code the file description specifications as shown below:



Column 15 must contain U to indicate that the file is an update file.

Entries in the other columns are the same as those for reading randomly by relative record number.

Example of Updating Randomly by Relative Record Number

Each day you want to prepare invoices for customer orders for the file described in the *Example of Reading Consecutively*. You use information from the invoices to update the customer file CUSTFILE. The records in the invoice file (INFILE) are unordered, so you process it randomly.

The records in the invoice file contain the date and total amount of transactions for each customer. The records also contain new addresses when the customer addresses change. The program reads each record entered at a display station and uses the customer number (CUSTMR) to chain to the direct file. The amount of the transaction is added to the total sales for the current period (CURPER) and to the accounts receivable amount (ARLT30). The transaction date is placed in the date of last order (LSTORD) field in the customer record.

If an address change is indicated (by X in column 18 of the input record), the new customer address replaces the old address.

If a record is not found in CUSTFILE because of an invalid relative record number, the input record and the message ABOVE RECORD NOT FOUND--INVALID CUSTOMER NUMBER are printed.

Figure 5-12 shows how to code the specifications for this example.

Line	Form Type	Filename	File Type	File Designation	File Format	Block Length	Record Length	Length of Key Field or of Record Address Field	Record Address Type	Device
02	F	INFILE	I	P	F	96	96			DISK
03	F	CUSTFILE	U	C	F	256	128		R	DISK
04	F	PRINT	O		F	96	96			PRINTER
05	F									

Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator	Position	Not (N)	C/Z/D	Character	Field Location From	Field Location To	Decimal Positions	RPG Field Name
01	I	INFILE	NS	10	1		C	V				
02	I								2	5	0	CUSTMR
03	I								6	11		DATE
04	I								12	17	2	TOCOST
05	I								18	18		NEWADR
06	I								19	73		NAMADD
07	I								1	96		RECORD
08	I	CUSTFILE	NS									
09	I								3	6	0	CUSTNO
10	I								82	88	2	CURPER
11	I								96	101	2	ARLT30
12	I											

Figure 5-12 (Part 1 of 2). Updating a Direct Customer File Randomly by Relative Record Number

Line	Form Type	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Resulting Indicators (High)	Resulting Indicators (Low)	Resulting Indicators (Equal)
01	C	10	CUSTMR	CHAIN	CUSTFILE		13		
02	C	10 N13		ADD	TOCOST	CURPER			
03	C	10 N13		ADD	TOCOST	ARLT30			
04	C	10 N13	NEWADR	COMP	'X'				15
05	C								

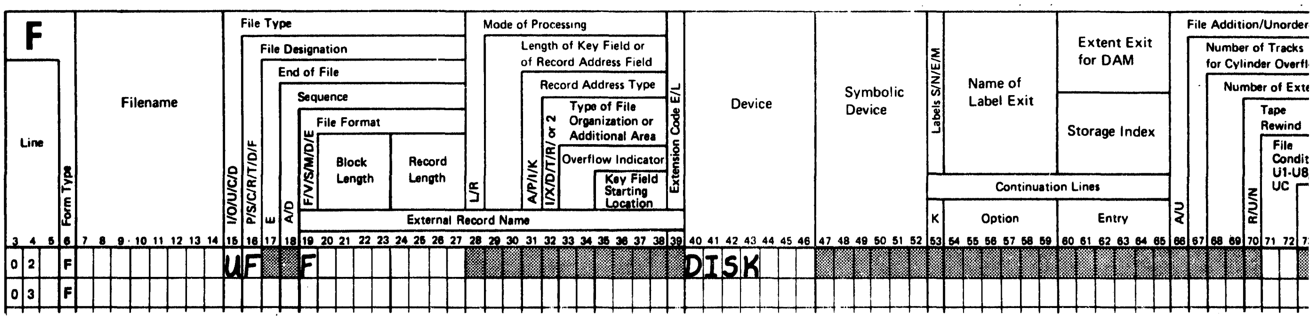
Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Space After	Output Indicators	Field Name or EXCPT Name	End Position in Output Record	Constant or Edit Word
01	O	CUSTFILE	D		10 N13			
02	O					DATE	75	
03	O					CURPER	88	
04	O					ARLT30	101	
05	O				15	NAMADD	67	
06	O	PRINT	D	1	10 13			
07	O					RECORD	96	
08	O		D	2	10 13			
09	O						24	'ABOVE RECORD NOT FOUND--'
10	O						47	'INVALID CUSTOMER NUMBER'
11	O							

Figure 5-12 (Part 2 of 2). Updating a Direct Customer File Randomly by Relative Record Number

Updating Randomly by Relative Record Number and/or Consecutively

You can update records in a direct file randomly by relative record number and/or consecutively. That is, the file is defined as a full-procedural file (F in column 16 of the file description specifications). The record to be updated can be read either randomly by relative record number with a CHAIN operation or consecutively with a READ or READP operation. The record cannot be read consecutively with a READE operation because the READE operation cannot read by relative record number. The output operation to update the record can occur during detail output, total output, or exception output time of the RPG program cycle.

Code the file description specifications as shown below:



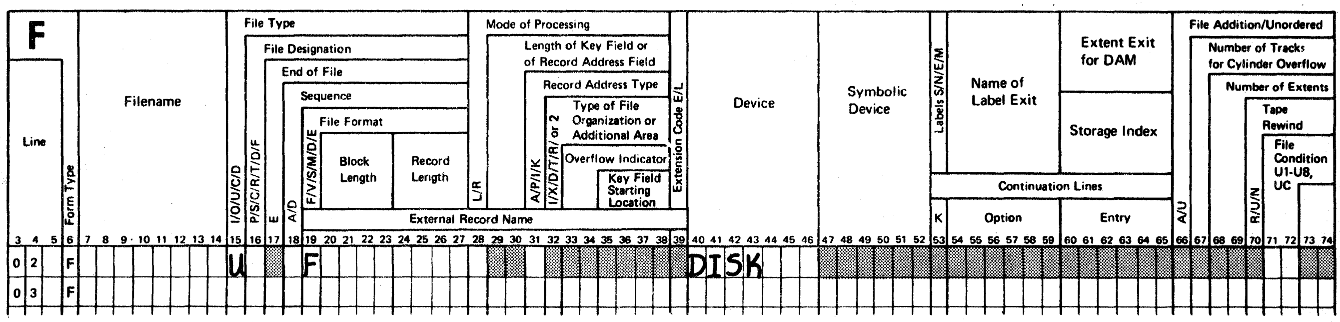
Column 15 must contain U to indicate that the file is an update file.

Entries in the other columns are the same as those for reading randomly by relative record number and/or consecutively.

Updating Randomly by Address Output (Addrout) File

You can update records in a direct file processed by an address output (addrout) file. The direct file can be a primary, secondary, or full-procedural file (P, S, or F in column 16 of the file description specifications). If the file is a primary or secondary file, the program reads a record from the file at input time of the RPG program cycle, and program writes a record to update the file at detail output or exception output time of the program cycle. If the file is a full-procedural file, the program reads a record from the file when a READ operation occurs in the calculation specifications, and the program writes a record to update the file at detail output, total output, or exception output time of the program cycle.

Code the file description specifications as shown below:



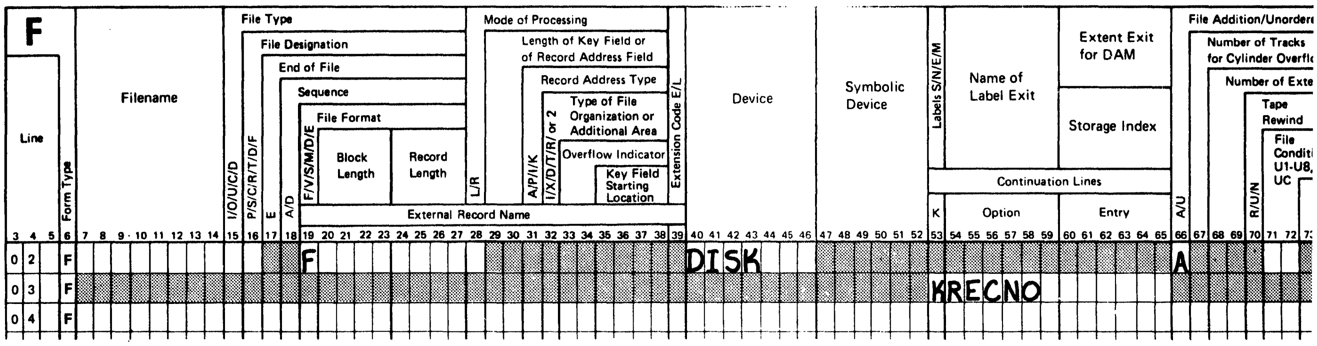
Column 15 must contain U to indicate that the file is an update file.

Entries in the other columns are the same as those for reading randomly by address output file.

Adding Records to a Direct File

You can add records to a direct file that is processed randomly by relative record number. The file must be one that allows records to be deleted; that is, when the file was created, the control language FILE statement must have had DFILE-YES specified. For information about the FILE statement, see the *System Reference* manual.

To add records to a direct file, code entries in the unshaded columns of the file description specifications shown below:



In the first line of the file description specifications:

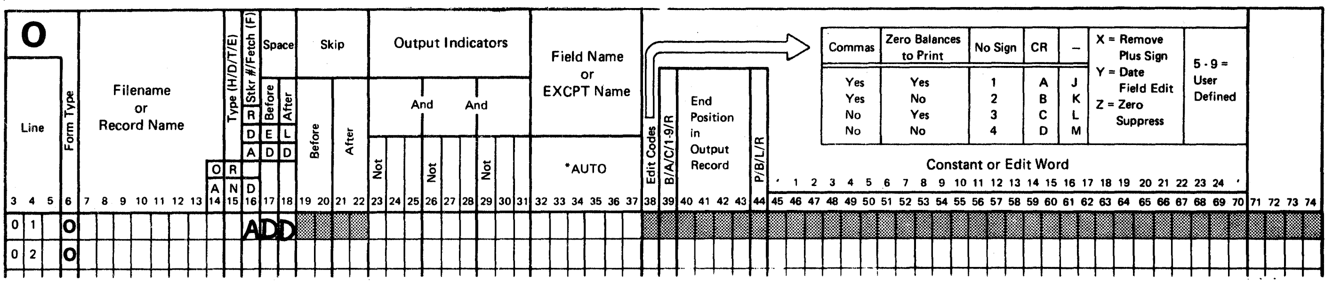
- Columns 7 through 14 must contain the name of the file.
- Column 15 must contain I or U to indicate that the file is an input file or an update file.
- Column 16 must contain C or F to indicate that the file is a chained file or a full-procedural file.
- Column 19 must contain F or blank.
- Columns 20 through 23 must contain the block length or blanks.
- Columns 24 through 27 must contain the record length.
- Column 28 must contain R if column 16 contains C.
- Columns 40 through 43 must contain DISK.
- Column 66 must contain A to indicate that records are to be added to the file.
- Columns 71 and 72 can contain an external indicator, U1 through U8.

In the second line:

- Column 53 must contain K to indicate that this line is a continuation line.

- Columns 54 through 58 must contain RECNO, which stands for relative record number.

- Columns 60 through 65 must contain the name of the field into which the relative record number is placed. The field must be defined on either the input specifications or the calculation specifications as a 7-position numeric field with zero decimal positions. That field is called the RECNO field.



On the output specifications for the record to be added, columns 7 through 14 must contain the name of the output file.

Column 15 can contain D, T, or E, to indicate that the record is to be added at detail, total, or exception output time of the program cycle.

Columns 16 through 18 must contain ADD to indicate that the fields defined on the following lines form the record to be added to the file.

Columns 23 through 31 can contain output indicators.

Columns 32 through 37 can contain an EXCPT name if column 15 of the output specifications contains E.

The RECNO field identifies the position in the file where the output record is to be added. (That record is the one described on the output specification that contains ADD in columns 16 through 18.) You must place into the RECNO field the relative record number of the record to be added to the file. It must be the relative record number of a deleted record. One way to place the relative record number into the RECNO field is to code the following sequence of operations in the calculation specifications:

1. Code a CHAIN operation with the relative record number in factor 1, the name of the chained file in factor 2, and a resulting indicator in columns 54 and 55 that turns on when a record is not found.

2. Code a Z-ADD operation with the same indicator that you used for the CHAIN operation as a conditioning indicator (in columns 10 and 11, 13 and 14, or 16 and 17), the relative record number in factor 2, and the RECNO field in the result field.

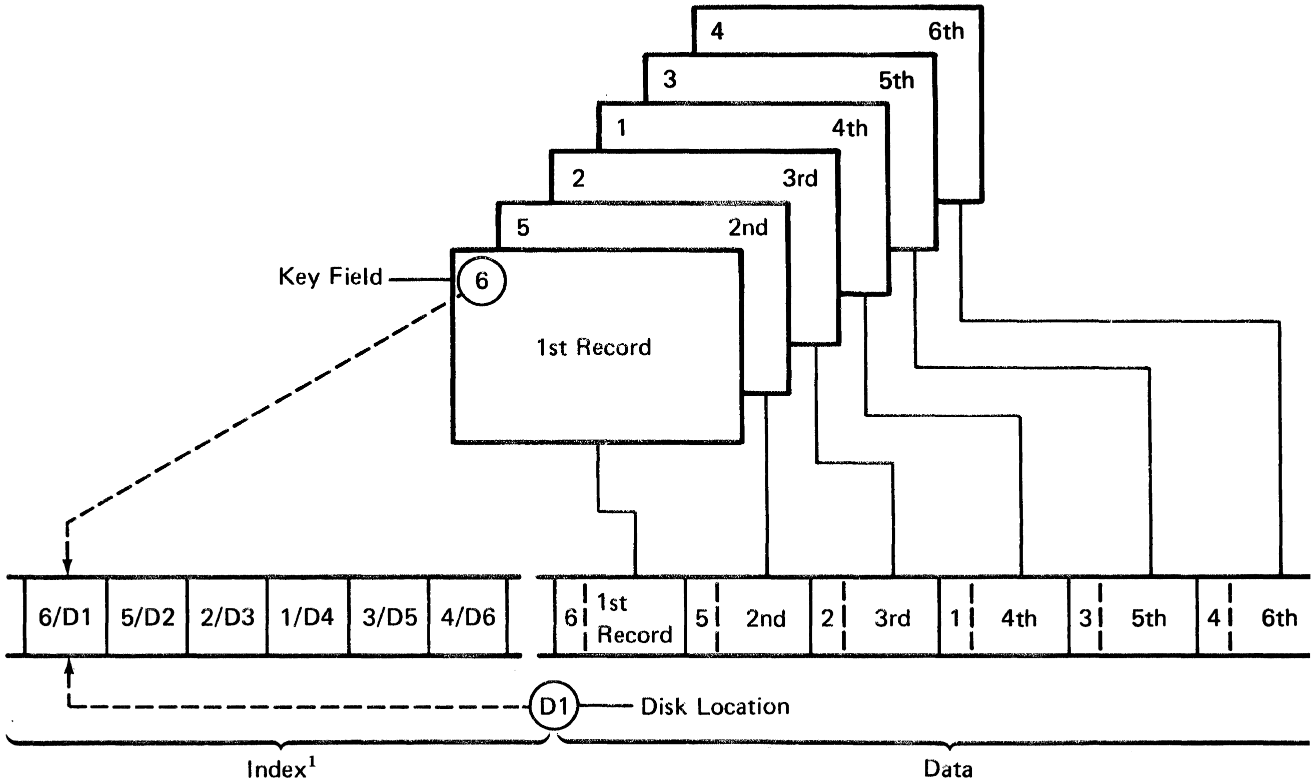
When a CHAIN operation (for a chained file) or a READ operation (for a full-procedural file) reads a nondeleted record, data management places into the RECNO field the relative record number of the record read.

When the program tries to add a record to the file, if the relative record number is not the number of a deleted record, the program stops and error message RPG-9070, OUTPUT TO A NONDELETED RECORD, is displayed. If you respond to the message by choosing option 1, the program continues but it does not add the record to the file.

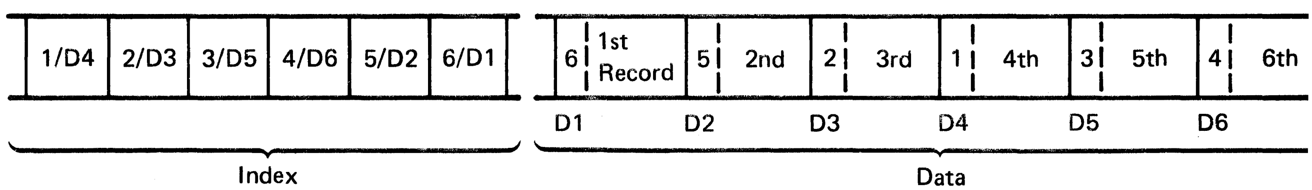
INDEXED FILES

An indexed file has two parts: an index and the data records (see Figure 5-13). The index contains an entry for each record in the file. Each index entry also has two parts: a key field and the disk address of the record for that index entry. The key field contains data that identifies each record individually. For example, the customer account number could be the key field to identify each record in a customer master file. The second part of the index contains the disk address of the record. The disk address tells where the record is stored on the disk. Thus, a program can go to the index, find the location of a record, go to that location, and find the record you want.

Records are stored in the data portion of the file in the same order in which they are written to the file. When a record is stored in the data portion, an entry for the record is made in the index. After the last entry is made in the index, the entries in the index are sorted into ascending order according to the key fields.



The order of the records in the data portion remains unchanged when the entries in the index are sorted.



---
[1] Entries are of the form key field/disk location (D1 = 1st disk location, D2=2nd disk location, and so on).

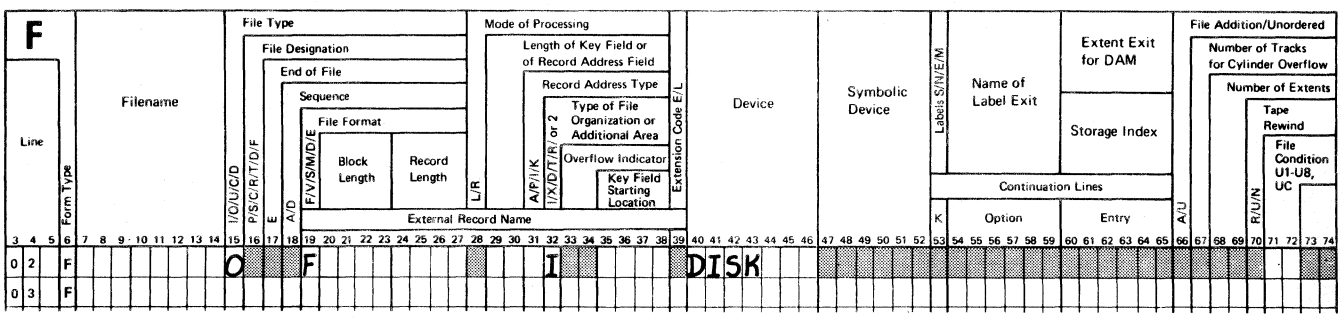
Figure 5-13. Organization of an Indexed File

Creating an Indexed File

You can create an indexed file by writing records in an ordered sequence or in an unordered sequence. *In an ordered sequence* means that the records are written in ascending order of key field. *In an unordered sequence* means that the records are written in no particular order of key field.

Creating an Indexed File by Writing Records in an Ordered Sequence

To create an indexed file by writing records in ascending order of key field, describe the file in the unshaded columns of the file description specifications shown below:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain O to indicate that the file is an output file.

Column Column 19 must contain F or blank to indicate that the record length is fixed. That is, all records in the file must be the same length.

Columns 20 through 23 must contain the block length or blanks. The maximum block length is 9999. If you do not enter any number in these columns, the block length equals the record length.

Columns 24 through 27 must contain the record length. The record length can be any number from 1 through 4096.

Columns 29 and 30 must contain the length of the key field in each record. The maximum length is 99 positions, unless an indexed file is processed sequentially within key-field limits using a CONSOLE device, in which case the maximum length is 29 positions. Key fields in packed-decimal format can be up to 8 positions in length.

Column 31 must contain A if the key field is in zoned-decimal or alphameric format, or P if the key field is in packed-decimal format. If you create a file with a key field in packed-decimal format, you must also code the key field in packed-decimal format on the output specifications (P in column 44).

Column 32 must contain I to indicate that the file is an indexed file.

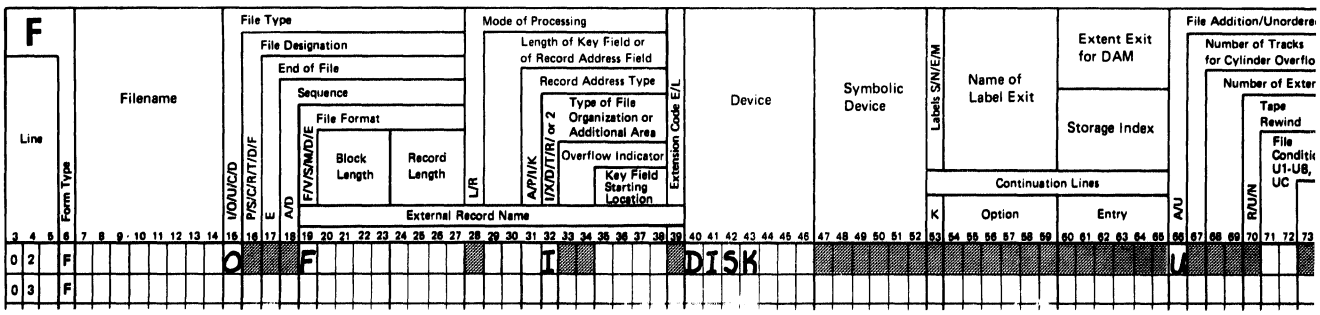
Columns 35 through 38 must contain the record position in which the key field begins. The maximum number you can use for the starting position of the key field is the record length minus the length of the key field + 1.

Columns 40 through 43 must contain DISK.

Columns 71 and 72 can contain an external indicator, U1 through U8.

Creating an Indexed File by Writing Records in an Unordered Sequence

To create an indexed file by writing the records in no particular order of key field, you code the same entries required to create an indexed file by writing the records in ascending order of key fields, but you also code U in column 66 to indicate that the records can be written in an unordered sequence.



Example of Creating an Indexed File

Figure 5-14 shows how to code the file description, input, and output specifications to create an indexed file by writing records in an unordered sequence.

```
  02 FINPUT   IPE F  96  96            DISK
  03 FMASTER  O   F 192  96  6AI     2 DISK                      U
  04 F
```

The output file, MASTER, is an indexed file to be loaded and processed by key fields. The U in column 66 of the file description specifications indicates that an unordered load is to be done. The input file, INPUT, is unsequenced.

```
  01 IINPUT   NS  Ø1
  02 I                                        1  96 RECORD
  03 I
```

The key fields from which the index is to be built appear as the first eight positions of the output record. As the DISK file is loaded, the key field is extracted from the record and an index entry containing the location of the record on the disk is built. After the entire file is loaded and an index entry is constructed for each record, the index entries are sorted into ascending sequence.

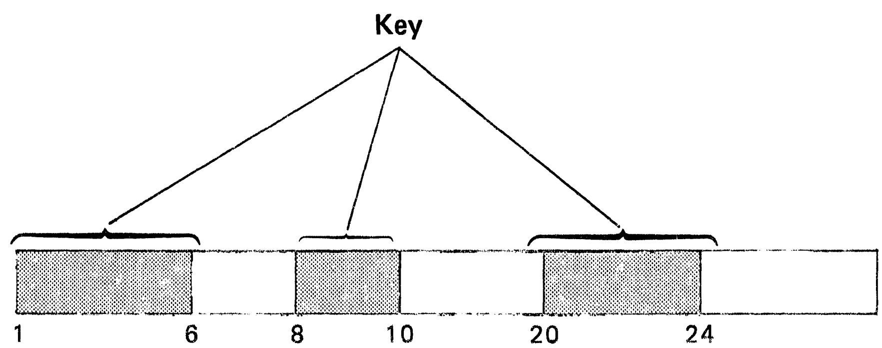
```
  01 OMASTER  D        Ø1
  02 O                         RECORD    96
  03 O
```

Figure 5-14. Creating an Indexed File by Writing Records in an Unordered Sequence

Creating an Alternative Index File for an Indexed File

After you create an indexed file, you can create alternative index files for that file. Using an alternative index file is equivalent to using an address output (addrout) file to process an indexed file. Each alternative index file uses one or more different portions of the record in the physical file as the key field. (An alternative index file may also be created for a sequential or direct physical file). Therefore, you can process records from the file in various sequences, depending on which index you use. For example, for a personnel file you can use the employee number as the primary index key and the department number as a key in an alternative index file.

An alternative index file has either a single field as its key, as have indexed files, or it can have up to three noncontiguous fields as its key. Only alternative index files can have keys with noncontiguous fields. For example, the fields in positions 1 through 6, 8 through 10, and 20 through 24 can be specified as one key:



One field cannot overlap another field of the key. For example, if one key is specified in positions 10 through 15, another field cannot be specified in positions 14 through 20. For details on coding for keys with noncontiguous fields, see *Coding the File Description Specifications for an Alternative Index File with Noncontiguous Fields as its Key* later in this chapter.

The existing indexed file from which you create alternative index files is called the *physical file.* To create an alternative index file for a physical file, use the control language procedure BLDINDEX. On the BLDINDEX procedure, you specify:

- The name of the alternative index file to be created
- The starting location of each field that is to be part of the key (1 to 3 fields may be used, and the value for each field must be a decimal number from 1 through 4096)
- The length of each field that is to be part of the key (the total length of all fields that make up the key cannot be greater than 99 bytes)
- The name of the physical file
- Creation date of the physical file

- Whether duplicate key fields are allowed for the alternative index file
- The preferred disk location

For more information about the BLDINDEX procedure, see the *System Reference* manual.

You can use an alternative-index file to do any of the following:

- Read records sequentially by key field
- Read records randomly by key field
- Read records within key-field limits by using the SETLL operation or a limits file
- Update records in the file
- Delete records from the file
- Add records to the file

These are exactly the same operations that you can do with the primary index. The program does not know if the indexed file defined in the program uses the primary index or an alternative index file. All the program knows is that the file is an indexed file. Therefore, the program treats an alternative index file the same as the primary index.

For more information about using alternative indexes, see the *Concepts and Programmer's Guide.*

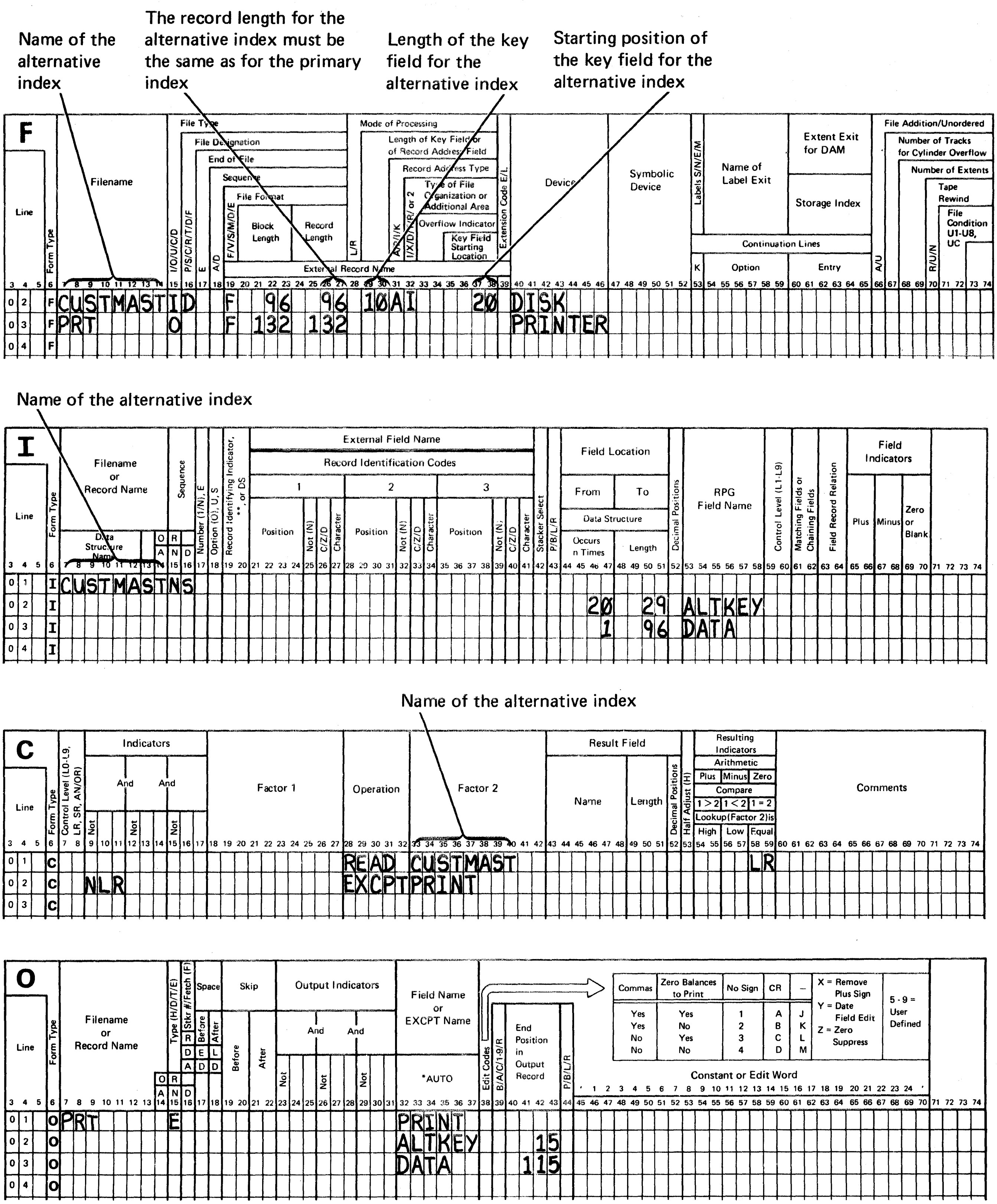
Example of Creating an Alternative Index File

Suppose that you want to create an alternative index for the file named MASTER that was created in Figure 5-14. To do so, you use the BLDINDEX procedure. For example, you might enter the following values:

```
                         BLDINDEX PROCEDURE                 Optional-*

               Creates an alternative index for a physical file

Name of file to be created  . . . . . . . . . . . . . . . .  CUSTMAST
Starting position for first field of key  . . . . . . . 1-4096  20
Length of first field . . . . . . . . . . . . . . . . . 1-120   10
Starting position for second field of key . . . . . . . 1-4096         *
Length of second field . . . . . . . . . . . . . . . .  1-120          *
Starting position for third field of key . . . . . . .  1-4096         *
Length of third field  . . . . . . . . . . . . . . . .  1-120          *
Name of physical file . . . . . . . . . . . . . . . . . .   MASTER
Creation date of physical file  . . . . . . . . . . . . .              *
Allow duplicate keys  . . . . . . . . . . . . DUPKEY,NODUPKEY  DUPKEY
Preferred disk location . . . . . . .  A1,A2,A3,A4,block number        *




Cmd3-Previous menu      Cmd4-Put on job queue             (c) 1985 IBM Corp.
```

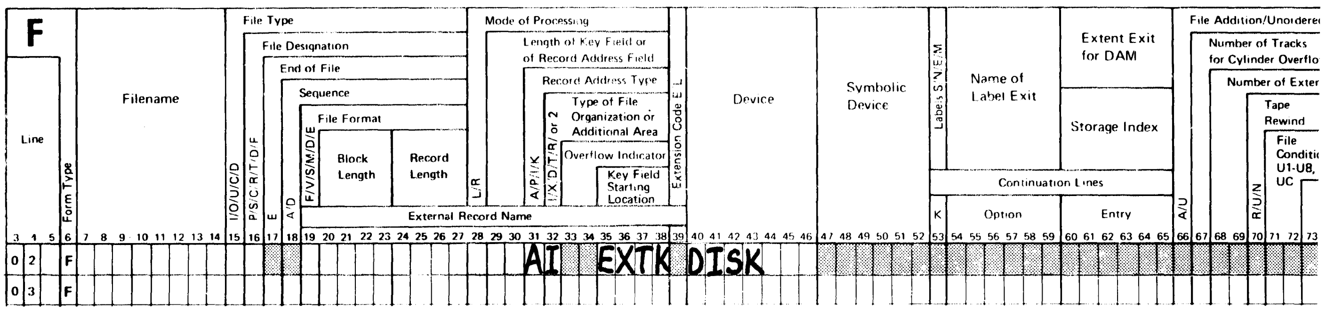
Example of Using an Alternative Index File with Only One Field as its Key

The following example shows how to use the alternative index named CUSTMAST, which we just created, to read records.



Using an Alternative Index File with Noncontiguous Fields as its Key

To use an alternative file index with noncontiguous fields as its key, describe the file in the unshaded columns of the file description specifications shown below:



Columns 7 through 14 must contain the name of the alternative index file with noncontiguous fields as its key.

Column 15 must contain I or U. I indicates that the file is an input file. U indicates that the file is an update file.

Column 16 must contain P, S or F. P indicates that the file is a primary file. S indicates that the file is a secondary file. F indicates that the file is a full-procedural file.

Columns 20 through 23 must contain the block length or blanks.

Columns 24 through 27 must contain the the record length. The record length for the alternative index must be the same as for the primary index.

Column 28 must contain blank, R or L.

Columns 29 and 30 must contain the total length of the key. The maximum total length a key can be is 99 positions.

Column 31 must contain A to indicate that the key is in zoned-decimal or alphameric format (packed-decimal format is not allowed).

Column 32 must contain I to indicate that the file is an indexed file.

Columns 35 through 38 must contain EXTK, to indicate that the file has noncontiguous fields as its key.

Columns 40 through 46 must contain DISK.

Reading an Indexed File

Records can be read from an indexed file in the following ways:

- Sequentially by key field
- Sequentially within key-field limits
- Randomly by key field
- Randomly and/or sequentially by key field
- Randomly by address output (addrout) file

Note: An indexed file can also be read without using the index. When this is done, only the data portion of the file is used. There are three ways to read an indexed file without using the index:

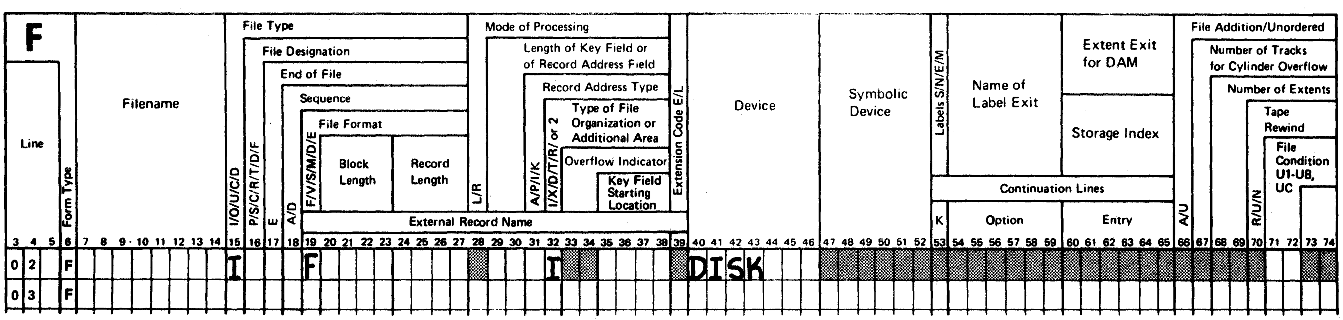
- Consecutively
- Randomly by relative record number
- Randomly by relative record number and/or consecutively

To read an indexed file in any of these ways without using the index, code the file as a sequential file. For more information about coding a sequential file, see *Sequential Files* earlier in this chapter.

Reading Sequentially by Key Field

When a program reads an indexed file sequentially by key field, it reads the records in the order in which the key fields are sequenced, not in the order in which the records exist in the file.

If you want to read records sequentially by key field, code entries in the file description specifications shown below:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain I to indicate that the file is an input file.

Column 16 can contain P, S, or D:

- If you code P (primary) or S (secondary), the file is read as part of the RPG program cycle. For an explanation of how primary and secondary files are read, see Chapter 11.

- If you code D (demand), you must code a READ operation in the calculation specifications in order to read the file. For information about the READ operation, see Chapter 28.

Column 17 can contain E or blank if column 16 contains P or S. E indicates that the program must process every record in the file before the program ends. Blank indicates that the program can end whether or not every record in the file is processed.

Column 18 can contain A, D, or blank if column 16 contains P or S:

- A indicates that the program checks that the records in the file are in ascending sequence.

- D indicates that the program checks that the records are in descending sequence.

- Blank indicates that the program does not check the record sequence.

Column 19 must contain F or blank.

Columns 20 through 23 must contain the block length or blanks.

Columns 24 through 27 must contain the record length.

Columns 29 and 30 must contain the total length of the key.

Column 31 must contain A or P. A indicates that the key fields are in zoned-decimal or alphameric format. P indicates that the key fields are in packed-decimal format.

Column 32 must contain I to indicate that this is an indexed file.

Columns 35 through 38 must contain the record position in which the key field begins, if the Key has only one field. If the Key has noncontiguous fields, columns 35 through 38 must contain EXTK.

Columns 40 through 43 must contain DISK.

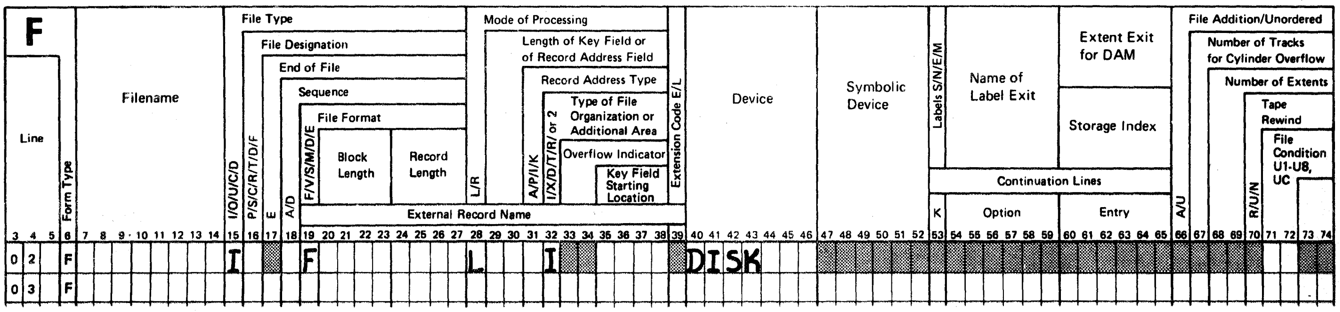
Columns 71 and 72 can contain an external indicator, U1 through U8.

Reading Sequentially within Key-Field Limits

A program can process an indexed file sequentially within key-field limits in either of two ways:

- Using a limits record
- Using the SETLL operation

Both methods allow you to limit the key fields of the records you want your program to process. If you want to read records sequentially within key-field limits, code entries in the unshaded columns of the file description specifications shown below:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain I to indicate that this file is an input file.

Column 16 must contain P, S, F, or D:

- If you code P (primary) or S (secondary), the program reads a record from the file at input time of the RPG program cycle. For an explanation of how primary and secondary files are read, see Chapter 11.

- If you code F (full-procedural), the program reads a record when a READ, READE, READP, or CHAIN operation occurs in the calculation specifications. Only the READ operation code may be used if the file is to be processed within limits using a limits record. When the file is to be processed within limits using the SETLL operation code, the SETLL must be immediately followed by a READ, READE, or READP operation. When a CHAIN operation occurs, the limits set by the SETLL operation are set off. For information about the READ, READE, READP and CHAIN operations, see Chapter 28.

- If you code D (demand), the program reads a record when a READ operation occurs in the calculation specifications. For information about the READ operation, see Chapter 28.

Column 18 can contain A, D, or blank if column 16 contains P or S:

- A indicates that the program checks that the records in the file are in ascending sequence.
- D indicates that the program checks that the records are in descending sequence.
- Blank indicates that the program does not check the record sequence.

Column 19 must contain F or blank.

Columns 20 through 23 must contain the block length or blanks.

Columns 24 through 27 must contain the record length.

Column 28 must contain L to indicate that the file is processed sequentially within limits.

Columns 29 and 30 must contain the total length of the key.

Column 31 must contain A or P. A indicates that the key fields are in zoned-decimal or alphameric format. P indicates that the key fields are in packed-decimal format.

Column 32 must contain I to indicate that this is an indexed file.

Columns 35 through 38 must contain the record position in which the key field begins, if the key has only one field. If the key has noncontiguous fields, columns 35 through 38 must contain EXTK.

Columns 40 through 43 must contain DISK.

Columns 71 and 72 can contain an external indicator, U1 through U8.

Using a Limits Record

When a program processes an indexed file sequentially within key-field limits by using a limits record, it first automatically reads a limits record from a record address file assigned to a DISK or CONSOLE device. A limits record contains the lowest key field and the highest key field of the records in the indexed file to be processed. The program reads records from the indexed file in ascending order of key field, from the low key field in the limits record to the high key field in the limits record. When the upper limit is passed, the program automatically reads another limits record from the record address file and sets new limits. This process continues until the program reaches the end of the record address file or until the program ends because another file reaches the end-of-file condition.

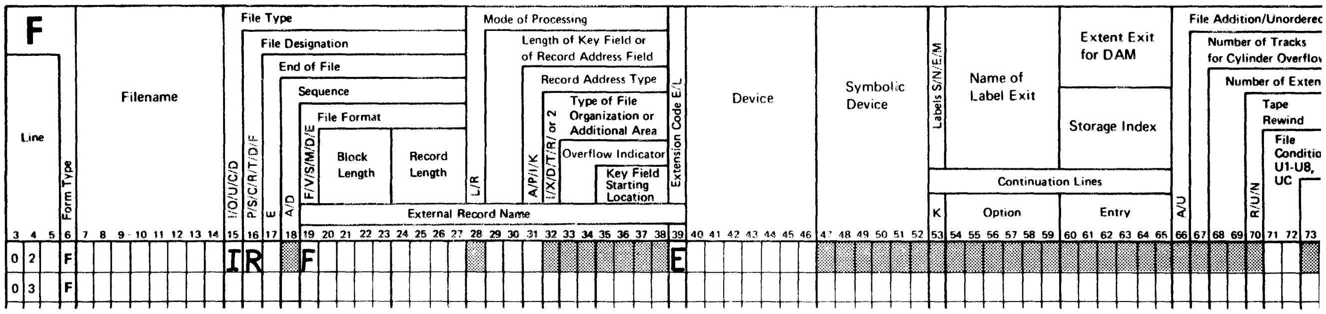
Each limits record contains only one set of limits (the low key field and the high key field). The low key field must begin in position 1 of the record, and the high key field must immediately follow the low key field. Therefore, the length of a limits record is twice the length of a key field. The length of a key field can be from 1 through 99 positions for a file processed using a DISK device, and 1 through 29 positions for a file processed using a console device. The low key field and the high key field must have the same length, and that length must be the same as the entry coded in columns 29 and 30 of the file description specifications. Therefore, you may have to code leading zeros when you use numeric key fields. An alphameric key field can contain blanks. A key field cannot contain any hexadecimal FF characters. For files with noncontiguous keys, the low and high key fields in the limits record contain all subfields that make up each noncontiguous key.

You can use the same set of limits in more than one limits record. Therefore, you can process data records within those limits as many times as you want. If the two key fields in a limits record are equal, the program reads only one data record.

The key fields in the limits records can have a different format from the key fields in the files being processed by limits. For example, one can have a packed-decimal format, and the other can have a zoned-decimal format. If the formats differ, the format of the key fields from each file must be indicated by A or P in column 31 of the file description specifications, and the length of the zoned-decimal key field must be twice the length of the packed-decimal key field, minus one or two. See *Column 43 (Packed or Binary Field)* in Chapter 25 for information about this calculation. While the program is running, the format of the key fields in the limits records is changed to the format of the key fields in the files being processed by limits.

To use the record address file from which the limits are read, entries are required in the file description and extension specifications. No input specifications are required for the record address file.

Code entries in the unshaded columns of the file description specifications shown below:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain I to indicate that the file is an input file.

Column 16 must contain R to indicate that the file is a record address file.

Column 17 must contain E or blank. E indicates that all records from the file must be processed before the program can end. Blank indicates that the program can end whether or not all records from the file are processed.

Column 19 must contain F or blank.

Columns 20 through 23 must contain the block length or blanks.

Columns 24 through 27 must contain the record length, which should be two times the length of the key field.

Columns 29 and 30 must contain the length of the key field.

Column 31 must contain A, P, or blank. A or blank means that the key field is in zoned-decimal or alphameric format. P means that the key field is in packed-decimal format. Column 31 must contain A or blank if columns 40 through 46 contain CONSOLE.

Column 39 must contain E to indicate that the file is further described on extension specifications.

Columns 40 through 46 must contain DISK or CONSOLE.

Columns 71 and 72 can contain an external indicator, U1 through U8.

Two entries are required on the extension specifications:

Line (3-5)	Form Type (6)	Record Sequence of the Chaining File (7-8)	Number of the Chaining Field (9-10)	From Filename (11-18)	To Filename (19-26)	Table or Array Name (27-32)	Number of Entries Per Record (33-35)	Number of Entries Per Table or Array (36-39)	Length of Entry (40-42)	P/B/L/R (43)	Decimal Positions (44)	Sequence (A/D) (45)	Table or Array Name (Alternating Format) (46-51)	Length of Entry (52-54)	P/B/L/R (55)	Decimal Positions (56)	Sequence (A/D) (57)	Comments (58-74)
01	E																	
02	E																	

Columns 11 through 18 must contain the name of the limits file. This name must be the same one coded on the file description specifications for the limits file.

Columns 19 through 26 must contain the name of the indexed file to be processed by the limits file. This name must be the same one coded on the file description specifications for the indexed file.

Using the SETLL Operation

The SETLL (set lower limit) operation allows you to set the lower limit for the key field during the calculations part of the RPG program cycle. You can use the SETLL operation to process any indexed file that is used as a demand or full-procedural file (that is, any file that has D or F in column 16, L in column 28, and I in column 32 of the file description specifications). However, you cannot process an indexed demand or full-procedural file with the SETLL operation if you are using a record address file to set the key-field limits for the file. The maximum number of files that you can process with the SETLL operation is limited by the number of demand and full-procedural files that an RPG program can use. The number of demand, chained, and full-procedural files that an RPG program can use cannot total more than 15.

Factor 1 of the SETLL operation must contain either the name of a field that contains the lower limit being set or a literal that is used as the lower limit. The name of the field containing the lower limit may be the name of a data structure subfield. Noncontiguous keys may be created to specify the lower limit by using alternative indexes and data structures to process the file. See *Creating an Alternative Indexed File for an Indexed File* in this chapter.

Factor 2 must contain the name of the file for which the lower limit is being set.

The SETLL operation must be followed by a READ, READE, or READP operation. Otherwise, the SETLL operation is ignored. The READE operation reads the next record if factor 1 of READE matches that record's key. The READP operation reads the record prior to the lower limit specified. Other operations (except for input operations) can come between the SETLL and READ, READE, or READP operations.

Figure 5-15 shows an example of SETLL coding for the SMASTER file and an example of using a limits record to process the MASTER file.

```
  02 FLIMITS  IR  F  16  16  8        ECONSOLE
  03 FMASTER  IP  F 256  64L 8AI     1 DISK
  04 FPRINT   O   F 132 132     OF     PRINTER
  05 FSMASTER ID  F 256  64L 8AI     1 DISK
  06 F
```

The input DISK file, MASTER, is an indexed file to be processed within the key-field limits contained in the record address file, LIMITS. The LIMITS file, which if further described on the extension specifications, is entered from the CONSOLE device.

Each set of limits read from LIMITS consists of the low and high account numbers to be processed. Because the account number key field (ACCT) is 8 positions long, each set of limits includes two 8-position key fields.

```
  01 E    LIMITS  MASTER
  02 E
```

As MASTER is processed within to each set of limits, the corresponding records are written to the PRINTER output file, PRINT. Processing is complete when all sets of limits have been processed.

Figure 5-15 (Part 1 of 2). Processing an Indexed File Sequentially within Key-Field Limits by Using the SETLL Operation for SMASTER and by Using a Limits Record for MASTER

Line	Form Type	Filename or Record Name	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, ** , or DS	Field Location From	Field Location To	Decimal Positions	RPG Field Name
01	I	MASTER	NS			Ø1				
02	I						1	8		ACCT
03	I						9	64		NAMADR
04	I	SMASTER	NS			Ø2				
05	I						1	8		ACTT
06	I						9	64		NAME
07	I									

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Resulting Indicators High (Plus, 1>2)	Low (Minus, 1<2)	Equal (Zero, 1=2)
01	C	LR		START	TAG						
02	C	LR	N99	'AAAAAAAA'	SETLL	SMASTER					
03	C	LR	N99		SETON				99		
04	C	LR			READ	SMASTER					98
05	C	LR	N98		GOTO	START					
06	C										

File SMASTER is processed by the SETLL operation code. It uses no extension specifications, and its filename appears in factor 2 of the SETLL operation code. In this example the first record read from file SMASTER would be the one whose key field is equal to or the next higher than the literal 'AAAAAAAA'. Records are read sequentially to end of file unless the cycle is interrupted by additional SETLL operations.

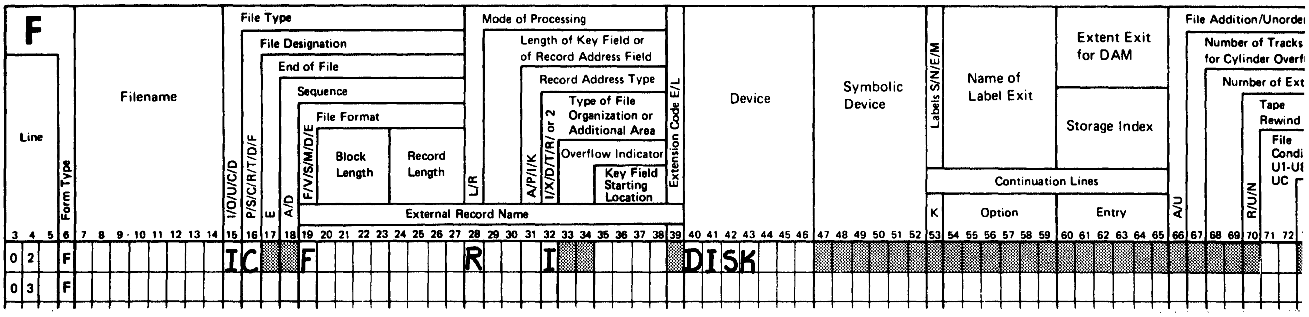
Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Space Before	Space After	Skip Before	Skip After	Output Indicators	Field Name or EXCPT Name	End Position in Output Record	Constant or Edit Word
01	O	PRINT	D		1			Ø1			
02	O								ACCT	8	
03	O								NAMADR	7Ø	
04	O		T		1			LR			
05	O									16	'SMASTER NO. ='
06	O								ACTT	25	
07	O								NAME	9Ø	
08	O										

Figure 5-15 (Part 2 of 2). Processing an Indexed File Sequentially within Key-Field Limits by Using the SETLL Operation for SMASTER and by Using a Limits Record for MASTER

Reading Randomly by Key Field

You can process an indexed file randomly by key field only if it is a chained file (that is, if it has C in column 16 of the file description specifications). You must use the CHAIN operation to read a record from the file during the calculation part of the RPG program cycle.

If you want to read an indexed file randomly by key field, code entries in the unshaded columns of the file description specifications shown below:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain I to indicate that the file is an input file.

Column 16 must contain C to indicate that the file is a chained file. Note that column 16 must not contain a C if the key has noncontiguous fields.

Column 19 must contain F or blank.

Columns 20 through 23 must contain the block length or blanks.

Columns 24 through 27 must contain the record length.

Column 28 must contain R to indicate that the file is processed randomly.

Columns 29 and 30 must contain the total length of the key.

Column 31 must contain A or P. A indicates that the key field is in zoned-decimal or alphameric format. P indicates that the key field is in packed-decimal format.

Column 32 must contain I to indicate that the file is an indexed file.

Columns 35 through 38 must contain the record position in which the key field begins, if there is only one field in the key. If the key has noncontiguous fields, columns 35 through 38 must contain EXTK.

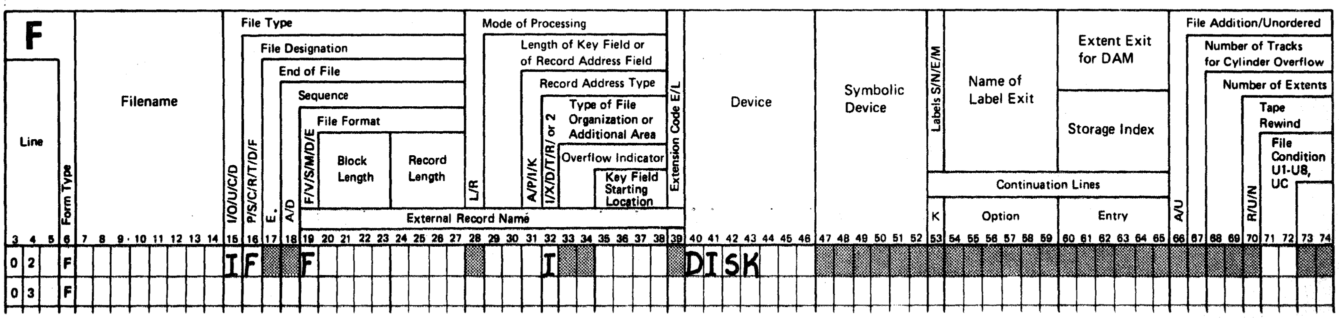
Columns 40 through 43 must contain DISK.

Columns 71 and 72 can contain an external indicator, U1 through U8.

Reading Randomly and/or Sequentially by Key Field

If you want to read records both randomly by key field and sequentially by key field, use a full-procedural file (F in column 16 of the file description specifications). You can read a full-procedural file randomly by key field by using the CHAIN operation, sequentially by key field by using the READ, READE, or READP operation, or both randomly by key field and sequentially by key field by using either the CHAIN and READ, READE, or READP operations. It is not necessary to use both a CHAIN and either a READ, READE or READP operation, but you must code at least one CHAIN, READ, READE, or READP operation in the calculation specifications in order to read a full-procedural file.

Code the unshaded columns of the file description specifications shown below:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain I to indicate that the file is an input file.

Column 16 must contain F to indicate that the file is a full-procedural file.

Column 19 must contain F or blanks.

Columns 20 through 23 must contain the block length or blanks.

Columns 24 through 27 must contain the record length.

Columns 29 and 30 must contain the total length of the key.

Column 31 must contain A or P. A indicates that the key field is in zoned-decimal or alphameric format. P indicates that the key field is in packed-decimal format.

Column 32 must contain I to indicate that the file is an indexed file.

Columns 35 through 38 must contain the record position in which the key field begins, if the key has only one field. If the key has noncontiguous fields, columns 35 through 38 must contain EXTK.

Columns 40 through 43 must contain DISK.

Columns 71 and 72 can contain an external indicator, U1 through U8.

Example of Reading an Indexed File Randomly and Sequentially by Key Field

Suppose you want to print a list of all the employees in a particular department. Figure 5-16 shows how to do so. The department number is entered in the first 5 positions of the local data area. The program uses this department number as the key field. The CHAIN operation reads the first record that has the desired department number in the DEPTNO field of the EMPLOYEE file. The READ operation then reads sequentially all the other records that have the same department number.

F

Line	Form Type	Filename	File Type	File Designation	End of File	Sequence	File Format	Block Length	Record Length	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
0 2	F	EMPLOYEE	I	F			F	256	256	5	A	I		1		DISK
0 3	F	LIST	O				F	132	132				OA			PRINTER
0 4	F															

I

Line	Form Type	Filename or Record Name	Sequence	Option (O), U, S	Record Identifying Indicator, or DS	Field Location From	Field Location To	RPG Field Name
0 1	I	EMPLOYEE	NS					
0 2	I					1	5	DEPTNO
0 3	I					20	50	EMPNAM
0 4	I							
0 5	I							
0 6	I							
0 7	I			U	DS			
0 8	I					1	5	DNUM
0 9	I							

The number of the department to be printed is in positions 1 through 5 of the local data area.

C

Line	Form Type	Indicators	Factor 1	Operation	Factor 2	Resulting Indicators (High)
0 1	C					
0 2	C					
0 3	C					
0 4	C					
0 5	C					
0 6	C		DNUM	CHAIN	EMPLOYEE	99
0 7	C	99		EXCPT	BADNUM	
0 8	C	99		GOTO	END	
0 9	C					

The CHAIN operation reads the first record that has the requested department number. If the department number is not found, an error message is printed.

Figure 5-16 (Part 1 of 2). Reading an Indexed File Randomly by Key Field and Sequentially by Key Field

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust	Resulting Indicators High/Plus	Low/Minus	Equal/Zero	Comments
01	C													
02	C		Now, reading sequentially, we can read the records for the rest of the employees in the department.											
03	C													
04	C													
05	C			LOOP	TAG									
06	C				EXCPT	PRTEMP								
07	C				READ	EMPLOYEE							98	
08	C		98		GOTO	END								IF EOF, DONE.
09	C			DNUM	COMP	DEPTNO							97	
10	C		97		GOTO	LOOP								IF DEPT NUMBERS
11	C	*												SAME, LOOP.
12	C			END	TAG									
13	C				SETON						LR			
14	C													

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	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	LIST	D			2	Ø6		1P						
02	O	OR							0A						
03	O												2Ø		'DEPT NUMBER'
04	O												35		'EMPLOYEE NAME'
05	O		E	F		1				PRTEMP					
06	O									DEPTNO			17		
07	O									EMPNAM			54		
08	O		E			1				BADNUM					
09	O												3Ø		'* * * DEPARTMENT NUMBER '
10	O									DNUM			35		
11	O												51		'NOT FOUND * * *'
12	O														

Figure 5-16 (Part 2 of 2). Reading an Indexed File Randomly by Key Field and Sequentially by Key Field

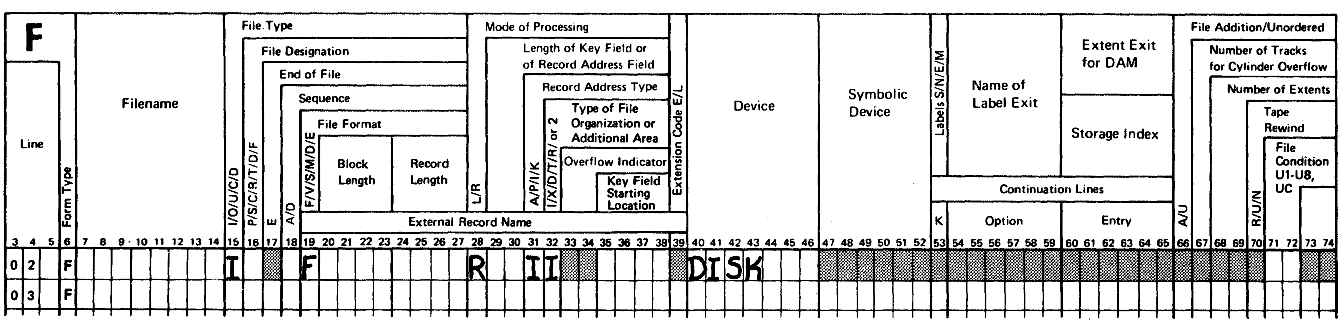
Reading Randomly by Address Output (Addrout) File

An *address output (addrout) file* is a record address file produced by a sort program. (A *record address file* is an input file that tells the program which records to read from a DISK file and the order in which to read them.) An address output file contains the relative record numbers of the records in a DISK file. The advantages of an address output file are that:

- The space required for the address output file is much less than the space required for a sorted sequential file.
- The sort runs much faster.
- The original file is unchanged.

You can have only one address output file in a program. When an RPG program uses an address output file, it reads the relative record numbers consecutively from the address output file. Then, using the relative record number, the program randomly reads the DISK file to process the corresponding record. In this way, the program can process an indexed DISK file in a new sequence without actually sorting the records and creating a new file. Also, once the file description and extension specifications are coded for the DISK file and for its associated address output file, you can code the DISK file as an ordinary indexed file. No input specifications are required for the address output file.

If you want to read records randomly by an address output file, code the file description specifications as shown below:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain I to indicate that the file is an input file.

Column 16 must contain P (primary) or S (secondary). For an explanation of how primary and secondary records are read, see Chapter 11.

Column 18 must contain A, D, or blank:

- A indicates that the program checks that the records in the file are in ascending sequence.
- D indicates that the program checks that the records in the file are in descending sequence.

- Blank indicates that the program does not check the sequence of the records in the file.

Column 19 must contain F or blank.

Columns 20 through 23 must contain the block length or blanks.

Columns 24 through 27 must contain the record length.

Column 28 must contain R to indicate that the file is processed randomly by an address output file.

Columns 29 and 30 must contain the total length of the key.

Column 31 must contain I to indicate that relative record numbers from the address output file are used to process the file.

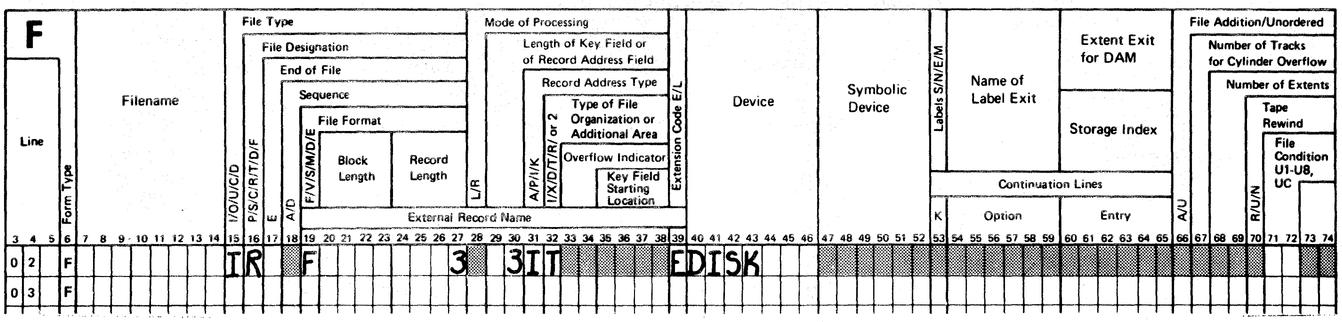
Column 32 must contain I to indicate that this file is an indexed file.

Columns 35 through 38 must contain the record position in which the key field begins, if there is only one field in the key. If the key has noncontiguous fields, columns 35 through 38 must contain EXTK.

Columns 40 through 43 must contain DISK.

Columns 71 and 72 can contain an external indicator, U1 through U8.

For the address output file, code entries in the unshaded columns of the file description specifications shown below:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain I to indicate that the file is an input file.

Column 16 must contain R to indicate that the file is a record address file.

Column 17 must contain E to indicate that all records from the file must be processed before the program can end, or a blank to indicate that the program can end whether or not all records from the file are processed.

Column 19 must contain F or blank.

Columns 20 through 23 must contain the block length or blanks.

Column 27 must contain 3 because each record in an address output file is a relative record number, which is always three positions long.

Column 30 must contain 3 because relative record numbers in address output files are always three positions long.

Column 31 must contain I to indicate that the file is an address output file.

Column 32 must contain T to indicate that the file is an address output file.

Column 39 must contain E to indicate that the file is further described on extension specifications.

Columns 40 through 43 must contain DISK.

Columns 71 and 72 can contain an external indicator, U1 through U8.

Two entries are required on the extension specifications:

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
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
0 1	E																	
0 2	E																	

Columns 11 through 18 must contain the name of the address output file. This name must be the same one coded on the file description specifications for the address output file.

Columns 19 through 26 must contain the name of the indexed file to be processed by the address output file. The name must be the same one coded on the file description specifications for the indexed file.

Updating an Indexed File

Updating records in an indexed file involves reading a record, changing some data in the record, and writing the record back to its original location in the file. If you try to update a record that was not the last record read, error message RPG-9043, FILE TRIED RECORD UPDATE BEFORE INPUT, is displayed. The fields to be updated must be described on both the input and the output specifications.

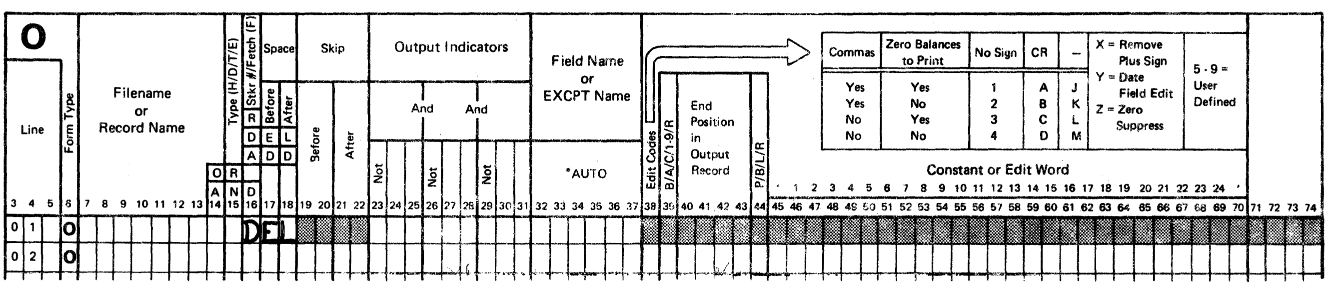
When you update records in an indexed file, the file can be processed in any of the following ways:

- Sequentially by key field
- Sequentially within key-field limits
- Randomly by key field
- Randomly and/or sequentially by key field
- Randomly by address output (addrout) file

Deleting Records from an Indexed File

Updating a file can include deleting records from the file. To allow records to be deleted from the file, the control language FILE statement coded when the file was created must use the DFILE-YES parameter. For information about the FILE statement, see the *System Reference* manual. If you try to delete a record from a file that does not allow deletions, error message RPG-9067, INVALID OPERATION ATTEMPTED, is displayed.

To delete a record, you first read the record (either randomly by key field or sequentially by key field). Then, with DEL coded in the output specifications, you write the record back to the same file. Code entries in the unshaded columns of the output specifications shown below:



Columns 7 through 14 must contain the name of the output file.

Column 15 must contain D, T, or E, to indicate that the record is to be deleted at detail, total, or exception output time of the program cycle.

Columns 16 through 18 must contain DEL to indicate that the record is to be deleted.

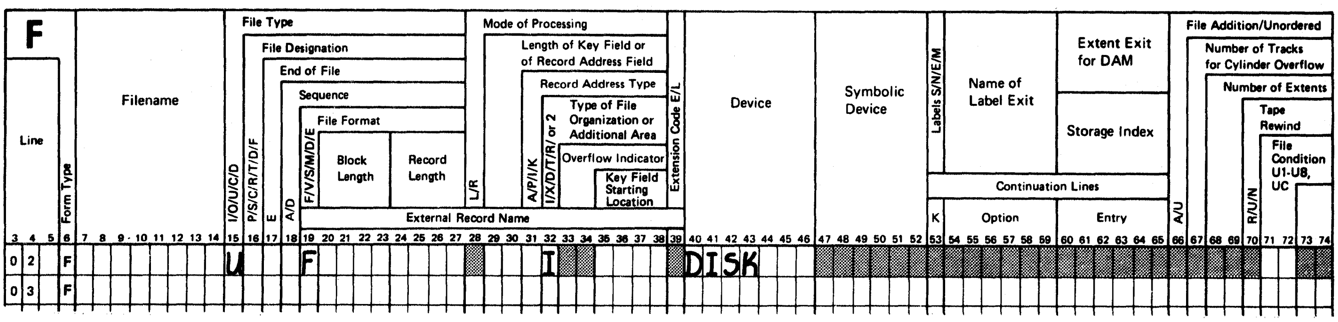
Columns 23 through 31 can contain output indicators.

Columns 32 through 37 can contain an EXCPT name if column 15 contains E.

Records are not physically removed from a file when they are deleted. Instead, deleted records are filled with hexadecimal FFs. That is, all the bits for every character in the deleted record are set on.

Updating Sequentially by Key Field

When you want to update most of the records in a file, process the file sequentially by key field. Code entries in the unshaded columns of the file description specifications shown below:

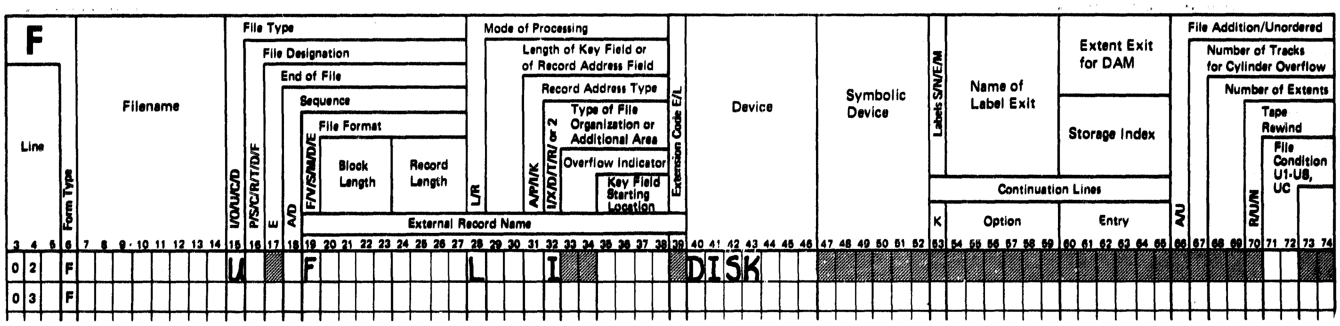


Column 15 must contain U to indicate that the file is an update file.

The other entries are the same as those for reading an indexed file sequentially by key.

Updating Sequentially within Key-Field Limits

You can update records in an indexed file sequentially within key-field limits. Code entries in the unshaded columns of the file description specifications shown below:

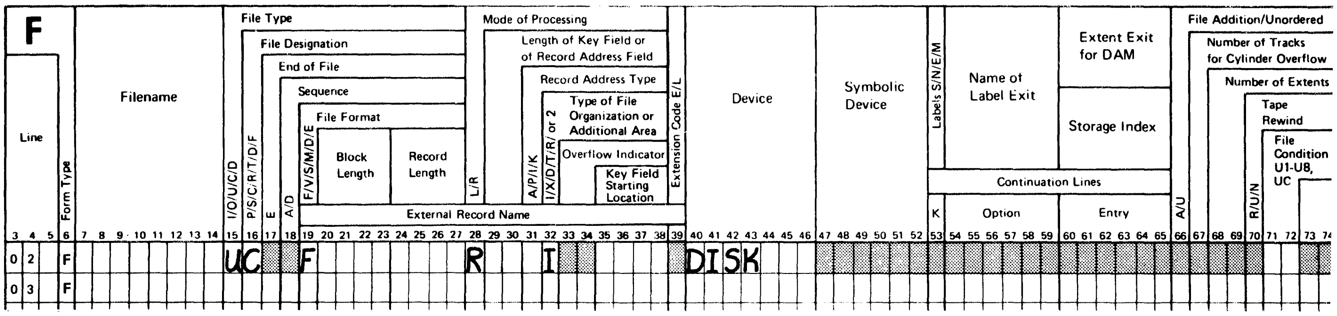


Column 15 must contain U to indicate that this file is an update file.

The other entries are the same as those for reading an indexed file sequentially within key-field limits.

Updating Randomly by Key Field

You can update an indexed file randomly by key field if it is a chained file (that is, if it has C in column 16 of the file description specifications). Code entries in the unshaded columns of the file description specifications shown below:



Column 15 must contain U to indicate that the file is an update file.

The other entries are the same as those for reading an indexed file randomly by key field.

Example of Updating an Indexed File Randomly by Key Field

Figure 5-17 shows sample program OE400R, which updates an indexed file randomly by key. The program reads the customer number (CUSNO) and the amount due for a new order (EXAMT) from the customer order file, CUSTORDS. In line 05 of the calculation specifications, the amount due for each item ordered (EXAMT) is added, and the total for the current order is stored in result field TOTDUE. In line 12, the program uses the customer number (CUSNO) to chain to the corresponding record in the customer master file, CUSTMAST. In line 13, the total amount due for the current order (TOTDUE) is added to the customer's previous amount due (AMDUE), and the result is stored again in AMDUE. On the output specifications, when the program writes to the customer master file (CUSTMAST), the AMDUE field is updated for the customer record. UDATE is used to update the field that indicates the most recent date that the record was updated.

```
02 FCUSTORDSIP  F 256 128            DISK
03 FCUSTMASTUC  F 256 256R 8AI     2 DISK
```

Name = OE400R

Function = Update the amount-due and the date-of-order fields in the customer master file (CUSTMAST) from the customer order file (CUSTORDS).

Input = From the CUSTORDS file, the customer number and the amount due for each item ordered. From the CUSTMAST file, the customer's current amount due.

Output = In the CUSTMAST file, the record for a customer who placed the order. The amount-due field is updated to the current amount due. The date that the record was updated is also updated.

Indicator Definitions:

- 01 Record-identifying indicator
- 02 Record-identifying indicator
- 03 Record-identifying indicator
- 20 Turns on when a customer number is not found in the CUSTMAST file
- L1 Control-level indicator that turns on when the customer number changes in the CUSTORDS file

Running This Program

To run this program, file CUSTORDS must already have been created by program OE140R. Also code the following procedure:

```
// LOAD OE400R
// FILE NAME-CUSTMAST
// FILE NAME-CUSTORDS
// RUN
```

Figure 5-17 (Part 1 of 2). Sample Program OE400R (Updating an Indexed File Randomly by Key Field)

```
I  Line  Form  Filename    Seq  Record  Record Identification Codes     Field Location   RPG Field  Control
           Type              Ident.  1          2          3            From    To  Dec  Name       Level
   01    I     CUSTORDS    NS   01      2 CC       3 CU
   02    I                                                                 4    11      CUSNO      L1
   03    I                 NS   02      2 CC       3 CS
   04    I                                                                 4    11      CUSNO      L1
   05    I                 NS   03      2 CI       3 CT
   06    I                                                                 4    11      CUSNO      L1
   07    I                                                                34    42  2   EXAMT
   08    I     CUSTMASTNS
   09    I                                                           P   131   135  2   AMDUE
   10    I
```

```
C  Line  Form  Control  Indicators  Factor 1  Operation  Factor 2   Result Field  Length  Dec  Resulting
           Type  Level                                                                           Indicators
   01    C
   02    C
   03    C
   04    C
   05    C              03                    ADD        EXAMT      TOTDUE        9       2
   06    C
   07    C
   08    C
   09    C
   10    C
   11    C
   12    C     L1                   CUSNO     CHAIN      CUSTMAST                                20
   13    C     L1       N20                   ADD        TOTDUE     AMDUE
   14    C     L1                             Z-ADD      *ZERO      TOTDUE
   15    C
```

When a line-item record is read from the CUSTORDS file, add the total amount due for this customer order.

When a control break occurs (the customer number changes), read the previous customer's record from the CUSTMAST file. Add the amount due from this order to the amount already due.

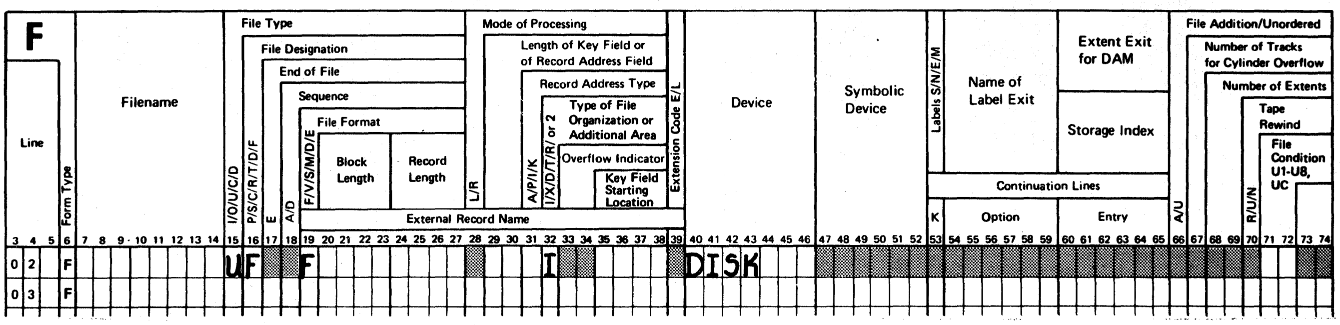
```
O  Line  Form  Filename    Type  Output Indicators  Field Name  End Position  P/B/L/R
           Type
   01    O
   02    O
   03    O
   04    O     CUSTMAST    T     L1N20
   05    O                                          AMDUE       135           P
   06    O                                          UDATE       188           P
   07    O                                          UDATE       192           P
   08    O
```

Update the customer record in the CUSTMAST file.

Figure 5-17 (Part 2 of 2). Sample Program OE400R (Updating an Indexed File Randomly by Key Field)

Updating Randomly and/or Sequentially by Key Field

You can update records in an indexed file randomly by key field, sequentially by key field, or both randomly and sequentially by key field if the file is defined as a full-procedural file (F in column 16). Code entries in the unshaded columns of the file description specifications shown below:

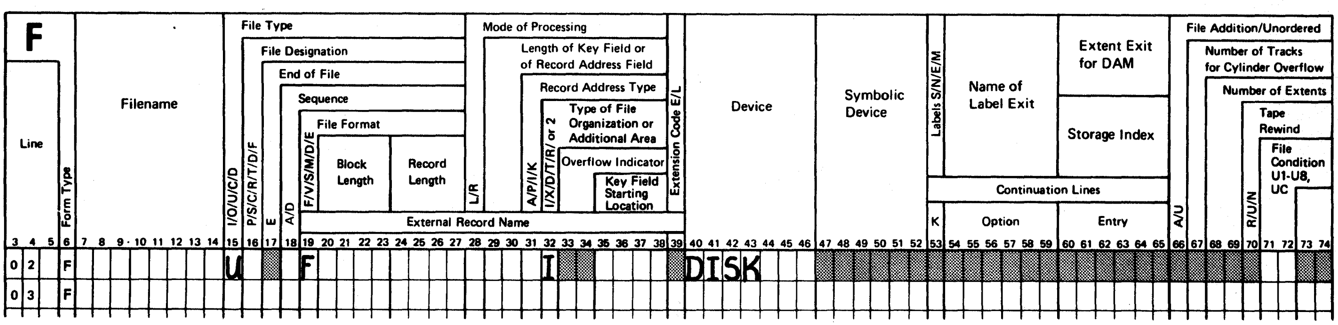


Column 15 must contain U to indicate that the file is an update file.

The other entries are the same as those for reading an indexed file randomly by key and/or sequentially by key field.

Updating Randomly by Address Output (Addrout) File

To update records randomly by an address output file, code entries in the unshaded columns of the file description specifications shown below:



Column 15 must contain U to indicate that the file is an input file.

The other entries are the same as those for reading an indexed file randomly by address output file.

Adding Records to an Indexed File

When a record is added to an indexed file, the program writes the *data part* of the record at the end of the records already in the file. The *index part* of the added record is written at the end of the indexes for the records already in the file.

You can add records to an indexed file in either of two ways:

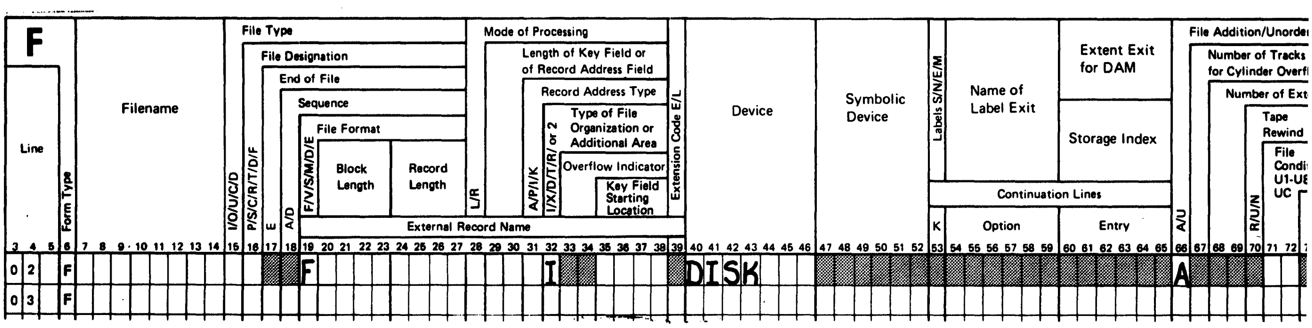
- Randomly by key field
- Sequentially by key field

Adding Records Randomly by Key Field

You can add records randomly by key field to an indexed file with chaining. *Chaining* means comparing the key field of the record to be added with the key fields already in the index. The reason for this comparison is to make sure that the record to be added is not a duplicate of a record already in the file. Chaining allows you to design your program so that, if a duplicate key field is found, your program can handle it appropriately without requiring the person using the display station to decide how to respond to an error message. If the program has a logic error that would allow a record with a duplicate key field to be added to the file, or if another program tries to add a record with a duplicate key field during the time between the CHAIN operation and the output operation in this program, the system ensures that the duplicate record is not added. However, a record with a duplicate key field can be added if the DUPKEYS-YES parameter was specified on the control language FILE statement when the file was created, or if the BYPASS-YES parameter is specified on the current FILE statement.

To add records randomly by key field, entries are required in the file description and output specifications.

Code entries in the unshaded columns of the file description specification shown below:



Columns 7 through 14 must contain the file name.

Column 15 must contain I or U to indicate that the file is an input or update file.

Column 16 must contain C or F to indicate that the file is a chained or full-procedural file. If column 16 contains C, you must code a CHAIN operation in the calculation specifications in order to read the file. If it contains F, you must code a READ, READE, READP, or CHAIN operation in the calculation specifications in order to read the file.

Column 19 must contain F or blank to indicate that every record in the file has the same length.

Columns 20 through 23 must contain the length of the block of records.

Columns 24 through 27 must contain the length of each record.

Column 28 must contain R to indicate that the file is processed randomly by key if this is a chained file (C in column 16), or a blank if this is a full-procedural file (F in column 16). If this is a full-procedural file, column 28 can contain L to indicate that the file is processed within key-field limits by using the SETLL operation.

Columns 29 and 30 must contain the total length of the key.

Column 31 must contain A or P. A indicates that the key field is in alphameric or zoned-decimal format. P indicates that the key field is in packed-decimal format.

Column 32 must contain I to indicate that the file is an indexed file.

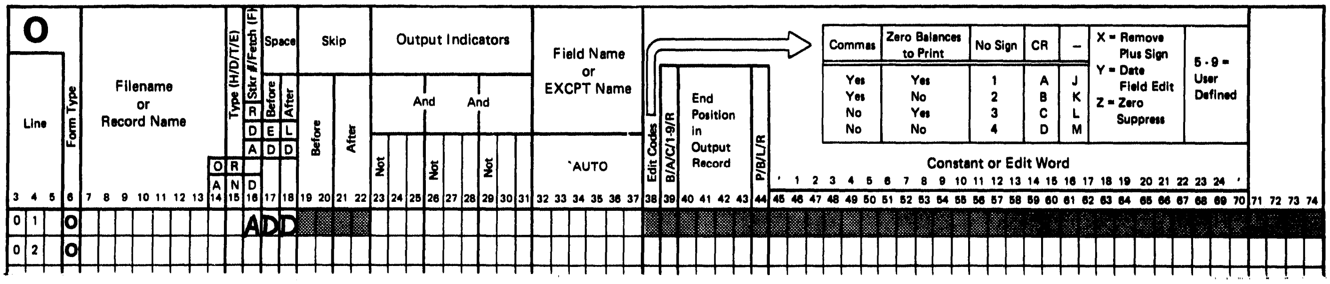
Columns 35 through 38 must contain the starting position of the key field, if the key has only one field. If the key has noncontiguous fields, columns 35 through 38 must contain EXTK.

Columns 40 through 43 must contain DISK.

Column 66 must contain A to indicate that records are added to the file.

Columns 71 and 72 can contain an external indicator, U1 through U8.

Entries are also required in the unshaded columns of the output specifications shown below:



Columns 7 through 14 must contain the name of the file to which the records are added.

Column 15 must contain D, T, or E to indicate whether the record is to be added at detail, total, or exception output time of the RPG program cycle.

Columns 16 through 18 must contain ADD to indicate that the fields coded on the following lines form the record to be added to the file named in columns 7 through 14.

Columns 23 through 31 can contain output indicators.

Columns 32 through 37 can contain an EXCPT name if column 15 of the output specifications contains E.

Example of Adding Records Randomly by Key Field

Suppose you want to add new inventory items to the indexed inventory file created in the *Example of Creating an Indexed File.* The new records are not in sequence. Key fields in the new records can be lower than, between, or higher than key fields in the records already in the file. Input and output records will be in the same format as the records used to create the file.

Figure 5-18 shows how to code the file description, input, calculation, and output specifications to all records added randomly by key field.

```
  02 FADDFILE IPE F  96  96            DISK
  03 FMASTER  UC  F 192  96RØ6AI     2 DISK                      A
  04 F
```

```
  01 IADDFILE AA  Ø1   1 CR
  02 I                                        2   7 ITEMNO
  03 I                                        1  96 RECORD
  04 IMASTER  BA
  05 I
```

```
  01 C   Ø1      ITEMNO    CHAINMASTER               99
  02 C
  03 C
  04 C
  05 C
  06 C
  07 C
  08 C
```

If 99 is on, the item is not in the file, so add the record. If 99 is off, the item has a duplicate in the file, so do not add the record.

```
  01 OMASTER  DADD     Ø1 99
  02 O                         RECORD    96
  03 O
  04 O
```

New records are added to the file.

Figure 5-18. Adding Records to an Indexed File Randomly by Key Field

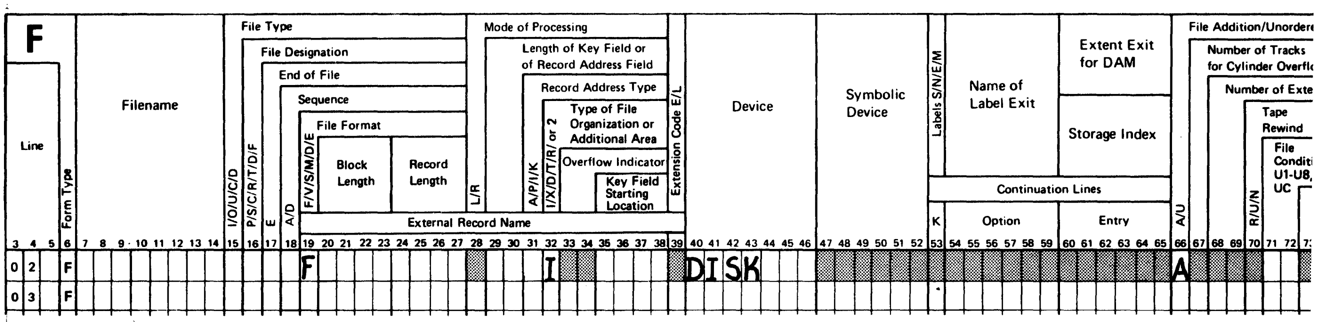
Adding Records Sequentially by Key Field

Physically, all records added to an indexed file are placed at the end of the file. However, depending on the value of the key field, a record added sequentially by key can be processed as if it were added in either of two places: between records already in the file or at the end of the file. A key field added between existing key fields must have a value that is lower than the key field in the record currently being processed and higher than the key field in the last record processed. A key field added at the end of the index must have a value that is higher than the highest key field in any record already in the file. If the key field of the record to be added does not meet either of these conditions, error message RPG-9037, TRIED TO ADD KEY NOT IN SEQUENCE, is displayed. If the person using the display station responds to that error message by entering option 0 (zero), the add operation is skipped.

Adding records sequentially can be faster than adding records randomly with chaining if the records to be added are already sorted into ascending order by key field. The reason is that you can use a large block containing many records when you add records sequentially.

To add records sequentially by key field, entries are required in the file description and output specifications.

Code entries in the unshaded columns of the file description specifications shown below:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain I or U to indicate that the file is an input or update file.

Column 16 can contain P, S, or D:

- If you code P (primary) or S (secondary), the file is read as part of the RPG program cycle. For an explanation of how primary and secondary files are read, see Chapter 11.
- If you code D (demand), you must code a READ operation in the calculation specifications in order to read the file. For information about the READ operation, see Chapter 28.

Column 17 can contain E if column 16 contains P or S. E indicates that the program must process every record in the file before the program ends. Blank indicates that the program can end whether or not every record in the file is processed.

Column 18 can contain A, D, or blank if column 16 contains P or S:

- A indicates that the program checks that the records in the file are in ascending sequence.
- D indicates that the program checks that the records are in descending sequence.
- Blank indicates that the program does not check the record sequence.

Column 19 must contain F or blank to indicate that all records in the file must be the same length.

Columns 20 through 23 must contain the block length or blanks.

Columns 24 through 27 must contain the record length.

Columns 29 and 30 must contain the total length of the key.

Column 31 must contain A or P. A indicates that the key field is in alphameric or zoned-decimal format. P indicates that the key field is in packed-decimal format.

Column 32 must contain I to indicate that the file is an indexed file.

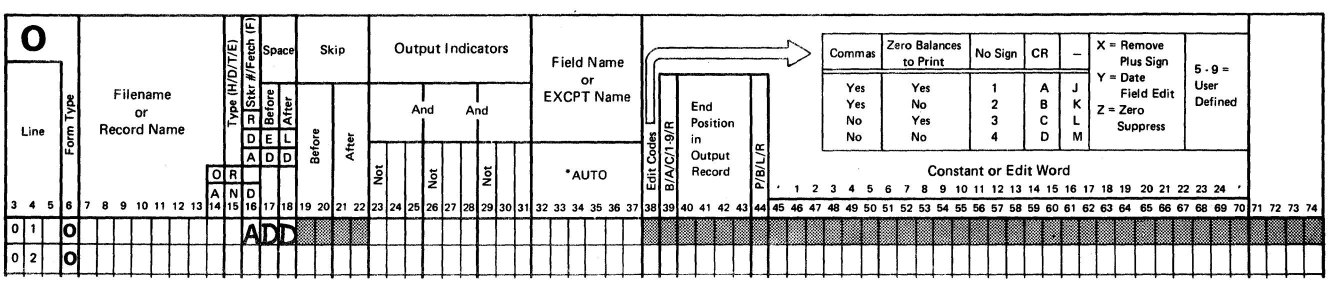
Columns 35 through 38 must contain the record position in which the key field begins, if the key has only one field. If the key has noncontiguous fields, columns 35 through 38 must contain EXTK.

Columns 40 through 43 must contain DISK.

Column 66 must contain A to indicate that records are added to the file described on this line.

Columns 71 and 72 can contain an external indicator, U1 through U8.

Entries are also required on the output specifications:



Columns 7 through 14 must contain the name of the file to which the records are added.

Column 15 must contain D, T, or E to indicate that the records are to be added at detail, total, or exception output time of the RPG program cycle.

Columns 16 through 18 must contain ADD to indicate that the fields coded on the following lines form the record to be added to the file named in columns 7 through 14.

Columns 23 through 31 can contain output indicators.

Columns 32 through 37 can contain an EXCPT name if column 15 contains E.

Example of Adding Records Sequentially by Key Field

Suppose you want to add new inventory items to the indexed inventory file created in *Example of Creating an Indexed File* earlier in this chapter. You want to add records from a transaction file that contains both new items and new shipments received of existing items. The transaction file is sorted into ascending sequence by key field (item number), and the records in the transaction file are in the same format as the records already in the inventory file.

Figure 5-19 shows how to code the specifications to update the inventory file and add new item records to the file.

Line	Form Type	Filename	File Type	File Designation	End of File	Sequence	File Format	Block Length	Record Length	L/R	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	Storage Index	File Addition/Unordered
02	F	TRANS	I	P	E	A	F	96	96								DISK					
03	F	MASTER	U	S		A	F	192	96		Ø6	A	I		2		DISK					A
04	F																					

Line	Form Type	Filename or Record Name	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, **, or DS	Position (1)	Not (N) (1)	C/Z/D (1)	Character (1)	Position (2)	Not (N) (2)	C/Z/D (2)	Character (2)	From	To	Decimal Positions	RPG Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields
01	I	TRANS	AA			Ø1	1		C	I										
02	I														2	7		ITEMNO		M1
03	I														1	96		RECORD		
04	I																			
05	I		AB			Ø2	1		C	R										
06	I														2	7		ITEMIN		M1
07	I														8	12	Ø	QTY		
08	I														13	18	2	COST		
09	I														19	24		DATE		
10	I		AC			Ø3	1	N	C	I	1	N	C	R						
11	I																			
12	I																			
13	I																			
14	I	MASTER	BA			Ø4	1		C	I										
15	I														2	7		ITEM		M1
16	I														23	28	2	UCOST		
17	I														37	41	Ø	ONHAND		
18	I														42	46	Ø	ONORD		
19	I														57	62		LDATE		
20	I																			

New Item Records

Receipts of Existing Inventory Items

Catch-all for Unidentified Records

Inventory Master File

Figure 5-19 (Part 1 of 2). Adding Records Sequentially by Key Field

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators			Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators	Comments
01	C		Ø4	MR			Z-ADD	COST	UCOST					Update inventory master record.
02	C		Ø4	MR			SUB	QTY	ONORD					
03	C		Ø4	MR			ADD	QTY	ONHAND					
04	C		Ø4	MR			MOVE	DATE	LDATE					
05	C													

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			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	MASTER	D	ADD					Ø1	NMR							Add new item to inventory master file.	
02	O											RECORD			96			
03	O		D						Ø4	MR							Update existing record in inventory master file.	
04	O											UCOST			28			
05	O											ONHAND			41			
06	O											ONORD			46			
07	O											LDATE			62			
08	O																	

Figure 5-19 (Part 2 of 2). Adding Records Sequentially by Key Field

Chapter 6. Using a WORKSTN File

Chapter 6. Using a WORKSTN File

A WORKSTN file allows you to interact with your RPG program at a display station. That is, while the program is running, information is displayed on the screen and you can enter data at the keyboard. Several people at different display stations can interact with the same program at the same time.

A program can use only one WORKSTN file. If a program uses a WORKSTN file, it cannot use a KEYBORD, CRT, or CONSOLE file.

You must use a WORKSTN file to communicate with other systems through the Interactive Communications Feature of the System Support Program (SSP-ICF). For more information about SSP-ICF, see the *Interactive Communications Feature: Reference* and the *Interactive Communications Feature: Guide and Examples.*

Note: Throughout this chapter, the term *device* means either a display station or an SSP-ICF session.

This chapter contains six sections, organized as follows:

- The first section contains a simple example of a program that uses a WORKSTN file.
- The second section explains the two steps in all programs that use a WORKSTN file (creating the displays and coding the RPG specifications).
- The third section presents some additional ways that programs commonly use a WORKSTN file.
- The fourth section covers the differences between programs that use only one display station and those that use more than one display station.
- The fifth section discusses some advanced topics relating to WORKSTN files.
- The sixth section contains several sample programs.

EXAMPLE OF USING A WORKSTN FILE

Suppose you want to create a program that allows a person to display accounts receivable information about your customers. The program displays the name and address of the customer, the current balance, the credit limit, the amount due more than 30, 60, and 90 days, and the date of last payment.

From the point of view of the person who uses this program, the program involves three steps:

1. Seeing a display that prompts the person to enter the customer number

2. Entering the customer number in response to the prompt

3. Seeing the accounts receivable information for the customer chosen

From your point of view, the program involves two basic steps:

1. Creating the two displays

2. Coding the specifications for the program

Creating the Displays

The first display, which prompts the person to enter a customer number, looks like the one shown below:

```
                          Customer Inquiry

Please enter customer number.

Customer number

Press the Enter key to see accounts receivable information.




Press Cmd3 to return to the main menu.
```

The second display, which shows the accounts receivable information for the customer chosen, looks like this:

```
                          Customer Inquiry


Customer number  ********

Customer name    *************************
Address          *************************
                 *************************
                 *************************
State and zip    ** *********


Current amount due        ************  Credit limit        ********
Amount due over 30 days   ************
Amount due over 60 days   ************  Last amount paid    ***********
Amount due over 90 days   ************  Last date paid      ********


Press the Enter key to continue.
```

For information on how to use SDA to create displays see the manual *Creating Displays.*

Coding the RPG Specifications

File Description Specifications

This program requires the following file description specifications:

```
  02 FCUSTNMBRCP  F     256            WORKSTN
  03 FCUSTMASTIC  F 256 256R 8AI     2 DISK
  04 F
```

The customer number file, CUSTNMBR, is a WORKSTN file (line 02, columns 40 through 46) and is described as a combined file (C in column 15). It is used as the primary file (P in column 16) because it is the main file from which the program reads records. The F in column 19 indicates that all records in the file have the same length. The 256 in columns 25 through 27 indicate the number of positions in each record.

The customer master file, CUSTMAST, is a DISK file (line 03, columns 40 through 43). It is used only for input (I in column 15), and it is a chained file (C in column 16). All records in the file contain 256 positions (column 19 and columns 25 through 27). CUSTMAST is an indexed file (I in column 32) that is processed randomly by key field (columns 28 and 31). The key field, CUSNO, is 8 positions long (column 30), beginning in position 2 (column 38).

Input Specifications

The following input specifications describe the two files that provide input:

Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator	Position	Not (N)	C/Z/D	Character	P/B/L/R	From	To	Decimal Positions	RPG Field Name
01	I	CUSTNMBR	NS	Ø2	1		C						
02	I		NS	Ø3	1		C	A					
03	I									2	9		CUSNO
04	I		NS	Ø4	1		C	B					
05	I	CUSTMAST	NS	1Ø									
06	I									2	9		CUSNO
07	I									1Ø	34		CUSNM
08	I									35	59		CUSA1
09	I									6Ø	84		CUSA2
10	I									85	1Ø9		CUSA3
11	I									11Ø	111		STATE
12	I								P	112	116	Ø	ZIPCD
13	I								P	131	135	2	AMDUE
14	I								P	124	127	Ø	CRLIM
15	I								P	155	159	2	OVR3Ø
16	I								P	16Ø	164	2	OVR6Ø
17	I								P	14Ø	144	2	LSTAP
18	I								P	165	169	2	OVR9Ø
19	I								P	136	139	Ø	DLTPM
20	I												

The WORKSTN file, CUSTNMBR, contains three types of records, identified by blank, A, and B in position 1 (lines 01, 02, and 04; columns 24 through 27). These record-IDs (blank, A, and B) are the IDs of user-created display screens. The three record types above turn on record-identifying indicators 02, 03, and 04, respectively (lines 01, 02, and 04; columns 19 and 20). The CUSNO field is in positions 2 through 9 of the record type identified by record-identifying indicator 03 (line 03).

The DISK input file, CUSTMAST, also has a field named CUSNO in positions 2 through 9 of each record (line 06). In addition, CUSTMAST contains the accounts receivable information indicated by the field names in lines 07 through 19. The P in column 43 of lines 12 through 19 indicates that the data in these fields is in packed-decimal format.

Calculation Specifications

This program requires only two calculation specifications:

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators Plus / >2 / High (54-55)	Minus / <2 / Low (56-57)	Zero / =2 / Equal (58-59)	Comments
01	C				SETOF						99			
02	C		Ø3	CUSNO	CHAIN	CUSTMAST					99			
03	C													

The first specification turns off error indicator 99 before the program reads a record from the customer master file, CUSTMAST.

If indicator 03 is on (line 02, columns 10 and 11), meaning that the program reads a record containing an A in position 1, the program uses the CHAIN operation to compare the number that the person entered on the first display with the CUSNO field in the records in the CUSTMAST file. If no record in the CUSTMAST file has a CUSNO field that matches the number entered, resulting indicator 99 turns on to signal an error (line 02, columns 54 and 55). Therefore, indicator 99 is also used on the output specifications to condition the error messages (lines 06 through 09, columns 24 and 25).

Output Specifications

The following output specifications describe the two displays:

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Output Indicators	Field Name or EXCPT Name	Edit Codes	End Position in Output Record	Constant or Edit Word
01	O	CUSTNMBR	D	Ø2NKC				
02	O	OR		99NKC				
03	O	OR		Ø4NKC				
04	O						K8	'AR23ØRD1'
05	O				CUSNO		8	
06	O			99				'CUSTOMER NUMBER DOES NOT'
07	O			99				' EXIST.'
08	O			99				'ENTER A DIFFERENT CUSTOM'
09	O			99				'ER NUMBER.'
10	O		D	Ø3N99NKC				
11	O						K8	'AR23ØRD2'
12	O				CUSNO		8	
13	O				CUSNM		33	
14	O				CUSA1		58	
15	O				CUSA2		83	
16	O				CUSA3		1Ø8	
17	O				STATE		11Ø	
18	O				ZIPCD		119	
19	O				AMDUE	L	131	'$'
20	O				CRLIM	3	139	'$'
21	O				OVR3Ø	L	151	'$'
22	O				OVR6Ø	L	163	'$'
23	O				LSTAP	3	174	'$'
24	O				OVR9Ø	L	186	'$'
25	O				DLTPM	Y	194	

*Number of sheets per pad may vary slightly.

In line 01, columns 7 through 14 show that the name of the output file is CUSTNMBR. The first display, AR230RD1 (line 04, columns 46 through 53), is a detail record (D in line 01, column 15) that is written if record-identifying indicator 02, 99, or 04 is on (lines 01 through 03, columns 24 and 25) and if command key 3 is not pressed (NKC, lines 01 through 03, columns 26 through 28).

If the person enters a customer number that equals the CUSNO field in a record in the CUSTMAST file, the program shows the second display, AR230RD2 (line 11, columns 46 through 53) at detail output time (line 10, column 15) if record-identifying indicator 03 is on, indicator 99 is not on, and command key 3 is not pressed (line 10, columns 24 through 31). The second display shows the accounts receivable information indicated by the fields in lines 12 through 25. The L edit code in column 38 of lines 19, 21, 22, and 24 displays a minus sign after any negative balance. The Y edit code in line 25 edits the date of last payment.

Reaching End of File

A program that uses a WORKSTN file, as this example does, can end in any of several ways:

- One way to end the program is simply to turn on the last-record indicator (LR).

- Another way is to have the person using the display station press a command key. In this example, command key 3 is used. Both display formats are conditioned so that they are not written after command key 3 is pressed. On the next input operation, the WORKSTN file reaches end of file.

  - If the WORKSTN file is a primary file, as in this example, the program automatically turns on the last-record indicator at input time of the next program cycle, and the program goes to end of job.
  - If the WORKSTN file is a demand file, and if an indicator is coded in columns 58 and 59 of the calculation specification containing the READ operation for the file, the indicator turns on.

For more information about reaching the end of a WORKSTN file, see *Reaching End of File for a MRT Program* later in this chapter.

This concludes the simple example of using a WORKSTN file. Although the example does not show how to change the information displayed, WORKSTN files do allow you to interact with the program. The other sample programs at the end of this chapter show how to use a WORKSTN file to update data.

The next section of this chapter explains all the entries you can code for a WORKSTN file.

STEPS IN USING A WORKSTN FILE

There are two general steps in using a WORKSTN file:

1. Create the format of each display from which your program will read input and to which your program will write output.

2. Code the necessary file description, input, calculation, and output specifications for your program.

The reason for creating the displays first is that the format of each display (that is, the location and characteristics of each field on the display) determines the coding required for the input and output specifications for the WORKSTN file.

Creating the Display Formats

Creating a display format includes designing the format, entering the specifications for the format, and compiling the specifications for the format. For a complete explanation of how to create display formats, see the manual *Creating Displays*.

There are two ways to create a display format:

- Use the screen design aid (SDA) utility to design the format and to enter and compile the specifications for the format. For information about using SDA, see the manual *Creating Displays*.

- Use the display format layout sheet and the display format S and D specifications to design the format. Use the $MAINT utility, the development support utility (DSU), if it is installed on the system, or the source entry utility (SEU) to enter the specifications for the format. Use the $SFGR utility to compile the specifications for the format. For information about the display format S and D specifications, the $MAINT utility, and the $SFGR utility, see the manual *Creating Displays*. For information about DSU, see the *Development Support Utility Guide*, and for information about SEU, see the *Source Entry Utility (SEU) Guide*.

SDA is the easier and recommended way to create a display format. SDA offers two advantages:

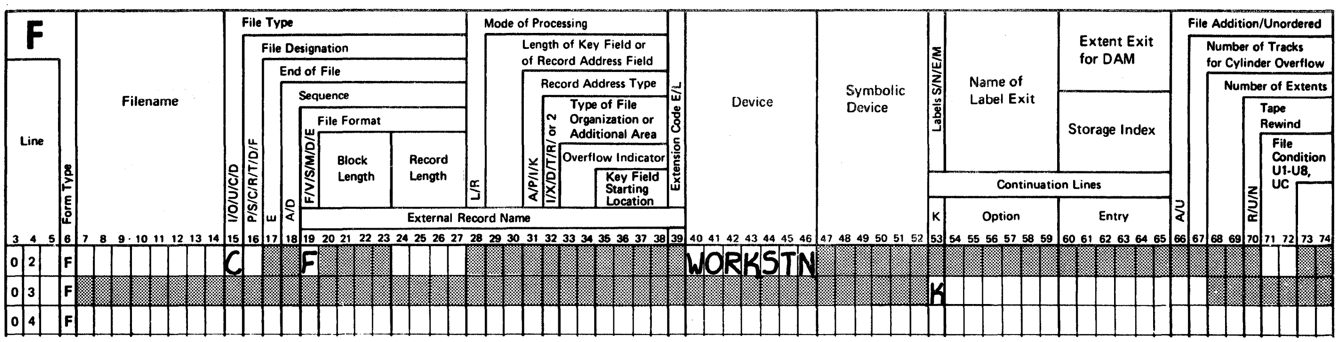
- It allows you to design formats right on the screen, so you do not need to fill out the display format S and D specifications.

- It allows you to choose an option that creates file description and input specifications for the WORKSTN file, so you do not need to code those RPG specifications for that file.

Coding the RPG Specifications

File Description Specifications

To use a WORKSTN file, code entries in the unshaded columns of the file description specifications shown below:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain C to indicate that the file is a combined (input and output) file.

Column 16 must contain P (primary) or D (demand) to indicate how the program uses the file:

- If column 16 contains P, the WORKSTN file is automatically read during input time of the RPG program cycle. Record-identifying indicators are automatically set off at input time of the program cycle. If the WORKSTN file is a primary file, no secondary files are allowed.

- If column 16 contains D, you must code a READ operation in the calculation specifications in order to read the file. Record-identifying indicators are not set off when the READ operation occurs, so it is possible to have more than one record-identifying indicator on if the WORKSTN file is a demand file.

Column 19 must contain F or blank to indicate that all the records in the file have the same length.

Columns 20 through 23 must be blank. The block length equals the record length.

Columns 24 through 27 must contain the length of the longest record. This number is equal to the highest end position coded on the input or output specifications. The maximum record length is 9999.

Columns 40 through 46 must contain the device name WORKSTN.

Columns 71 and 72 can contain an external indicator (U1 through U8).

Continuation-Line Options

The file description specifications can also include one or more continuation lines. Continuation lines are coded on the lines immediately following the file description specification for a WORKSTN file. Continuation lines are used to provide additional information about the file.

Column 53 must contain K to identify this as a continuation line.

Columns 54 through 58 must contain the continuation-line option. Valid entries for the continuation-line option are NUM, SAVDS, IND, SLN, FMTS, ID, INFSR, INFDS, and CFILE.

Columns 60 through 65 (columns 60 through 67 if the option is FMTS or CFILE) must contain the value for the continuation-line option.

Figure 6-1 shows sample values coded for each option.

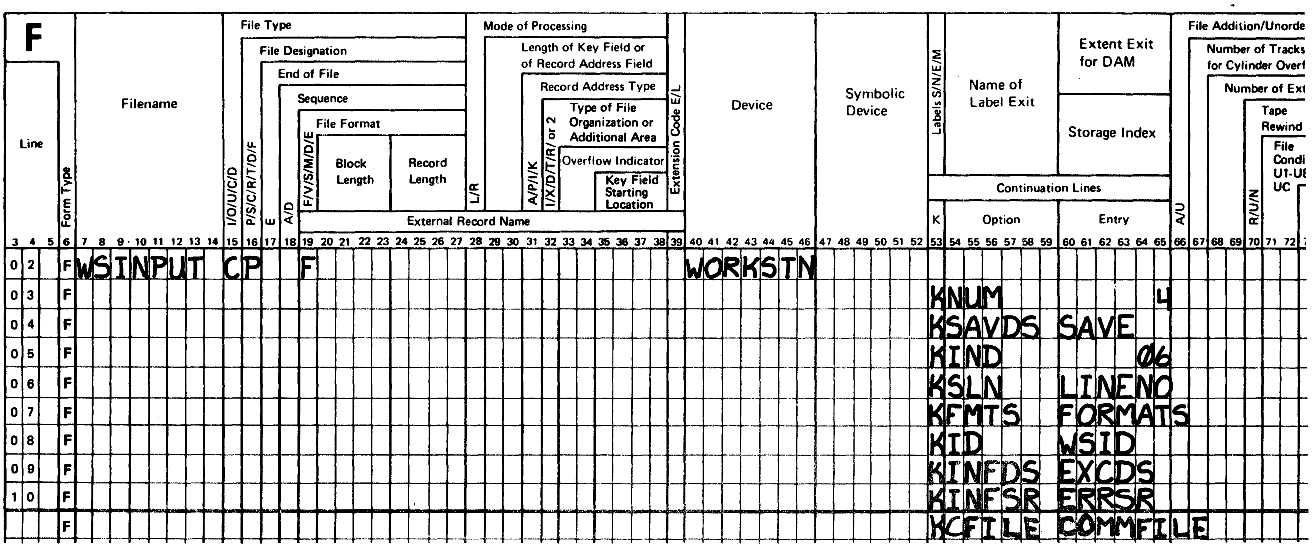


Figure 6-1. Continuation-Line Options for a WORKSTN File

Continuation-line options are explained in the following paragraphs.

NUM

The entry coded in columns 60 through 65 for the NUM option indicates the maximum number of devices that can use this file at the same time. If no number is coded, 1 is assumed. If a number is coded, it must be at least as large as the sum of the number of acquired devices plus the number you enter in response to the prompt *Maximum number of requesting display stations* on the RPGONL, RPGC, or AUTOC procedure.

Note: If the value for NUM is greater than 1, use caution when updating a file (see *Updating DISK Files in a MRT Program* later in this chapter).

SAVDS

The name coded in columns 60 through 65 for the SAVDS option identifies the data structure that is to be saved and restored for each device. This data structure allows you to save the contents of a field that is unique to each display station. Therefore, it allows you to save your place in the program while another requester is using the program. For example, it can contain a field that is used to accumulate the number of records read or to store a field that is not used until later cycles, such as a credit limit. The data structure cannot be a display station local data area, a compile-time array, or a preexecution-time array. If SAVDS is not coded, no data areas are swapped.

IND

The entry coded in columns 60 through 65 for the IND option indicates the number of indicators, beginning with 01, that are to be saved and restored by the display station. IND allows you to save the state of an indicator that is unique to the display station. Therefore, it allows you to save your place in the program while another requester is using the program. For example, you can use separate error indicators or security clearance indicators for each display station. If IND is not coded, no indicators are swapped.

Indicators may need to be reset in the program; they are not always reset by RPG in time to be useful to the programmer.

The following types of fields and indicators do not need to be placed in SAVDS and IND:

- Work fields that are used during one cycle (between input operations for the WORKSTN file), but can then be destroyed.

- Job fields that are used by all display stations but are not destroyed.

For SAVDS and IND, only one copy of the data structure and indicators is available at a time. The indicators and data structure that are available are those associated with the device from which the last input was read. The data structure and indicators that are available change each time the program does an input operation (either by the RPG program cycle for a primary file or by the READ operation code for a demand file). On an input operation, the program writes the present copy of the data structure and indicators in the program to a save area for the device from which the previous input was read. Then, for the device from which the current input is being read, the program writes the data structure and indicators from the save area associated with the device to the program SAVDS and IND areas. After the first input operation for each device, all the restored indicators will be off and all the fields in the SAVDS data structure will be blank.

SLN

The entry coded in columns 60 through 65 for the SLN option identifies a 2-digit numeric field whose value determines the line on the screen at which the display is to begin if a variable starting-line number was specified in SDA or in the display format S specifications. If a variable starting-line number is not specified, all displays having a variable starting-line number begin on line 01.

FMTS

The name coded in columns 60 through 67 for the FMTS option identifies the load member containing the display formats. If a name is not entered, the compiler assumes that the name of the load member containing the display formats is the same as the program name (from columns 75 through 80 of the control specification) with FM added to the end of the name. The constant *NONE coded in columns 60 through 64 indicates that the only formats in this program are SSP-ICF formats, or IDDU communication formats.

ID

The name coded in columns 60 through 65 for the ID option identifies a 2-character alphameric field that contains the identification of the device that supplied the record currently being processed in this file. This field does not have to be coded as an input or result field. The ID field is updated whenever a record is read from the WORKSTN file. Therefore, it always contains the identification of the device from which the last record was read (unless your program moves a different identification into the ID field). If the NUM option has a value of more than 1, you can direct output to various devices by changing the value in the ID field to the identification of another device in the file.

Display station identifiers are in the form AX, where A is an alphabetic character (A through Z, or one of the special characters #, @, or $) and X is any character. If a control language WORKSTN statement exists for the display station, the value of ID is the same as the value of the SYMID parameter on the WORKSTN statement.

SSP-ICF session identifiers can be in either of two formats: NN or NA, where N is numeric (0 through 9) and A is alphabetic (A through Z, #, @, or $). If the format is NA, a control language SESSION statement must be specified with a SYMID parameter whose value is also in the NA format.

INFSR

The name coded in columns 60 through 65 for the INFSR option identifies the WORKSTN exception/error-processing subroutine. For more information about this subroutine, see *Handling Exceptions and Errors* later in this chapter.

INFDS

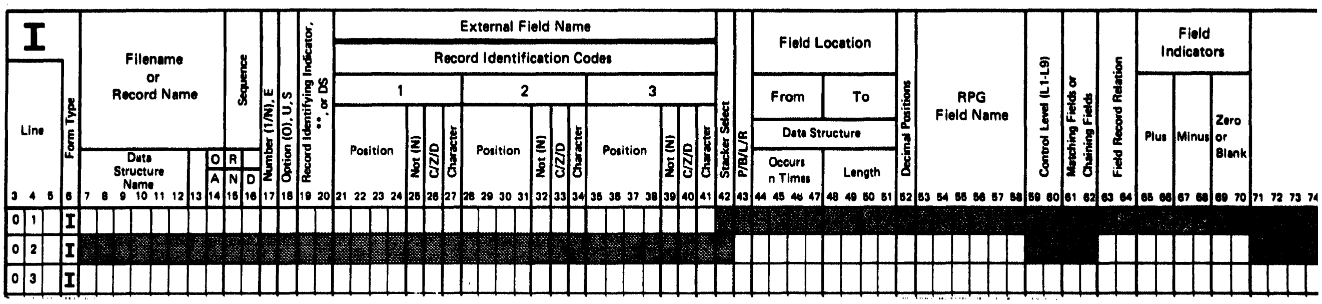
The name coded in columns 60 through 65 for the INFDS option identifies the WORKSTN file information data structure. For more information about this data structure, see *Handling Exceptions and Errors* later in this chapter.

CFILE

The name coded in columns 60 through 67 for the CFILE option identifies a communication format file. This file associates a WORKSTN file with a communication format file defined through the interactive data definition utility (IDDU). For more information about IDDU, see Interactive Data Definition Utility in chapter 6, and see the manual *Getting Started with the Interactive Data Definition Utility (IDDU)*, GC21-8003.

Input Specifications

Code entries in the unshaded columns of the input specifications as shown below:



In the first line:

- Columns 7 through 14 must contain the name of the WORKSTN file unless the preceding input specifications are for the same file.
- Columns 14 and 15 can contain OR, or columns 14 through 16 can contain AND, to indicate a relationship between record-identifying indicators or record types on consecutive lines.
- Columns 15 and 16 must contain a numeric or alphabetic entry. A numeric entry indicates that the program checks the sequence of input records. An alphabetic entry indicates that the program does not check the sequence of input records.
- Column 17 can contain 1 or N if columns 15 and 16 contain a numeric entry. 1 indicates that only one record of this type can be present in the sequenced group. N indicates that one or more records of this type can be present in the sequenced group.
- Column 18 can contain blank or O. Blank indicates that the record type must be present if columns 15 and 16 contain a numeric entry. O indicates that the record type is optional if columns 15 and 16 contain a numeric entry.
- Columns 19 and 20 can contain a record-identifying indicator.
- Columns 21 through 41 can contain record identification codes.

In the second line of the input specifications:

- Column 43 can contain blank, P, or B:

  - Blank indicates that the field is in zoned-decimal format or is alphameric.
  - P indicates that the field is in packed-decimal format. Use the P entry only if the input is from an SSP-ICF session.
  - B indicates that the field is in binary format. Use the B entry only if the input is from an SSP-ICF session.

- Columns 44 through 51 must contain the location of the field in the input record. These entries do not refer to the location of the field as it is displayed. The input fields are placed in the input record in the order in which they are described in SDA (top to bottom, left to right) or in the order in which they are described in the display format specifications. However, you can use the line number and horizontal position columns on the display format specifications to change the order in which the fields appear on the display. Figure 6-2 shows the relationship between the display format specifications and the RPG input specifications.

- Column 52 can contain a digit to indicate the number of decimal positions in a numeric field named in columns 53 through 58.

- Columns 53 through 58 must contain the name of a field, array, or array element in the input record.

- Columns 63 and 64 can contain a field-record-relation indicator.

- Columns 65 through 70 can contain field indicators.

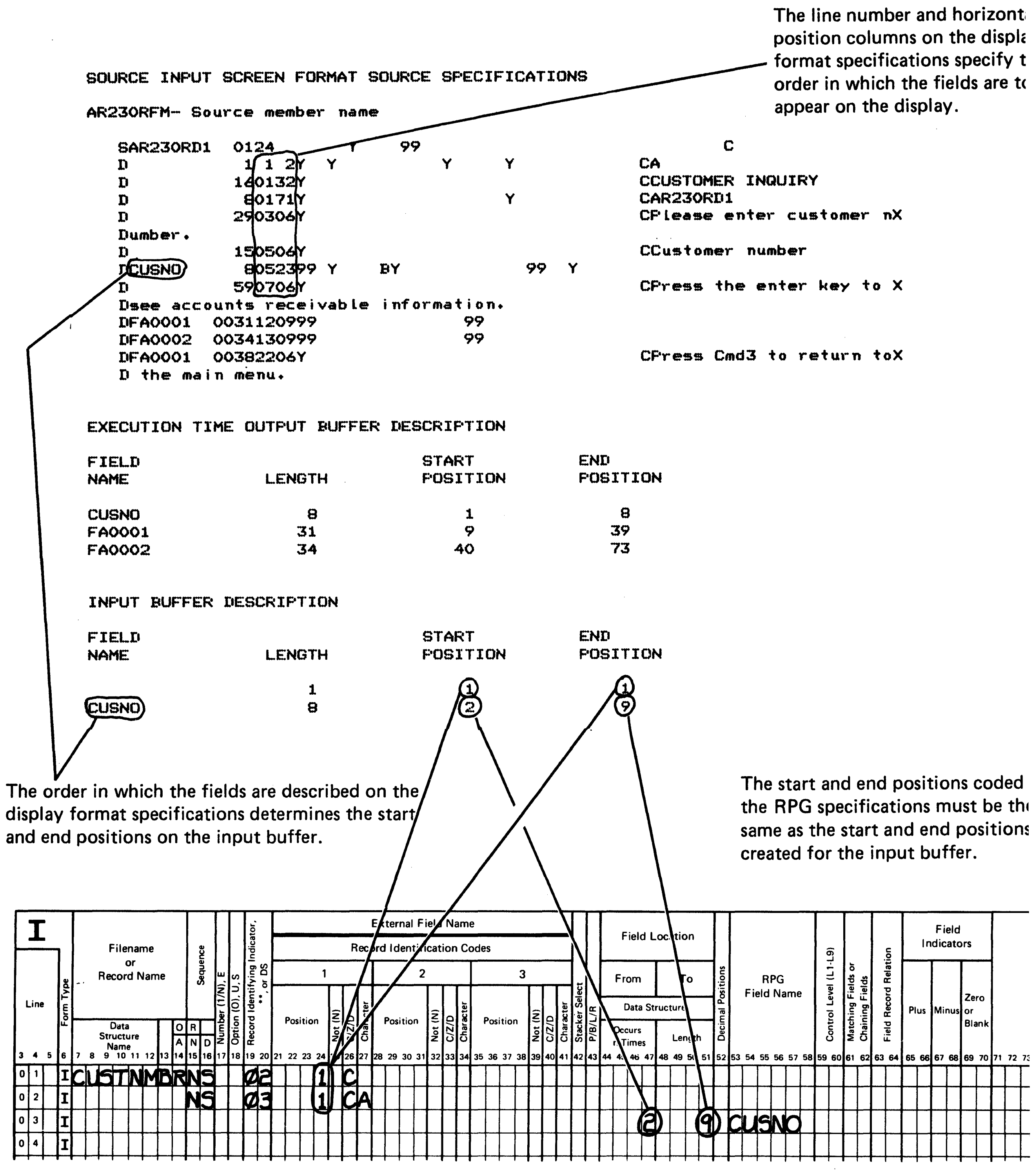


Figure 6-2. Relationship Between the Display Format Specifications and the RPG Input Specifications

Each record, including the blank record at the first read to a device, should be identified on the input specifications. For displays, specify a nondisplayed, protected output/input field as the record code on each display format.

The first input record read from a device is blank except in the following cases:

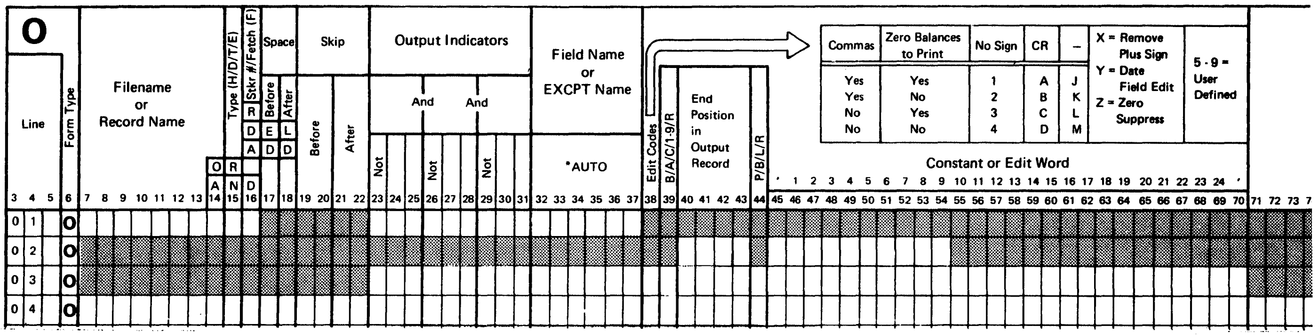
- The program reads a format that was displayed by a procedure (PDATA-YES was specified on the control language PROMPT statement in the procedure). For more information on creating procedures, see *Making Procedures* in the *System Reference* manual.

- The program reads a format that was displayed by a different program (see *Reading Data from a Display Shown by a Previous Program* later in this chapter).

- The program is a single requesting terminal (SRT) program and writes a display before reading for the first time.

Output Specifications

Code entries in the unshaded columns of the output specifications as shown below:



On the first line:

- Columns 7 through 14 must contain the name of the WORKSTN file unless the preceding output specifications are for the same file.
- Columns 14 and 15 can contain OR, or columns 14 through 16 can contain AND, to indicate a relationship between output indicators on consecutive lines.
- Column 15 must contain H (heading), D (detail), T (total), or E (exception) to indicate the type of record to be written.
- Column 16 can contain R to indicate that the device is to be released from the program after output to that device occurs. If OR is coded in column 14 and 15, column 16 must contain an R for each OR line.
- Columns 23 through 31 can contain output indicators other than the first-page (1P) indicator. For information about output indicators, see Chapter 12.
- Columns 32 through 37 can contain an EXCPT name if column 15 contains E.

On the second line:

- Columns 40 through 43 must contain K and the number of characters in the name of the display format.

- Columns 45 through 54 must contain the name of the display format, enclosed in apostrophes. One and only one format name is required for each output record for a WORKSTN file.

On the third and following lines:

- Columns 23 through 31 can contain output indicators other than the first-page (1P) indicator.

- Columns 32 through 37 can contain the name of an output field. The fields must be coded on the output specifications in the order in which they are described on the display format S and D specifications.

- Column 38 can contain an edit code. If you use the Z edit code for a signed numeric field that has a value of zero, the RPG program sends a blank field to the System Support Program, which places a zero in the rightmost position of the field. For more information about edit codes, see Chapter 16.

- Column 39 can contain B or blank. B indicates that the field named in columns 32 through 37 is to be set to blanks or zero after it is written.

- Columns 40 through 43 must contain the end position of the field in the output record. The end position does not refer to the end position of the field as it appears on the display. Use SDA or the output from the $SFGR utility as a guide when coding the end position of the field (see Figure 6-3).

- Column 44 can contain blank, P, or B:

  - Blank indicates that the field is in zoned-decimal or alphameric format.
  - P indicates that the field is in packed-decimal format. Use the P entry only if the output is to an SSP-ICF session.
  - B indicates that the field is in binary format. Use the B entry only if the output is to an SSP-ICF session.

- Columns 45 through 70 can contain an edit word or a constant. For information about edit words, see Chapter 16.

```
SOURCE INPUT SCREEN FORMAT SOURCE SPECIFICATIONS

AR230RFM- Source member name

    SAR230RD1   0124         Y      99                                C
    D              1 1 2Y  Y              Y        Y          CA
    D            160132Y                                      CCUSTOMER INQUIRY
    D             80171Y                           Y          CAR230RD1
    D            290306Y                                      CPlease enter customer nX
    Dumber.
    D            150506Y                                      CCustomer number
    DCUSNO        8052399 Y     BY           99  Y
    D            590706Y                                      CPress the enter key to X
    Dsee accounts receivable information.
    DFA0001    0031120999            99
    DFA0002    0034130999            99
    DFA0001    00382206Y                                      CPress Cmd3 to return toX
    D the main menu.
```

The line number and horizont position columns on the displ format specifications can be u to change the order in which t fields appear on the display.

EXECUTION TIME OUTPUT BUFFER DESCRIPTION

FIELD NAME	LENGTH	START POSITION	END POSITION
CUSNO	8	1	8
FA0001	31	9	39
FA0002	34	40	73

The end positions created for the output buffer and the end positions coded on the RPG output specificati must be the same for each field.

INPUT BUFFER DESCRIPTION

FIELD NAME	LENGTH	START POSITION	END POSITION
	1	1	1
CUSNO	8	2	9

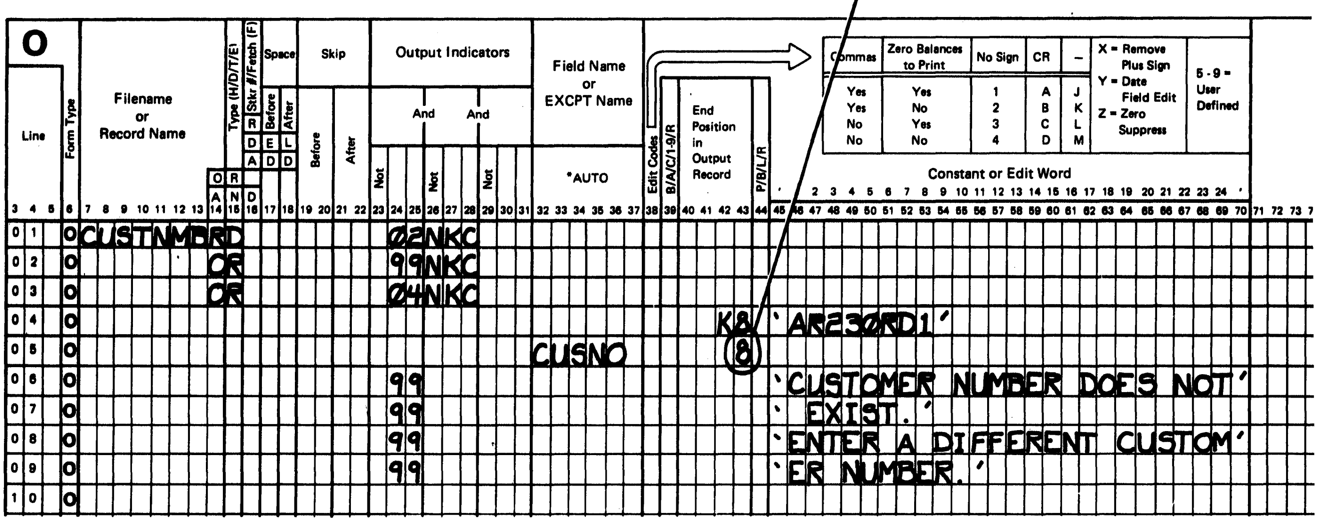


Figure 6-3. Relationship Between the Output Buffer and the RPG Output Specifications

COMMON PROCESSING VARIATIONS

RPG programs that use a WORKSTN file commonly include one or more of the following processing variations:

- Using command keys
- Handling exceptions and errors
- Reading data from a display shown by a previous program

Using Command Keys

There are 24 command keys. Each one corresponds to a separate command-key indicator:

Cmd Key	Command-Key Indicator	Keyboard Keys to Press	Cmd Key	Command-Key Indicator	Keyboard Keys to Press
1	KA	Cmd, 1	13	KM	Cmd, Shift, |
2	KB	Cmd, 2	14	KN	Cmd, Shift, @
3	KC	Cmd, 3	15	KP	Cmd, Shift, #
4	KD	Cmd, 4	16	KQ	Cmd, Shift, $
5	KE	Cmd, 5	17	KR	Cmd, Shift, %
6	KF	Cmd, 6	18	KS	Cmd, Shift, ¬
7	KG	Cmd, 7	19	KT	Cmd, Shift, &
8	KH	Cmd, 8	20	KU	Cmd, Shift, *
9	KI	Cmd, 9	21	KV	Cmd, Shift, (
10	KJ	Cmd, 0	22	KW	Cmd, Shift,)
11	KK	Cmd, -	23	KX	Cmd, Shift, _
12	KL	Cmd, =	24	KY	Cmd, Shift, +

Note: The keyboard keys may vary, depending on what type of keyboard you have.

You can use SDA or column 28 and columns 64 through 79 of the display format S specifications to allow a command key to be used in a program (see the manual *Creating Displays* for more information). You can use any of the 24 command keys with a WORKSTN file. If a person presses a command key that can be used in that program, the corresponding command-key indicator turns on. You can then use the command-key indicator to condition calculation and output operations.

For example, you can specify that the person press command key 2 (rather than the Enter key) when the last item for an invoice has been typed at the display station. You can then use command-key indicator KB in the program to condition calculation operations and output operations, such as presenting the next display.

If the person using the display station presses a command key that is not allowed by the format, error message KBD-0099, KEY NOT VALID AT THIS TIME, is displayed. The person can press the Error Reset key and then press the correct command key.

For a discussion of how to determine whether a command key was pressed, see *Coding the INFDS Data Structure* in this chapter.

Note: Each time an input operation occurs from a WORKSTN file, all command-key indicators are reset, unless an exception or error occurs during the input operation.

To document the use of the command keys for the person using the display station, you can use the template assignment form on the IBM 5251 Display Station Keyboard Template Assignment Sheet and Display Screen Layout Sheet.

Handling Exceptions and Errors

For a display station, the term *exception* means input from a function key (Print, Roll Up, Roll Down, Clear, Help, or Home) to the program. This input is an exception because no data is read into the program, so the program cannot go through normal processing such as record identification.

To allow function keys to be used, you must do three things:

- You must define which function keys you will allow input from. You define them with either SDA or column 27 and columns 64 through 79 of the display screen format S specifications when you create the display for your program. No specific function is automatically associated with any function key, so you can define their functions. For information about how to define function keys, see the manual *Creating Displays*.
- Your program must include the file information data structure (INFDS), which contains an indication of the exception or error. If the program does not contain the INFDS, it cannot tell whether one of the function keys was pressed.
- Your program must also include either the exception/error-processing subroutine (INFSR) or a resulting indicator in columns 56 and 57 of the calculation specification for a READ operation. If neither the INFSR subroutine nor resulting indicators are specified, the program halts and an error message is displayed.

The term *error* means an error that occurs during an input or output operation (either a program cycle input/output operation or an ACQ, REL, NEXT, READ, or EXCPT operation).

If an exception or error occurs while your program is processing a WORKSTN file, you can use the INFDS and either the INFSR or resulting indicators in columns 56 and 57 of the calculation specification for an ACQ, REL, NEXT, POST, or READ operation to control the program logic.

First, your program can check the information in the INFDS. The INFDS contains an identification of the exception or error that occurred and an identification of the operation for which it occurred. The INFDS also contains status information on normal conditions (not exceptions or errors) such as whether a command key was pressed or whether end of file has occurred. The information in the INFDS is updated for each ACQ, REL, NEXT, POST, READ, or EXCPT operation or for each input or output operation in the program cycle.

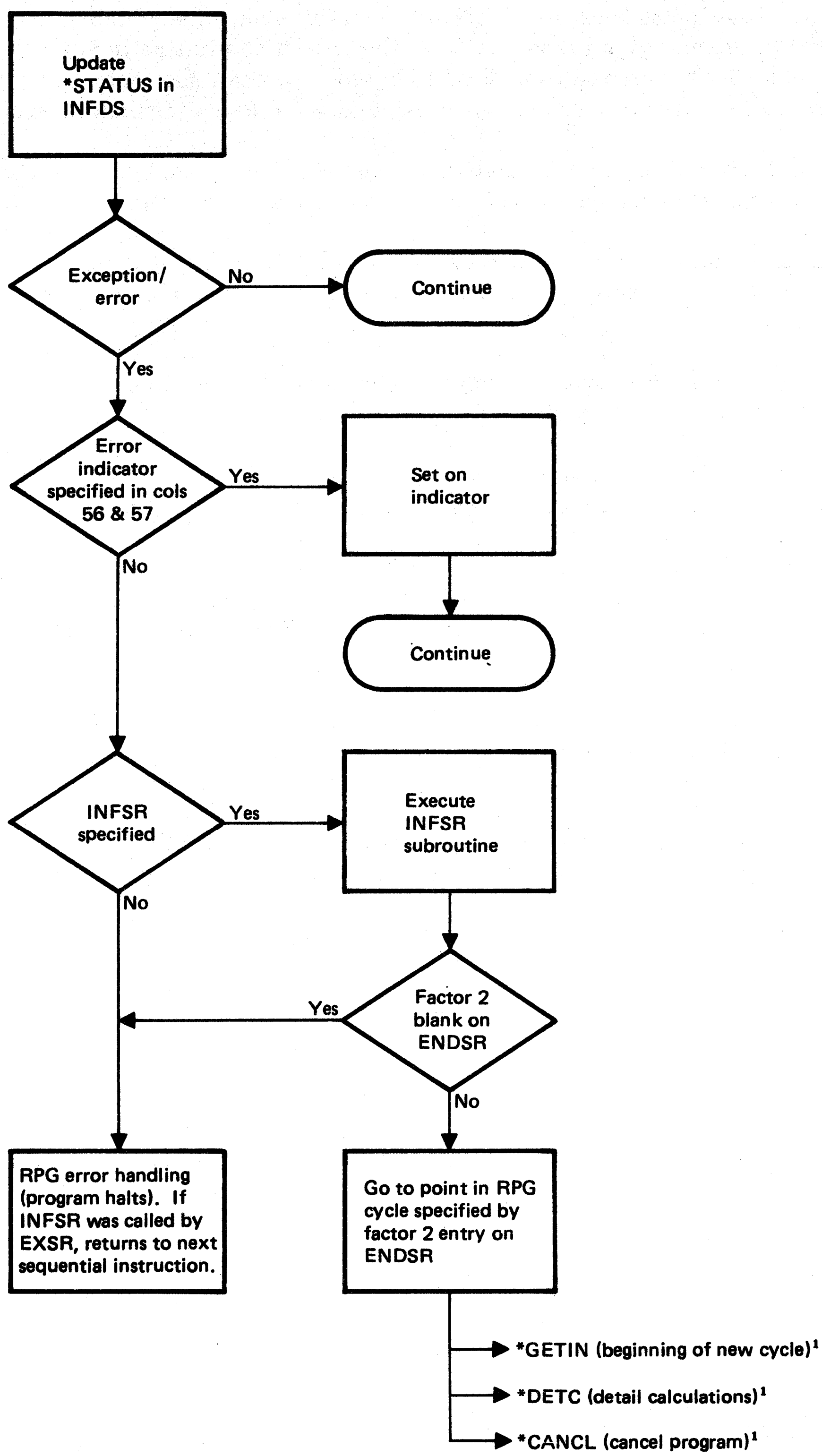
Then, using that information in the INFDS, you can determine which exceptions or errors you want to handle in the INFSR subroutine in your program and which ones you want RPG to handle. Control automatically passes to the INFSR subroutine, if specified, under the following conditions:

- If an exception or error occurs at input time of the program cycle for a primary file, at exception output time, or at normal output time
- If an exception or error occurs on an ACQ, REL, NEXT, POST, or READ operation that does not have an indicator coded in columns 56 and 57

In addition, the INFSR subroutine can be called directly from detail or total calculations by the EXSR operation.

The indicator specified in columns 56 and 57 for an ACQ, REL, NEXT, POST, or READ operation turns on if an exception or error occurs on that operation. Control then passes to the next executable operation in the program. In order to use the INFSR subroutine, the next operation must be an EXSR operation that calls the INFSR subroutine. Control does not automatically pass to the INFSR subroutine if the EXSR operation is not specified.

The relationship between INFDS, INFSR, and indicators in columns 56 and 57 is shown in Figure 6-4. These exception/error-handling techniques are optional and can be used individually or in any combination. However, if function keys are allowed for the program, the INFDS data structure and either the INFSR subroutine or an error indicator on the READ operation must be specified. You can choose the technique that best suits your own program.



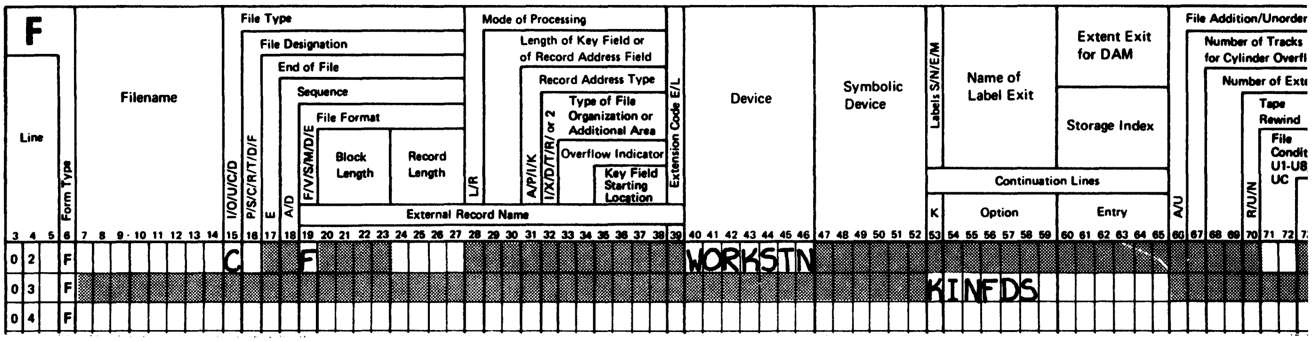
[1] For the exact point in the cycle that is specified by these keywords, see Chapter 19.

Figure 6-4. Handling Exceptions and Errors in a WORKSTN File

Coding the INFDS Data Structure

File Description Specifications

Three entries are required on the continuation line of the file description specifications for the WORKSTN file:



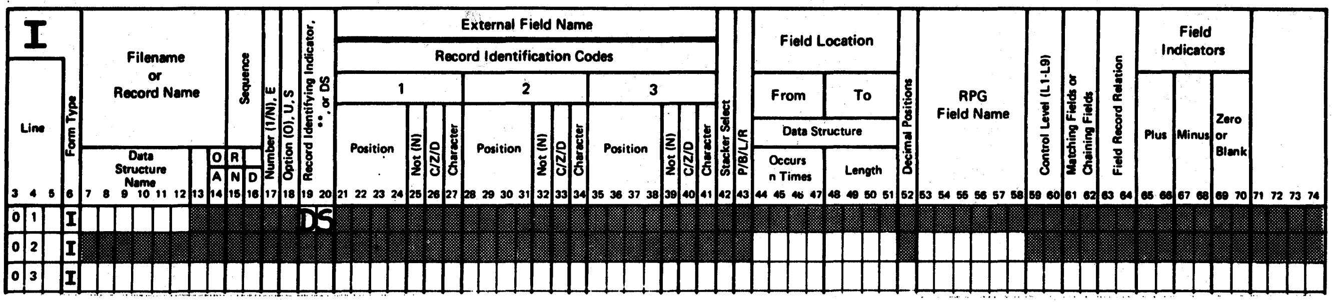
Column 53 must contain K to indicate that this is a continuation line.

Columns 54 through 58 must contain INFDS.

Columns 60 through 65 must contain the name of the data structure to be used as the INFDS.

Input Specifications

The following input specifications are required for the INFDS:



On the first line of the input specifications:

- Columns 7 through 12 must contain the name of the data structure. The name must be the same as the name coded in columns 60 through 65 of the file description specifications continuation line with the INFDS keyword.
- Columns 19 and 20 must contain DS to indicate data structure.

On the second and following lines:

- Columns 44 through 51 must contain a keyword that identifies the location of the subfields containing the status information. The valid keywords are *STATUS, *OPCODE, *SIZE, *RECORD, *MODE, *INP, and *OUT. The keywords are not labels and cannot be used to refer to the subfields.
- Columns 53 through 58 must contain a subfield name. You can use the subfield names to refer to the subfields.

Figure 6-5 shows all the subfield keywords and subfield names.

Line	Form Type	Filename or Record Name (Data Structure Name)	Record Identifying Indicator, **, or DS	Field Location From	Field Location To	RPG Field Name
01	I	EXCPDS	DS			
02	I			*STATUS		STATUS
03	I			*OPCODE		OPCODE
04	I			*RECORD		FMTNM
05	I			*SIZE		SIZE
06	I			*MODE		MODE
07	I			*INP		INP
08	I			*OUT		OUT
09	I			23	26	RCODE
10	I					

The keywords and predefined from and to locations in columns 44 through 51 define the location and size of the subfields in the INFDS data structure, which contain the status information. Field names must be assigned in columns 53 through 58 so the subfields can be referred to in the program.

Figure 6-5. Subfield Keywords for the INFDS Data Structure

***STATUS Keyword**: The name in columns 53 through 58 for the *STATUS keyword identifies a 5-digit numeric subfield with zero decimal positions within the INFDS data structure. This subfield contains a code that identifies the exception or error that occurred. The codes are as follows:

Exception/Error Conditions	Code
Function Keys	
Print	01121
Roll Up	01122
Roll Down	01123
Clear	01124
Help	01125
Home	01126
Error Status Codes	
Input was rejected because the buffer is too small	01201
Permanent I/O error occurred	01251
Invalid device, or maximum number of requesting and/or acquired display stations	01261
Device is busy	01271
Display station was released by operator	01275
Input was rejected, device is not available, or device was not found	01281
Attempt to acquire a device already owned	01285
Other input errors	01299
Change of direction was received with no data	01311
Request for change of direction was received	01321
Time interval expired	01331

If an exception or error occurs, RPG bypasses the move field logic, no fields are changed, no record-identifying indicators are turned on, and **the command-key indicators are not reset**.

You also have access to the following successful status codes that are placed in *STATUS after any input/output operation:

Condition	Code
No exception (with a display station, either the Enter or Auto Record Advance key was pressed)	00000
Any of the 24 command keys	00002
End of file (input rejected, no display stations ready)	00011

Any code in *STATUS greater than 99 is considered to be an exception or error, and the resulting error indicator, if specified, turns on. If no resulting error indicator is specified on an ACQ, REL, NEXT, POST, or READ operation or if the program cycle is at input time for a primary file, at exception output time, or at normal output time, control automatically passes to the INFSR subroutine.

For information on return codes resulting from the use of the Interactive Communications Feature, see the *Interactive Communications Feature: Reference*.

***OPCODE Keyword**: The name coded in columns 53 through 58 for the *OPCODE keyword identifies a 5-character alphameric subfield within the INFDS data structure. This subfield contains a value that identifies which operation was being performed when the exception or error occurred. The value inserted in the *OPCODE subfield is READ, ACQ, REL, NEXT, POST, or WRITE (for output operations). A value is inserted in the *OPCODE subfield when a value greater than 99 is placed in *STATUS.

***RECORD Keyword**: The name coded in columns 53 through 58 for the *RECORD keyword identifies an 8-character alphameric subfield within the INFDS data structure. This subfield contains the format name if *OPCODE contains WRITE. If *OPCODE does not contain WRITE, *RECORD is blank.

***SIZE Keyword**: The name coded in columns 53 through 58 for the *SIZE keyword identifies a 4-digit numeric subfield within the INFDS data structure. The digits in this subfield indicate the size of the display. The subfield contains either 1920 (24 rows x 80 columns = 1920 characters), or, if you are using a 3180 model 2 workstation, 3564 (27 rows x 132 columns = 3564 characters). The subfield is reset each time the POST operation occurs.

***MODE Keyword**: The name coded in columns 53 through 58 for the *MODE keyword identifies a 2-digit numeric field that indicates if ideographic support was requested when you signed on.

Value	Explanation
10	Ideographic support was requested.
00	Ideographic support was not requested.

***INP Keyword**: The name coded in columns 53 through 58 for the *INP keyword identifies a 2-digit numeric field that indicates whether the ideographic or the alphameric/katakana keyboard is being used with this display station.

Value	Explanation
10	The ideographic keyboard is being used.
00	The alphameric/katakana keyboard is being used.

***OUT Keyword**: The name coded in columns 53 through 58 for the *OUT keyword identifies a 2-digit numeric field that indicates whether this display station's screen is capable of displaying ideographic characters.

Value	Explanation
10	The screen can display ideographic characters.
00	The screen cannot display ideographic characters, or the display is not output-capable.

Positions 23-26: Positions 23 through 26 of the INFDS data structure contain the 4-character return code for WORKSTN files. These positions are filled in for all WORKSTN files. This subfield is similar to *STATUS except that *STATUS values are the same for RPG on all IBM systems. Positions 23 through 26 must be coded as the beginning and ending positions of the return-code subfield on the input specifications (see Figure 6-5). The subfield is referred to by the name coded in columns 53 through 58 of the input specifications. Figure 6-6 shows the RPG return codes for display stations.

Note: This subfield is not updated for a *STATUS value of 01261 because RPG does not call the System Support Program. If the *STATUS value is 01281, this subfield is not updated unless the error occurs on a read or ACQ operation.

For information about major and minor return codes that result from the use of the Interactive Communications Feature, see the *Interactive Communications Feature: Reference* manual.

Major Return Code	Minor Return Code	Explanation
00	00	Operation was successful
01	00	Program successfully acquired a new requester
02	00	System operator requested a halt
04	00	Output exception occurred
08	00	Program attempted to acquire a display station that was already acquired. No error
11	00	Input operation was attempted but no input was available (end of file)
18	00	Acquire failed temporarily
24	00	Display station was released by option 2 chosen on inquiry display
28	00	Operation was rejected because the program previously released the single requester
32	00	Acquire failed because the user is unauthorized
34	01	Input was rejected because the buffer is too small
38	00	Attempt to acquire the display station failed
40	00	Requested display station is offline
80	00	Permanent device error occurred

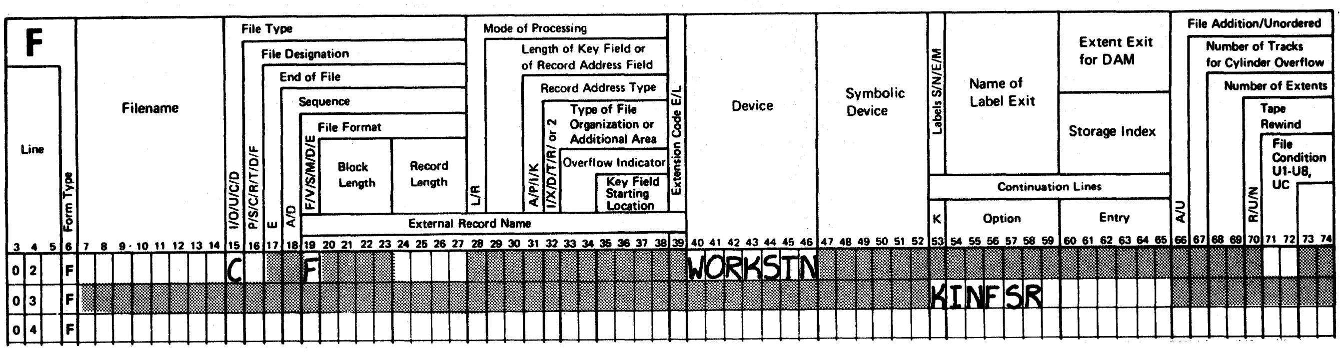
Figure 6-6. WORKSTN Return Codes

Coding the INFSR Subroutine

The INFSR subroutine can perform any function normally allowed in calculations, including exits to other calculation subroutines and input/output operations. The INFSR subroutine returns control to the point specified by the optional factor 2 entry for the ENDSR operation.

File Description Specifications

Three entries are required on the continuation line of the file description specifications for the WORKSTN file:



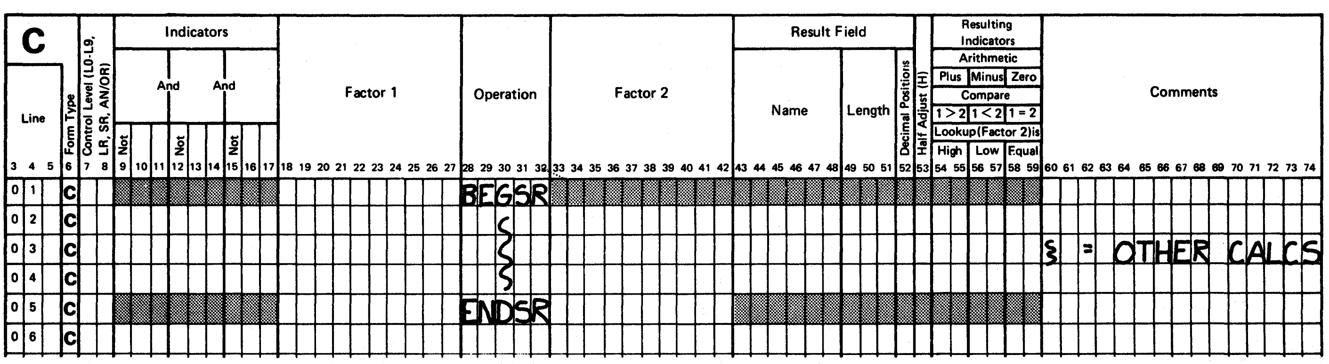
Column 53 must contain K to indicate that this is a continuation line.

Columns 54 through 58 must contain INFSR.

Columns 60 through 65 must contain the name of the calculation subroutine that is to be performed if a WORKSTN exception or error occurs on an ACQ, REL, NEXT, POST, or READ operation for which no error indicator is coded in columns 56 and 57 or of program cycle input/output operations.

Calculation Specifications

The INFSR subroutine requires the following entries in the calculation specifications:



BEGSR Operation: Factor 1 of the BEGSR operation must contain the name of the exception/error-processing subroutine. This must be the same as the name coded in columns 60 through 65 of the file description specification continuation line.

ENDSR Operation: Factor 2 of the ENDSR operation can contain a literal, an array element, or a field name that identifies the point to which the INFSR returns control. INFSR can be accessed using EXSR whether factor 2 contains a blank or any of the allowed literals: '*GETIN', '*DETC', or '*CANCL'.

- If factor 2 contains blanks and if the INFSR subroutine was called directly by the EXSR operation, control returns to the operation following the EXSR statement.

- If factor 2 contains blanks and if the subroutine was called indirectly (that is, control was automatically passed to the subroutine because of an exception or error), the subroutine is performed, and control is passed to the RPG error-handling routine, which causes the program to halt and prompts the person at the display station to choose the appropriate option.

- If factor 2 contains a literal, the literal must be one of the following keywords. The keyword must be enclosed in apostrophes.

  '*GETIN' Control returns to the beginning of a new cycle.

  '*DETC' Control returns to the beginning of detail calculations within the same cycle.

  '*CANCL' Files are closed and the program is canceled.

  ' ' A literal value of blanks is the same as no entry.

  If an exception or error occurs on an operation that attempts to read data from a file and the exception/error-handling subroutine receives control, you must ensure that an output operation to the WORKSTN file occurs before another read occurs. Two consecutive read operations cannot be performed to the WORKSTN file. For example, if the WORKSTN file is a primary file and the exception/error subroutine ENDSR statement specifies a return point of *GETIN, an output operation must be performed to the file before the ENDSR operation occurs. The *GETIN routine will attempt to read from a WORKSTN primary file.

- If factor 2 is an array element or field name, it must identify a 6-character alphameric field that contains one of the keywords *GETIN, *DETC, or *CANCL or blanks that define the return point from the subroutine. By specifying the return point in a field, you can use the subroutine to process all types of exceptions and errors that occur in the WORKSTN file.

  If a field name or array element is specified in factor 2, the field or array element is set to blanks upon each exit from the subroutine. Therefore, you can control the return point of the INFSR within the program by placing the return point in the field that best fits the particular exception or error that occurred. If factor 2 is blank and if the subroutine was called indirectly, the subroutine is performed and control passes to the RPG error-handling routine. If factor 2 is blank and if the INFSR subroutine was called directly by the EXSR operation, control returns to the calculation immediately following the EXSR operation.

Reading Data From a Display Shown by a Previous Program

When one program in a procedure uses a normal output operation to show a display and then goes to end of job or releases the display station, the next program in the procedure can read that display. The person using the display station can enter data into the display while the second program is initiating. When the person presses the Enter key, the data entered into the display is sent to the second program. This technique is called a *read under format.*

There are two ways to do a read under format:

- One way is to use the control language PROMPT statement to display the format. The PROMPT statement is in the procedure, so no other program is involved. For information about the PROMPT statement, see the *System Reference* manual.

- The other way is to display the format in one program, then end that program, and load and run a second program, which reads the format. This method does not involve any control language statements to display the format.

USING ONE OR MORE DISPLAY STATIONS

A WORKSTN file allows people to interact with the program from one or more display stations at the same time. A program that can process requests from only one display station or SSP-ICF session is called a *single requester terminal (SRT)* program. A program that can process requests from more than one display station or SSP-ICF session at the same time, using a single copy of the program, is called a *multiple requester terminal (MRT)* program. Whether a program is a SRT or a MRT depends on the number you enter in response to the prompt *Maximum number of requesting display stations* on the RPGONL, RPGC, or AUTOC procedure when you compile the program. For information about these procedures, see Chapter 3, *Entering and Compiling an RPG Program.* For a complete description of SRT and MRT programs, see the *Concepts and Programmer's Guide.*

Using a SRT Program

Although a SRT program allows only one requester, it is possible for more than one requesting display station or SSP-ICF session to use a SRT program at the same time if each display station or SSP-ICF session uses a *separate* copy of the program.

Using a MRT Program

Each requester of a MRT program uses the same copy of the program. The first requester loads and initiates the program. If the WORKSTN file is a primary file, each succeeding requester begins to use the program at the beginning of an input cycle. If the WORKSTN file is a demand file, each succeeding requester begins to use the program when the READ operation for the WORKSTN file occurs. If the program is handling the maximum number of requesters, the System Support Program places the next requester of the program in a queue. When the program releases one of its requesters, the program can process the queued request.

If the program is called by more than one requester, the first requester:

- Initiates the program
- Provides the external indicators (U1 through U8)
- Provides the display station local data area for the data structure defined by a U in column 18 of the input specifications

Each requester can access any display station local data area and external indicators attached to the program by using SUBR20 and SUBR21. For information about SUBR20, see *Setting and Restoring External Indicators* later in this chapter. For information about SUBR21, see *Reading and Writing the Local Data Area for a Display Station* later in this chapter.

Program error messages go to the system operator.

Requesters can leave the program without suspending the program or other devices.

If the Attn or Sys Req key is pressed while a MRT program is running, the processing of information is suspended at the display station where the Attn or Sys Req key was pressed. The program continues to process information from other requesters. However, if the program must write a display to the suspended requester, the entire program is suspended (all requesters are suspended).

File Description Specifications

A MRT program can include one or more continuation lines for the WORKSTN file description specifications. For more information, see *Continuation-Line Options* earlier in this chapter.

Calculation Specifications

Two operations that are commonly used in MRT programs are NEXT and REL.

NEXT Operation

The NEXT operation forces the next input to the program to come from the device specified in factor 1. Factor 1 must contain the name of a 2-character field that contains the device identification or a 2-character alphameric literal that is the device identification. Factor 2 must contain the name of the WORKSTN file.

If NEXT is coded more than once between input operations, only the last operation occurs. If you code an indicator in columns 56 and 57, it turns on if an exception or error occurs on the NEXT operation.

If columns 56 and 57 do not contain an indicator and an exception or error occurs, the program halts unless the INFSR subroutine is specified. If the INFSR subroutine is specified, the subroutine automatically receives control when an exception or error occurs. For more information on the INFSR subroutine, see *Handling Exceptions and Errors* earlier in this chapter.

REL Operation

The REL operation releases the device specified in factor 1 from the program. Factor 2 must contain the name of the WORKSTN file.

You can release either a requester or an acquired device with the REL operation code. The program releases the specified device when the REL operation occurs during calculations unless the device is the requester of a SRT program. If the device specified in factor 1 is the requester of a SRT program, the device is released at end of job, not when the REL operation occurs in calculations, but the device is no longer available to the program except to log messages. If the device is a display station, the display station is no longer available to the program.

If you code an indicator in columns 56 and 57, it turns on if an exception or error occurs on the REL operation. If columns 56 and 57 do not contain an indicator and an exception or error occurs, the program halts unless the INFSR subroutine is specified.

If the INFSR subroutine is specified, the subroutine automatically receives control when an exception or error occurs. For more information on the INFSR subroutine, see *Handling Exceptions and Errors* in this chapter.

Output Specifications

You can release a device from a MRT program by coding R in column 16 of the output specifications. The device is released when the output specification occurs during the output operations. If a format name is coded in columns 46 through 53 of the next output specification, the display appears or the SSP-ICF operation is performed, and then the device is released.

Acquiring One or More Display Stations by the Program

Both SRT programs and MRT programs can acquire one or more display stations or SSP-ICF sessions while the program is running. The program acquires other display stations or SSP-ICF sessions by using the control language WORKSTN statement or the ACQ operation. For information about the WORKSTN statement, see the *System Reference* manual.

A SRT program that acquires any display stations or SSP-ICF sessions must include a NUM option in the continuation line of the file description specifications for the WORKSTN file.

An acquired device cannot supply external indicators U1 through U8, and SUBR20 cannot be used to read or write them.

An acquired device does not provide the display station local data area for the data structure defined by a U in column 18 of the input specifications, and SUBR21 cannot be used to read or write a local data area for an acquired device.

Program error messages go to the requester. If the requester is an SSP-ICF session, program error messages go to the system console.

An acquired display station must have a STANDBY display. To change a display station from a COMMAND display to a STANDBY display so that it can be acquired, enter MODE on the command line at the display station.

ACQ Operation

The ACQ operation acquires the device specified in factor 1 for the program. Factor 2 must contain the name of the WORKSTN file.

If the device is available, it is acquired by the program. If it is not available or was already acquired by the program, the indicator coded in columns 56 and 57 is set on.

If no indicator is coded in columns 56 and 57 but the program contains the INFSR (WORKSTN exception/error-processing) subroutine, the INFSR subroutine automatically receives control when an exception or error occurs on the ACQ operation.

If no indicator is specified and the program does not contain the INFSR subroutine, the program halts when an exception or error condition occurs. Then you can continue the job or try the ACQ operation again. No input or output operation occurs when you use the ACQ operation.

Requesting the Program by One or More Display Stations

Each requester of a MRT program uses the same copy of the program. The first requester provides the external indicators (U1 through U8) and the display station local data area for the data structure defined by a U in column 18 of the input specifications. Other requesters can access any external indicators and display station local data area attached to the program by using SUBR20 and SUBR21.

Setting and Restoring External Indicators (SUBR20)

The IBM-written subroutine SUBR20 allows you to set and restore the external indicators (U1 through U8) for each requesting display station when more than one display station requests the same program. To call SUBR20, code the EXIT SUBR20 operation, followed by **exactly three** RLABL operations (see the example below):

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators	Comments
01	C				EXIT	SUBR20						
02	C				RLABL		OP	1				
03	C				RLABL		TNAME	2				
04	C				RLABL		RCODE	1				
05	C											

OP is a 1-character field that contains I or O to indicate whether the external indicators are to be input to or output from the program for this display station. To enter the appropriate character in the OP field, you can use a MOVE operation before calling the subroutine.

TNAME is a 2-character field that contains the identification of the display station. This field is normally the field whose name is coded in columns 60 through 65 for the ID option on the WORKSTN file description continuation line. If TNAME is not the field whose name was coded in columns 60 through 65 on the WORKSTN file description continuation line, you can use a MOVE operation to enter the appropriate display station identification in the TNAME field before calling the subroutine.

RCODE is a 1-character field that contains one of the following return codes:

0 = Successful

1 = Unsuccessful (the display station is not attached to the program)

2 = Unsuccessful (the display station is not a requester)

The external indicators for the requester of a SRT program are automatically available to the program without the use of SUBR20 and are written out at end of job. The external indicators for the first requester of an MRT program are available without the use of SUBR20, but they are not automatically written out at end of job.

Reading and Writing the Local Data Area for a Display Station (SUBR21)

The IBM-written subroutine SUBR21 allows you to read and write the local data area for each display station when more than one display station requests the same WORKSTN file. To call SUBR21, code the EXIT SUBR21 operation followed by **exactly four** RLABL operations (see the example below):

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators	Comments
01	C				EXIT	SUBR21						
02	C				RLABL		OP	1				
03	C				RLABL		TNAME	2				
04	C				RLABL		RCODE	1				
05	C				RLABL		AREA					
06	C											

OP is a 1-character field that contains I or O to indicate whether the display station local data area is to be input to or output from the program for this display station.

TNAME is a 2-character field that contains the identification for the display station.

RCODE is a 1-character field that contains one of the following return codes:

0 = Successful

1 = Unsuccessful (the display station is not attached to the program)

2 = Unsuccessful (the display station is not a requester)

AREA is a field or data structure from which the local data area for the display station is read or to which it is written. AREA can be up to 512 characters long. If AREA is between 257 and 512 in length, then AREA must be the name of a data structure. Position 1 of the local data area is always placed in position 1 of this field. If you use AREA to pass options to control language statements, use caution when you use the characters ? (question mark) and / (slash).

If a single display station is used, the program writes the information from the data structure to the local data area (LDA). To print, write, or display the contents of the LDA, use the contents as data to be passed to the desired output device. See Figure 6-13, *Sample Program AR935R* for an example of information transfer between programs.

The local data area for the requester of a SRT program or for the **first** requester of a MRT program can be referred to in your program if you define a data structure with a U in column 18 of the input specifications. For a MRT program, the local data area is not automatically written out at the end of the job. (For more information see the section *Local Data Area for a Display Station* in Chapter 14, *Using Data Structures*)

Compiling and Running a MRT Program

Compiling the Program

When you use the RPGONL, RPGC, or AUTOC procedure to compile a MRT program, you must respond to the prompt *Maximum number of requesting display stations* by entering a number less than or equal to the number of all requesting devices.

Respond NEP to the *Never-ending program* prompt if the program is to be used often or if initializing the program is time-consuming. The program is initialized once and remains in main storage until the STOP SYSTEM command is entered from the system console. Respond NONEP to the prompt if the program is seldom used, if initialization time is negligible, or if DISP-SHR is not specified on the control language FILE statements for the files used by your program and needed often by other programs.

For information about the RPGONL, RPGC, and AUTOC procedures, see Chapter 3.

Running the Program

When you use the development support utility, DSU, (or the source entry utility, SEU, if DSU is not installed) to enter the procedure to run a program, and you press command key 7 to end DSU (or SEU), the DSU (or SEU) display asks whether the program is a MRT program. Respond Y to that prompt.

Updating Disk Files in a MRT Program

Possible Errors

Use care when updating DISK files in a MRT program. If a file is shared by two or more display stations in a program and if the record being processed is not updated before the next record is read, the following errors can occur:

- An update can be lost. For example, suppose a record is read from file X and displayed at display station 1, then the same record is read from file X and displayed at display station 2. The update performed at one display station might be destroyed by an update performed at the other display station. If this condition occurs and if DISP-SHR was specified on the control language FILE statement for file X, an error message is displayed and the second update is not performed. If this condition occurs and if DISP-SHR was not specified, the second update overlays the first update.

- An update performed by another program sharing the file can be lost. For example, suppose a record is read from file X and is displayed at display station 1, then a record in a different disk sector is read from file X and displayed at display station 2. The second read operation from file X causes the System Support Program to free the sector containing the first record. Another program sharing file X can then update the first record. If display station 1 also tries to update that record by using the original field values, the updates made by the other program may be lost.

- The wrong record can be updated. For example, suppose a record is read from file X and displayed at display station 1, then a different record is read from file X and displayed at display station 2. If display station 1 tries to update the first record but the program does not reread that record, the program tries to update the last record read from file X. If this condition occurs during an attempt to update an indexed file, an error message is displayed and the requested update is not performed. Otherwise, the wrong record is updated.

Avoiding These Errors

You can avoid the preceding error conditions by using one of the following techniques:

- Before you update a record, read the record again and check that none of the fields being updated have been changed after the record was displayed for updating. If any of the fields were changed, you might want to display the field again for updating or, if possible, update the record by using the field values currently in the record.

- Within the program, define an array for each DISK file. The array should contain one element for each display station. When a person using a display station enters a relative record number or the key field of a record to be updated, the program should check the array to ensure that no other display station is updating that record. If no other display station is updating the requested record, the program should place the specified relative record number or key field into the array element corresponding to the display station. The program can then read the record and display it at the display station. If another display station is updating the record, you can display a message saying so. After the person at the display station enters the updates, the program must read the record again and use the information entered at the display station to update the record. The program should then blank out the array element corresponding to the display station.

- There is another way to solve the problem of more than one display station in a MRT program updating the same record. This method also applies if another program is updating the same file and causes updates for the MRT program to fail. That method is to define each file on a separate control language FILE statement for each display station using the MRT program. Each of the FILE statements should specify a different name but the same label, and each FILE statement should specify DISP-SHR. This method causes each record for each display station and each program to be locked or protected when other display stations or programs try to access it.

Reaching End of File for a MRT Program

Primary File

If a MRT program is defined as a never-ending program, if all devices have been released or no input-capable records are in the WORKSTN file, and if the program tries to read another record from the WORKSTN file, end of file does not occur and RPG does not set on the LR indicator until the system operator enters the STOP SYSTEM command (P S). However, if you set on the LR indicator, the program goes to end of job (that is, the system operator does not have to enter the STOP SYSTEM command).

Note: A MRT program should not set on the LR indicator until end of file is reached for the WORKSTN file. If the LR indicator is set before end of file is reached, undesirable results occur for requesters that are signing on or are still signed on.

Demand File

If the program is defined as a never-ending program and if all devices have been released or no input-capable records are in the WORKSTN file, the end-of-file indicator on the READ operation is not set on until another READ operation occurs and the system operator enters a STOP SYSTEM command. However, if you set on the LR indicator based on some condition other than the end-of-file indicator on the READ operation, the program goes to end of job (that is, the system operator does not have to enter the STOP SYSTEM command).

ADVANCED TOPICS

Processing the Duplicate Character Value

If you specify Enable Dup for a field in a display format (by using SDA or a Y entry in column 34 of the display format D specifications), the person at the display station can press the Dup key to indicate to the program that the contents of the field are to be duplicated from the field in the previous record. When the Dup key is pressed, the field, from the position of the cursor to the end of the field, is filled with the duplicate character value (hexadecimal 1C), which is displayed as the character *. The Dup key does not duplicate any characters; therefore, you must process the duplicate character values in your program.

If you want the person at the display station to either duplicate the entire field or type in the entire field, you need to test only one character in the field to determine whether the Dup key was pressed. For example, you can test the last character in an alphameric field for the duplicate character value by using the TESTB operation code. If the last character in the field is not a duplicate character value, move the contents of the test field to the processing field (see Figure 6-7).

You can also write your program to allow the person using the display station to change the first part of a field and duplicate the latter part of the field. For example, if the person changes the first four characters in a 10-character field and then presses the Dup key, positions 5 through 10 of the field will contain the duplicate character value (hexadecimal 1C). In your program, you then have to test each character in the field to determine where the first duplicate character occurs, and replace the appropriate positions with the data to be duplicated.

For a list of the hexadecimal value of each character, see Chapter 17.

Line	Form Type	Filename or Record Name	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, **, or DS	Record Identification Codes 1	Record Identification Codes 2	Record Identification Codes 3	Stacker Select	P/B/L/R	Field Location From	Field Location To	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																			
02	I											1	6		TSTFLD						
03	I											6	6		TEST						
04	I																				

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 Plus / 1>2 / High	Minus / 1<2 / Low	Zero / 1=2 / Equal	Comments
01	C						TESTB	'345'	TEST						99	ALL ON?
02	C		99				TESTB	'Ø1267'	TEST				99			ALL OFF?
03	C		N99				MOVE	TSTFLD	OLDFLD							
04	C															

Figure 6-7. Testing for a Duplicate Character Value

Using Message Identification Codes

When a message identification code (MIC) is to be displayed for a WORKSTN file, the length of the message must be entered in the field length column of the display format specifications, and the constant type column must contain an M. The name of a 6-character field or a 6-character constant must then be coded on the output specifications. The contents of the field or the constant must be in the form xxxxyy, where xxxx is the MIC number and yy is the 2-character message member identifier. For a complete description of the message member identifier, see the manual *Creating Displays: Screen Design Aid and System Support Program* (SC21-7902).

Overriding Fields in a Display Format

An override operation allows you to override fields in a display format when you redisplay the same format. You can specify an override operation when you design the format with SDA or with the display format S specification (by coding an indicator in columns 33 and 34). An override operation occurs if the indicator is on when the format is displayed. A normal output operation is performed if the indicator is off when the format is displayed (see Figure 6-8).

During an override operation (the indicator in columns 33 and 34 is on), the following occurs:

- A field is unchanged if you code an indicator in columns 23 and 24 of the D specification for that field and that indicator is off. If data was typed into the field, that data is unchanged. Any field that had Y, N, or blank coded in columns 23 and 24 is also unchanged.

- A field is displayed if you code an indicator in columns 23 and 24 of the D specification for that field and that indicator is on. Any data that was typed into the field is lost. Output information is displayed from the *same locations in the output record area* as for a normal display.

- For all fields, the use of indicator-controlled characteristics such as highlight or reverse image is determined by the state of that indicator. All field characteristics that are not controlled by indicators are unchanged.

For example, you may want to override fields in a display if a person types incorrect data into a field. To override fields, code an indicator in columns 33 and 34 of the S specification to allow the format to be overridden. If the person types incorrect data into a field, you can then set on the indicator in columns 33 and 34 and display the format again. If the indicator coded in columns 23 and 24 of the D specification for the field is off, the incorrect data is unchanged. If the indicator is on, data from the program is displayed. You can also use indicators for field characteristics such as highlight and reverse image and can set these indicators on when the override indicator is set on.

For more information about overriding fields in a display format, see the manual *Creating Displays.*

		Indicator in Columns 33 and 34 of the S Specification	
		OFF	**ON**
Indicator in Columns 23 and 24 of the D Specification	**OFF**	Output data comes from D specification (columns 57 through 79).	No change occurs to data on the screen.
	ON	Output data comes from the RPG program.	Output data comes from the RPG program
		Normal Output Operation	Override Operation

Figure 6-8. Effect of Indicators on Output Data during an Override Operation

Using the POST Operation

The POST operation allows you to retrieve status information from the subfield named in columns 53 through 58 of the input specifications for the *SIZE, *MODE, *INP, and *OUT keywords in the INFDS data structure. After a POST operation is performed, the *SIZE subfield will contain one of the following:

- 1920, to indicate that the specified display station has a 1920-character display screen (24 rows x 80 columns).

- 3564, if you are using a 3180 model 2 workstation, to indicate that the specified display station has a 3564-character display screen capability (27 rows x 132 columns).

After a POST operation is performed, the *MODE subfield will contain either 10, to indicate ideographic support was requested, or 00, to show it was not requested.

Following a POST operation, the *INP subfield will contain either 10, to indicate the ideographic keyboard is being used, or 00, to show the alphameric/katakana keyboard is being used.

Following a POST operation, the *OUT subfield will contain either 10, to indicate the display station's screen is capable of displaying ideographic characters, or 00, to show the screen can't display ideographic characters, or the display is not output capable.

Factor 1 must contain a variable or an alphameric literal that identifies the display station whose status is being requested. The result field must contain the name of the INFDS data structure in which this information is to be posted.

Columns 56 and 57 can contain an indicator that turns on if an error occurs on the POST operation. An error occurs if the specified display station is not attached to the WORKSTN file for which the INFDS data structure is specified.

If columns 56 and 57 do not contain an indicator but the program contains the INFSR subroutine, the subroutine automatically receives control when an error occurs.

If columns 56 and 57 do not contain an indicator and the program does not contain the INFSR subroutine, the program halts when an exception or error occurs.

How WORKSTN Files Are Processed

Figure 6-9 shows how the RPG program cycle processes WORKSTN input files. All steps in the cycle except steps 1, 3, 11, 12, 13, and 15 are the same as those for the regular RPG program cycle.

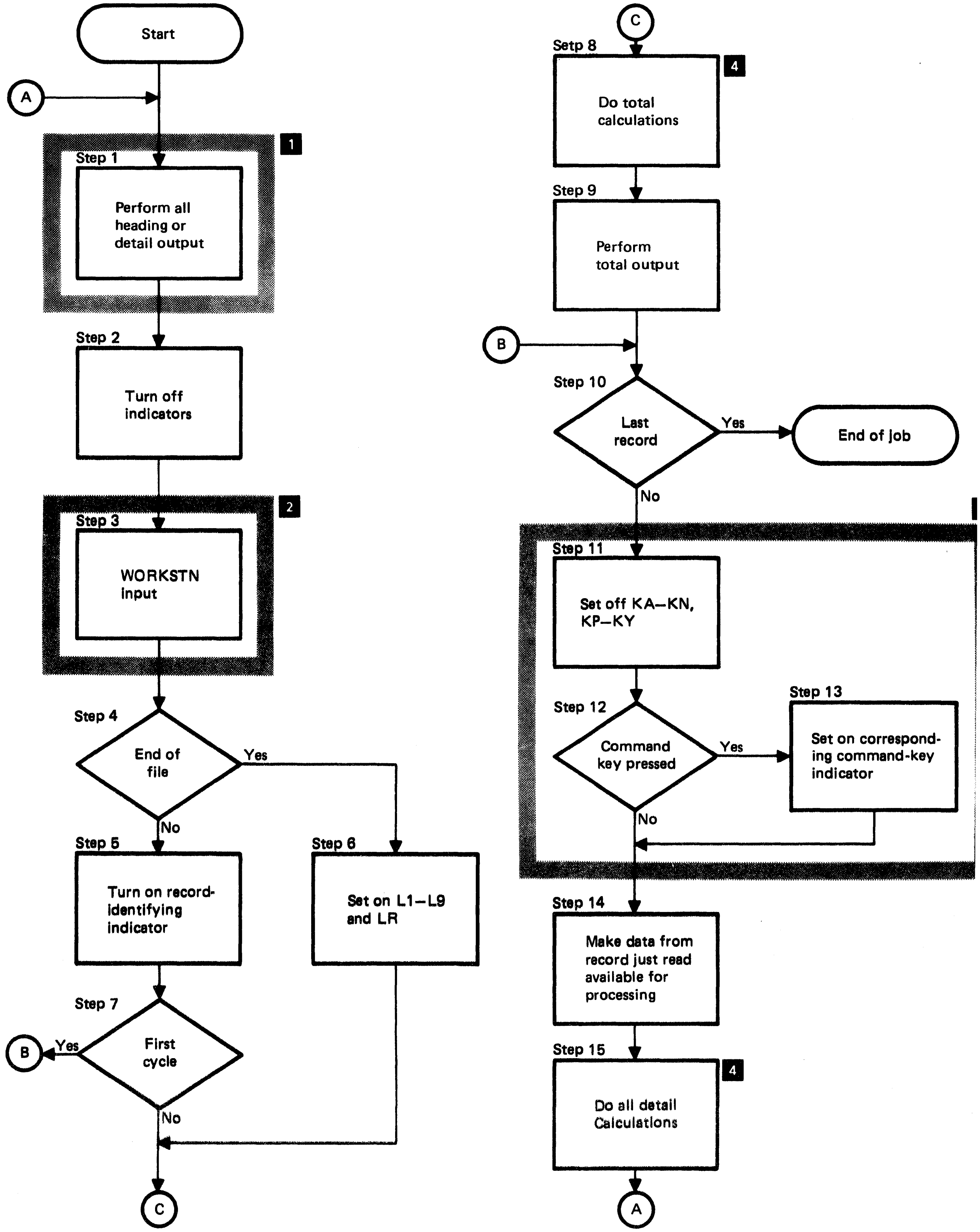


Figure 6-9 (Part 1 of 2). RPG Program Cycle for a WORKSTN File

1 1P output to a WORKSTN file is not allowed.

2 WORKSTN input processing can include:

- Saving the common IND/SAVDS area in the IND/SAVDS area for the device from which the last input record was read (if specified).
- Getting the display station record. If the display station is new to the file, the record may be blank. Only the last input-capable display format can be read into the program.
- Restoring the IND/SAVDS area of the device from which the last input record was read (if specified).
- Inserting value into ID field (if specified).

For a detailed explanation of processing WORKSTN input files, see Figure 6-10 (an expansion of step 3).

Data keyed at a display station is returned (input) to the RPG program for processing when the operator presses a command key or the Enter key. The operator can also cause the data to return to the program by pressing the Field Exit, Field +, or Field - key if the last input field in the format is defined as an auto record advance field (column 36 of the D specification).

3 All command-key indicators are turned off; then the appropriate one, if any, is turned on.

Note: If an exception/error occurs on the read, the command-key indicators are not reset.

4 If the READ operation code is used, it combines steps 3, 5, 11, 12, 13, and 14. If the EXCPT operation code is specified, it uses the ID field to direct output to the display station whose ID is contained in the field.

Figure 6-9 (Part 2 of 2). RPG Program Cycle for a WORKSTN File

Step 3 of Figure 6-9, which is the WORKSTN input processing step, is expanded and shown in Figure 6-10. The following explanation refers to the steps shown in Figure 6-10.

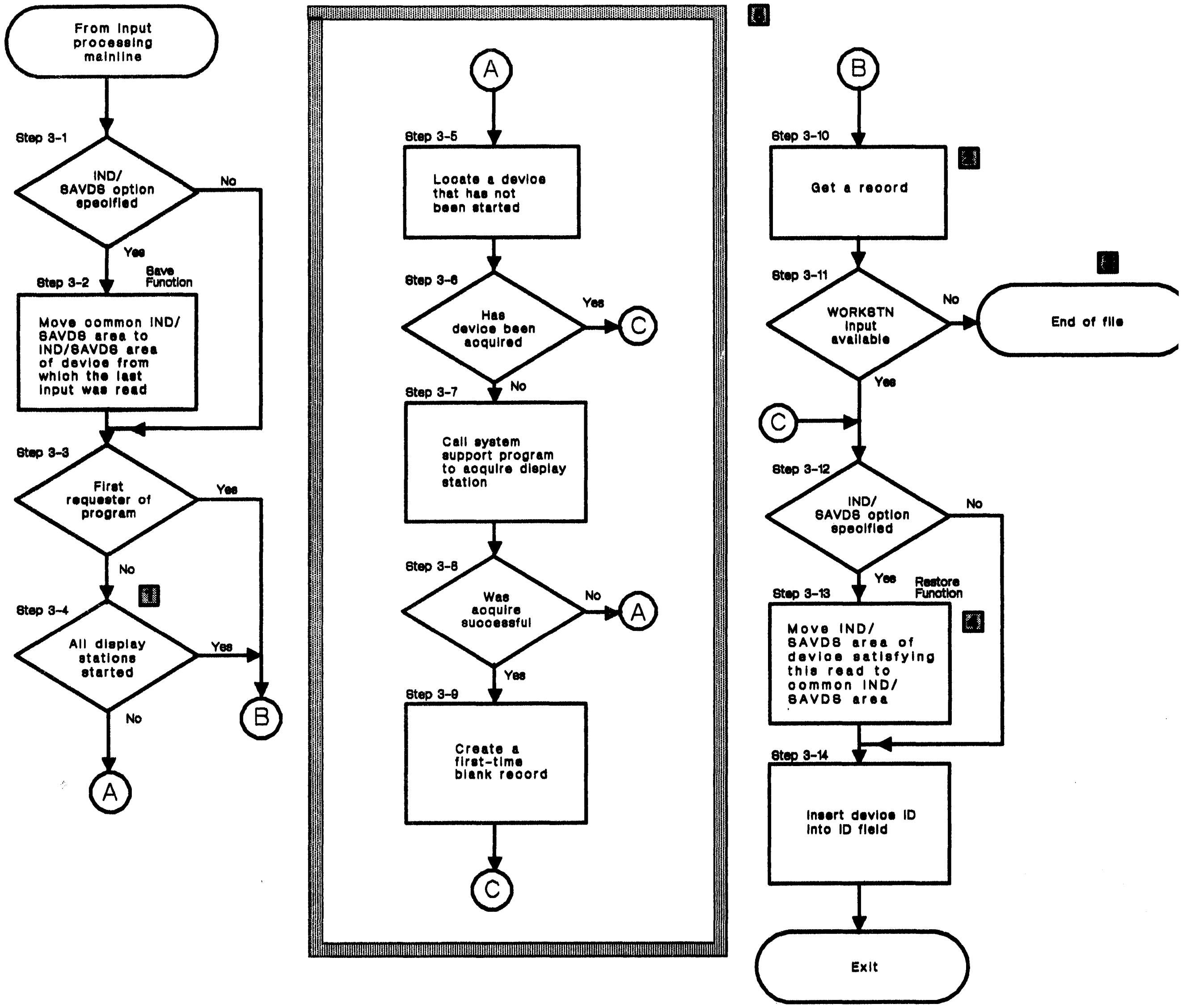


Figure 6-10 (Part 1 of 2). Processing of WORKSTN Input Files

1 One RPG cycle is used to start each acquired device. If no input or output operation to the device has previously been specified, RPG acquires the device if necessary and creates a blank record to satisfy the first read.

2 All requesters of the program except the first enter the program at this step.

3 End of file occurs for a WORKSTN file at the time of the read if:

- All devices have been released.
- Input is not allowed from any of the attached display stations because:
  - A new format has not been displayed at the display station since data was last keyed.
  - Suppress input-yes is specified in column 35 of the S specification for the format currently displayed and no input-capable formats are concurrently displayed at the display station.
- The program is an NEP, if all devices have been released, and if the operator has entered a STOP SYSTEM command.

4 If this is the first input for this device, the indicators specified in the IND field are off and the SAVDS field is blank.

5 Steps 3-5 through 3-9 occur only for acquired devices.

Figure 6-10 (Part 2 of 2). Processing of WORKSTN Input Files

Step 3-1. RPG determines whether the IND and/or SAVDS continuation line option is coded on the file description specifications for this file. If neither option is coded, RPG goes to step 3-3.

Step 3-2. If the IND and/or SAVDS option is coded, RPG moves the common IND/SAVDS area to the IND/SAVDS area for the device from which the last input record was read.

Step 3-3. RPG determines whether this is the first cycle for the first requester of the program. If it is, RPG goes to step 3-10. All requesters of the program except the first enter the program at step 3-10.

Step 3-4. If the device is not a requester, RPG determines whether all devices in the internal device table have been started. If all are started, the program goes to step 3-10.

A device is started when it has been acquired and a successful input or output operation has occurred. If a device is acquired by the ACQ operation, the device is not considered to be started unless output is sent to the device in the same cycle.

Step 3-5. If not all devices have been started, RPG locates a device that has not been started.

Step 3-6. If the device located is a display station, RPG determines whether it has been acquired.

Step 3-7. If the display station has not been acquired, RPG calls the System Support Program to acquire the display station.

Step 3-8. RPG determines whether the acquire was successful. If it was not successful, RPG goes back to step 3-5.

Step 3-9. If the device is acquired, RPG creates a first-time blank record to satisfy the first read to the device. RPG then goes to step 3-11.

Step 3-10. RPG reads in the record from the device. Remember that all requesters of the program except the first enter the program at this point.

Step 3-11. RPG determines whether WORKSTN input is available. If it is not, the program has reached the end of the file.

Step 3-12. RPG determines whether the IND and/or SAVDS continuation line option is coded on the file description specifications for this file. If an option is not coded, RPG goes to step 3-14.

Step 3-13. If the IND and/or SAVDS option is coded, RPG moves the IND/SAVDS area for the device that satisfied the read to the common IND/SAVDS area.

Step 3-14. RPG inserts the device ID of the device that satisfied the read into the ID field if the ID continuation line option is specified on the file description specifications.

After the WORKSTN input file processing, RPG goes to step 4 as shown in Figure 6-9.

Interactive Data Definition Utility (IDDU)

The Interactive Data Definition Utility (IDDU) is a feature that enables you to define communications file formats. You must define these formats through IDDU in order to use the confirm function of the advanced program-to-program communications (APPC) sub-system of the SSP-ICF. The APPC confirm function gives two communicating programs access to return codes that confirm the completion status of ICF functions.

Communications file formats are stored in IDDU's data dictionaries. A data dictionary is a container for field, format, and file definitions, where the definitions consist of data attributes such as length and number of decimal places.

You must indicate at compile time that your program uses IDDU format files containing communication formats, by using either RPGONL, RPGC, or AUTOC procedure. The displays of these procedures have a prompt for specifying a data dictionary name. If you specify a data dictionary name on the RPGONL display, the dictionary must exist at compile time. If you do not use the prompt displays, you must specify a data dictionary name as the nineteenth parameter, for example:

RPGC progname, library,,,,,,,,,,,,,,,,,dictionary

AUTOC progname, library,,,,,,,,,,,,,,,,,dictionary

In order to associate an IDDU format file containing communication formats with a WORKSTN file, you must use the CFILE continuation line option. CFILE must be coded in columns 54 through 59 of the file description specifications, and the name of the communications file must be specified in the leftmost of columns 60 through 67.

If your WORKSTN file includes both SFGR screen formats and IDDU communication formats, you should also specify the ID continuation option on the file description specifications.

On output specifications, IDDU communications formats are specified in the same way as SFGR screen format names.

IDDU-defined communication formats can have the same names as SFGR screen formats, even in the same program. However, if you plan to use a System/36 program in System/38, it is best to use unique format names in the program. This will allow easier conversion to System/38's mixed device files, in which screen and communication formats are found in the same file.

For more information on IDDU, see the following manuals:

Getting Started with the Interactive Data Definition Utility (IDDU) (GC21-8003)

Using System/36 Communications (SC21-9082)

ICF Reference Manual (SC21-7910)

Guide and Examples (SC21-7911)

IDDU online information also provides help.

Example of Using the Interactive Data Definition Utility (IDDU)

The following example shows how to associate an IDDU-defined communication format file with a WORKSTN file, in order to use the confirm function of the APPC subsystem.

```
01 H     0641
```

The ID continuation-line option must be specified when both IDDU communication and SFGR screen formats are included in a WORKSTN file.

FMTS indicates that SFGR screen formats are also included in the WORKSTN file.

SCREEN is the load member that holds all SFGR screen formats used for input and output with a display screen.

```
02 FWSFILE  CD         256              WORKSTN
03 F                                                KNUM           2
04 F                                                KFMTS   SCREEN
05 F                                                KID     WSID
06 F                                                KINFDS  INFODS
07 F                                                KCFILE  COMM
08 FPRTFILE O    F 132  132                PRINTER
09 F
```

The CFILE continuation-line option associates the WORKSTN file with an IDDU-defined communication format file.

COMM is the IDDU format file that holds all communication formats used during a communications session.

INFODS is that data structure that holds the return code information.

Line	Form Type	Filename or Record Name / Data Structure Name	Sequence	Record Identifying Indicator, or DS	Position (1)	C/Z/D	Character	Field Location From / Occurs n Times	To / Length	Decimal Positions	RPG Field Name
01	I	WSFILE	NS	11	79	C	0				
02	I							7Ø	75	Ø	QTY1
03	I	INFODS		DS							
04	I							*STATUS			STATUS
05	I							23	24		MAJCOD
06	I							25	26		MINCOD
07	I										

FIELDA, FIELDB, and FIELDC, are numeric fields to be transmitted using the IDDU-defined communication formats and displayed using the SFGR screen formats.

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	High	Low	Equal	Comments
01	C				Z-ADD	888888	QTY1	6	Ø					
02	C			QTY1	SUB	238Ø5	QTY2	6	Ø					
03	C			QTY2	SUB	1	QTY3	6	Ø					
04	C	*												
05	C			QTY1	MULT	QTY2	FIELDA	15	1					
06	C			FIELDA	DIV	QTY3	FIELDB	15	1	H				
07	C			QTY3	ADD	QTY1	FIELDC	15	3					
08	C				MOVE	'1S'	WSID							
09	C	*												
10	C				EXCPTCOMM									
11	C	*												
12	C				MOVE	'W1'	WSID							
13	C				EXCPTDISPL									
14	C	*												
15	C				READ	WSFILE						95	95	
16	C		95		SETON						LR			
17	C		11		SETON						LR			
18	C													

The '1S' session ID makes RPG look for the IDDU-defined communication format in COMM.

The 'W1' session ID makes RPG look for the SFGR screen format in SCREEN.

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Fetch (F)	Space Before	Skip Before	Skip After	Output Indicators	Field Name or EXCPT Name	End Position in Output Record	Constant or Edit Word
01	O	WSFILE	E						COMM		
02	O									K5	'FORM1'
03	O								FIELDA	19	
04	O								FIELDB	49	
05	O								FIELDC	79	
06	O		E						DISPL		
07	O									K5	'SCRN1'
08	O								FIELDA	19	
09	O								FIELDB	39	
10	O								FIELDC	59	
11	O									71	'THIS WILL BE'
12	O									80	'DISPLAYED'
13	O	PRTFILE	T	F	3	04	19	LR			
14	O									45	'THIS PROG TRYOUT'
15	O									60	'HAS'
16	O							95		70	'FAILED'
17	O							11		70	'PASSED'
18	O							95	MAJCOD	75	
19	O							95	MINCOD	77	
20	O										

SAMPLE PROGRAMS

AR230R (Inquiring into an Accounts Receivable File)

Figure 6-11 shows sample program AR230R, an accounts receivable inquiry program. This program uses two displays, AR230RD1 and AR230RD2. The manual *Creating Displays* explains how to create displays using SDA.

The first display, which prompts the person using the display station to enter a customer number, looks like this:

```
                         Customer Inquiry

Please enter customer number.

Customer number

Press the Enter key to see accounts receivable information.









Press Cmd3 to return to the main menu.
```

If the display read into the program contains an A in the first position of the record, the CUSNO field from the CUSTNMBR record is used to chain to the CUSTMAST file. If the customer number entered is equal to a customer number in the CUSNO field in the CUSTMAST file, the second display is shown. In all other cases, the first display is shown again with the error message CUSTOMER NUMBER DOES NOT EXIST. ENTER A DIFFERENT CUSTOMER NUMBER.

The second display looks like this:

```
                          Customer Inquiry


Customer number  ********

Customer name    *************************
Address          *************************
                 *************************
                 *************************
State and zip    ** *********


Current amount due       ************   Credit limit       ********
Amount due over 30 days  ************
Amount due over 60 days  ************   Last amount paid   ***********
Amount due over 90 days  ************   Last date paid     ********


Press the Enter key to continue.
```

It shows the customer number, name, address, amount due, credit limit, amount due more than 30, 60, and 90 days, and the date of last payment. The person using the display station cannot update this information in this program.

```
01 H
02 FCUSTNMBRCP  F     256            WORKSTN
03 FCUSTMASTIC  F     256R 8AI     2 DISK
```

Name = AR230R

Function = Inquires into an accounts receivable file. This program displays the customer number, name, address, amount due, credit limit, amount due more than 30, 60, and 90 days, and date of last payment for the customer requested.

Input = From disk, the customer master file, CUSTMAST. From a single display station, the customer number.

You can press command key 3 to end this program.

Output = The requested record from the customer master file is displayed. The record cannot be updated in this program.

Indicator Definitions:

02 = Record ID for the blank read for the WORKSTN file.
03 = Record ID for format AR230RD1.
04 = Record ID for format AR230RD2.
10 = Record ID for CUSTMAST file.
99 = Error indicator (Turns on if the customer number requested does not exist in the CUSTMAST file.)
KC = Command key 3 ends the program.

Running This Program

To run AR230R, code the following procedure:

```
// LOAD AR230R
// FILE NAME-CUSTMAST
// RUN
```

Figure 6-11 (Part 1 of 3). Sample Program AR230R (Inquiring Into An Accounts Receivable File)

Line	Form Type	Filename or Record Name (Data Structure Name)	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, or DS	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 (Occurs n Times)	To (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	CUSTNMBR	NS			02	1		C																					
02	I		NS			03	1		C	A																				
03	I																				2	9		CUSNO						
04	I		NS			04	1		C	B																				
05	I	CUSTMAST	NS			10																								
06	I																				2	9		CUSNO						
07	I																				10	34		CUSNM						
08	I																				35	59		CUSA1						
09	I																				60	84		CUSA2						
10	I																				85	109		CUSA3						
11	I																				110	111		STATE						
12	I																			P	112	116	0	ZIPCD						
13	I																			P	131	135	2	AMDUE						
14	I																			P	124	127	0	CRLIM						
15	I																			P	155	159	2	OVR30						
16	I																			P	160	164	2	OVR60						
17	I																			P	140	144	2	LSTAP						
18	I																			P	165	169	2	OVR90						
19	I																			P	136	139	0	DLTPM						
20	I																													

Figure 6-11 (Part 2 of 3). Sample Program AR230R (Inquiring Into An Accounts Receivable File)

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust	Resulting Indicators High	Low	Equal	Comments
01	C				SETOF						99			
02	C		03	CUSNO	CHAIN	CUSTMAST					99			
03	C													

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Output Indicators	Field Name or EXCPT Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	Constant or Edit Word
01	O	CUSTNMBR	D	02NKC					
02	O	OR		99NKC					
03	O	OR		04NKC					
04	O							K8	'AR230RD1'
05	O				CUSNO			8	
06	O			99					'CUSTOMER NUMBER DOES NOT'
07	O			99					' EXIST.'
08	O			99					'ENTER A DIFFERENT CUSTOM'
09	O			99					'ER NUMBER.'
10	O		D	03N99NKC					
11	O							K8	'AR230RD2'
12	O				CUSNO			8	
13	O				CUSNM			33	
14	O				CUSA1			58	
15	O				CUSA2			83	
16	O				CUSA3			108	
17	O				STATE			110	
18	O				ZIPCD			119	
19	O				AMDUE	L		131	'$'
20	O				CRLIM	3		139	'$'
	O				OVR30	L		151	'$'
	O				OVR60	L		163	'$'
	O				LSTAP	3		174	'$'
	O				OVR90	L		186	'$'
	O				DLTPM	Y		194	

*Number of sheets per pad may vary slightly.

Figure 6-11 (Part 3 of 3). Sample Program AR230R (Inquiring Into An Accounts Receivable File)

AR330R (Maintaining a Customer Master File)

Figure 6-12 shows sample program AR330R, which maintains a customer master file. This is a MRT program that allows two requesting display stations. The first display looks like this:

```
                  Maintain Customer Master File

    Date ********

      Enter a customer number . . . . . . . . . . . ********

      Choose an option. . . . . . . . . . . . . A,C,R *
                      A = Add a customer record
                      C = Change a customer record
                      R = Remove a customer record









  Cmd7 = End program
```

It prompts the person to enter a customer number and an A, C, or R to indicate whether he wants to add, change, or remove a record from the customer master file CUSTMAST.

If the person enters an invalid number, one of several error messages is displayed.

If the customer number is valid, the program shows one of three displays that allows the person to add, change, or remove a record from the customer master file, depending on whether the person entered A, C, or R on the first display.

The display to add a customer record looks like this:

```
                    Add a Customer Record

   Date ********

     Customer number . . . . . . . . ********

     Customer name . . . . . . . . .
     Customer address. . . . . . . .


     State and Zip code. . . . . . .
     Phone number. . . . . . . . . .

     Customer type . . . . . . . . .
     Credit limit. . . . . . . . . .

     Salesman number . . . . . . . .




 Cmd7 = End program     Cmd3 = Do not add this customer record
```

The display to change a customer record looks like this:

```
                 Change a Customer Record

   Date ********

     Customer number . . . . . . . . ********

     Customer name . . . . . . . . . *************************
     Customer address. . . . . . . . *************************
                                     *************************
                                     *************************
     State and Zip code. . . . . . . **  *********
     Phone number. . . . . . . . . . ***  *******

     Customer type . . . . . . . . . *
     Credit limit. . . . . . . . . . *******

     Salesman number . . . . . . . . *****




 Cmd7 = End program     Cmd3 = Do not change this customer record
```

The display to remove a customer record looks like this:

```
                    Remove a Customer Record

 Date ********

   Customer number . . . . . . . . *********

   Customer name . . . . . . . . . **************************
   Customer address. . . . . . . . **************************
                                   **************************
                                   **************************
   State and Zip code. . . . . . . **  *********
   Phone number. . . . . . . . . . ***  ******

   Customer type . . . . . . . . . *
   Credit limit. . . . . . . . . . *******

   Salesman number . . . . . . . . *****


   Press enter to remove this customer record.

 Cmd7 = End program     Cmd3 = Do not remove this customer record
```

```
01H          14
02FWORKSTN CP        256            WORKSTN
03F                                                 KNUM        2
04F                                                 KSAVDS SAVEDS
05FCUSTMASTUC  F 256 256R 8AI      2 DISK                           A
```

Name = AR330R

Function = Maintains a customer master file. This program can add new customer records, change existing customer records, and remove customer records.

Input = From disk, the customer master file, CUSTMAST. From one or two display stations, the customer number and one of the following options for that customer number:

A = Add a new customer record. A second display is shown, on which you can enter the customer's name, address, etc.

C = Change an existing customer record. A second display is shown, on which you can change the customer's name, address, etc.

R = Remove an existing customer record. A second display shows the record to be deleted, so you can verify that the correct customer number was entered.

From any display, you can press command key 7 to end this program. From any of the second displays, you can press command key 3 to show the first display again and not add, change, or remove the customer record.

Output = Customer records are added to, changed in, or removed from the file CUSTMAST.

Figure 6-12 (Part 1 of 10). Sample Program AR330R (Maintaining a Customer Master File)

Indicator Definitions:

01 = Record ID for the blank read for the WORKSTN file.
02 = Record ID for format AR330RD1.
03 = Record ID for format AR330RD2.
04 = Record ID for format AR330RD3.
05 = Record ID for format AR330RD4.
08 = First-time switch for code needed only for the first cycle.
10 = Turns on if option A is chosen on format AR330RD1 and is used to display format AR330RD2.
20 = Turns on if option C is chosen on format AR330RD1 and is used to display format AR330RD3.
30 = Turns on if option R is chosen on format AR330RD1 and is used to display format AR330RD4.
50 = Turns on if the customer record is not found in CUSTMAST.
60 = Record ID for records read from CUSTMAST.
90 = Error (The CUSNO field read from format AR330RD1 is blank.)
91 = Error (The option chosen on AR330RD1 was not A, C, or R.)
92 = Error (The add option was chosen, but customer number is already in the CUSTMAST file.)
93 = Error (The change option was chosen, but customer number is not in the CUSTMAST file.)
94 = Error (The remove option was chosen, but the customer number is not in the CUSTMAST file.)
95 = Error (When the add option was chosen, the customer number was not in the CUSTMAST file. But after format AR330RD2 was displayed, another person added the customer record. The record is not added now.)
96 = Error (When the change option was chosen, the customer number was in the CUSTMAST file. But after format AR330RD3 was displayed, another person deleted the customer record. The record is not changed now.)
97 = Error (When the remove option was chosen, the customer was in the CUSTMAST file. But after format AR330RD4 was displayed, another person deleted the customer record. The record is not removed now.)
99 = Error indicator for formats (Turns on any time one of the error indicators 90-97 turns on.)
KD = Command key 3 displays format AR330RD1 again after format AR330RD2, AR330RD3, or AR330RD4 is displayed.
KG = Command key 7 ends the program.

Running This Program

To run AR330R, code the following procedure:

```
// LOAD AR330R
// FILE NAME-CUSTMAST
// RUN
```

Figure 6-12 (Part 2 of 10). Sample Program AR330R (Maintaining a Customer Master File)

Line	Form Type	Filename or Record Name	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, **, or DS	Position (1)	Not (N) (1)	C/Z/D (1)	Character (1)	Position (2)	Not (N) (2)	C/Z/D (2)	Character (2)	Position (3)	Not (N) (3)	C/Z/D (3)	Character (3)	Stacker Select	P/B/L/R	Field Location From	Field Location To	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	WORKSTN	NS			Ø1	1		C																					
02	I		NS			Ø2	1		C	A																				
03	I																				2	9		CUSNO						
04	I																				1Ø	1Ø		OPTION						
05	I		NS			Ø3	1		C	B																				
06	I																				2	26		CUSNMW						
07	I																				27	51		CUSA1W						
08	I																				52	76		CUSA2W						
09	I																				77	1Ø1		CUSA3W						
10	I																				1Ø2	1Ø3		STATEW						
11	I																				1Ø4	112	Ø	ZIPCDW						
12	I																				113	115	Ø	AREACW						
13	I																				116	122	Ø	PHONEW						
14	I																				123	123		CUSTPW						
15	I																				124	13Ø	Ø	CRLIMW						
16	I																				131	135	Ø	SLSNOW						
17	I		NS			Ø4	1		C	C																				
18	I																				2	26		CUSNMW						
19	I																				27	51		CUSA1W						
20	I																				52	76		CUSA2W						
	I																				77	1Ø1		CUSA3W						
	I																				1Ø2	1Ø3		STATEW						
	I																				1Ø4	112	Ø	ZIPCDW						
	I																				113	115	Ø	AREACW						
	I																				116	122	Ø	PHONEW						
01	I																				123	123		CUSTPW						
02	I																				124	13Ø	Ø	CRLIMW						
03	I																				131	135	Ø	SLSNOW						
04	I		NS			Ø5	1		C	D																				
05	I	CUSTMAST	NS			6Ø	1		C																					
06	I																				1Ø	34		CUSNM						
07	I																				35	59		CUSA1						
08	I																				6Ø	84		CUSA2						
09	I																				85	1Ø9		CUSA3						
10	I																				11Ø	111		STATE						
11	I																			P	112	116	Ø	ZIPCD						
12	I																			P	117	118	Ø	AREAC						
13	I																			P	119	122	Ø	PHONE						
14	I																				123	123		CUSTP						
15	I																			P	124	127	Ø	CRLIM						
16	I																			P	128	13Ø	Ø	SLSNO						
17	I																													

Figure 6-12 (Part 3 of 10). Sample Program AR330R (Maintaining a Customer Master File)

Line	Form Type	Filename or Record Name / Data Structure Name	Record Identifying Indicator, **, or DS	From (Occurs n Times)	To (Length)	Decimal Positions	RPG Field Name
01	I						
02	I						
03	I						
04	I						
05	I						
06	I						
07	I						
08	I	SAVEDS	DS				
09	I			1	8		CUSNO
10	I						
11	I						
12	I						
13	I						
14	I						
15	I						
16	I						
17	I						
18	I		DS				
19	I			1	1		ARCOD
20	I			2	10	2	AMDUE
	I			2	10	2	LSTAP
	I			2	10	2	PRBAL
	I			2	10	2	CHGTD
	I			2	10	2	OVR30
	I			2	10	2	OVR60
01	I			2	10	2	OVR90
02	I			2	10	2	CRDTD
03	I			2	10	2	ADJTD
04	I			2	10	2	SLSLY
05	I			11	16	0	DLTPM
06	I			11	16	0	DLTOR
07	I			17	23	0	RRFST
08	I			17	23	0	RRLST
09	I			24	79		BUFFR
10	I						

The save data structure is used to save the customer number for each display.

This data structure is used to initialize all the fields not used by this program when a record is added to file CUSTMAST.

Figure 6-12 (Part 4 of 10). Sample Program AR330R (Maintaining a Customer Master File)

Initialize the fields used for adding a new customer record.

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust	High	Low	Equal	Comments
01	C													
02	C													
03	C													
04	C													
05	C													
06	C													
07	C		NØ8		Z-ADD	*ZERO	AMDUE							
08	C		NØ8		Z-ADD	*ZERO	DLTPM							
09	C		NØ8		SETON						Ø8			
10	C													
11	C													
12	C													
13	C													
14	C													
15	C													
16	C													
17	C				SETOF						1Ø	2Ø	3Ø	
18	C				SETOF						9Ø	91	92	
19	C				SETOF						93	94	95	
20	C				SETOF						96	97	99	

These indicators turn on during each cycle as they are needed.

If command key 3 is pressed, skip calculations and display format AR330RD1. If command key 7 is pressed, skip calculations and release this display station.

If format AR330RD1 was read, indicator 02 is on. If the CUSNO field is blank, display a message to ask for a nonblank customer number.

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust	High	Low	Equal	Comments
01	C													
02	C													
03	C													
04	C													
05	C		KG											
06	C	OR	KG		GOTO	ENDCAL								
07	C													
08	C													
09	C													
10	C													
11	C													
12	C													
13	C													
14	C													
15	C													
16	C		Ø2	CUSNO	COMP	*BLANK							9Ø	
17	C		Ø2 9Ø		SETON						99			
18	C													

Figure 6-12 (Part 5 of 10). Sample Program AR330R (Maintaining a Customer Master File)

If format AR330RD1 was read, indicator 02 is on. If the option chosen is not A, C, or R, indicators 90 and 99 turn on to display format AR330RD1 again and to display an error message.

```
09 C   02N99   OPTION    COMP 'A'                     10
10 C   02N99   OPTION    COMP 'C'                     20
11 C   02N99   OPTION    COMP 'R'                     30
12 C   02N99N10
13 CANN20N30             SETON                     9199
```

If a valid option was chosen on format AR330RD1 and the CUSNO field is not blank, check for the following errors:

Option A – Display error message if customer number is found.
Option C – Display error message if customer number is not found.
Option R – Display error message if customer number is not found.

```
   C   02N99   CUSNO     CHAINCUSTMAST             50
01 C   02 10N50          SETON                     9299
02 C   02 20 50          SETON                     9399
03 C   02 30 50          SETON                     9499
```

If format AR330RD2 is read, add the customer record to the file. If the customer record is found when the add is attempted, display an error message.

```
13 C   03      CUSNO     CHAINCUSTMAST             50
14 C   03 50             EXCPTADD
15 C   03 50             SETON                     01
16 C   03N50             SETON                     9599
```

Figure 6-12 (Part 6 of 10). **Sample Program AR330R (Maintaining a Customer Master File)**

Line	Form Type	Control Level	Indicators (Not)	Indicators	Indicators (Not)	Indicators	Indicators (Not)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	High	Low	Equal	Comments
01	C																		
02	C																		
03	C																		
04	C																		
05	C																		
06	C																		
07	C																		
08	C																		

If format AR330RD3 is read, change the customer record. If the customer's record is not found when the change is attempted, display an error message.

Line	Form Type	Control Level	Indicators (Not)	Indicators	Indicators (Not)	Indicators	Indicators (Not)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	High	Low	Equal	Comments
09	C			Ø4					CUSNO	CHAIN	CUSTMAST					5Ø			
10	C			Ø4	N	5Ø				EXCPT	UPDATE								
11	C			Ø4	N	5Ø				SETON						Ø1			
12	C			Ø4		5Ø				SETON						96	99		
13	C																		
14	C																		
15	C																		
16	C																		
17	C																		
18	C																		
19	C																		
20	C																		

If format AR330RD4 is read, remove the customer record. If the customer's record is not found when the delete is attempted, display an error message.

Line	Form Type	Control Level	Indicators (Not)	Indicators	Indicators (Not)	Indicators	Indicators (Not)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	High	Low	Equal	Comments
	C																		
	C			Ø5					CUSNO	CHAIN	CUSTMAST					5Ø			
	C			Ø5	N	5Ø				EXCPT	DELETE								
	C			Ø5	N	5Ø				SETON						Ø1			
	C			Ø5		5Ø				SETON						97	99		
01	C								ENDCAL	TAG									
02	C																		

Figure 6-12 (Part 7 of 10). Sample Program AR330R (Maintaining a Customer Master File)

Line	Form Type	Filename or Record Name	Type / AND/OR	Output Indicators	Field Name or EXCPT Name	Edit Codes	End Position in Output Record	Constant or Edit Word
01	O	WORKSTN	D	Ø1				
02	O		OR	99				
03	O		OR	KC				
04	O						K8	'AR33ØRD1
05	O				UDATE	Y	8	
06	O			99	CUSNO		16	
07	O			99	OPTION		17	
08	O			99N91			36	'THE CUSTOMER NUMBER'
09	O			9Ø			6Ø	' CANNOT BE BLANK. ENTER'
10	O			9Ø			79	' A CUSTOMER NUMBER.'
11	O			91			41	'CHOOSE OPTION A, C, OR R'
12	O			91			42	'.'
13	O			92			6Ø	' YOU ARE TRYING TO ADD I'
14	O			92			82	'S ALREADY IN THE FILE.'
15	O			93			6Ø	' YOU ARE TRYING TO CHANG'
16	O			93			81	'E IS NOT IN THE FILE.'
17	O			94			6Ø	' YOU ARE TRYING TO REMOV'
18	O			94			81	'E IS NOT IN THE FILE.'
19	O			95			6Ø	' WAS ADDED BY SOMEONE EL'
20	O			95			84	'SE AFTER YOU CHOSE THE A'
	O			95			94	'DD OPTION.'
	O			96			6Ø	' WAS REMOVED BY SOMEONE '
	O			96			84	'AFTER YOU CHOSE THE CHAN'
	O			96			94	'GE OPTION.'
	O			97			6Ø	' WAS REMOVED BY SOMEONE '
01	O			97			84	'AFTER YOU CHOSE THE REMO'
02	O			97			94	'VE OPTION.'
03	O		D	1ØN99				
04	O						K8	'AR33ØRD2'
05	O				UDATE	Y	8	
06	O				CUSNO		16	
07	O		D	2ØN99				
08	O						K8	'AR33ØRD3'
09	O				UDATE	Y	8	
10	O				CUSNO		16	
11	O				CUSNM		41	
12	O				CUSA1		66	
13	O				CUSA2		91	
14	O				CUSA3		116	
15	O				STATE		118	
16	O				ZIPCD		127	
17	O				AREAC		13Ø	
18	O				PHONE		137	
19	O				CUSTP		138	
20	O				CRLIM		145	
	O				SLSNO		15Ø	
	O							

Figure 6-12 (Part 8 of 10). Sample Program AR330R (Maintaining a Customer Master File)

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)	Output Indicators	Field Name or EXCPT Name	Edit Codes	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O		D	30 N99					
02	O						K8		'AR330RD4'
03	O				UDATE	Y	8		
04	O				CUSNO		16		
05	O				CUSNM		41		
06	O				CUSA1		66		
07	O				CUSA2		91		
08	O				CUSA3		116		
09	O				STATE		118		
10	O				ZIPCD		127		
11	O				AREAC		130		
12	O				PHONE		137		
13	O				CUSTP		138		
14	O				CRLIM		145		
15	O				SLSNO		150		
16	O								
17	O								
18	O								
19	O								
20	O								

Release the display station at which command key 7 was pressed.

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Output Indicators	Field Name or EXCPT Name	Edit Codes	End Position in Output Record	P/B/L/R	Constant or Edit Word
	O								
	O								
	O		DR	KG					
	O								
	O								
01	O								
02	O								
03	O								
04	O								
05	O								
06	O	CUSTMASTE			UPDATE				
07	O				CUSNMW		34		
08	O				CUSA1W		59		
09	O				CUSA2W		84		
10	O				CUSA3W		109		
11	O				STATEW		111		
12	O				ZIPCDW		116	P	
13	O				AREACW		118	P	
14	O				PHONEW		122	P	
15	O				CUSTPW		123		
16	O				CRLIMW		127	P	
17	O				SLSNOW		130	P	
18	O				UDATE		192	P	
19	O								

Change the customer record.

Figure 6-12 (Part 9 of 10). Sample Program AR330R (Maintaining a Customer Master File)

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Field Name or EXCPT Name	End Position in Output Record	P/B/L/R
01	O					
02	O					
03	O					
04	O					
05	O					
06	O					
07	O					
08	O		E ADD	ADD		
09	O			ARCOD	1	
10	O			CUSNO	9	
11	O			CUSNMW	34	
12	O			CUSA1W	59	
13	O			CUSA2W	84	
14	O			CUSA3W	1Ø9	
15	O			STATEW	111	
16	O			ZIPCDW	116	P
17	O			AREACW	118	P
18	O			PHONEW	122	P
19	O			CUSTPW	123	
20	O			CRLIMW	127	P
	O			SLSNOW	13Ø	P
	O			AMDUE	135	P
	O			DLTPM	139	P
	O			LSTAP	144	P
	O			PRBAL	149	P
01	O			CHGTD	154	P
02	O			OVR3Ø	159	P
03	O			OVR6Ø	164	P
04	O			OVR9Ø	169	P
05	O			CRDTD	174	P
06	O			ADJTD	179	P
07	O			SLSLY	184	P
08	O			DTLOR	188	P
09	O			UDATE	192	P
10	O			RRFST	196	P
11	O			RRLST	2ØØ	P
12	O			BUFFR	256	
13	O					
14	O					
15	O					
16	O					
17	O					
18	O					
19	O					
20	O		E DEL	DELETE		
	O					

Add the customer record. All fields in this record are written to initialize them properly.

Delete the customer record.

Figure 6-12 (Part 10 of 10). Sample Program AR330R (Maintaining a Customer Master File)

AR935R (Requesting a Printout of Accounts Receivable)

Figure 6-13 shows sample program AR935R, which allows the person using the display station to choose the type of accounts receivable information to be printed by sample program AR936R, shown in Chapter 7.

The first display looks like this:

```
              Print aged trial balance report.

     Choose report option . . . . 1,2,3,4 *

              1 = All customers.
              2 = Customers with balances.
              3 = Customers with balances over a certain amount.
              4 = Customers with overdue balances.








     Cmd7 = End program
```

The person enters a 1, 2, 3, or 4 to request a printed report for all customers, for customers with balances, for customers with balances over an amount that the person enters on the second display, or for customers with overdue balances.

If the person enters 3 on the first display, the second display is shown. The second display looks like this:

```
Print report for balances over what whole
  dollar amount? . . . . . . . . . . . . . . . $         .00



Cmd7 = End program      Cmd3 = Choose a different report option
```

The information that the person enters is written to the display station local data area (the data structure coded on line 07 of the input specifications). The information in this local data area is read by program AR936R, which prints the report requested.

```
01 H           14
02 FWORKSTN CP         256            WORKSTN
```

Name = AR935R

Function = Allows you to choose a type of aged trial balance report and passes that choice to AR936R in the local data area.

Input = Your choice of which type of report to print. The options are:

1 = All customers
2 = All customers with balances
3 = All customers with balances over the amount entered in field OVBAL
4 = All customers with overdue balances

Output = The following information is written to the local data area:

Position	Information
1	= Report option
2-8	= OVBAL if option 3 is chosen
9	= C if command key 7 is pressed. This tells the procedure not to run AR936R.

Indicator Definition:

01 = Record ID for format AR935D1
02 = Record ID for format AR935D2
03 = Checks for a valid option
04 = Checks for a valid option
05 = Conditions first-time processing
30 = Turns on if option 3 is chosen and displays format AR935RD2
99 = Turns on if the option chosen was not 1, 2, 3, or 4 and displays an error message
KC = Command key 3. Displays format AR935RD1 again when format AR935RD2 has been displayed.
KG = Command key 7. Cancels the report request and ends the program.
LR = Turns on when the last record is processed. The program goes to end of job.

Figure 6-13 (Part 1 of 4). Sample Program AR935R (Requesting a Printout of Accounts Receivable)

Running This Program:

AR935R must be run before AR936R. To run both AR935R and AR936R, code the following procedure:

```
// LOAD AR935R
// WORKSTN UNIT-?WS?,RESTORE-YES
// RUN
*  IF COMMAND KEY 7 IS PRESSED IN AR935R, A NONBLANK CHARACTER
*    IS PLACED IN POSITION 9 OF THE LOCAL DATA AREA AND AR936R
*    IS NOT RUN.
// IFF ?L'9,1'?/ GOTO NOPRT
// LOAD AR936R
// FILE NAME-CUSTMAST
// RUN
// TAG NOPRT
```

Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator, or DS	Position	C/Z/D	Character	From	To	RPG Field Name
01	I	WORKSTN	NS	Ø1	1	C	A			
02	I							2	2	OPTION
03	I		NS	Ø2	1	C	B			
04	I							2	8Ø	OVBAL
05	I		NS	Ø5						
06	I									
07	I									
08	I									
09	I									
10	I									
11	I									
12	I									
13	I									
14	I			UDS						
15	I							1	1	OPTION
16	I							2	8Ø	OVBAL
17	I							9	9	CMD7
18	I							1	9	CLEAR
19	I									

This is the local data area information passed from AR935R to AR936R.

Figure 6-13 (Part 2 of 4). Sample Program AR935R (Requesting a Printout of Accounts Receivable)

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators High (Plus, 1>2)	Low (Minus, 1<2)	Equal (Zero, 1=2)	Comments
01	C													
02	C													
03	C													
04	C													
05	C													
06	C													
07	C		Ø5		MOVE	*BLANKS	CLEAR							
08	C													
09	C													
10	C													
11	C													
12	C													
13	C													
14	C													
15	C													
16	C				SETOF						3Ø	99		
17	C		KG		MOVE	'C'	CMD7							
18	C		KG		GOTO	END								
19	C													
20	C													

First-time logic to clear the local data area.

If command key 7 is pressed, move 'C' to field CMD 7, which is placed in the local data area.

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators High (Plus, 1>2)	Low (Minus, 1<2)	Equal (Zero, 1=2)	Comments
	C													
	C													
	C													
	C													
	C													
01	C													
02	C													
03	C		Ø1	OPTION	COMP	'1'					Ø3		Ø3	
04	C		Ø1	OPTION	COMP	'4'						Ø4	Ø4	
05	C		Ø1NØ3											
06	C	OR	Ø1NØ4		SETON						99			
07	C													
08	C													
09	C													
10	C													
11	C													
12	C													
13	C													
14	C													
15	C													
16	C		Ø1N99	OPTION	COMP	'3'							3Ø	
17	C													

If the option chosen is not 1, 2, 3, or 4, indicator 99 turns on to display an error message and allows you to choose another option.

If option 3 is chosen, indicator 30 turns on and displays format AR935RD2, which allows you to enter the minimum balance for customers to be included in the report.

Figure 6-13 (Part 3 of 4). Sample Program AR935R (Requesting a Printout of Accounts Receivable)

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators (High / Low / Equal)	Comments
01	C											
02	C											
03	C											
04	C											
05	C											
06	C											
07	C											
08	C											
09	C			END	TAG							
10	C		Ø1N3ØN99									
11	C	OR	Ø2NKC		SETON						LR	
12	C											

If option 1, 2, or 4 is chosen or if option 3 is chosen and a value is entered in OVBAL, display format AR935RD3 and go to end of job. If command key 7 is pressed, go to end of job.

Line	Form Type	Filename or Record Name	Type (H/D/T/E) / AND / OR	Space Before	Space After	Skip Before	Skip After	Output Indicators	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	WORKSTN	D					Ø5						
02	O		OR					99						
03	O		OR					KC						
04	O											K8		'AR935RD1'
05	O							99	OPTION			1		
06	O							99				25		'CHOOSE OPTION 1, 2, 3, O'
07	O							99				29		'R 4.'
08	O							99				53		'OR PRESS CMD7 IF YOU DO '
09	O							99				71		'NOT WANT A REPORT.'
10	O		D					3Ø						
11	O											K8		'AR935RD2'
12	O		D					LRNKG						
13	O											K8		'AR935RD3'
14	O													

Figure 6-13 (Part 4 of 4). Sample Program AR935R (Requesting a Printout of Accounts Receivable)

OE140R (Entering Orders from Customers)

Figure 6-14 shows sample program OE140R, which allows a person to enter customer orders at a display station. The person enters a customer number and an order number on the first display:

```
                         Order Entry


          Enter the customer number. . . .

          Enter the order number . . . . .











Cmd7 = End program
```

On the second display, he enters the item number and the quantity ordered. When this information is entered, the program displays the item number, the quantity ordered, a description of the item, the unit cost, and the total cost for the quantity of that item ordered. Up to four items are displayed on the same display:

```
                         Order Entry

   Customer Number ********              Order Number ******

   Name    **************************
   Address **************************
           **************************
           ************************** ** *********

  Item No.   Qty    Description                       Price        Amount
  ********  ******  ******************************  ***********  ************
  ********  ******  ******************************  ***********  ************
  ********  ******  ******************************  ***********  ************
  ********  ******  ******************************  ***********  ************


 Enter Item Number. . . ********
 Enter Quantity . . . . ******


 Cmd2 = End of Order   Roll = Roll through line items
```

The *STATUS keyword for the file information data structure INFDS (line 25) and the exception/error-processing subroutine INFSR (line 67) allow the person to use the Roll Up and Roll Down keys to display additional orders.

```
  01 H     064
```

```
  02 FCUSTMASTIC  F 256 256R 8AI     2 DISK
  03 FITEMMSTRIC  F 128 128R 8AI     2 DISK
  04 FCUSTORDSO   F 128 128            DISK                      A
  05 FWORKSTN CP        500            WORKSTN
  06 F                                              KINFDS DSINF
  07 F                                              KINFSR SUBINF
```

Name = OE140R

Function = Allows the user to enter customer orders.

Input = The customer number and the order number are entered on display OE140RD1. The item number and the quantity are entered on display OE140RD2. After the item number and quantity have been entered, they are displayed with a description, price, and total price for that item. The roll keys can be pressed to roll through the orders and make corrections. Press command key 2 to have the order written to the file CUSTORDS. If the first array entry is blank (no items entered), then no order is written to the file CUSTORDS. Press command key 7 on display OE140RD1 to end the program.

Output = The order entered is written to the file CUSTORDS. The following records are written:

1) Customer record
2) Ship-to record
3) Line-item record (one for each line item entered)

Figure 6-14 (Part 1 of 12). Sample Program OE140R (Entering Orders from Customers)

Indicator Definitions:

01 = Record ID for the blank read for the WORKSTN file.
02 = Record ID for format OE140RD1.
03 = Record ID for format OE140RD2.
04 = Is turned on by the information subroutine, passes control to the beginning of detail calculations, and skips to the end of detail calculations.
11 = Turns off when no line items have been entered and is used to display array line one.
12 = Turns off when only one item has been entered and is used to display array line two.
13 = Turns off when only two items have been entered and is used to display array line three.
14 = Turns off when only three items have been entered and is used to display array line four.
20 = Used in subroutines UPINDX and DNINDX to determine when 1 is to be added to or subtracted from the array indexes A, B, C, and D. The indexes are not changed if I is less than 4.
25 = Used in subroutines ROLLUP and ROLLDN to indicate that a roll cannot be done because the top or bottom of an array has been reached.
60 = Turns on when the Roll Up key is pressed.
61 = Turns on when the Roll Down key is pressed.
70 = Used in the subroutine ADDORD to determine when all line items are written to the file CUSTORDS.
71 = Used in subroutine ADDORD to write the customer record and the ship-to record to the file CUSTORDS for each order.
90 = Error (The customer number entered on format OE140RD1 was blank. It cannot be blank.)
91 = Error (The order number entered on format OE140RD1 was blank. It cannot be blank.)
92 = Error (The customer number entered on format OE140RD1 was not found in the file CUSTMAST.)
93 = Error (The item number entered on format OE140RD2 was blank. It cannot be blank.)
94 = Error (The quantity entered on format OE140RD2 was zero. It cannot be zero.)
95 = Error (The item number entered on format OE140RD2 was not found in the file ITEMMSTR.)
96 = Error (The operator tried to enter more than 98 items for one order on format OE140RD2.)
99 = Error indicator for formats (Turns on any time one of the error indicators 90-96 turns on.)
KB = Command key 2 writes the order placed in the file CUSTORDS.
KG = Command key 7 ends the program.
Roll Up = Used to roll forward through the line items.
Roll Down = Used to roll backwards through the line items.

Figure 6-14 (Part 2 of 12). Sample Program OE140R (Entering Orders from Customers)

Line	Form Type	
02	F	
03	F	Running This Program:
04	F	
05	F	To run this program, the file CUSTORDS must already have been created. Also code the
06	F	following procedure:
07	F	
08	F	// LOAD OE140R
09	F	// FILE NAME-CUSTMAST
10	F	// FILE NAME-ITEMMSTR
	F	// FILE NAME-CUSTORDS
	F	// RUN
02	F	
03	F	

Line (3-5)	Form Type (6)	Record Sequence of the Chaining File (7-8) / Number of the Chaining Field (9-10) / From Filename (11-18)	To Filename (19-26)	Table or Array Name (27-32)	Number of Entries Per Record (33-35)	Number of Entries Per Table or Array (36-39)	Length of Entry (40-42)	P/B/L/R (43)	Decimal Positions (44)	Sequence (A/D) (45)	Table or Array Name (Alternating Format) (46-51)	Length of Entry (52-54)	P/B/L/R (55)	Decimal Positions (56)	Sequence (A/D) (57)	Comments (58-74)
01	E			ANBR		99	8									ITEM-NUMBER ARRAY
02	E			AQTY		99	6		Ø							QUANTITY ORDERED
03	E	*ARRAY														
04	E			ADSC		99	3Ø									ITEM DESCRIPTION
05	E	*ARRAY														
06	E			APRI		99	9		2							ITEM-PRICE ARRAY
07	E			AEXA		99	9		2							TOTAL-PRICE ARRAY
08	E															

Figure 6-14 (Part 3 of 12). Sample Program OE140R (Entering Orders from Customers)

Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator, or DS	Position (1)	C/Z/D (1)	Character (1)	Position (2)	Character (2)	P/B/L/R	From	To	Decimal Positions	RPG Field Name	Field Indicators (Zero or Blank)
01	I	CUSTMAST	NS												
02	I										1Ø	111		NAMAD	
03	I									P	112	116	Ø	ZIPCD	
04	I	ITEMMSTR	NS												
05	I										1Ø	39		ITDSC	
06	I									P	57	61	2	PRICE	
07	I	WORKSTN	NS	Ø1	1	C									
08	I														
09	I	Format OE140RD1													
10	I														
11	I		NS	Ø2	1	CD		2	C1						
12	I										3	1Ø		CUSNO	9Ø
13	I										11	16		ORDNO	91
14	I	Format OE140RD2													
15	I														
16	I														
17	I		NS	Ø3	1	CD		2	C2						
18	I										3	1Ø		ITNBR	93
19	I										11	16	Ø	QTYOR	94
20	I														
	I	Information data structure													
	I														
	I	DSINF		DS											
	I										*STATUS			STATUS	
	I														

Line	Form Type	Indicators	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators	Comments
01	C	Initialize error indicators to off.						
02	C							
03	C							
04	C			SETOF			929596	
05	C			SETOF			99	
06	C							
07	C							
08	C							
09	C	If command key 2 was pressed on format OE140RD2, add the items to file CUSTORDS.						
10	C							
11	C	KB		EXSR	ADDORD			
12	C							

Figure 6-14 (Part 4 of 12). Sample Program OE140R (Entering Orders from Customers)

If command key 7 was pressed, go to the end of calculations and turn on LR. Indicator 04 is turned on by the information subroutine to indicate that a roll key had been pressed and that detail output should be performed. If command key 2 had been pressed, skip over detail calculations.

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Resulting Indicators High	Low	Equal	Comments
07	C		Ø4										
08	C	OR	KB										
09	C	OR	KG		GOTO	ENDCAL							

If format OE140RD1 was read, indicator 02 is on. If field CUSNO or ORDNO is blank or zero, display an error message. Note: Only one error message can be displayed at a time.

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Resulting Indicators High	Low	Equal	Comments
16	C		Ø2 9Ø										
17	C	OR	Ø2 91		SETON					99			
18	C		Ø2 9Ø		SETOF					91			

If CUSNO and ORDNO were not blank or zero, check to see if the customer number is in the CUSTMAST file. If it is not found, display an error message.

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Resulting Indicators High	Low	Equal	Comments
	C		Ø2N99	CUSNO	CHAIN	CUSTMAST				92			
01	C		Ø2 92		SETON					99			

If the customer number was valid, initialize the array index fields to display format OE140RD2.

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Resulting Indicators High	Low	Equal	Comments
07	C		Ø2N99		Z-ADD	1	A	2	Ø				
08	C		Ø2N99		Z-ADD	2	B	2	Ø				
09	C		Ø2N99		Z-ADD	3	C	2	Ø				
10	C		Ø2N99		Z-ADD	4	D	2	Ø				
11	C		Ø2N99		Z-ADD	Ø	I	2	Ø				
12	C		Ø2N99		Z-ADD	1	J	2	Ø				
13	C		Ø2N99		SETOF					11	12	13	
14	C		Ø2N99		SETOF					14			

Figure 6-14 (Part 5 of 12). Sample Program OE140R (Entering Orders from Customers)

If format OE140RD2 was read, indicator 03 is on. If field ITNBR or QTYOR is blank or zero, display an error message. Note: Only one error message can be displayed at a time.

```
     C   03 93
     COR 03 94             SETON                     99
     C   03 93             SETOF                     94
```

Check to make sure the maximum number of lines has not been entered. If 98 orders have been entered, the operator can enter no more. An error message is displayed.

```
     C   03N99   I         COMP 98                       96
     C   03 96             SETON                     99
```

If ITNBR and QTYOR were not blank or zero, check to see if the item number is in the ITEMMSTR file. If it is not found, display an error message.

```
     C   03N99   ITNBR     CHAINITEMMSTR             95
     C   03 95             SETON                     99
```

If the item number was valid, move the item number, the quantity ordered, the description, and the price (obtained from file ITEMMSTR), and the total cost for the item (product of the quantity and the price) into arrays for display and for later output to CUSTORDS.

```
     C   03N99             ADD  1         I
     C   03N99             ADD  1         J
     C   03N99             MOVE ITNBR     ANBR,I
     C   03N99             Z-ADDQTYOR     AQTY,I
     C   03N99             MOVE ITDSC     ADSC,I
     C   03N99             Z-ADDPRICE     APRI,I
     C   03N99   PRICE     MULT QTYOR     AEXA,I
     C   03N99             MOVE ANBR,J    ITNBR
     C   03N99             Z-ADDAQTY,J    QTYOR
```

Figure 6-14 (Part 6 of 12). Sample Program OE140R (Entering Orders from Customers)

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Plus / 1>2 / High	Minus / 1<2 / Low	Zero / 1=2 / Equal	Comments
01	C													
02	C													

Indicators 11 through 14 display up to four line items on the display.

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust	High	Low	Equal	Comments
03	C													
04	C													
05	C		Ø3N99		EXSR	SETIND								
06	C													
07	C													
08	C													

Add 1 to the index for A, B, C, and D if more than four items have been ordered.

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust	High	Low	Equal	Comments
09	C													
10	C													
11	C		Ø3N99		EXSR	UPINDX								
12	C													
13	C													
14	C													

End of detail calculations.

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust	High	Low	Equal	Comments
15	C													
16	C													
17	C			ENDCAL	TAG									
18	C		KG		SETON						LR			
19	C				SETOF						Ø4			
20	C													

This is the information subroutine. Control comes directly to here when one of the roll keys is pressed. It allows the operator to press the roll keys and roll through the items already entered.

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust	High	Low	Equal	Comments
01	C													
02	C	SR		SUBINF	BEGSR									
03	C	SR		STATUS	COMP	Ø1122							6Ø	
04	C	SR		STATUS	COMP	Ø1123							61	
05	C	SR	6Ø		EXSR	ROLLUP								
06	C	SR	61		EXSR	ROLLDN								
07	C	SR			SETON						Ø3	Ø4		
08	C	SR			ENDSR	'*DETC'								
09	C													

Figure 6-14 (Part 7 of 12). Sample Program OE140R (Entering Orders from Customers)

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators Plus / High 1>2	Minus / Low 1<2	Zero / Equal 1=2	Comments
01	C													
02	C													
03	C													
04	C													
05	C													
06	C	SR		ROLLUP	BEGSR									
07	C	SR		I	COMP	98							25	
08	C	SR	N25		ADD	1	I							
09	C	SR	N25		ADD	1	J							
10	C	SR	N25		MOVE	ANBR,J	ITNBR							
11	C	SR	N25		Z-ADD	AQTY,J	QTYOR							
12	C	SR	N25		EXSR	UPINDX								
13	C	SR	N25		EXSR	SETIND								
14	C	SR			ENDSR									
15	C													
16	C													
17	C													
18	C													
19	C													
20	C													
	C	SR		ROLLDN	BEGSR									
	C	SR		I	COMP	*ZERO							25	
	C	SR	N25		MOVE	ANBR,I	ITNBR							
	C	SR	N25		Z-ADD	AQTY,I	QTYOR							
	C	SR	N25		SUB	1	I							
01	C	SR	N25		SUB	1	J							
02	C	SR	N25		EXSR	DNINDX								
03	C	SR	N25		EXSR	SETIND								
04	C	SR			ENDSR									
05	C													

When the Roll Up function key is pressed, this subroutine moves the content of the arrays (which is two higher than what is displayed) into fields ITNBR and QTYOR so the operator can change that item or get back to the top of the list.

When the Roll Down function key is pressed, this subroutine moves the bottom array element displayed to fields ITNBR and QTYOR so the operator can change those fields.

Figure 6-14 (Part 8 of 12). Sample Program OE140R (Entering Orders from Customers)

This subroutine adds 1 to the indexes used to display the array data on the screen if more than four line items have been entered.

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust	High (1>2)	Low (1<2)	Equal (1=2)	Comments
06	C	SR		UPINDX	BEGSR									
07	C	SR		I	COMP	4					20			
08	C	SR	20		ADD	1	A							
09	C	SR	20		ADD	1	B							
10	C	SR	20		ADD	1	C							
11	C	SR	20		ADD	1	D							
12	C	SR			ENDSR									

This subroutine subtracts 1 from the indexes used to display the array data on the screen when rolling down and when the item moved into ITNBR is greater than the fourth element in the arrays.

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust	High (1>2)	Low (1<2)	Equal (1=2)	Comments
19	C	SR		DNINDX	BEGSR									
20	C	SR		I	COMP	4					20		20	
	C	SR	20		SUB	1	A							
	C	SR	20		SUB	1	B							
	C	SR	20		SUB	1	C							
	C	SR	20		SUB	1	D							
	C	SR			ENDSR									

This subroutine turns on indicator 11 if one or more line items have been entered, indicator 12 if two or more line items have been entered, indicator 13 if three or more line items have been entered, indicator 14 if four or more line items have been entered.

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust	High (1>2)	Low (1<2)	Equal (1=2)	Comments
09	C	SR		SETIND	BEGSR									
10	C	SR		I	COMP	4					14		14	
11	C	SR		I	COMP	3					13		13	
12	C	SR		I	COMP	2					12		12	
13	C	SR		I	COMP	1					11		11	
14	C	SR			ENDSR									

Figure 6-14 (Part 9 of 12). Sample Program OE140R (Entering Orders from Customers)

C (Calculation Specifications)

This subroutine puts the order entered in file CUSTORDS. It writes the customer order record, then the ship-to record. Next it writes one line-item record for each line item entered. If no line items have been entered, no records are written.

```
01 C
02 C
03 C
04 C
05 C
06 C
07 CSR         ADDORD    BEGSR
08 CSR                   SETOF                        7071
09 CSR                   Z-ADD1         I
10 CSR         LOOP      TAG
11 CSR         ANBR,I    COMP *BLANK                      7Ø
12 CSR  7Ø               GOTO END
13 CSRN71                EXCPTCUSSHP
14 CSRN71                SETON                        71
15 CSR                  EXCPTLINEIT
16 CSR                   ADD  1         I
17 CSR                   GOTO LOOP
18 CSR         END       TAG
19 CSR                   MOVE *BLANK    ANBR
20 CSR                   MOVE *ZERO     AQTY
   CSR                   MOVE *BLANK    ADSC
   CSR                   MOVE *ZERO     APRI
   CSR                   MOVE *ZERO     AEXA
   CSR                   ENDSR
   C
```

O (Output Specifications)

```
01 OWORKSTN D        Ø1
02 O       OR        Ø2 99
03 O       OR        KB
04 O       OR        KH
05 O                                  K8 'OE14ØRD1'
06 O                 99     CUSNO      8
07 O                 99     ORDNO     14
08 O                 99N91            33 'THE CUSTOMER NUMBER'
09 O                 91               33 '    THE ORDER NUMBER'
10 O                 99N92            5Ø ' CANNOT BE BLANK.'
11 O                 92               48 ' WAS NOT FOUND.'
12 O
```

Figure 6-14 (Part 10 of 12). Sample Program OE140R (Entering Orders from Customers)

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Output Indicators	Field Name or EXCPT Name	Edit Codes	End Position in Output Record	Constant or Edit Word
01	O		D	02 N99 NKG				
02	O		OR	03 NKB NKH				
03	O						K8	'OE140RD2'
04	O				CUSNO		8	
05	O				ORDNO		14	
06	O				NAMAD		116	
07	O				ZIPCD		125	
08	O			11	ANBR,A		133	
09	O			11	AQTY,A	3	139	
10	O			11	ADSC,A		169	
11	O			11	APRI,A	3	180	'$'
12	O			11	AEXA,A	3	191	'$'
13	O			12	ANBR,B		199	
14	O			12	AQTY,B	3	205	
15	O			12	ADSC,B		235	
16	O			12	APRI,B	3	246	'$'
17	O			12	AEXA,B	3	257	'$'
18	O			13	ANBR,C		265	
19	O			13	AQTY,C	3	271	
20	O			13	ADSC,C		301	
	O			13	APRI,C	3	312	'$'
	O			13	AEXA,C	3	323	'$'
	O			14	ANBR,D		331	
	O			14	AQTY,D	3	337	
	O			14	ADSC,D		367	
01	O			14	APRI,D	3	378	'$'
02	O			14	AEXA,D	3	389	'$'
03	O				ITNBR		397	
04	O				QTYOR		403	
05	O			93			418	'THE ITEM NUMBER'
06	O			93			435	'CANNOT BE BLANK.'
07	O			94			427	'THE QUANTITY CANNOT BE Z'
08	O			94			431	'ERO.'
09	O			95			418	'THE ITEM NUMBER'
10	O			95			433	' WAS NOT FOUND.'
11	O			96			427	'TOO MANY LINE ITEMS HAVE'
12	O			96			441	' BEEN ENTERED.'

Figure 6-14 (Part 11 of 12). Sample Program OE140R (Entering Orders from Customers)

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	ADD/DEL	Field Name or EXCPT Name	End Position in Output Record	Constant or Edit Word
01	O	CUSTORDS	E	ADD	CUSSHP		
02	O					3	'CU'
03	O				CUSNO	11	
04	O				ORDNO	17	
05	O		E	ADD	CUSSHP		
06	O					3	'CS'
07	O				CUSNO	11	
08	O				ORDNO	17	
09	O		E	ADD	LINEIT		
10	O					3	'IT'
11	O				CUSNO	11	
12	O				ORDNO	17	
13	O				I	19	
14	O				ANBR,I	27	
15	O				AQTY,I	33	
16	O				AEXA,I	42	
17	O						

Figure 6-14 (Part 12 of 12). Sample Program OE140R (Entering Orders from Customers)

Chapter 7. Using a PRINTER File

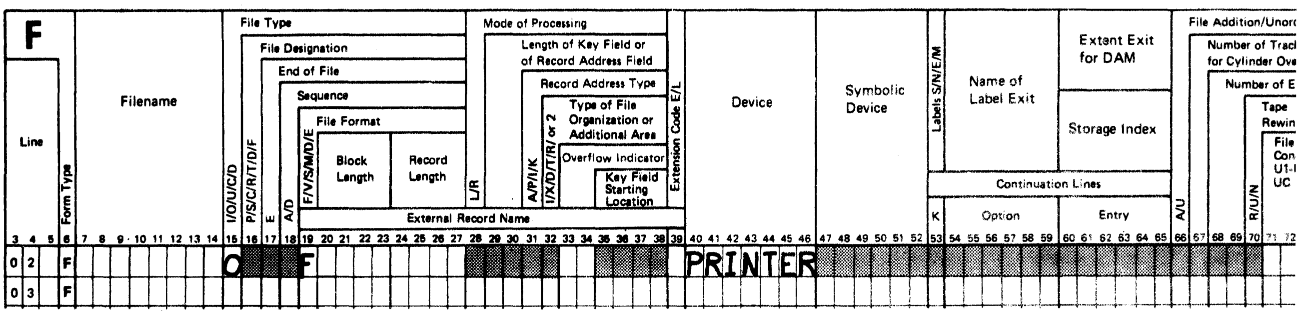
Chapter 7. Using a PRINTER File

A PRINTER file provides output to a printer. A program can use a maximum of eight PRINTER files, each with a separate name. Use the control language PRINTER statement to assign a PRINTER file to a particular printer. For information about the PRINTER statement, see the *System Reference* manual.

To use a PRINTER file, code entries on the file description, line counter, and output specifications.

File Description Specifications

Code entries in the unshaded columns of the file description specifications shown below:



Column 7 through 14 must contain the name of the PRINTER file.

Column 15 must contain O to indicate that the file is an output file.

Column 19 must contain F or blank to indicate that all the records in the file have the same length.

Columns 20 through 23 must contain the block length or blanks. If you enter the block length, it must equal the record length coded in columns 24 through 27. If you leave the entry blank, the program assumes that the block length equals the record length.

Columns 24 through 27 must contain the length of the largest record in the file. The record length can be 1 through 132 or 1 through 198, depending on the number of print positions your printer has.

Columns 33 and 34 can contain an overflow indicator (OA through OG or OV) or blank. For more information about overflow indicators, see *Overflow Indicators* later in this chapter.

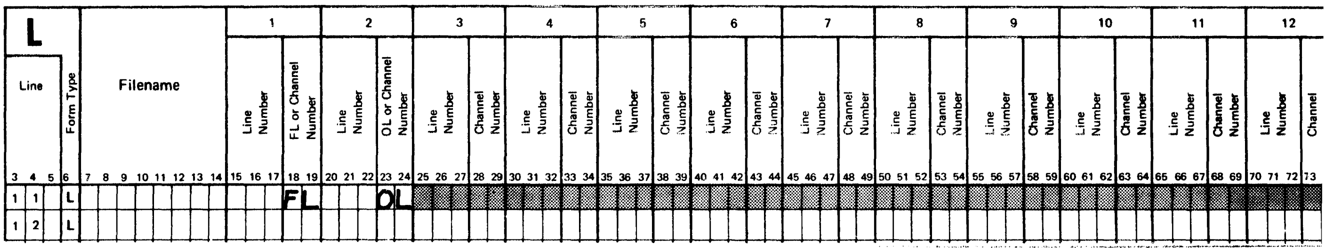
Column 39 must contain L if the file is further described on the line counter specifications.

Columns 40 through 46 must contain the device name PRINTER.

Columns 71 and 72 can contain U1 through U8 to indicate that the file is conditioned by an external indicator. For more information about external indicators, see Chapter 12.

Line Counter Specifications

Code entries in the unshaded columns on the line counter specifications shown below:



Columns 7 through 14 must contain the name of the PRINTER file. This name must be the same as the name coded in columns 7 through 14 of the file description specifications.

Columns 15 through 17 must contain the number of lines per page. The number of lines can be 1 through 112.

Columns 18 and 19 must contain FL to indicate that the entry in columns 15 through 17 is the form length.

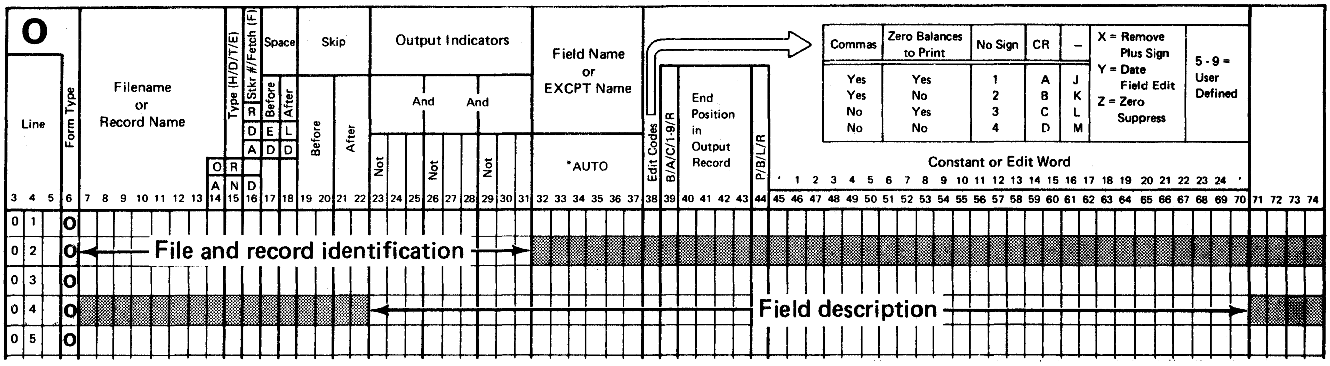
Columns 20 through 22 must contain the line number that is the overflow line. This entry can be 1 through 112, but it must be less than or equal to the number coded in columns 15 through 17. If the entry in columns 20 through 22 equals the entry in columns 15 through 17, overflow does not occur. If you leave these columns blank, the overflow line is six lines from the bottom of the page (line 60 if you have 66 lines per page). You cannot override the entry in columns 20 through 22 by using the LINES option of the control language PRINTER statement.

For more information about overflow, see *Handling Overflow* later in this chapter.

Columns 23 and 24 must contain OL to indicate that the entry in columns 20 through 22 is the overflow line.

Output Specifications

Code entries in the unshaded columns of the output specifications shown below:



File- and record-identification entries in columns 7 through 37 describe the output file, the records in the file, the values that control the spacing and skipping by the printer, and the indicators that condition the output.

Field-description entries in columns 23 through 70 describe the position and format of data on the output record. These entries must begin one line below the file- and record-identification entries.

File- and Record-Identification Entries

Columns 7 through 14 must contain the name of the PRINTER file. This name must be the same as the name given to the file in columns 7 through 14 of the file description specifications.

Columns 14 through 16 can contain AND if more than three indicators are needed to condition an output operation. Columns 14 and 15 can contain OR if an output operation is conditioned by any one of two or more output indicators or sets of output indicators. For more information about AND and OR lines, see *AND and OR Lines* later in this chapter.

Column 15 must contain H (heading), D (detail), T (total), or E (exception) to indicate the type of record to be written.

Column 16 must contain F if you use the fetch overflow routine. For more information, see *Fetch Overflow Routine* later in this chapter.

Column 17 can contain 0 through 3 to indicate the number of lines to be spaced before a line is printed. *Spacing* means advancing the form in the printer a specified number of lines. For more information about spacing, see *Spacing and Skipping* later in this chapter.

Column 18 can contain 0 through 3 to indicate the number of lines to be spaced after a line is printed.

Columns 19 and 20 can contain 01 through 99, A0 through A9, or B0 through B2 to indicate the line number that the printer should skip to before printing. A0 through A9 means 100 through 109. B0 through B2 means 110 through 112. *Skipping* means advancing the page in the printer to a specified line. For more information about skipping, see *Spacing and Skipping* later in this chapter.

Columns 21 and 22 can contain 01 through 99, A0 through A9, or B0 through B2 to indicate the line number that the printer should skip to after printing. A0 through A9 means 100 through 109. B0 through B2 means 110 through 112.

Columns 24 and 25, columns 27 and 28, and columns 30 and 31 can contain output indicators to specify the conditions under which a line is written. If these indicators are on, the output operation occurs.

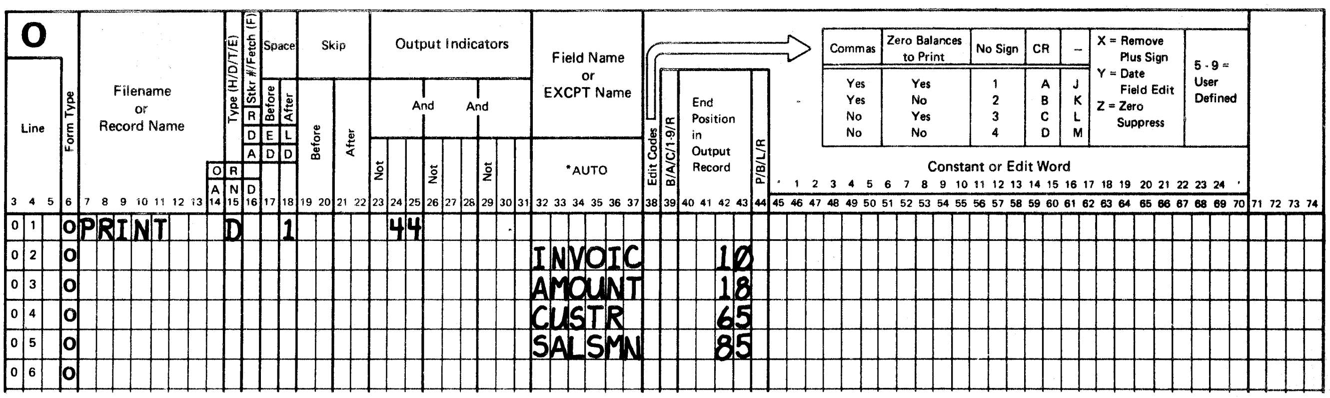
Figure 7-1 shows how output indicators condition the printing of an entire line or of a single field.

If no output indicators are specified, the line is written every time that record is checked for output. If no output indicators are specified on a heading or detail line, that line is also written at the beginning of the program cycle.

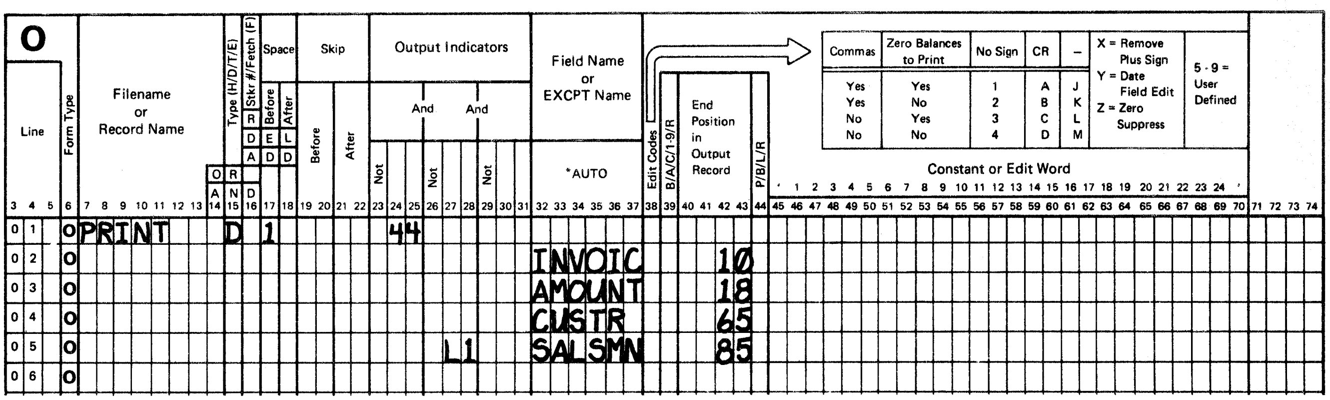
For more information about indicators, see Chapter 12.

Column 23, 26, or 29 can contain N to indicate that the output operation occurs only if the indicator coded in columns 24 and 25, 27 and 28, or 30 and 31 is *not* on. An N plus an indicator is called a *negative indicator*. No output line should be conditioned by negative indicators only; at least one of the output indicators should be positive. If a heading or detail line is conditioned by negative indicators only, the line is written at the beginning of the program cycle when the first-page lines (those conditioned by the 1P indicator) are written.

Columns 32 through 37 can contain an EXCPT name if column 15 contains E.



One indicator is used to condition an entire line of printing. When 44 is on, the fields named INVOIC, AMOUNT, CUSTR, and SALSMN are all printed.



A control-level indicator is used to condition when one field should be printed. When indicator 44 is on, fields INVOIC, AMOUNT, and CUSTR are always printed. However, SALSMN is printed for the first record of a new control group only if 44 and L1 are on.

Figure 7-1. Using Output Indicators to Condition the Printing of an Entire Line or of a Single Field

Field-Description Entries

Columns 23 through 31 can contain output indicators to specify the conditions under which a field is written.

Columns 32 through 37 can contain one of the following to specify each field that is to be written:

- Any field name or data structure name that was used in this program
- The reserved words PAGE, PAGE1 through PAGE7, *PLACE, UDATE, UDAY, UMONTH, or UYEAR
- An array name, an array element, or a table name

For more information, see *Columns 32-37 (Field Name)* in Chapter 27, *Output Specifications*.

Column 38 can contain an edit code. For information about edit codes, see Chapter 16, *Editing Numeric Fields*.

Column 39 can contain B to indicate that the field named in columns 32 through 37 is reset to blanks or zeros after the output operation is complete. Leave these columns blank if the field is not to be reset.

Columns 40 through 43 can contain the end position of a field or constant in the output record. This entry cannot exceed the record length, which is specified in columns 24 through 27 of the file description specifications for this file.

Columns 45 through 70 can contain a constant or edit word. For information about edit words, see Chapter 16, *Editing Numeric Fields*.

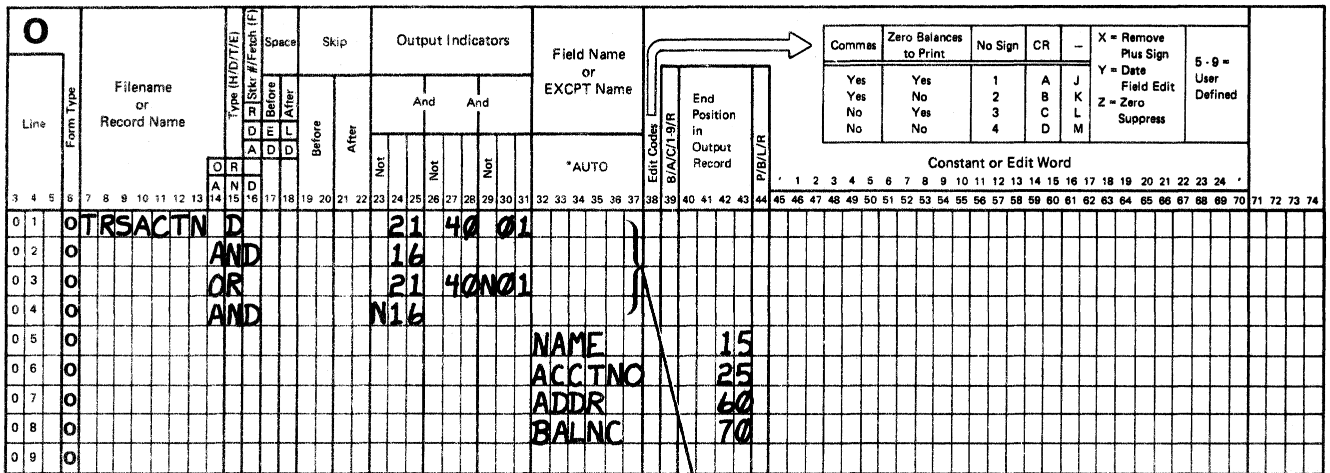
AND and OR Lines

Use an AND line if you need more than three indicators to condition an output operation. Code the word AND in columns 14 through 16 of each additional line. The conditions for all indicators in an AND relationship must be satisfied in order for the output operation to occur. You can use any number of AND lines for an output operation.

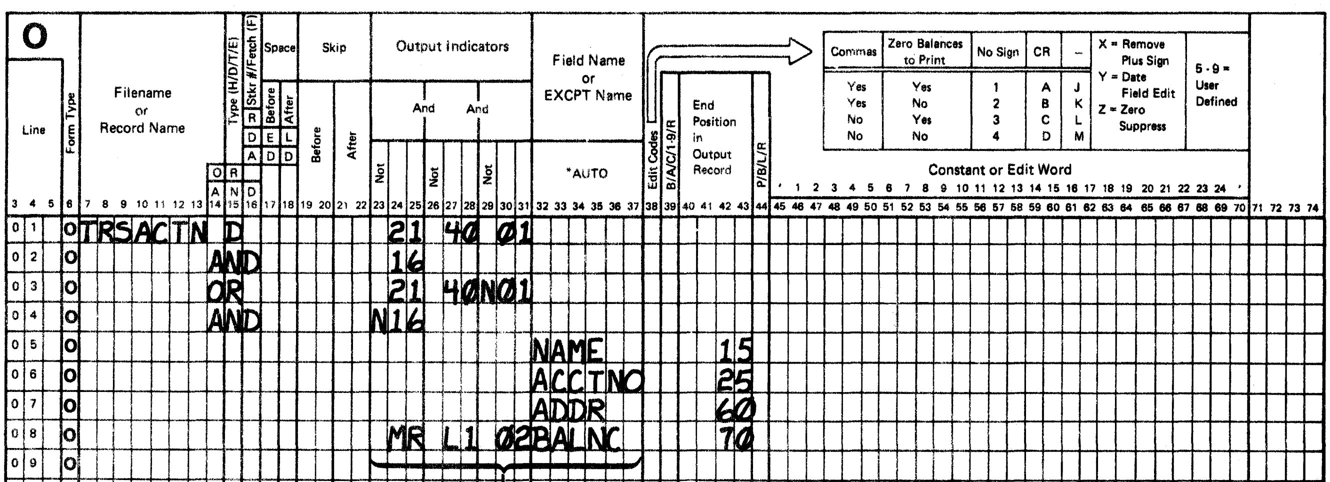
Use an OR line if you want an output operation to occur if any one of two or more conditions is satisfied. Code the word OR in columns 14 and 15 of each additional line. You can use a maximum of 20 OR lines for an output operation.

If you use a combination of AND and OR lines for an output operation, you can use any number of AND lines and a maximum of 20 OR lines.

AND and OR lines can condition entire output lines, but they cannot condition individual fields (see Figure 7-2). However, you can use more than three indicators to condition an output field by using the SETON operation code in calculations. For example, suppose you use indicators 10, 12, 14, 16, and 18 to condition an output field named PAY. In the calculation specifications you can set on indicator 20 if indicators 10, 12, and 14 are on, and then in the output specifications you can use indicators 16, 18, and 20 to condition the output field PAY.



The detail line is printed if either of two sets of conditions is met. If 21, 40, 01, and 16 are all on, the line is printed; if 21 and 40 are on and 01 and 16 are off, the line is also printed.



A maximum of three indicators can be used to condition a field.

Figure 7-2. Output Indicators in AND and OR Lines

If you use a control-level indicator (L0 through L9) in an OR relationship with the last-record (LR) indicator, the output operation might occur twice when the last-record indicator is on. One operation occurs when the last record is processed, and the other occurs at detail or total time. Figure 7-3 shows how to use control-level indicators and the last-record indicator correctly in an OR relationship.

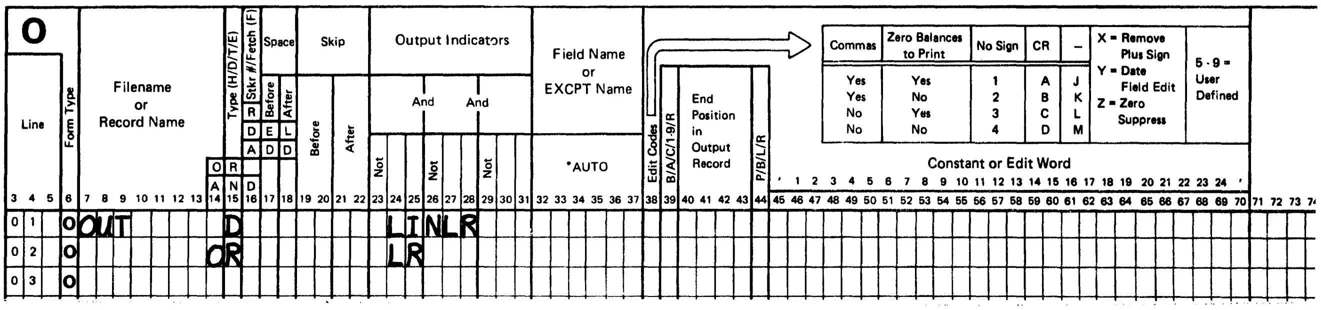


Figure 7-3. Correct Use of Control-Level Indicators and the Last-Record Indicator in an OR Relationship

Handling Overflow

Overflow is the condition that exists when a printer reaches the last line to be printed on a form. There are three ways to handle overflow:

- Automatic overflow
- Overflow indicators
- Fetch overflow routine

Automatic Overflow

If columns 33 and 34 of the file description specifications are blank and you do not use the fetch overflow routine, the program automatically advances the forms when overflow occurs and continues printing on the line one inch from the top of the new page (line 06 if you have 66 lines per page and six lines per inch).

The following steps occur during automatic overflow:

1. All remaining detail lines in that program cycle are printed if a printer operation spaced or skipped to the overflow area.

2. All remaining total lines in that program cycle are printed.

3. The printer skips to the line one inch from the top of the new page. Therefore, detail lines begin on that line (normally line 06) for all pages after the first.

If you use line counter specifications, overflow occurs at the line coded in columns 20 through 22 of those specifications. If you do not use line counter specifications, overflow occurs six lines before the line number coded as the LINES option of the PRINTER control language statement.

Overflow Indicators

You can use OA through OG or OV as an overflow indicator in columns 24 and 25, 27 and 28, or 30 and 31 of the output specifications. An overflow indicator conditions which lines in the PRINTER file print when overflow occurs. No more than one overflow indicator can be assigned to each PRINTER file in a program, and no overflow indicator can be assigned to more than one PRINTER file in a program. To use an overflow indicator in the output specifications, you must also assign the same overflow indicator to the PRINTER file in columns 33 and 34 of the file description specifications.

The RPG program cycle allows the overflow indicator to turn on at three different times: at total time, at detail time, and at calculation time if exception output is used. However, the only time that the program checks to see if the overflow indicator is on is right after all total records are printed, unless the fetch overflow routine was specified by an F in column 16.

When the overflow indicator turns on, the following steps occur:

1. Detail lines are printed (if that part of the program cycle is not already complete).

2. Total lines are printed (if conditions are met).

3. Total lines conditioned by the overflow indicator are printed.

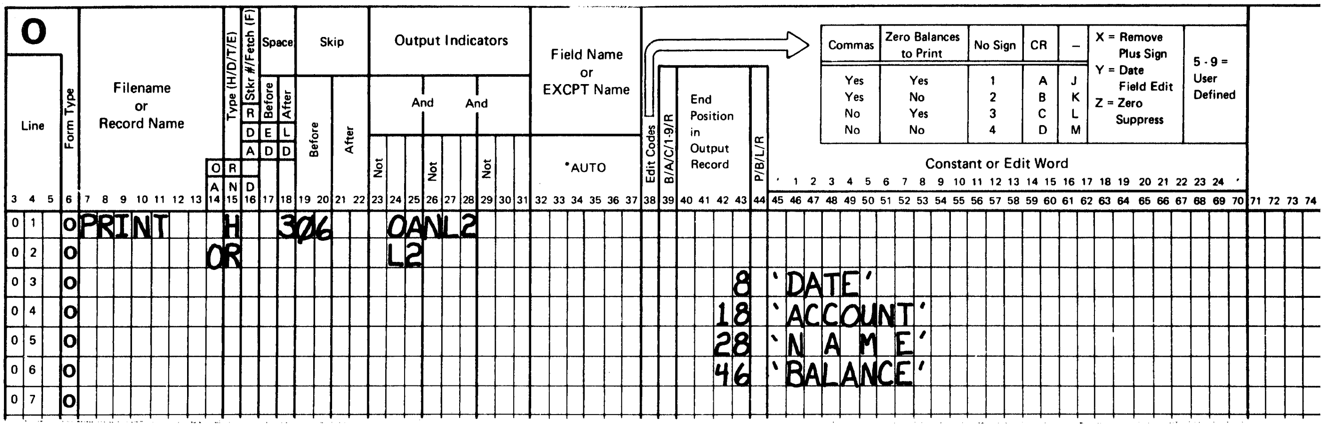
4. Heading lines and detail lines conditioned by the overflow indicator are printed.

5. The overflow indicator turns off.

Coding Overflow Indicators

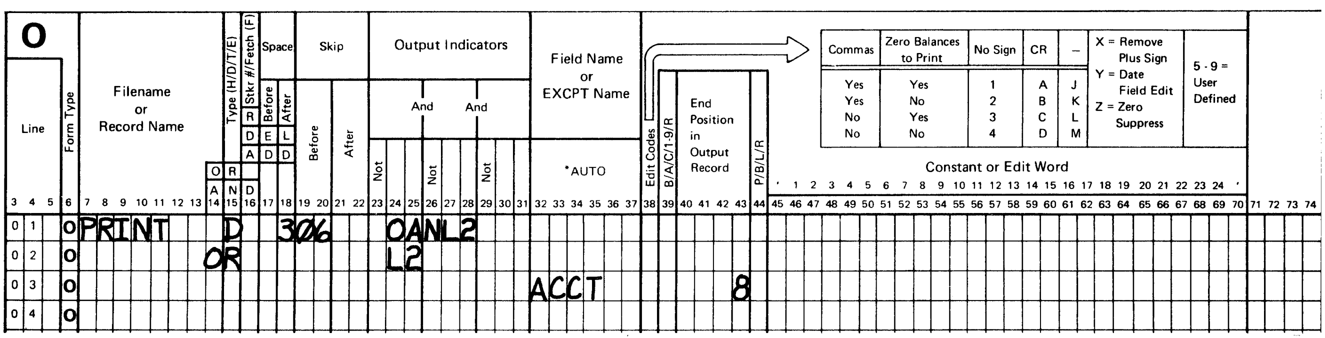
When you code overflow indicators in the output specifications, consider the following:

- Spacing past the overflow line turns on the overflow indicator.
- Skipping past the overflow line to any line on the same page turns the overflow indicator on.
- Skipping past the overflow line to any line on a new page does *not* turn the overflow indicator on.
- A skip to a new page coded on a line not conditioned by an overflow indicator turns the overflow indicator off before the forms advance to a new page.
- Control-level indicators can be used with an overflow indicator so that each page contains information from only one control group (see Figure 7-4).
- You can code an overflow indicator on AND or OR lines. However, only one overflow indicator can be associated with one group of output indicators.
- If you use an overflow indicator in an AND relationship with a record-identifying indicator, you may get unusual results because the record type might not be the one read when overflow occurred. In that case, the record-identifying indicator would not be on, so all lines conditioned by both overflow and record-identifying indicators would not print.
- An overflow indicator can be specified on the record-identification line of the output specifications only for a heading, detail, or total record (column 15 contains H, D, or T).
- An overflow indicator cannot be specified on the record-identification line for an exception record (E in column 15). However, an overflow indicator can condition fields within the exception record.
- You can turn overflow indicators on and off by using the SETON and SETOF operation codes.



This is the coding necessary for printing headings on every page: first page, every overflow page, and each new page to be started because of a change in control fields (L2 is on). Line 01 allows the headings to be printed at the top of a new page (skip to 06) only when an overflow occurs (OA is on and L2 is not on).

Line 02 allows printing of headings on the new page only at the beginning of a new control group (L2 is on). This way, duplicate heading caused by both L2 and OA being on at the same time do not occur. Line 02 allows headings to be printed on the first page after the first record is read because the first record always causes a control break (L2 turns on) if control fields are specified on the record.



This is the necessary coding for the printing of certain fields on every page; a skip to 06 is done either on an overflow condition or on a change in control level (L2). The NL2 indicator in line 01 prevents the line from printing and skipping twice in the same cycle.

Figure 7-4. Using Control-Level Indicators with an Overflow Indicator

Figure 7-5 shows the setting of overflow indicators when overflow is handled by overflow indicators and when it is handled by the fetch overflow routine. Both normal output and exception output are shown for each case. The solid lines show when the overflow indicator is on. The dashed lines show connections between the end of one program cycle and the beginning of the next.

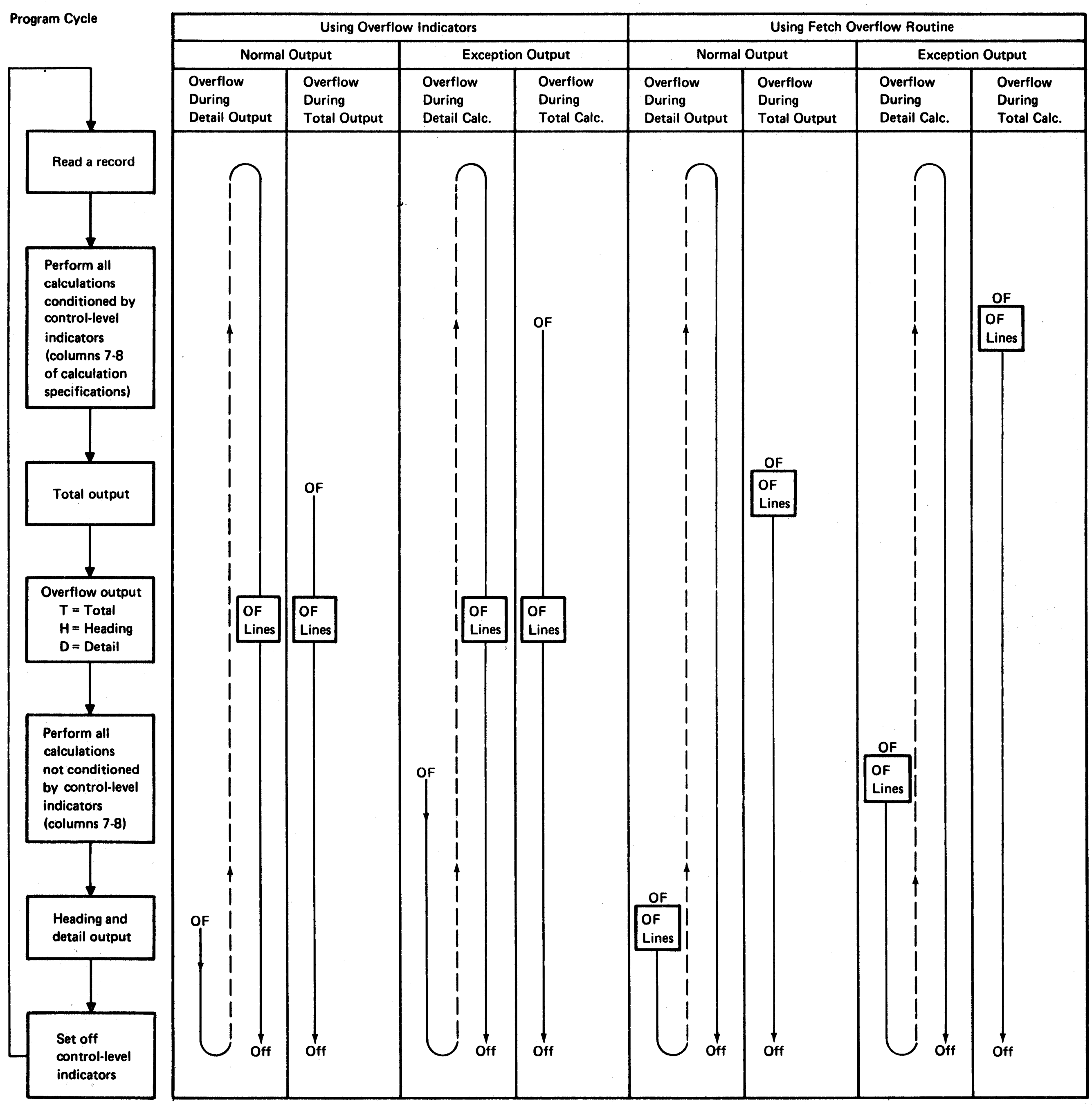


Figure 7-5. The Setting of Overflow Indicators during Overflow Handled by Overflow Indicators and by the Fetch Overflow Routine

Fetch Overflow Routine

The fetch overflow routine allows you to change the overflow logic of the RPG program cycle. You can advance forms when total, detail, or exception records are printed instead of waiting for the usual time in the program cycle.

Use the fetch overflow routine if printing a particular line would cause overflow and if not enough space is left on the page to print the remaining detail, exception, or total output lines. The fetch overflow routine can prevent printing over the page perforation and can ensure use of as much of the page as possible. To determine when to use the fetch overflow routine, study all possible overflow situations. By counting lines and spaces, you can calculate what happens if overflow occurs on each detail and total line.

To use the fetch overflow routine, code F in column 16 of the output specifications. Each time the program encounters the F in column 16, it tests whether the overflow indicator assigned to the PRINTER file is on. If the overflow indicator is on, the fetch overflow routine occurs in the following sequence:

1. All total lines conditioned by the overflow indicator are printed.

2. Forms advance to a new page when a skip to a line number less than the line number the printer is currently on is specified in a line conditioned by the overflow indicator.

3. Heading lines and detail lines conditioned by the overflow indicator are printed.

4. The line containing the F in column 16 is printed.

5. Any detail, exception, and total lines left to be printed for that output cycle are printed.

The fetch overflow routine does not automatically cause forms to advance; forms advance only if columns 19 and 20 or columns 21 and 22 of the output specifications for the overflow-conditioned line contain a 2-digit entry that is less than the line number that the printer is currently on.

Column 16 of each OR line must contain an F if the fetch overflow routine is to be used for each record in an OR relationship. The fetch overflow routine cannot be used when an overflow indicator is coded in columns 23 through 31 of the same specification line. If this occurs, the overflow routine is not called.

Spacing and Skipping

Spacing means advancing the form in the printer a specified number of lines. *Skipping* means advancing the form in the printer to a specified line. Spacing and skipping can be specified both before and after a line is printed. If both spacing and skipping are specified on the same line, they occur in this order:

1. Skip before
2. Space before
3. Skip after
4. Space after

With spacing, the maximum number of blank lines that can occur between two lines of print is five. If six spaces are specified (three after the preceding print line and three before the current print line), the printer spaces six lines and begins printing on the sixth line.

Spacing or skipping to the overflow line or past the overflow line turns the overflow indicator on. However, skipping past the overflow line to a line on the next page does not turn the overflow indicator on. Therefore, if you want to turn the overflow indicator on when you skip to the next page, use a SETON operation to turn on the overflow indicator to condition overflow operations.

Skipping is usually done when a new page is needed. A skip to a lower line number means advancing to a new page. Skipping can also be specified when more than five blank lines are required between two lines of print. The entry for skipping must be a 2-digit number that indicates the number of the next line to be printed. The skip entry must not be a higher number than the form length coded in columns 15 through 17 of the line counter specifications. If you code a skip to the line number that the forms are already positioned on, the forms do not move.

If columns 17 through 22 of the output specifications are blank, single spacing occurs after each line is printed. Separate spacing and skipping entries can be coded for each record in an OR relationship. If no spacing or skipping entries are coded for an OR line, spacing and skipping are done according to the specifications for the line before that OR line. No spacing or skipping can be specified on AND lines.

Sample Program

Figure 7-6 shows sample program AR936R, which prints a report of accounts receivable information. Depending on which option the user chooses, the program prints the balance due from each customer, from only those customers whose balance is not zero, from only those customers whose balances exceed an amount specified by the user, or from only those customers whose balance is overdue.

Sample program AR935R, which maintains the accounts receivable information in the customer master file CUSTMAST, is shown in Chapter 6. AR935R must be run before AR936R can be run.

IBM International Business Machines Corporation

RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

GX21-9092-7 UM/050*
Printed in U.S.A.

Program		Keying Instruction	Graphic		Card Electro Number
Programmer	Date		Key		

Page 1 2 of __ Program Identification (75 76 77 78 79 80) AR936R

Control Specifications

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

Line (3–5)	Form Type (6)	Size to Compile (7–9)	Object Output (10)	Listing Options (11)	Size to Execute (12–14)	Debug (15)	Reserved (16–17)	Currency Symbol (18)	Date Format (19)	Date Edit (20)	Inverted Print (21)	Reserved (22)	Number of Print Positions (23–25)	Alternate Collating Sequence (26)	Reserved (27–36)	Inquiry (37)	Reserved (38–39)	Sign Handling (40)	1P Forms Position (41)	Indicator Setting (42)	File Translation (43)	Punch MFCU Zeros (44)	Nonprint Characters (45)	Reserved (46)	Table Load Halt (47)	Shared I/O (48)	Field Print (49)	Formatted Dump (50)	RPG to RPG II Conversion (51)	Number of Formats (52–53)	S/3 Conversion (54)	Subprogram (55)	CICS/DL/I (56)	Transparent Literal (57)	(58–74)
01	H				Ø14																														

Name = AR936R

Function = Print aged trial balance report.

Input = CUSTMAST (customer master file) on disk.
Report option passed from program AR935R by local data area.
The options are 1, 2, 3, and 4.
1 = All customers
2 = All customers with balances
3 = All customer with balances over the amount entered in field OVBAL
4 = All customers with overdue balances

Output = The printed report chosen.

Indicator Definitions:
01 Record-identifying indicator for the CUSTMAST file
02 Conditions first-time processing
03 Conditions first-time headings
05 Conditions printing and record count
10 Turns on if option 1 is chosen
20 Turns on if option 2 is chosen
30 Turns on if option 3 is chosen
40 Turns on if option 4 is chosen
LR Last record processed
OA PRINTER file page overflow

Running This Program
AR935R must be run before AR936R. To run both AR935R and AR936R, code the following procedure:

```
// LOAD AR935R
// WORKSTN UNIT-?WS?,RESTORE-YES
// RUN
*    IF CMD7 IS PRESSED IN AR935R, A NONBLANK CHARACTER
*    IS PLACED IN POSITION 9 OF THE LOCAL DATA AREA AND
*    AR936R IS NOT RUN.
// IFF?L'9,1'?/GOTO NOPRT
// LOAD AR936R
// FILE NAME-CUSTMAST
// RUN
// TAG NOPRT
```

Figure 7-6 (Part 1 of 4). Sample Program AR936R (Printing Accounts Receivable)

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	Record Address Type	Type of File Organization	Overflow Indicator	Key Field Starting Location	Extension Code	Device	Symbolic Device	Labels	Name of Label Exit	Continuation Lines
02	F	CUSTMAST	I	P			F	256	256	8	A	I		2		DISK				
03	F	PRT	O					132	132				OA			PRINTER				
04	F																			

Line	Form Type	Filename or Record Name	Sequence	Number	Option	Record Identifying Indicator, **, or DS	Position (1)	Not (1)	C/Z/D (1)	Character (1)	Stacker Select	P/B/L/R	From	To	Decimal Positions	RPG Field Name	Control Level	Matching Fields	Field Record Relation	Plus	Minus	Zero or Blank
01	I	CUSTMAST	NS			01	1		C													
02	I												2	9		CUSNO						
03	I												10	34		CUSNM						
04	I											P	124	127	0	CRLIM						
05	I											P	131	135	2	AMDUE						
06	I											P	150	154	2	CHGTD						
07	I											P	155	159	2	OVR30						
08	I											P	160	164	2	OVR60						
09	I											P	165	169	2	OVR90						
10	I																					
11	I																					
12	I																					
13	I																					
14	I																					
15	I				UDS																	
16	I												1	1		OPTION						
17	I												2	80		OVBAL						
18	I																					

This is the local data area information passed to AR936R from AR935R.

Figure 7-6 (Part 2 of 4). Sample Program AR936R (Printing Accounts Receivable)

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators High (1>2) Plus	Low (1<2) Minus	Equal (1=2) Zero	Comments
01	C													
02	C	First-time logic to determine which report to print.												
03	C													
04	C													
05	C		Ø2		SETON						Ø3			
06	C		NØ2	OPTION	COMP	'1'							1Ø	
07	C		NØ2	OPTION	COMP	'2'							2Ø	
08	C		NØ2	OPTION	COMP	'3'							3Ø	
09	C		NØ2	OPTION	COMP	'4'							4Ø	
10	C		NØ2		SETON						Ø2			
11	C													
12	C	Calculate the total overdue payments for each customer.												
13	C													
14	C													
15	C													
16	C				Z-ADD	OVR3Ø	OVRTOT	9	2					
17	C				ADD	OVR6Ø	OVRTOT							
18	C				ADD	OVR9Ø	OVRTOT							
19	C													
20	C	If option 1 is chosen, print a report of all customers.												
	C													
	C													
	C													
	C		1Ø		SETON						Ø5			
	C													
01	C													
02	C	If option 2 is chosen, print a report only of customers with a balance.												
03	C													
04	C													
05	C		2Ø	AMDUE	COMP	*ZERO					Ø5	Ø5		
06	C													
07	C	If option 3 is chosen, print a report of customers with balances over a certain amount.												
08	C													
09	C													
10	C													
11	C		3Ø	AMDUE	COMP	OVBAL					Ø5			
12	C													
13	C	If option 4 is chosen, print a report of customers with overdue balances.												
14	C													
15	C													
16	C													
17	C		4Ø	OVRTOT	COMP	*ZERO					Ø5	Ø5		
18	C													
19	C	Count the number of records printed.												
20	C													
	C													
	C		Ø5		ADD	1	CUSCNT	5	Ø					
	C													

Figure 7-6 (Part 3 of 4). Sample Program AR936R (Printing Accounts Receivable)

Edit Codes	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	AND/OR	Type (H/D/T/E)	Space Before	Space After	Skip Before	Skip After	Output Indicators	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	PRT		D		2	Ø6		Ø1NØ3						
02	O		OR						OA						
03	O												24		'AGED TRIAL BALANCE REPOR'
04	O												25		'T'
05	O									UDATE	Y		34		
06	O								1Ø				54		'FOR ALL CUSTOMERS'
07	O								2Ø				61		'FOR CUSTOMERS WITH A BAL'
08	O								2Ø				65		'ANCE'
09	O								3Ø				61		'FOR CUSTOMERS WITH A BAL'
10	O								3Ø				7Ø		'ANCE OVER'
11	O								3Ø	OVBAL	J		81		'$'
12	O								4Ø				61		'FOR CUSTOMERS WITH OVERD'
13	O								4Ø				72		'UE BALANCES'
14	O												1ØØ		'PAGE'
15	O									PAGE			1Ø5		
16	O			D		2			Ø1NØ3						
17	O		OR						OA						
18	O												8		'CUST NUM'
19	O												24		'CUSTOMER NAME'
20	O												46		'AMOUNT DUE'
	O												57		'CHARGES'
	O												7Ø		'OVERDUE 3Ø'
	O												82		'OVERDUE 6Ø'
	O												94		'OVERDUE 9Ø'
	O												1Ø9		'TOT OVERDUE'
01	O												12Ø		'CRED LIM'
02	O			D		1			Ø1 Ø5						
03	O									CUSNO			8		
04	O									CUSNM			34		
05	O									AMDUE	J		46		'$'
06	O									CHGTD	J		58		'$'
07	O									OVR3Ø	J		7Ø		'$'
08	O									OVR6Ø	J		82		'$'
09	O									OVR9Ø	J		94		'$'
10	O									OVRTOT	J		1Ø8		'$'
11	O									CRLIM	J		12Ø		'$'
12	O			T	2				LR						
13	O									CUSCNT	1		6		
14	O												31		'CUSTOMER RECORDS PRINTED'
15	O														

Figure 7-6 (Part 4 of 4). Sample Program AR936R (Printing Accounts Receivable)

Chapter 8. Using a SPECIAL File

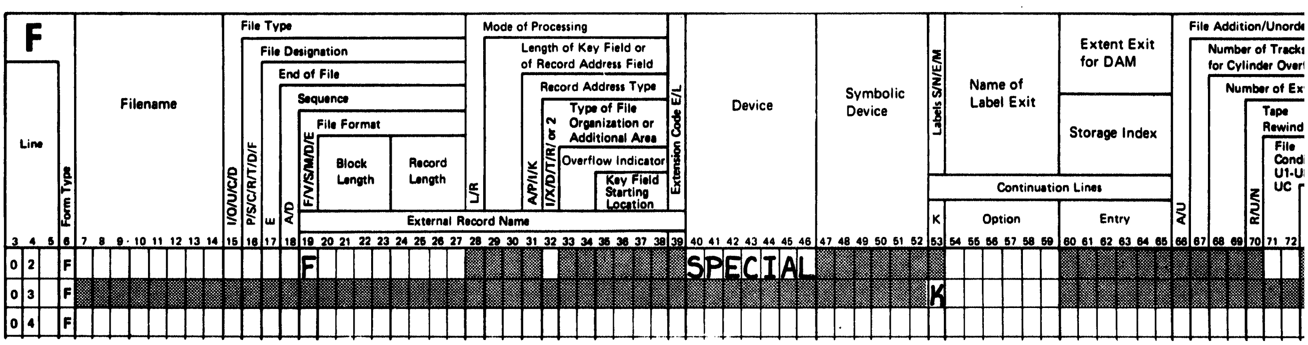
Chapter 8. Using a SPECIAL File

An RPG program can process files that use input and output devices not directly supported by RPG. To use such a file, code the device name SPECIAL on the file description specifications and provide a subroutine to transfer data between the SPECIAL device and main storage. That subroutine can be SUBR01, supplied by IBM, or a subroutine that you write in assembler language.

You can also use a SPECIAL file with subroutine SUBR22, provided by IBM, to read a transaction file created by the Work Station Utility (WSU). See *Reading and Updating a Work Station Utility Transaction File* in this chapter.

File Description Specifications

To use a SPECIAL file, code the unshaded columns in line 02 on the file description specifications shown below. If you use a continuation line, also code the unshaded columns in line 03:



Columns 7 through 14 must contain the name of the file.

Column 15 can contain I, O, U, or C to indicate that the file is an input, output, update, or combined file.

Column 16 can contain P, S, or D to indicate that the file is a primary, secondary, or demand file. Column 16 must be blank if column 15 contains O.

Column 17 must contain E if the program must process every record from the file before the program can end. Leave column 17 blank if the program can end before it processes every record in the file. Column 17 applies only to input, update, or combined files used as primary or secondary files.

Column 18 can contain A, D, or blank. A indicates that the program checks that the records are in ascending sequence. D indicates that the program checks that the records are in descending sequence. Blank indicates that the program does not check the record sequence. Column 18 applies only to input, update, or combined files used as primary or secondary files.

Column 19 must contain F or blank to indicate that all records in the file have the same length.

Columns 20 through 23, the block length, must contain a number from 1 through 9999. The block length must be greater than or equal to the record length.

Columns 24 through 27, the record length, must contain a number from 1 through 4096.

Columns 28 through 31 must be blank because SPECIAL files can be processed only consecutively.

Column 32 can contain any number from 1 through 9 to indicate that the program uses two input/output areas, or a blank to indicate that the program uses only one input/output area.

Columns 40 through 46 must contain the device name SPECIAL.

Columns 54 through 59 must contain the name of the subroutine that does the input and output operations between the SPECIAL device and main storage. You can use SUBR01 or SUBR22 for this purpose. The subroutine name must be in the form SUBRxx, where x is any alphabetic character (numeric characters are reserved for subroutines supplied by IBM), or in the form SRyzzz, where y is any of the following 15 characters: B, C, D, F, G, H, I, L, M, O, P, R, S, T, or U, and z is any of the following 16 characters: A, B, C, D, F, G, H, I, L, M, O, P, R, S, T, or U. Subroutine names in the form SUBRxx cannot be overlaid; subroutine names in the form SRyzzz can be overlaid.

Columns 71 and 72 can contain an external indicator, U1 through U8.

Continuation Line

Column 53 must contain K to indicate that this continuation line provides more information about the SPECIAL file coded on the preceding line. Only one continuation line can be used for each SPECIAL file.

Column 54 through 59 must contain the name of a table or array used by the subroutine that you wrote in assembler language to do input and output for the file.

Restrictions for SPECIAL Files

You can use the following with SPECIAL files:

- FORCE operation in the calculation specifications
- READ operation in the calculation specifications
- File translation (column 43 of the control specification)

You cannot use the following with SPECIAL files:

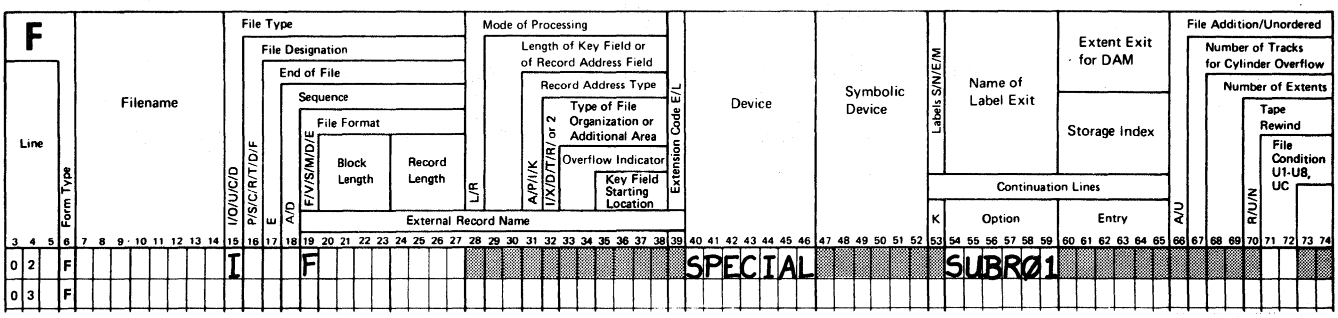
- CHAIN operation in the calculation specifications
- Spacing and skipping (columns 17 through 22 of the output specification)
- *PRINT

Using a Subroutine for Input and Output

Because RPG does not support a SPECIAL device directly, you must provide a subroutine to transfer data between the SPECIAL device and main storage. You can use SUBR01 or SUBR22, supplied by IBM, or you can write your own subroutine in assembler language.

Using IBM's Subroutine, SUBR01

Subroutine SUBR01 reads records from the system source of input. (If you enter control language statements from the display station keyboard, the display station is the source of input. If the control language statements are in a procedure, the procedure is the source of input.) The records can be 120 or 512 characters long. RPG treats the records read by SUBR01 as data records. To use this subroutine, code SUBR01 in columns 54 through 59 of the file description specifications for a SPECIAL device:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain I to indicate that the file is an input file.

Column 16 must contain P, S, or D to indicate that the file is a primary, secondary, or demand file.

Column 19 must contain F or blank to indicate that all records in the file have the same length.

Columns 20 through 23 (block length) and 24 through 27 (record length) must contain 120 or 512. If you do not specify a block length, the block length is assumed to equal the record length. If you enter the control language statements to run the program from the keyboard, the records to be made available to the file must also be entered from the keyboard. If you call a procedure to run the program, the records to be made available to the file must follow the RUN statement in the procedure.

The last input record in the procedure should be followed by a control language END statement. If the program uses only one display station and if there is no END statement in the procedure, the program treats the next control language statements (entered from the keyboard) as input to the SPECIAL file in the program. If the program uses more than one display station, you must use a procedure for the data records, and an END statement must follow the last data record.

If a program is to be run from the input job queue, you must use a procedure to run the program. If the program that uses SUBR01 also uses a CONSOLE file, the control language statements and the data records for SUBR01 must be contained in a procedure. Otherwise, undesirable results can occur.

See the *System Reference* manual for information on how to create a procedure.

Using Your Own Subroutine

If you write your own assembler language subroutine to do input and output for a SPECIAL device, you must link your subroutine to the RPG program and perform your file input and output by using the DTF (define the file) created by the RPG compiler. The format of the DTF is as follows:

Bytes (Hex)	Description
0	Device code (hex 00)
1-2	Address of data management
3	Mask for external indicators
4	Must be hex 80
5-6	Backward chain pointer
7	Must be hex 80
8-9	Forward chain pointer
A	Must be hex 80
B-C	Logical record address
D	Completion code: Hex 42 = End of file Hex 41 = Controlled cancel Hex 40 = Normal completion
E	Operation code: Hex 80 = Get Hex 40 = Put Hex 20 = Update Hex 10 = Close
F-12	Attributes Byte 1: Hex 20 = Update file Hex 40 = Output file Hex 80 = Input file Hex C0 = Combined file Byte 2: Hex 08 = Dual input/output Hex 01 = DTF open Bytes 3 and 4 must be hex 00
13-14	Record length
15-1C	Filename
1D-1E	Physical input address
1F-20	Physical output address
21-22	Block length
23-24	Address of area in storage used to define the array (DTT) if array linkage is used

The compiler passes the address of byte 0 of the area in storage used to define the SPECIAL file (DTF) to the subroutine in index register 2. The compiler fills in bytes 0 through 9 and F through 24 at compilation time. The content of these fields depends on the entries coded in the file description specifications for the SPECIAL device. The input address (bytes 1D and 1E) and the output address (bytes 1F and 20), when present, point to the physical buffer that the compiler reserves within the load module for use by the SPECIAL device. The subroutine inserts the completion code (byte D) when control returns to RPG. The operation code (byte E) and the logical record address (bytes B and C) are inserted when the RPG program is running.

Figure 8-1 shows the equates for the DTF fields created by the RPG compiler for an assembler subroutine.

```
      243+***********************************************************************>
      244+*                                                                      >
      245+*              RPG II SPECIAL DTF OFFSETS AND EQUATES                  >
      246+*                                                                      >
      247+***********************************************************************>

      249+*        **** SPECIAL DTF LAYOUT ****
0000  250+SPDEV    EQU   0              DEVICE CODE (X'00')
0002  251+SPDMA    EQU   SPDEV+2        ADDRESS OF D.M.
0003  252+SPUPS    EQU   SPDMA+1        UPSI INDICATORS
0006  253+SPCHA    EQU   SPUPS+3        BACKWARD CHAIN POINTER
0009  254+SPCHB    EQU   SPCHA+3        FORWARD CHAIN POINTER
000C  255+SPLRA    EQU   SPCHB+3        LOGICAL RECORD ADDRESS
000D  256+SPCMP    EQU   SPLRA+1        COMPLETION CODE
000E  257+SPOPC    EQU   SPCMP+1        OPERATION CODE
000F  258+SPAT1    EQU   SPOPC+1        ATTRIBUTE BYTE ONE
0010  259+SPAT2    EQU   SPAT1+1        ATTRIBUTE BYTE TWO
0011  260+SPAT3    EQU   SPAT2+1        ATTRIBUTE BYTE THREE
0012  261+SPAT4    EQU   SPAT3+1        ATTRIBUTE BYTE FOUR
0014  262+SPRCL    EQU   SPAT4+2        RECORD LENGTH
001C  263+SPNAM    EQU   SPRCL+8        FILE NAME
001E  264+SPPBI    EQU   SPNAM+2        PHYSICAL INPUT I/O ADDRESS
0020  265+SPPBO    EQU   SPPBI+2        PHYSICAL OUTPUT I/O ADDRESS
0022  266+SPBKL    EQU   SPPBO+2        BLOCK LENGTH
0024  267+SPDTT    EQU   SPBKL+2        ADDR OF ARRAY DTT IF SPECIFIED
0025  268+SPLEN    EQU   SPDTT+1        LENGTH OF SPECIAL DTF

      270+*        **** SPCMP EQUATES ****
0040  271+SPNORM   EQU   X'40'          NORMAL
0042  272+SPEOF    EQU   X'42'          END OF FILE
0041  273+SPCCNL   EQU   X'41'          CONTROLLED CANCEL

      275+*        **** SPAT1 EQUATES ****
0020  276+SPUPDT   EQU   X'20'          UPDATE FILE
0040  277+SPOUT    EQU   X'40'          OUTPUT FILE
0080  278+SPINP    EQU   X'80'          INPUT FILE
00C0  279+SPCMB    EQU   X'C0'          COMBINED

      281+*        **** SPAT2 EQUATES ****
0008  282+SPDIO    EQU   X'08'          DUAL I/O
0001  283+SPOPEN   EQU   X'01'          DTF OPEN

      285+*        **** SPOPC EQUATES ****
0040  286+SPPUT    EQU   X'40'          PUT
0080  287+SPGET    EQU   X'80'          GET
0020  288+SPUPD    EQU   X'20'          UPDATE
0010  289+SPCLS    EQU   X'10'          CLOSE
      290 *** END OF EXPANSION **
```

Figure 8-1. Equates for the DTF Fields Created by RPG for an Input and Output Subroutine Written in Assembler Language by a User

If your subroutine uses an array, the RPG compiler creates the following area in storage to define the array (DTT):

Bytes	Description
0-1	Address of rightmost byte of the first element of the array
2-3	Address of rightmost byte of the last element of the array
4-5	Work area
6-7	Length of array element
8-13	Array name

Points to Remember When You Write an Assembler-Language Subroutine

The input and output subroutine must save and restore the registers changed in the subroutine. Control should be returned to the address in the address recall register (ARR).

When an input operation is done, the subroutine must move the address of the physical buffer currently being used to the logical record address location in the DTF (bytes B and C). This logical record address points to the record within the physical buffer that is to be processed by the RPG program.

When an output operation is requested, the subroutine must move the data from the logical buffer (address in bytes B and C of the DTF) to the physical buffer (address in bytes 1F and 20 of the DTF). If the record length is less than or equal to 144, the logical record address (bytes B and C) points to the RPG common output buffer that contains the record to be written by the SPECIAL device.

The subroutine must open the SPECIAL file the first time the subroutine is called. It must also close the file.

Reading and Updating a Work Station Utility Transaction File (SUBR22)

Subroutine SUBR22, provided by IBM, allows an RPG program to read and update records from a transaction file created by the work station utility (WSU). To link to this subroutine, use the SPECIAL device and an array with one 13-character element.

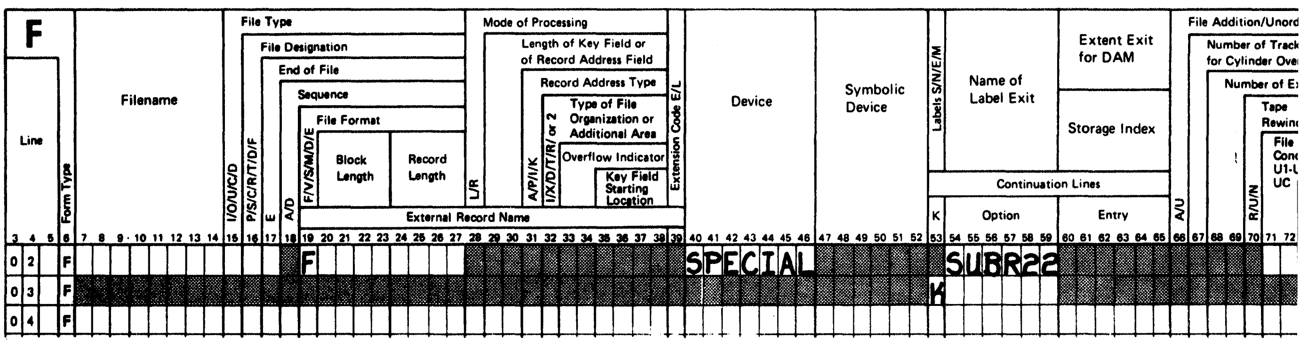
Your program must initialize positions 1 through 11 of the array to the values described later in this chapter under *Contents of the Array* before SUBR22 reads the first data record or begins to read a new logical chain.

The last 13 bytes of the record in the transaction file (the trailer information) are not returned to the RPG program and cannot be updated.

Your program should check position 13 of the array (the error indicator) after each data record is read to determine if any errors occurred.

File Description Specifications

To use SUBR22, the following entries must be made on the file description specifications:



Columns 7 through 14 must contain the name of the transaction file.

Column 15 can contain I or U to indicate that the file is an input file or an update file.

Column 16 can contain P, S, or D to indicate that the file is a primary, secondary, or demand file.

Column 17 must contain E if the program must process every record in the file before the program can end. Leave this column blank if the program can end before it processes every record in the file. The E entry applies only to primary and secondary files.

Column 19 must contain F or blank to indicate that every record in the file has the same length.

Columns 20 through 23 must contain the block length, which is determined as follows:

- The block length = 256 if the record length is a number that divides evenly into 256
- The block length = the record length if the record length is a number that can be divided evenly by 256
- Otherwise, the block length = the record length plus 255, rounded up to the next number that can be divided evenly by 256

Columns 24 through 27 must contain the record length. The record length can be from 14 through 4096. The record length must include 13 bytes for the work station utility file trailer information. Bytes 11 and 12 of the trailer contain the display station ID.

Columns 40 through 46 must contain the device name SPECIAL.

Columns 54 through 59 must contain SUBR22.

Columns 71 and 72 can contain an external indicator, U1 through U8.

Continuation Line

Column 53 must contain K to indicate that this continuation line contains more information about the file coded on the preceding specification line.

Columns 54 through 59 must contain the name of an array with one 13-character element. This array must also be coded on the extension specifications.

Contents of the Array

The array named in columns 54 through 59 of the continuation line is used to pass parameters from the RPG program to SUBR22 and from SUBR22 to the RPG program. The best way to address these array elements is to redefine the array as a data structure. The array can contain the following entries:

Positions 1 through 7 (Starting Record Number)

Entry: Any valid zoned-decimal relative record number or blank

To read part of a logical chain in the transaction file, code the relative record number of the first record to be read. SUBR22 reads records from the file until it reaches the end of the logical chain or until it finds a restart parameter (R in position 11). SUBR22 uses the starting record number to process records, and then it changes the starting record number to blanks.

Positions 8 and 9 (Display Station Identifier)

Entry: 2-character display station identifier or blank

To read one logical chain from the transaction file, code the identifier of the display station whose logical chain is to be read. SUBR22 reads records from the file until it reaches the end of the logical chain or until it finds a restart parameter (R in position 11). SUBR22 uses the display station identifier to process records, and then it changes the display station identifier to blanks.

When the file is read, this same value is in positions 11 and 12 of the trailer portion of the transaction file record.

Position 10 (Type)

Entry: A or blank

To read all the logical chains in the transaction file, code A in position 10. SUBR22 reads all logical chains in the file in the sequence in which they are chained (that is, it first reads all records for the first display station in the chain, then all records for the second display station in the chain, and so on). SUBR22 uses the type entry to process records, and then it changes the type entry to blank.

Position 11 (Restart/Active)

Entry: A, R, or blank

The restart parameter allows the RPG program to read or update more than one logical chain and to read or update active logical chains or chains from sessions that ended abnormally. When R is coded in position 11, the program starts processing records from the file as specified by the parameter list. However, you should not move R into position 11 before the program reads the first record. When A is coded in position 11, the program also starts processing records from the file as specified by the parameter list. An entry of A allows the program to process records from active work sessions and work sessions that ended abnormally. If position 11 is blank, the program reads the next record in the logical chain.

Position 12 (Last-Record Flag)

Entry: E, L, or blank

SUBR22 returns an E to the RPG program in position 12 of the array when A is coded in position 11 of the array and one of the following conditions occurs:

- The program reaches the end of the logical chain when a starting record number is coded in positions 1 through 7.
- The program reaches the end of the logical chain when a display station identifier is coded in positions 8 and 9.

- The program reaches the last record of the transaction when A is coded in position 10.

SUBR22 returns a blank record to the program whenever an E is returned to the array.

SUBR22 returns an L to the RPG program in position 12 of the array when one of the following conditions occurs:

- The current record is the last record entered for a work session that ended normally when a starting record number is specified in positions 1 through 7.
- The current record is the last record entered for a work session that ended normally when a display station identifier is specified in positions 8 and 9.
- The current record is the last record entered for a work session that ended normally, the work session is the last work session in the transaction file, and A is coded in position 10.

When SUBR22 returns an E or L to the RPG program, you can specify the restart option (R in position 11). If the program ignores the E or L parameter, the subroutine returns a normal end-of-file indication to the RPG program the next time the program asks the subroutine to read. After the subroutine returns the end-of-file indication to the RPG program, additional attempts to read the file also cause the subroutine to return an end-of-file indication.

SUBR22 blanks out the last-record flag if a restart is specified.

Position 13 (Error Flag)

Entry: Blank, W, J, N, P, or D

Blank

No errors were found. The subroutine returned a good data record to the program.

W

This display station session ended abnormally. The last sequence set may be incomplete, or some inserted records may have been lost. A data record is returned to the program with this error flag. SUBR22 does not read any records that were added during the work session that ended abnormally.

J

This file did not yet close normally and contains display station sessions that may be incomplete. However, the data being processed by the program comes from a completed display station session. A data record is returned to the program with this flag. This flag does not necessarily mean that an error occurred; it could mean only that another work station utility session is active and is using the same file.

N

No record was found for one of the following reasons:

- The display station identifier coded in the parameter list is not allowed.

- The parameter list does not contain a starting record number, type, or display station identifier parameter for a first-time or restart option.

- The relative record number coded is not allowed.

A blank record is returned with this error flag. If no record is found, you must code the restart option the next time the subroutine is called, and you must reset the no-record-found indicator.

P

The program tried to update a record, but no record was retrieved for the update.

D

The program tried to process a file from which records can be deleted. The work station utility does not allow such files. The subroutine cancelled the processing of this file and closed the file.

Note: When the subroutine is accessed the first time or with a restart parameter coded, one of the three first-time options (starting record number, display station identifier, or type) must be coded in the parameter list. If a valid option is not coded, the subroutine returns to the program without reading a record. If the program accesses the subroutine twice consecutively with a valid option coded, the subroutine returns a normal end-of-file indication to the RPG program. If more than one valid option is coded, the subroutine processes only the first valid option found. The subroutine checks for the first valid option in the following order: starting record number, type, and display station identifier.

Example of SUBR22

Figure 8-2 shows an example of an RPG program that reads a work station utility transaction file.

```
  02 FDISPLAY CD  F      5Ø            WORKSTN
  03 FWSUXAC  ID  F 512  77            SPECIAL       SUBR22
  04 F                                              KCONTRL
  05 FTRANS   O   F  64  64            DISK
  06 F
```

```
  01 E                    CONTRL      1 13
  02 E
```

```
  01 IDISPLAY NS.
  02 I                                       1   2 WSID            Ø6
  03 IWSUXAC  NS  Ø1   1 C
  04 I       OR   Ø1   1 CC
  05 I                                       1  64 DATA
  06 I            DS
  07 I                                       1  13 CONTRL
  08 I                                       8   9 WSID
  09 I                                      11  11 RESTRT
  10 I                                      12  12 LAST
  11 I                                      13  13 ERROR
  12 I
```

Figure 8-2 (Part 1 of 2). Reading a Work Station Utility Transaction File

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Name	Length	High	Low	Equal	Comments
01	C				READ	DISPLAY						BLANK RECORD
02	C				GOTO	RSTART						GET WSID
03	C			LOOP	TAG							
04	C				SETOF				Ø1			SET OFF RCD ID
05	C				READ	WSUXAC						READ WSU FILE
06	C			ERROR	COMP	'J'					Ø3	CHECK
07	C			ERROR	COMP	'W'					Ø4	FOR
08	C			ERROR	COMP	'N'					Ø5	ERROR
09	C			LAST	COMP	'L'					Ø7	CONDITIONS
10	C				MOVE	' '	LAST					
11	C		Ø3		GOTO	END						
12	C		NØ4NØ5		EXCPT	DSKOUT						OUTPUT TO DISK
13	C		NØ4NØ5NØ7		GOTO	LOOP						CONT IF READ C
14	C		Ø4									
15	C	OR	Ø5									
16	C	OR	Ø7		SETON				99			SET ON MSG FLA
17	C				MOVE	'R'	RSTART					SET UP RESTART
18	C			RSTART	TAG							
19	C				EXCPT	PROMPT						DISPLAY PROMPT
20	C				SETOF				Ø4	Ø5		SET OFF ERROR
	C				SETOF				Ø6	Ø7	99	INDICATORS
	C				SETON				LR			
	C				READ	DISPLAY						READ WS ID
	C				MOVE	WSID	WSIDH	2				SAVE WS ID
	C		NKG Ø6		GOTO	RSTART						ID BLANK, RETR
01	C		NKG		GOTO	LOOP						VALID ID, CONT
02	C			END	TAG							CMD 7, EOJ
03	C				EXCPT	EOJ						DISPLAY END
04	C											

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Output Indicators	Field Name or EXCPT Name	End Position in Output Record	Constant or Edit Word
01	O	DISPLAY	E		PROMPT		
02	O					K6	'PROMPT'
03	O			99		6	'WS ID'
04	O			99	WSIDH	9	
05	O			Ø6 LR		12	'WS ID BLANK'
06	O			Ø4		19	'BAD CHAIN'
07	O			Ø5		19	'NOT FOUND'
08	O			Ø7NØ4		22	'CHAIN COPIED'
09	O		E		EOJ		
10	O					K3	'END'
11	O			Ø3		8	'BAD FILE'
12	O	TRANS	E	Ø1	DSKOUT		
13	O				DATA	64	
14	O						

Figure 8-2 (Part 2 of 2). Reading a Work Station Utility Transaction File

This program uses SUBR22 to read chains of records from a work station utility transaction file (WSUXAC) and to write them to a permanent disk file (TRANS). The person using the display station enters the identifier of the display station whose transactions he wishes to copy. The program displays error messages when a file or chain containing an error occurs, or when the display station identifier entered cannot be found. The program also displays a message after all the records in a chain have been copied. The program ends when a file containing an error is found, or when command key 7 is pressed.

SUBR22 requires an array (in this program, an execution-time array is used) to contain control information to be passed back and forth between SUBR22 and the program with each data record. This array (CONTRL) contains the display station identifier, the error indications, and the last-record flag.

The first READ operation reads the first record, which is blank, from the display. Then the program branches to the EXCPT operation, which displays the first prompt, WS ID. The program also turns off indicators 04, 05, 06, 07, and 99.

When the second READ operation occurs, one of the following two conditions is true:

- *The Enter key is pressed (WS ID is blank).* Indicator 06 turns on because the input field is blank. The program then goes to RSTART. The EXCPT operation occurs, displaying the prompt ENTER WS ID and the error message WS ID BLANK. The program sets off indicators 04, 05, 06, 07, and 99, and waits for the READ operation to occur.

- *Two characters are entered in the WS ID field.* The program attempts to read a record from the transaction file (WSUXAC). The SUBR22 error flag (ERROR) is checked for the following:

  - If the WSU transaction file contains an error, indicator 03 turns on.
  - If the record returned to the program is from a display station session that ended abnormally, indicator 04 turns on.
  - If no records exist for the display station identifier that was entered, indicator 05 turns on.

After these comparisons are done, the last-record flag (LAST) is checked to determine whether this record is the last record in the logical chain. If it is the last record, indicator 07 turns on.

If indicators 04 and 05 are not on, the record is written to the permanent disk file TRANS if indicator 01 is on. This is done by the EXCPT output named DSKOUT.

Depending on which indicators turned on by the previous comparisons, the EXCPT operation causes one of the following output combinations to occur:

- The prompt ENTER WS ID and the error message WS ID xx BAD CHAIN are displayed if indicator 04 is on.
- The prompt ENTER WS ID and the error message WS ID xx NOT FOUND are displayed if indicator 05 is on.
- The prompt ENTER WS ID and the message WS ID xx CHAIN COPIED are displayed if indicator 07 is on.

If indicator 04, 05 or 07 is on, the program goes through logic to get a new identifier after writing the output. Otherwise, the program goes to LOOP and continues reading records from the WSU transaction file until an error occurs or until the last record in the chain is reached.

The program ends in one of two ways. Either the person at the display station presses command key 7, which turns on indicator KG, or a bad file is found, which turns on indicator 03. Either condition causes the program to branch to END and show the end-of-job display. If indicator 03 is on, the error message BAD FILE is also displayed.

Chapter 9. Using a CONSOLE, KEYBORD, or CRT File

Chapter 9. Using a CONSOLE, KEYBORD, or CRT File

The device names WORKSTN, CONSOLE, KEYBORD, and CRT all refer to the same object: a display station, which consists of a display screen and a keyboard. However, WORKSTN, CONSOLE, KEYBORD, and CRT files differ in the ways that they use a display station.

Whenever possible, use a WORKSTN file instead of a CONSOLE, KEYBORD, or CRT file. WORKSTN files offer much greater advantages. In fact, the only reason that System/36 allows the use of CONSOLE, KEYBORD, or CRT files is so that programs that used those files on earlier IBM computers can run on System/36 without being rewritten.

As Chapter 6 explains, a WORKSTN file is a combined (both input and output) file. It allows you to specify which fields on the display are input fields, which are output fields, and which are both input and output fields. You can also use a WORKSTN file in a program that allows one or more requesters.

By contrast, CONSOLE, KEYBORD, and CRT files can be used only in programs that allow only one requester. In addition:

- A CONSOLE file can be used only as an input file, so you cannot display the records in a CONSOLE file. A CONSOLE file can be used as an input data file to provide data to a program that is running or as a record address file to provide key fields for processing within key-field limits. A program can use only one CONSOLE file. (Along with these disadvantages, a CONSOLE offers one advantage. It provides an easy way to create a simple data-entry program, because the program creates input prompts automatically.)

- A KEYBORD file can be used as both an input and an output file when you use the KEY and SET operations. These operations allow you to display prompts and messages and to respond by entering one field at a time.

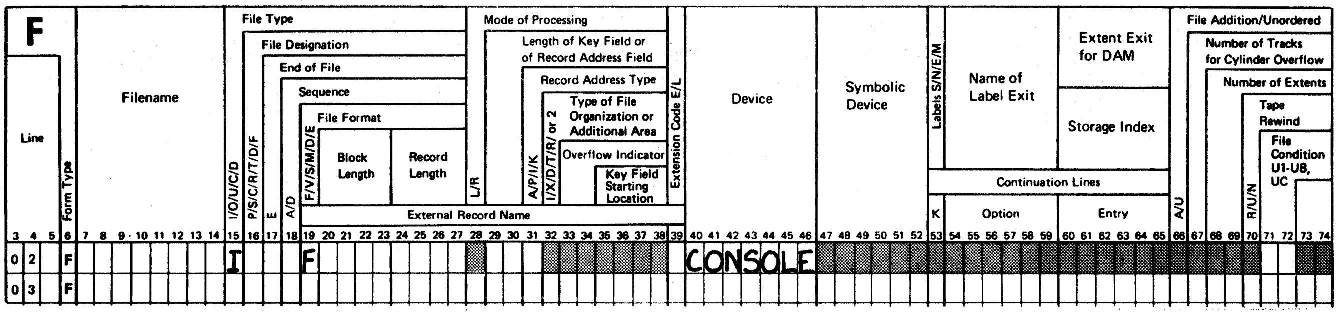
- A CRT file can be used only as an output file to display information. You cannot change this information by entering data at the keyboard.

Using a CONSOLE File

To use a CONSOLE file in a program, code entries in the file description specifications and in the input specifications.

File Description Specifications

Code entries in the unshaded columns of the file description specifications shown below:



Columns 7 through 14 must contain the name of the CONSOLE file.

Column 15 must contain I to indicate that the file is an input file.

Column 16 must contain P (primary), S (secondary), D (demand), or R (record address) to indicate how the program uses the file.

Column 17 can contain E if column 16 contains P, S, or R. It must be blank if column 16 contains D. E in column 17 indicates that the program must process every record from the file before the program can end. Blank indicates that the program can end whether or not every record is processed. If this column is blank for every file, the program must process every record from every file before the program can end. To indicate that all the records for a CONSOLE file have been entered, the person at the display station presses command key 12. For more information about command keys, see *Allowing Command Keys to Be Pressed* later in this chapter.

Column 18 can contain A or D if column 16 contains P or S. It must be blank if column 16 contains D or R. A indicates that the program checks that the records in the file are in ascending sequence. D indicates that the program checks that the records are in descending sequence. Blank indicates that the record sequence is not checked.

Column 19 must contain F or blank to indicate that every record in the file has the same length.

Columns 20 through 23 contain the length of a block of records. The block length must be equal to the record length, entered in columns 24 through 27, or be blank.

Columns 24 through 27 contains the record length. The record length must be the same as the highest number coded in columns 48 through 51 on the input specifications (the to field location). This record length cannot be less than 2 or greater than 1518. If the CONSOLE file is used as a record address file, determine the record length by multiplying the length of the record address field by 2. This record length cannot be less than 2 or greater than 58.

Columns 29 and 30 must be blank if column 16 contains P, S, or D. If column 16 contains R, columns 29 and 30 must contain the length of the key field of the indexed DISK file.

Column 31 is used only for record address files. Leave the column blank if the key fields in the record address file are the same as the key fields in the indexed DISK file. Enter A for an indexed DISK file with zoned-decimal key fields.

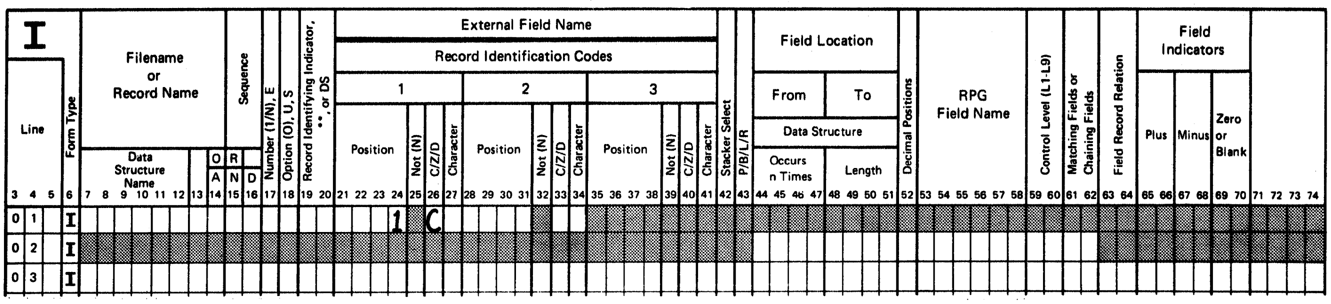
Column 39 must be blank if column 16 contains P, S, or D. If column 16 contains R for record address file, this column must contain E.

Columns 40 through 46 must contain CONSOLE as the device name.

Columns 71 and 72 can contain external indicators, U1 through U8.

Input Specifications

Input specifications are not required (and not allowed) for record address files. Therefore, if column 16 of the file description specifications contains R, do not code any input specifications for that file. However, if column 16 of the file description specifications contains P, S, or D, entries are required in the unshaded columns of the input specifications shown below:



File and Record Specifications

Columns 7 through 14 must contain the name of the CONSOLE file. The name must be the same as the file name on the file description specifications.

Columns 14 through 16 must not contain the characters AND; however, columns 14 and 15 can contain the characters OR. These OR lines can be used to indicate a relationship between record-identifying indicators or record types. If columns 14 and 15 contain OR, the same number of record identification codes must be described on this specification line as are described on the preceding line.

Columns 15 and 16 can contain any two alphabetic characters if you do not want the program to check the sequence of input records. Code a numeric entry (01 through 99) in these columns to assign a sequence number to each record type in the file. The maximum number of record types that you can use for a CONSOLE file is 10.

Column 17 must be blank if columns 15 and 16 contain alphabetic entries. If columns 15 and 16 contain numeric entries, code 1 in column 17 if the record type can consist of only one record, or code N if the record type can consist of one or more records.

Column 18 must be blank if columns 15 and 16 contain alphabetic entries. If columns 15 and 16 contain numeric entries, code O in column 18 if the record type is optional.

Columns 19 and 20 must contain a record-identifying indicator (01 to 10) to identify which command key the person at the display station enters to select this record type. You cannot use the same indicator to identify more than one record type within the input specifications for one program.

Column 24 must contain 1 to indicate that the record identification code is in position 1.

Column 26 must contain C to indicate that the entire character is used as the record identification code.

Column 27 must contain the character that is used as the record identification code in position 1 of the record. In an output only area of the display, the program automatically inserts a 1- or 2-character record identification code into positions 1 and 2 of each new record that is prompted.

Columns 28 through 34 must be blank if a 1-character record identification code is used. If a 2-character record identification code is used, code these columns the same as columns 24 through 27, except that column 31 must contain 2 to indicate record position 2.

Columns 44 through 47 must contain the record location in which the field begins.

Columns 48 through 51 must contain the record location in which the field ends. The maximum length for an alphameric field is 66 characters. The maximum length for a numeric field is 15 digits.

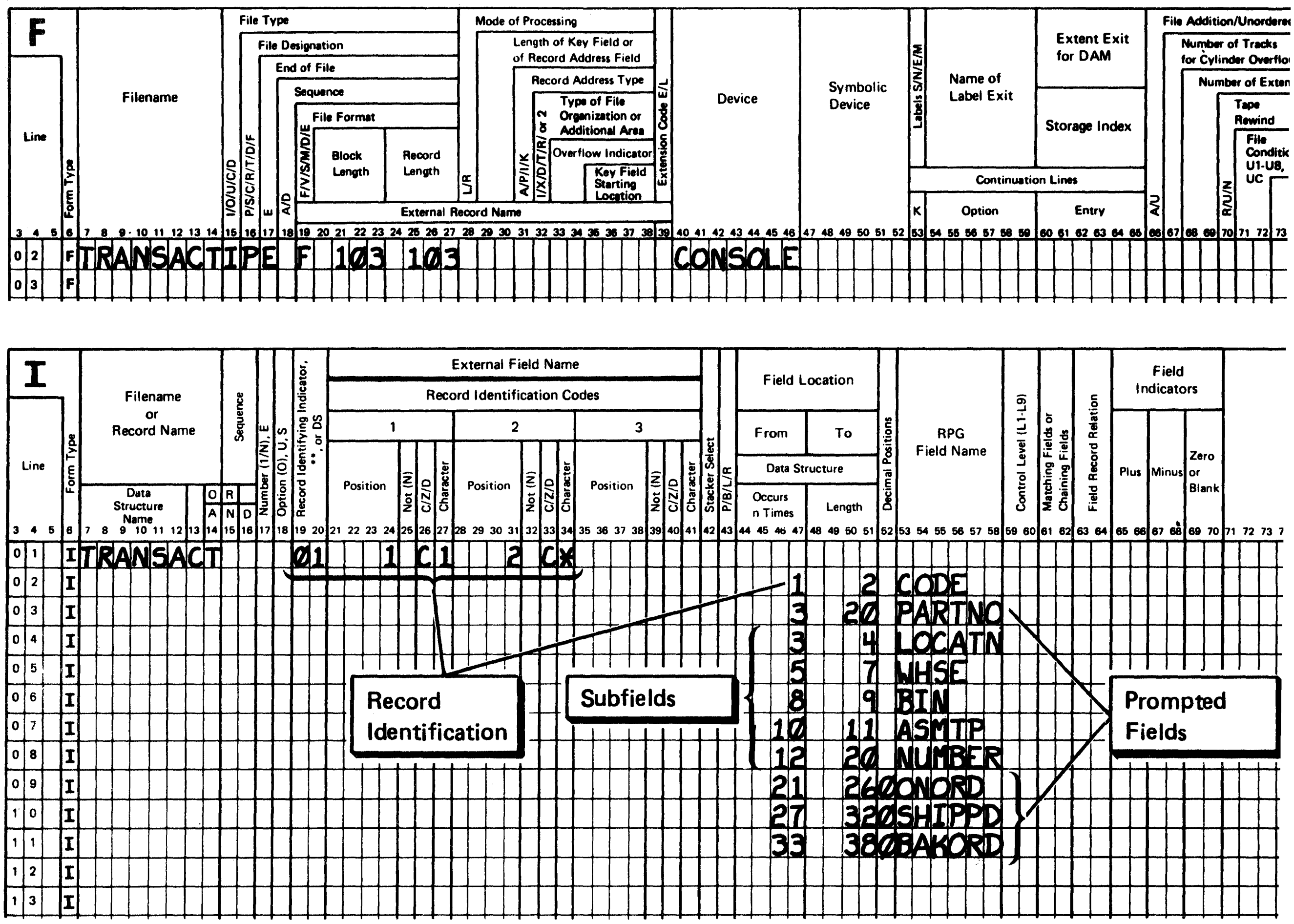
Subfields can be coded within the fields of a CONSOLE file record. The from and to field locations for subfields must not overlap the from and to field locations for another field. The program does not prompt for subfields, but it assigns values from the prompted field to subfields. You can use subfields in calculation and output specifications.

For example, in Figure 9-1, the part number 01ROC43CP843987831 is entered in response to the prompt field PARTNO. LOCATN, WHSE, BIN, ASMTP, and NUMBER are subfields within the PARTNO field. The values for the subfields are taken from the PARTNO field.

Columns 53 through 58 must contain a descriptive field name (one to six alphameric characters) to be used as a prompt for this data. To enter data into a whole array for a CONSOLE file, define the whole array as a subfield within a field of the CONSOLE file record, or define each element of the array with an index and place this entry in columns 53 through 58.

Columns 59 and 60 can contain a control-level indicator (L1 through L9) if this is a primary or secondary file. A control-level indicator indicates that a control break occurs when the contents of a field change.

Columns 61 and 62 can contain a match-field value (M1 through M9) if this is a primary or secondary file. Otherwise, leave these columns blank.



Prompted Field: PARTNO

<u>01ROC43CP843987831</u>

Subfields:

LOCATN:	01
WHSE:	ROC
BIN:	43
ASMTP:	CP
NUMBER:	843987831

Figure 9-1. Coding Subfields for a CONSOLE File

Creating Display Formats for CONSOLE Files

When you use a CONSOLE file in an RPG program, RPG automatically calls the RPGR procedure to create display formats for the file. The RPGR procedure uses the input specifications to create source input to the display format generator (the $SFGR utility) of the system support program (SSP). The display format generator compiles this source input and creates a load module containing the display formats for the program. For a complete description of display formats, see the manual *Creating Displays*.

To call the RPGR procedure directly, type RPGR on the command line of the display. Then you can press the Enter key or the Help key (in which case, the following display appears), or you can type RPGR and the parameters you want to use with the RPGR procedure and then press the Enter key (in which case, the following display does not appear):

```
                          RPGR PROCEDURE                      Optional-*

           Runs the RPG CONSOLE file display format generator.


Name of source program containing CONSOLE files . . . . . . . .  TEST

Size of $SOURCE file in blocks  . . . . . . . . . . . .  1-9999  40

Save display format source member . . . . . . . . . SAVE,NOSAVE  SAVE

Name of library containing source program . . . . . . . . . . .  YOURLIB

Name of library to contain format load member . . . . . . . . .             *

Replace duplicate members . . . . . . . . . .  REPLACE,NOREPLAC  REPLACE

Output option for format listing  . . . . . . . . PRINT,NOPRINT  PRINT


Cmd3-Previous menu     Cmd4-Put on job queue
                                                         © 1983 IBM Corp.
```

Respond to each prompt by entering the appropriate information.

Name of source program containing CONSOLE files. Enter the name of your RPG source program.

Size of $SOURCE file in blocks: Enter the number of blocks (each block is 2560 bytes) for the source file and SFGR file. If you do not specify a file size, 40 blocks is assumed.

Save display format source member. Enter SAVE or NOSAVE.

SAVE means that you want the source statements for the $SFGR utility to be saved in the library specified as the source input library. The name given to the saved source statements is the program name plus FM. (The program name is the name coded in columns 75 through 80 of the control specification.) For example, if the name of the program is PRNAME, the name of the display format source member and load module for that program is PRNAMEFM. The format name cannot be changed after compilation.

NOSAVE means that you do not want the source statements for the $SFGR utility to be saved.

If you do not choose an option, SAVE is assumed.

Name of library containing source program: Enter the name of the library that contains the RPG source program. If you do not specify a library name, the current library is assumed.

Name of library to contain format load member: Enter the name of the library that will contain the load module created by the $SFGR utility. If you do not specify a library name, the current library is assumed.

Replace duplicate members: Enter REPLACE or NOREPLAC.

REPLACE means that you want to replace an existing library member with the newly compiled library member that has the same name.

NOREPLAC means that, if another library member has the same name, you want an error message to be displayed.

If you do not choose an option, REPLACE is assumed.

Note: If you type the parameters for the RPGR procedure on the command line of your display instead of using the prompts, allow for the GEN parameter at this point. The only possible entry for this parameter is GEN. If you use the prompts, ignore this parameter. This parameter is included only for compatibility with the RPGR procedure on System/34.

Output option for format listing: Enter PRINT or NOPRINT.

PRINT means that you want a printed copy of the display formats created by the $SFGR utility and a listing of the $SFGR source specifications.

NOPRINT means that you do not want a printed copy of the display formats created by the $SFGR utility or a listing of the $SFGR source specifications.

If you do not choose an option, PRINT is assumed.

You can press the following keys from the RPGR Procedure display:

- Command key 4 to place the load module on the job queue
- Command key 7 to end the RPGR procedure

If more than one display format is created for the same program, the RPGR procedure adds FM to the program name to identify the entire set of display formats, and it adds the record-identifying indicator to the program name to identify each display format. Thus, if the program PRNAME contains three record types (identified by indicators 01, 02, and 03), the RPGR procedure creates the following names:

- PRNAMEFM, the name of the entire set of display formats
- PRNAME01, the name of the first display format in the set
- PRNAME02, the name of the second display format in the set
- PRNAME03, the name of the third display format in the set

If OR lines are used on the input specifications to identify the same record, only one format is associated with the record.

Using Displays

Displays prompt the person at the display station to enter data. The person presses command key 12 to indicate end of file (that is, there is no more data to enter). For information about command keys, see *Allowing Command Keys to Be Pressed* later in this chapter.

Display Formats

The top line on the display contains control information that is used by the person at the display station to identify the current record and to specify the next record type to be prompted (see Figure 9-2). The remaining 23 lines are used for the formatted record. The maximum number of input fields that can be displayed is 80.

For each field defined, the computer reserves 14 characters to contain the field name and its attributes. Therefore, the maximum record length is 1518 characters. (23 lines on the display format x 80 characters per line = 1840 characters on the display format. 23 lines x 14 characters reserved for each field (line) = 322 reserved characters. 1840 total characters - 322 reserved characters = 1518 characters available for the record.) The format actually created for a record depends on the size and number of fields in the record. The possible formats are:

- *One column.* The compiler creates a one-column display format whenever the number of fields prompted for is less than 24.

- *Two columns.* The compiler creates a two-column display format whenever the number of fields is 24 through 46 (see Figure 9-3). If any field is longer than 26 characters, the display format is changed to allow these fields.

- *Three columns.* The compiler creates a three-column display format whenever the number of fields is 47 through 69 (see Figure 9-3). If any field is longer than 12 characters, the display format is changed to allow these fields.

- *Four columns.* The compiler creates a four-column display format whenever the number of fields is 70 through 80 (see Figure 9-3). If any field is longer than 6 characters, the display format is changed to allow these fields.

If the format is changed so that the four-column format can not be used, error message RPG-1024, FORMAT FOR CONSOLE FILE DOES NOT FIT ON SCREEN, is displayed. This message means that you must reduce the number of fields in the record or change the order of the fields. Remember that all fields for the record must fit on one display format.

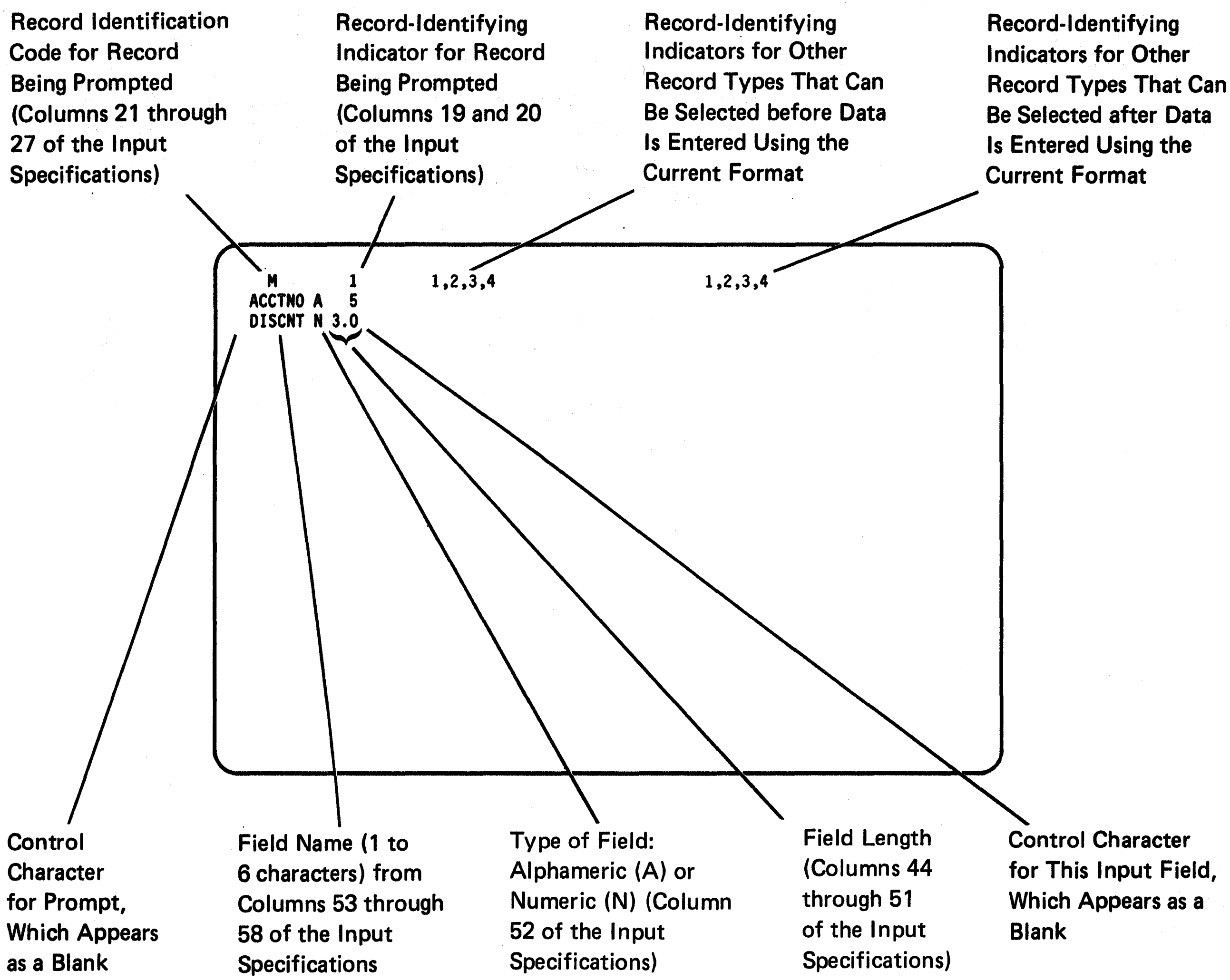


Figure 9-2. Display Format Created for a CONSOLE File

Two-Column Format

```
 M       1          1,2,3,4                        1,2,3,4
CUSTNO A  5                              PHONE  A  11
NAME   A  30
ADDR1  A  20                             ADDR2  A  20
ADDR3  A  20                             CITY   A  25
STATE  A  20                             ZIP    A   5
```

Three-Column Format

```
 M       1          1,2,3,4                        1,2,3,4
ACCTNO A  5               DISCNT N 3.3               COST    N10.2
SLSMAN A  40
DIST   A  3               REGION A   4
CITYST A  20                                         ZIP     A   5
```

Four-Column Format

```
 M       1          1,2,3,4                        1,2,3,4
ACCTNO A  5         DIST   A   3         REGION A   3         DISCNT N 4.4
SALES  N10.2                             TOTAL  N15.2
```

Figure 9-3. Display Formats Created for a CONSOLE File

Prompt Format

The display format generator uses the field names on the input specifications to create prompts for these display formats. The prompts are 14 positions long and have the following format:

Position	Explanation
1	Control character for the prompt. This character appears as a blank.
2-7	Field name.
8	Blank.
9	N for a numeric field, or A for an alphameric field.
10-13	Length of the field. For an alphameric field, positions 10 and 11 are blank, and positions 12 and 13 contain the length. For a numeric field, positions 10 and 11 contain the length of the field, position 12 contains a decimal point, and position 13 contains the number of decimal positions in the field.
14	Control character for the input field. This character appears as a blank.

Changing the Display Format

After the RPGR procedure has created the source input for the display format generator and the input has been cataloged in the library, you can change this source input, if you wish, by using the source entry utility (SEU) and the FORMAT procedure. For information about SEU, see the *Source Entry Utility Guide* For information about the FORMAT procedure, see the *System Reference* manual.

Erasing the CONSOLE File Buffer

To erase, or blank, the entire buffer for the CONSOLE file, use the SET operation with ERASE coded in the result field. Entries are required in the unshaded columns of the calculation specifications shown below:

C Line (3-5)	Form Type (6)	Control Level (L0-L9, LR, SR, AN/OR) (7-8)	Indicators: Not, And, Not, And, Not (9-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field Name (43-48)	Result Field Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Arithmetic Plus/Minus/Zero; Compare 1>2, 1<2, 1=2; Lookup (Factor 2) is High/Low/Equal (54-59)	Comments (60-74)
01	C				SET		ERASE					
02	C											

Columns 18 through 27 (factor 1) must be blank.

Columns 28 through 30 must contain the operation code SET.

Columns 33 through 42 (factor 2) must contain the name of the CONSOLE file.

Columns 43 through 47 (the result field) must contain ERASE.

ERASE causes the RPG program to change the contents of the buffer to blanks just before the program reads a record at the beginning of the next program cycle. Because the buffer is not erased until the beginning of the next program cycle, the program continues to process the current record after the ERASE operation occurs.

If the ERASE operation occurs because of invalid input data, you should insert code in your program to avoid further calculations and to return to the start of the program cycle. Then the person at the display station can enter a correct form of the record containing invalid input data and can reenter any records that were entered after that record.

Using a CONSOLE File with KEYBORD and CRT Files

When a program uses a CONSOLE file, a KEYBORD file, and a CRT file, and the person at the display station is entering data for the CONSOLE file, the following occurs when a KEY or SET operation for the KEYBORD file occurs:

1. The person must finish entering data for the current record in the CONSOLE file.

2. The prompt for the SET or KEY operation, or the output to the CRT file, is then displayed.

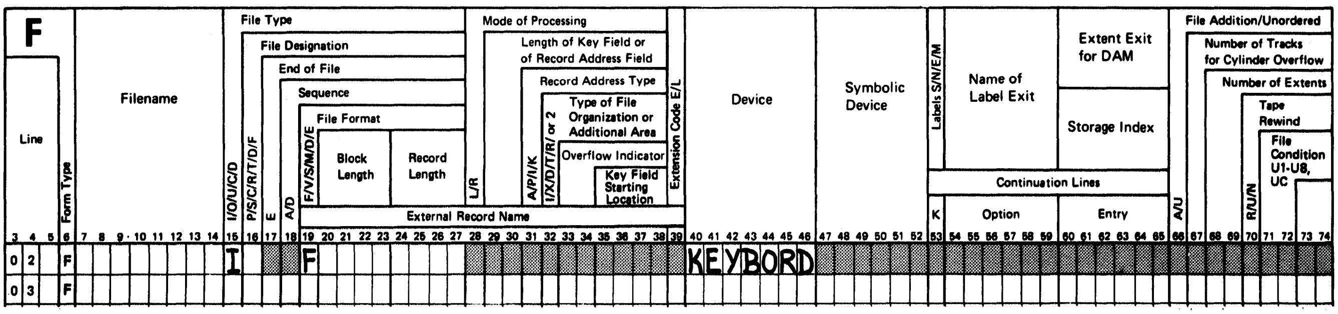
3. Normal processing of the CONSOLE file continues after the SET or KEY operation is completed. That is, the person at the display station can enter data for the next records in the CONSOLE file during the next program cycle.

Using A KEYBORD File

A KEYBORD file can be used as both an input and an output file. To use a KEYBORD file, you must code file description specifications. You do not code input or output specifications, however. Instead, you describe the data on the calculation specifications for the KEY operation or for the KEY and SET operations.

File Description Specifications

To create a KEYBORD file, code entries in the unshaded columns of the file description specifications shown below:



Columns 7 through 14 must contain the name of the file.

Column 15 must contain I to indicate that the file is an input file.

Column 16 can contain P (primary) or D (demand) to indicate how the program uses the file. If you use a KEYBORD file as a primary input file, no other files can be used as primary or secondary files. In this case, you must provide an exit for your program by turning on the last-record indicator in the calculation specifications. If you use a KEYBORD file as a demand file, you use the KEY operation, not the READ operation, to read records from the file.

Column 19 must contain F or blank to indicate that every record in the file has the same length.

Columns 20 through 23, the block length, must equal the record length coded in columns 24 through 27 or be blank.

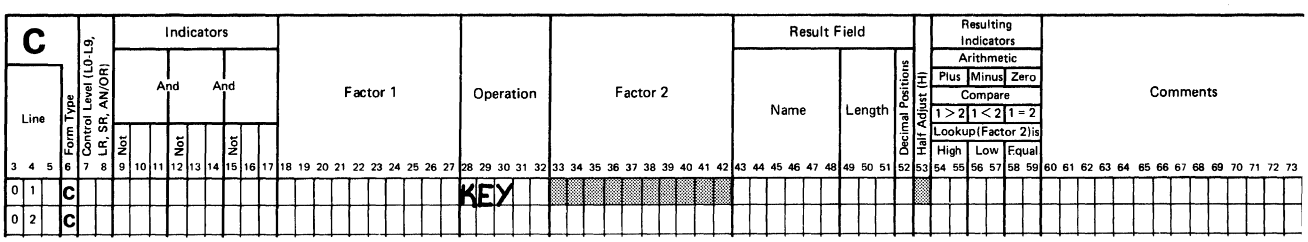
Columns 24 through 27 must contain the length of the largest field to be entered. This number must equal the largest field length coded in columns 49 through 51 of the calculation specification for the KEY operation. If you use the KEY operation to display a message, you must also consider the length of the message when you code the record length for the KEYBORD file. The maximum length for an alphameric field is 79 characters. The maximum length for a numeric field is 15. If the record length coded for a KEYBORD file is 40 or less, a display of six lines with 40 characters per line is centered both vertically and horizontally. If the record length is more than 40, the display consists of 24 lines with 79 characters per line.

Columns 40 through 46 must contain KEYBORD.

Calculation Specifications for a KEY Operation

Although a KEYBORD file is an input file, you do not code input specifications for a KEYBORD file. Instead, you define the input data on the calculation specifications for a KEY operation. The KEY operation causes a pause in calculations. During that pause, the person at the display station can enter data from the keyboard.

To use the KEY operation, code entries in the unshaded columns of the calculation specifications shown below:



Columns 7 and 8 can contain a control-level indicator (L1 through L9), AN, OR, or blanks. Leave these columns blank if the KEY operation is not part of a subroutine or if it occurs only at detail time.

Columns 9 through 17 can contain conditioning indicators, command-key indicators (KA through KN, KP through KY) coded in a SET or SETOF operation, or blanks.

Columns 18 through 27 (factor 1) can contain the constant, literal, field name, or table or array element to be displayed.

Columns 28 through 30 must contain the operation code KEY.

Columns 31 and 32 can contain the message identification code (01 through 99) corresponding to the message to be displayed. For information about message identification codes, see *Using a Message Member* later in this chapter. The message itself is in your message member. For information on how to create a message member, see *$MGBLD Utility Program* in the *System Reference* manual. This message prompts the person at the display station to do a KEY operation. An entry is required in columns 31 and 32 when columns 18 through 27 are blank. If you do not code a control language MEMBER statement that specifies your message member before you run the program, or if columns 31 and 32 contain a message identification code that does not correspond to a message in your message member, the computer displays the prompt nn-MESSAGE INDICATOR, where nn is the contents of columns 31 and 32. If factor 1 contains an entry that prompts the KEY operation, the message identification code in columns 31 and 32 is ignored.

Columns 33 through 42 (factor 2) must be blank.

Columns 43 through 48 (the result field) can contain the name of the field to be entered.

Columns 49 through 51 must contain the length of the field to be entered if the field is not defined somewhere else. The maximum length for a numeric field is 15. The maximum length for an alphameric field is 40 if the record length is 40 or less, or 79 if the record length is more than 40.

Column 52 must be blank for alphameric fields. For numeric fields, enter the number of decimal positions (0 through 9) in the field to be entered if that field is not defined somewhere else.

Columns 54 through 59 can contain resulting indicators (01 through 99) to test the condition of a numeric field (plus, minus, or zero) or to test an alphameric field for blanks (columns 58 and 59).

Figure 9-4 shows examples of KEY operations.

Keying operations with user message member prompts.

The following operations allow the operator to key a numeric field (FIELDA) and an alphabetic field (FIELDB). These fields have not been defined previously. The operations are prompted by messages 0001 and 0002 from the user message member, respectively.

Line	Form Type	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Resulting Indicators
10	C		KEYØ1		FIELDA	5	Ø	
11	C		KEYØ2		FIELDB	12		

The following operation allows the operator to key a numeric field defined previously. This field is tested for a plug, minus, or zero condition. The operation is prompted by user message 0030.

Line	Form Type	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Resulting Indicators
18	C		KEY3Ø		AMOUNT			Ø1Ø2Ø3

Display keying operations with factor 1 prompts.

The following operations cause the previously defined field (FIELDC) in factor 1 to be displayed and then allow the operator to key a numeric field (FIELDA). The numeric literal 40 is displayed and the operator is allowed to key an alphabetic field (FIELDB). FIELDA and FIELDB are not defined elsewhere. Note that factor 1 overrides user messages 0004 and 0005.

Line	Form Type	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Resulting Indicators
11	C	FIELDC	KEYØ4		FIELDA	5	Ø	
12	C	4Ø	KEYØ5		FIELDB	12		

The following operation displays the alphameric literal specified in factor 1 (ALTER) on the display screen. The operator is then allowed to key data into the numeric field specified in the result field defined elsewhere. Factor 1 overrides user message 0006.

Line	Form Type	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Resulting Indicators
20	C	'ALTER'	KEYØ6		AMOUNT			Ø4Ø5Ø6

Figure 9-4. Possible KEY Operations

Using a KEY Operation

As the person at the display station types data, it is displayed in one of two formats:

- If the record length is 40 or less, the display consists of six lines with 40 characters per line. The display is centered vertically and horizontally.
- If the record length is more than 40, the display consists of 24 lines with 79 characters per line. The computer reserves one character per line for field attributes.

When the person at the display station uses the KEY operation, the contents of the result field depend on the person's response. The possible responses are:

- *The person types the data and presses an entry key.* The person can use any of the following as an entry key: Field Exit, Field-, Field+, or Enter. However, if the person enters data into a numeric field, the Enter key cannot be used as an entry key. If the person does not type data into all positions of a numeric field, the computer moves the data into the rightmost positions of the field and puts zeros in the unused positions to the left. If the person does not type data into all positions of an alphameric field, the computer leaves the data in the leftmost positions of the field and puts blanks in the unused positions to the right.
- *The person presses only an entry key.* This action causes the computer to change any numeric data in the result field to zeros or any alphameric data to blanks.
- *The person presses the Dup key and then an entry key.* This action does not change the data in the result field.

Bypassing a KEY Operation

When the KEY operation causes a pause in the calculations, the person at the display station can go to the next calculation without entering any data for the current calculation. To do this, the person simply presses an entry key. This action causes the data in the result field to be changed to zeros or blanks. After each KEY operation (regardless of whether data is entered), the person must press an entry key before the next operation can occur.

Using a Message Member

You can create messages or prompts to be displayed during your RPG program. These messages or prompts must be in your message members (see *$MGBLD Utility Program* in the *System Reference* manual). The messages or prompts are displayed when you use a halt indicator (H1 through H9) or a message identification code on a KEY or SET operation. You must assign message identification codes 0001 through 0109 to specific kinds of messages in the message member:

Message Identification Code	Kind of Message
0001-0099	Message to be displayed as coded in columns 31 and 32 of a KEY or SET operation
0100	Message to be displayed at the end of a program cycle after all halt indicators are processed
0101-0109	Message to be displayed at the end of a program cycle in which a halt indicator (H1 through H9) occurs (0101 through 0109 correspond to H1 through H9)

For a message contained in a message member to be displayed, the message text must be in an object message member. The message member must be coded in the control language MEMBER USER1 statement, and the RPG program must use a KEY or SET operation or a halt indicator. (For information about the control language MEMBER USER1 statement, see *MEMBER Statement* in the *System Reference* manual.)

For each message 0101 through 0109 (corresponding to halt indicators H1 through H9), you can add a second-level message containing up to 225 characters. The second-level message must have the same message identification code as the first, and the second-level message member must be coded in the control language MEMBER USER2 statement. After halt indicators H1 through H9 turn on, the program does all calculations and detail output operations for the record before processing ends and a message is displayed. If the halt indicators turn on during the processing of the last record in a file, the program does not stop processing but continues to completion.

Figure 9-5 shows how to code the calculation specifications required to display a message.

The messages that are displayed as a result of the calculation specifications shown in Figure 9-5 depend on whether you coded a control language MEMBER statement before running the program.

If the control language statements are:

```
// LOAD USER
// RUN
```

then the messages are displayed from the system message member. 01-MESSAGE INDICATOR is the message displayed as the prompt for the KEY operation. If the person at the display station types HALT, halt indicator H1 turns on and the computer displays message 9101, RPG II INDICATOR H1 IS ON. If the person enters option 0 in response to that message, the computer displays message 9100, ALL HALT INDICATORS HAVE BEEN DISPLAYED.

If the control language statements include a MEMBER statement:

```
// LOAD USER
// MEMBER USER1-MESG1
// RUN
```

then the displayed messages come from your message member MESG1, which is coded on the MEMBER statement. The prompt TYPE HALT TO END THE PROGRAM is the text displayed for the KEY01 operation. This prompt is the contents of your message 0001 in MESG1. Later, when the person at the display station types the literal HALT, the message 0101, HALT HAS BEEN ENTERED WITH A KEY OPERATION, is displayed. If the second message, 0100, has not been loaded into your message member, it cannot be displayed. Instead, the message MESSAGE NOT FOUND IN SPECIFIED MESSAGE MEMBER is displayed.

Use the Source Entry Utility or the $MAINT utility to load the source member (MESG1 for this example) into a library. The message object member MESG1 must exist before you run the program. For information on creating the message source member, see the *Source Entry Utility (SEU) Guide*. For information on creating a message load member, see the *System Reference* manual.

Line (3-5)	Form Type (6)	Control Level (LO-L9, LR, SR, AN/OR) (7-8)	Indicators: Not (9)	(10-11)	And: Not (12)	(13-14)	And: Not (15)	(16-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field: Name (43-48)	Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Plus / High / 1>2 (54-55)	Minus / Low / 1<2 (56-57)	Zero / Equal / 1=2 (58-59)	Comments (60-74)
01	C									KEYØ1		FLD	4						
02	C								FLD	COMP	'HALT'							H1	
03	C																		

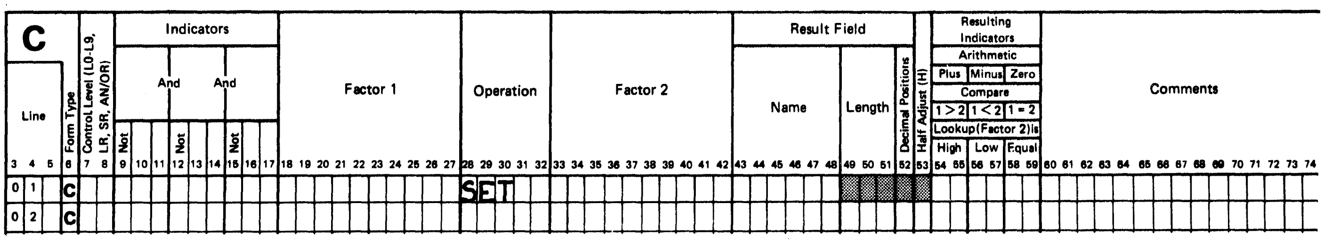
Figure 9-5. Calculation Specifications Required to Display a Message

Calculation Specifications for a SET Operation

The SET operation allows any or a combination of the following to occur for a KEYBORD file:

- Command keys to be pressed
- The field, literal, or array or table element coded in factor 1 to be displayed
- Messages from your message member to be displayed. The message that is displayed is determined by the message identification code in columns 31 and 32 of the SET operation.
- The buffer for a CONSOLE file to be erased if ERASE is coded in the result field of the SET operation.

To use the SET operation, code entries in the unshaded columns of the calculation specifications shown below:



Columns 7 and 8 can contain a control-level indicator (L1 through L9). However, leave these columns blank if the SET operation is not part of a subroutine or if it is to be done only at detail time.

Columns 9 through 17 can contain a conditioning indicator. However, leave these columns blank if the SET operation is to be done on every program cycle.

Columns 18 through 27 (factor 1) can contain the constant, literal, field name, or table or array element to be displayed.

Columns 28 through 30 must contain the operation code SET.

Columns 31 and 32 can contain the message identification code (01 through 99) corresponding to the message to be displayed. This message must be in your message member. (For information about how to create a message member, see *$MGBLD Utility Program* in the *System Reference* manual.) The message prompts the person at the display station to do a SET operation. An entry is required in columns 31 and 32 when columns 18 through 27 (factor 1) are blank and columns 54 through 59 contain a command key. However, if you code an entry in factor 1 and a message identification code in columns 31 and 32, the message identification codes are ignored. If you do not code a control language MEMBER statement that specifies your message member before you run the program, or if columns 31 and 32 contain a message identification code that does not correspond to a message in your message member, the prompt nn-MESSAGE INDICATOR is displayed, where nn is the contents of columns 31 and 32. If both factor 1 and a message identification code are specified, the message identification code is ignored.

Columns 33 through 42 (factor 2) must contain the name of the CONSOLE file if ERASE is coded in columns 43 through 48 (the result field). For all other SET operations, leave columns 33 through 42 blank.

Columns 43 through 48 (the result field) must contain ERASE if the name of a CONSOLE file is coded in columns 33 through 42 (factor 2). For all other SET operations, leave columns 43 through 48 blank.

Columns 49 through 53 must be blank.

Columns 54 through 59 can contain one, two, or three command keys (KA through KN, or KP through KY) that the person at the display station can press when the program is at this specification line. If only one or two command keys are used, you can code them in any of the three sets of columns. When the person at the display station presses a command key coded in these columns, that command key indicator turns on and stays on until it is used again in a SET operation or until it is turned off by the SETOF operation. If the person at the display station presses a command key other than those coded in columns 54 through 59 of a SET operation, the program stops. Several lines can be displayed before the program stops for input if you stack SET operations with factor 1 or a message identification code and no command key entries (see Figure 9-6). The program does not stop until a command key is pressed or a KEY operation occurs.

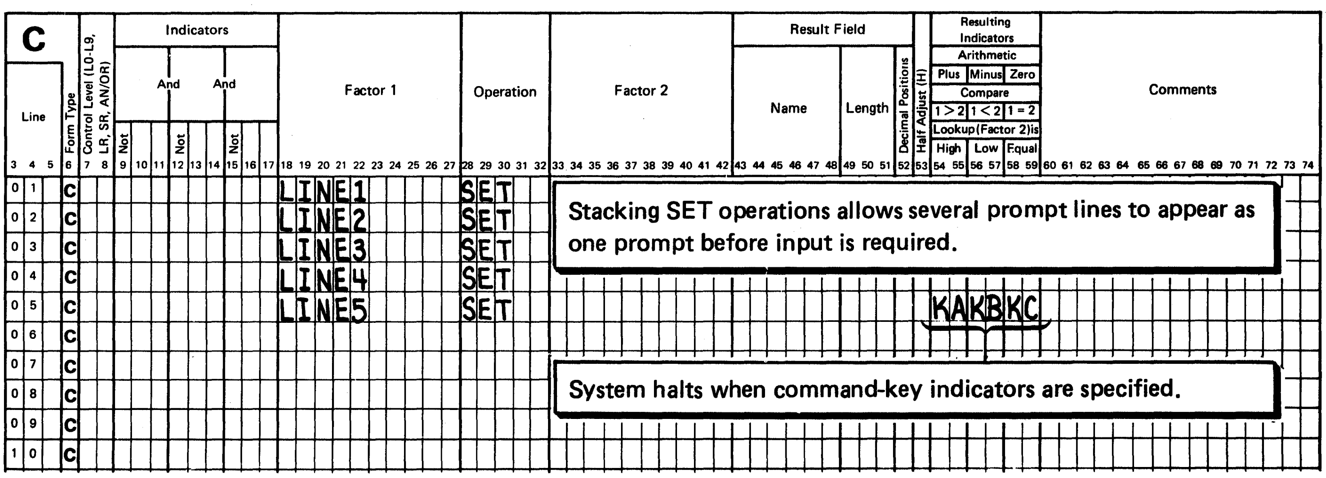


Figure 9-6. Using SET Operations to Display a Prompt with More Than One Line

Figure 9-7 is a summary of calculation specifications for SET operations

Line	Form Type	Factor 1	Operation	Factor 2	Result Field Name	Resulting Indicators
01	C					
02	C	Display the contents of the first element of the array PROMPT.				
03	C					
04	C					
05	C	PROMPT,1	SET			
06	C					
07	C	Allow command keys to be pressed. Prompt the operation by displaying the contents of the field SELECT.				
08	C					
09	C					
10	C	SELECT	SET			KAKB
11	C					
12	C	Display message 0013 from the user message member.				
13	C					
14	C		SET13			
15	C					
16	C	Erase or blank the existing CONSOLE buffer of the specified CONSOLE file.				
17	C					
18	C		SET	CONSFILE	ERASE	
19	C					
20	C	Allow command key 1, 2, or 3 to be pressed. Prompt the operation by MIC 0023.				
	C					
	C					
	C		SET23			KAKBKC
	C					

Figure 9-7. Summary of Calculation Specifications for SET Operations

Figure 9-8 shows possible combinations of uses for the SET operation.

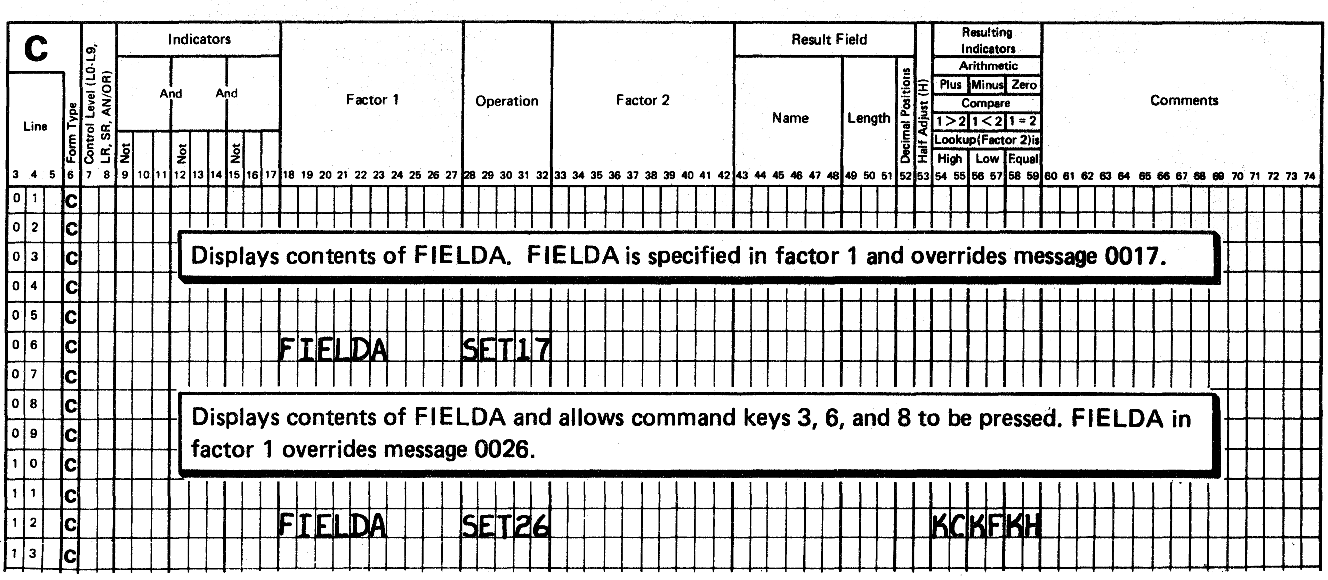


Figure 9-8. Possible Combinations of Uses for SET Operations

Allowing Command Keys To Be Pressed

The SET operation allows you to specify which command keys the person at the display station can press when the program is at a certain specification line. When the person presses a command key, the corresponding command-key indicator can be used to condition calculation or output operations that follow. Command-key indicators remain on until they are used again in a SET operation or until they are turned off by the SETOF operation.

When the program is at a certain specification line, you can allow the person at the display station to press one to three command keys. For each command key to be pressed, the person first presses the Cmd key and then presses the number key at the top of the keyboard that corresponds to the command-key indicator. After all command-key responses are entered, the person presses an entry key.

There are 24 command keys. Each one corresponds to a separate command-key indicator:

Command Key	Command-Key Indicator	Keybcard Keys to Press	Command Key	Command-Key Indicator	Keyboard Keys to Press
1	KA	Cmd, 1	13	KM	Cmd, Shift, |
2	KB	Cmd, 2	14	KN	Cmd, Shift, @
3	KC	Cmd, 3	15	KP	Cmd, Shift, #
4	KD	Cmd, 4	16	KQ	Cmd, Shift, $
5	KE	Cmd, 5	17	KR	Cmd, Shift, %
6	KF	Cmd, 6	18	KS	Cmd, Shift, ¬
7	KG	Cmd, 7	19	KT	Cmd, Shift, &
8	KH	Cmd, 8	20	KU	Cmd, Shift, *
9	KI	Cmd, 9	21	KV	Cmd, Shift, (
10	KJ	Cmd, 0	22	KW	Cmd, Shift,)
11	KK	Cmd, -	23	KX	Cmd, Shift, _
12	KL	Cmd, =	24	KY	Cmd, Shift, +

Note: The keyboard keys may vary, depending on what type of keyboard you have.

If the person at the display station presses the wrong command key and does not press an entry key, the person can reset all the command keys by pressing the Cmd key and then pressing the character backspace (Clear) key while holding down the Shift key. The person can then retype the correct keys. If the person presses a command key that is not specified in the SET operation, error message RPG-9049, COMMAND KEY PRESSED IS NOT DEFINED, is displayed.

If no command keys are to be pressed, the person responds to the SET operation by pressing only an entry key. This action turns off the command key indicators. Coding your program to allow this response is not recommended because the person at the display station could make this response accidentally. For example, the person could forget to press the Cmd key before pressing a number key and an entry key. This action turns off the command key indicator that the person actually wanted to use.

Using the SET and KEY Operations Together

Normally, the person at the display station must press an entry key after doing each KEY operation or after pressing command keys coded in a SET operation. However, it is possible to combine these operations so that the person can press command keys (coded in columns 54 through 59 of a SET operation), type a field (specified in a KEY operation), and press an entry key only once (see Figure 9-9).

This combination is possible only if:

- The SET operation immediately precedes the KEY operation.
- The SET and KEY operations are conditioned by the same indicators (columns 7 through 17), coded in the same order.
- The SET and KEY operations use the same message identification codes in columns 31 and 32. These columns can be blank in both operations if factor 1 is used to display messages.
- Factor 1 for the SET and KEY operations can be the same, different, or missing from one operation.

If factor 1 is coded for both the SET and KEY operations, the contents of both factor 1's are displayed.

If the data field is numeric, the person must first press the specified command key, type the field, and then press the Field Exit, Field+, or Field- key. The Enter key cannot be used as an entry key for a numeric field.

If the data field is alphameric, the person must perform the same sequence of steps if the Field Exit, Field+, or Field- is pressed. However, if the Enter key is used, the person can press the command key and then type the field, or type the field and then press the command key, before pressing the Enter key.

KEY operations can occur at several points in a program. Instead of coding these KEY operations and related SET operations every time they occur, you can code them once in a subroutine. Then, call the subroutine each time it is needed.

Line (3-5)	Form Type (6)	Control Level (LO-L9, LR, SR, AN/OR) (7-8)	Indicators (9-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field Name (43-48)	Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Plus / 1>2 / High (54-55)	Minus / 1<2 / Low (56-57)	Zero / 1=2 / Equal (58-59)	Comments (60-73)
01	C													
02	C			In the following operations the operator can respond to one or all three of the command keys specified in columns 54 through 59 and key the price field before pressing an entry function key. The operation is prompted by user message 0068.										
03	C													
04	C													
05	C													
06	C													
07	C				SET68						KA	KB	KC	
08	C				KEY68		PRICE	5	0					
09	C													

Figure 9-9. Using the SET and KEY Operations Together

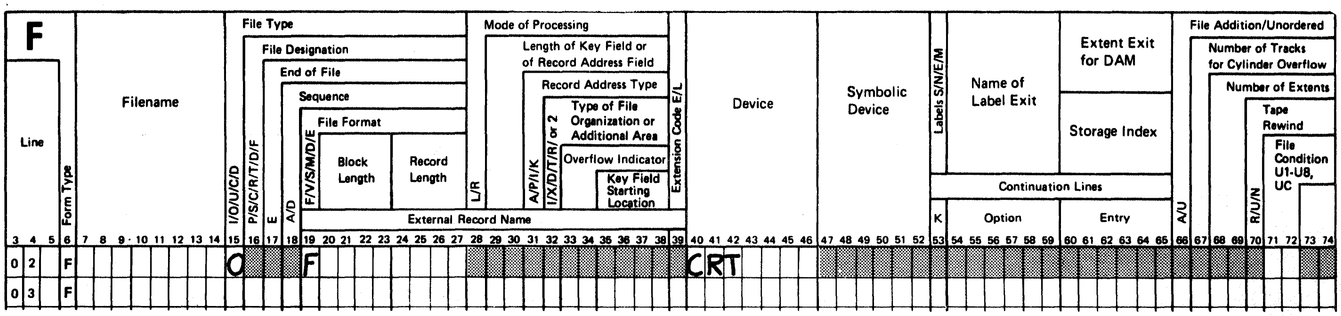
Using a CRT File

The CRT (cathode ray tube), or display screen, is designed to display messages and instructions to the person at the display station and to display that person's responses. You should not use it like the printer as a major output device because data moves on and off the screen too fast.

To use a CRT file, you must code both file description and output specifications.

File Description Specifications

Code entries in the unshaded columns of the file description specifications shown below:



Columns 7 through 14 must contain the name of the CRT file.

Column 15 must contain O to indicate that the file is an output file.

Columns 20 through 23, the block length, must equal the record length coded in columns 24 through 27 or be blank.

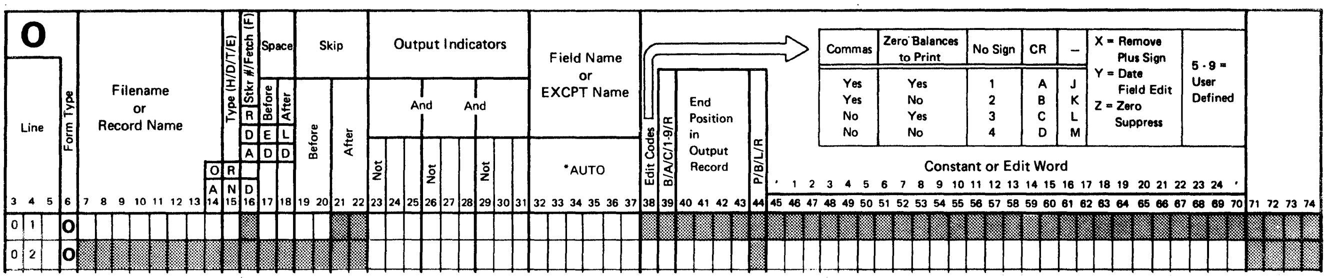
Columns 24 through 27 must contain the length of the largest record in the file. The maximum length is 79.

Columns 40 through 42 must contain CRT.

Columns 71 and 72 can contain an external indicator (U1 through U8).

Output Specifications

Because a CRT file is an output file, entries are also required on the output specifications. Code the unshaded columns on the output specifications shown below:



File- and Record-Identification Entries

Columns 7 through 14 must contain the name of the file.

Column 15 must contain H (heading), D (detail), T (total), or E (exception) to indicate the type of record to be written.

Column 17 can contain a number from 0 to 3 to indicate how many lines to leave blank before writing the current line.

Column 18 can contain a number from 0 to 3 to indicate how many lines to leave blank after writing the current line. If the CRT file has a record length (columns 24 through 27 of the file description specifications) of 40 or less, columns 17 and 18 of the output specifications cannot both contain 3. Data moves onto the screen from bottom to top. Therefore, if you code an entry in column 18 for the bottom line of a full screen, the top line moves off the screen.

Columns 19 and 20 can contain 01 or blanks. An entry of 01 tells the computer to clear the display before writing a record. If you code an entry other than 01 in columns 19 and 20, the computer assumes that the entry is 01 and erases the display.

Columns 23 through 31 can contain output conditioning indicators.

Columns 32 through 37 can contain an EXCPT name if column 15 contains E.

Field-Description Entries

Columns 23 through 31 can contain indicators.

Columns 32 through 37 can contain the names of the individual fields in the record.

Column 38 can contain an edit code.

Column 39 can contain B to indicate that the field is reset to blank or zero.

Columns 40 through 43 can contain the end position of each field in the output record.

Columns 45 through 70 can contain an edit word or literal constant.

Displaying Data

Data is displayed at the normal output times (total and detail) or at calculation time for exception output. (See *Column 15* in Chapter 27, *Output Specifications*, for information on exception output.) Any alphameric character can be displayed. If the record length is 40 or less, up to 40 characters can be written across the width of the screen, and a maximum of six such lines can appear at one time. The display is centered both vertically and horizontally. If the record length is more than 40, up to 79 characters can be written across the width of the screen, and a maximum of 24 lines can appear at one time.

Chapter 10. Using a BSCA File

Chapter 10. Using a BSCA File

A BSCA file is one way to send and receive data between your System/36 and another system. The letters BSCA stand for *binary synchronous communications adapter*. The *adapter* is part of the hardware. Batch BSC is part of the System Support Program (SSP). It allows you to *communicate binary* data (data represented as 0's and 1's) that is *synchronized* (the sending and receiving of data is controlled by timing signals).

Compared with another way to send and receive data, called the Interactive Communications Feature (SSP-ICF), batch BSC has several limitations:

- It is less efficient for interactive communications, so it is normally used for batch communications
- It can be used only between systems that use BSC (see *Systems That Use BSC* in this chapter).
- BSC does not allow a program sending data or receiving data to start programs on the other system.

To avoid these limitations, you may want to use SSP-ICF. For information about SSP-ICF, see the manuals *Interactive Communications Feature: Guide and Examples* and *Interactive Communications Feature: Reference.*

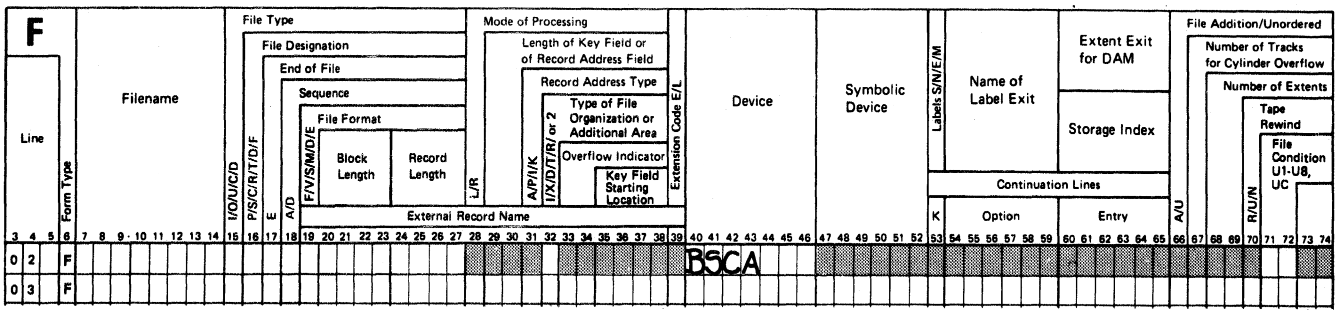
Defining a BSCA File

Defining a BSCA file requires entries on the file description specifications and on the telecommunications specifications.

In addition, the control specification must contain a blank or I in column 37 to indicate that the program does not recognize an inquiry request. A BSC program must not be interrupted, because an interruption might cause the remote system to stop communicating.

File Description Specifications

Code entries in the unshaded columns of the file description specifications shown below:



Columns 7 through 14 must contain the name of the BSCA file. The same name must be used in columns 7 through 14 of the telecommunications specification.

Column 15 must contain I or O to indicate that this file is an input (receive) file or an output (send) file.

Column 16 must contain P (primary), S (secondary), T (table), or D (demand) if column 15 contains I. If column 15 contains O, column 16 must be blank. D (demand) is the required entry when you use the file for interspersed sending and receiving of data. D should also be used for any receiving program that does not process the BSCA files immediately. For example, if the BSCA file is defined as a secondary file, the communications line opens as soon as the program begins; as a result, your wait time might be used up before you are ready to process the BSCA file. However, if the BSCA file is defined as a demand file, the line opens when the program is ready to receive the first record from the BSCA file.

Column 17 can contain E or blank if column 16 contains P, S, or T. Column 17 must be blank if column 16 contains D or blank. Enter E if end of file on the input (receive) file is to determine the end of the program. The BSCA file might be the only file with an E in column 17. However, if any other input file has an E in column 17, all BSCA input files should also have an E in column 17. This E is not necessary for the BSCA files; however, when it is not specified and the program reaches the end of another input file, the BSCA files close and the system on the other end of the communications line has no indication of what happened. When an E is specified for the BSCA files, all systems can end the program successfully.

Column 18 can contain A, D, or blank if column 16 contains P, S, or T. Column 18 must be blank if column 16 contains D or blank. A indicates that the program checks that the records in the file are in ascending sequence. D indicates that the program checks that the records in the file are in descending sequence. Blank indicates that the program does not check the record sequence.

Column 19 must contain F or blank to indicate that all records in the file have the same length.

Columns 20 through 23 must contain the block length of data processed by BSC. The block length must be a multiple of the record length. The maximum block length is 4075 positions.

Columns 24 through 27 must contain the record length. If you leave these columns blank, the program uses the maximum record length, which is 4075 positions. If you receive a record that has a length of zero, the record is ignored unless the other system is in 3740 mode, in which case the record is considered a file separator. If you receive a record that has a length greater than zero but shorter than the record size specified, the remainder of the record contains blanks.

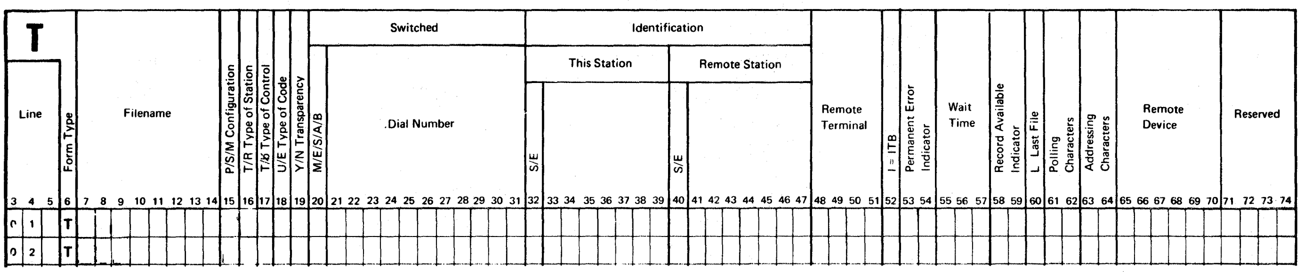
Column 32 can contain any number 1 through 9 or a blank. A number indicates that the program uses two input/output areas. A blank indicates that the program uses only one input/output area.

Columns 40 through 43 must contain the device name BSCA.

Columns 71 and 72 can contain an external indicator, U1 through U8.

Telecommunications Specifications

Code entries in the unshaded columns of the telecommunications specifications shown below:



Columns 7 through 14 must contain the name of the BSCA file. This must be the same name coded on the file description specifications for the BSCA file.

Column 15 must contain P, M, S, or blank. P or blank indicates that this is a point-to-point nonswitched network. M indicates that this is a multipoint network in which the control station selects the tributary station through polling or addressing. System/36 cannot be the control station. (If this column contains M, column 17 must contain T.) S indicates that this is a point-to-point switched network.

Column 16 must contain T or R. T indicates that this station transmits data from this BSCA file, which must be defined on the output specifications. R indicates that this station receives data in the BSCA file, which must be defined on the input specifications. The entry in column 16 is independent of the entry in column 20.

Column 17 must contain T or blank. T indicates that this is a tributary station on a multipoint network. (Column 17 must contain T if column 15 contains M.) A blank indicates that polling is not used. System/36 cannot be the control station.

Column 18 must contain A, U, E, or blank. A or U indicates that ASCII transmission control characters are used and that each station must provide file translation when it is required. E or blank indicates that EBCDIC transmission control characters are used.

Column 19 must contain Y, N, or blank. Y indicates that EBCDIC transparency is used. That is, the data being transferred can be packed-decimal numeric or alphameric and can contain transmission control characters. If column 19 contains Y, column 18 must contain E or blank. N or blank in column 19 indicates that EBCDIC transparency is not used. That is, the data being transferred is zoned-decimal numeric or alphameric and does not contain transmission characters.

Column 20 must contain M, A, B, or blank. M indicates that the person using this program makes the connection by dialing the number manually. A indicates that the program uses autoanswer. B indicates that the program uses manual answer. Blank indicates that this is not a switched network. The entry in column 20 is independent of the entry in column 16.

Column 32 must contain E, S, or blank. E indicates that this station's identification is the entry in columns 33 through 39. S indicates that this station's identification is at the position specified by the symbolic name in columns 33 through 39. Blank indicates that this station uses no identification.

Columns 33 through 39 must contain this station's actual identification (if column 32 contains E) or the symbolic name of the location of this station's identification (if column 32 contains S).

Column 40 must contain E, S, or blank. E indicates that the remote station's identification is the entry in columns 41 through 47. S indicates that the remote station's identification is at the position specified by the symbolic name in columns 41 through 47. Blank indicates that the remote station uses no identification.

Columns 41 through 47 must contain the remote station's actual identification (if column 40 contains E) or the symbolic name of the location of the remote station's identification (if column 40 contains S).

Column 52 must contain I or blank. I indicates that intermediate block checking is used. Blank indicates that intermediate block checking is not used.

Columns 53 and 54 can contain a permanent-error indicator (01 through 99, L1 through L9, LR, or H1 through H9) or blanks.

Columns 55 through 57 can contain the number of seconds (1 through 999) that BSC waits with no messages sent or received before a permanent error occurs. If you leave these columns blank, BSC waits 180 seconds before a permanent error occurs.

Columns 58 and 59 can contain a record-available indicator (01 through 99, L1 through L9, LR, or H1 through H9) or blanks. The record-available indicator turns on whenever a reverse interrupt is received.

Column 60 must contain L or blank. L indicates that this BSCA file is processed only after all other input files are processed. Blank indicates that this BSCA file may not be the last input file processed. The entry in column 60 does not affect demand files.

Columns 61 and 62 must contain the polling identification of this station if this station is part of a multipoint network and if the BSCA file is an output file. Otherwise, leave these columns blank.

Columns 63 and 64 must contain the addressing identification of this station if this station is part of a multipoint network and if the BSCA file is an input file. Otherwise, leave these columns blank.

Programming Considerations

First RPG Program Cycle

During the first RPG program cycle, all primary and secondary input files are opened. That is, the program reads one record from each primary and secondary input file before it processes any input file. However, depending on your particular program, you might want to delay the first-time logic for your BSCA input files. You can delay the first-time logic by designating each BSC input file as a demand file (D in column 16 of the file description specifications). One or more BSC input files can also be designated as the last file (L in column 60 of the telecommunications specifications). If 3740 multiple-file support is being used, all secondary input files should have the L in column 60. Remember that an entire BSC input file must be received before another BSC input file can be received or a BSC output file can be transmitted.

Autocall and X.21 Support

When the System/36 is configured with the multiline communications adapter and the autocall or X.21 feature, remote locations can be called automatically without operator intervention. Because autocall and X.21 are not specified in an RPG program, existing programs can already use autocall or X.21. You specify autocall or X.21 by using the PHONE option on the control language COMM statement. The COMM statement is described in the *System Reference* manual.

The phone list specified in the COMM statement can contain up to 120 phone numbers. The list is created by the DEFINEPN procedure for autocall or by the DEFINX21 procedure for X.21. These procedures are described in the *System Reference* manual.

When the first request is made to BSC during any BSC job step, the phone list is searched for a number to call. The search begins with the first number in the list. Each successive search begins with the next available number. If that number cannot be reached, a counter is decreased by 1 and the next number is called. If no number in the list can be reached, a return code indicating that no line connection was established is passed to the RPG program. A message is displayed at the system console, indicating each number that could not be reached. When a number is reached, a message is displayed at the system console, indicating the number reached, and communications proceed in the same way as for a manual call line. When the job step ends, you can use the control language IF statement to run the job step again and call the next number. You can also use the same phone list in a later step of the job. The IF statement is described in the *System Reference* manual.

If a batch BSC job is run on an autocall line and no phone list is specified in the COMM statement (or if there is no COMM statement), the call mode defaults to the configuration record specification. The mode can be manual answer, manual call, or autoanswer. If the phone list is specified in the

COMM statement but the line is not an autocall line, or if autocall was not requested when the system was defined at initial program load time, then the line is considered to be a manual answer, manual call, or autoanswer line, depending on the switch type defined for the line.

If a batch BSC job is run on an X.21 line and no phone list is specified, a switch type of autoanswer is assumed. If the X.21 task is not active on an X.21 switched line, an error message is displayed. If the phone list is specified but not an X.21 line, then the line is considered to be a manual answer, manual call, or autoanswer line, depending on the switch type defined for the line.

The ability to call multiple locations within a single BSC job step is primarily useful when the System/36 is receiving data from those locations. Because any number may be called during a request, you should use a single-number phone list to send data to a particular location.

If a permanent error occurs while you are receiving data, the phone number associated with the communications link is not reset. Because the number is not reset, it cannot be called again on later passes through the list. The recovery associated with that particular job step is your responsibility. You can use the RESTORE parameter on the control language COMM statement to determine whether the list is restored before you use it again. For information about the COMM statement, see the *System Reference* manual.

Removing Strings of Embedded Blanks

To use the communications line more efficiently and more cheaply, System/36 BSC allows RPG users to send and receive data with all strings of two or more embedded blanks removed. Removing strings of embedded blanks is called *compressing* the data. This is done by using the same format used by the IBM 3780 Data Communications Terminal.

For output files, data is moved from the logical buffer to the BSC input/output buffer with blanks removed and compression control characters inserted. After each record, an intermediate record separator character is inserted. The receiving station automatically inserts the same number of blanks where they were removed.

For input files, the procedure is reversed. The System/36 recognizes the intermediate record separator character, inserts the blanks removed by the remote station, and moves the record from the BSC input/output buffer to the logical buffer.

To remove strings of embedded blanks, specify COMPRESS on the ALTERCOM procedure before you run the BSC program.

Blanks cannot be inserted or removed if you use EBCDIC transparency (Y in column 19 of the telecommunications specifications) or intermediate block checking (I in column 52).

Removing Trailing Blanks

System/36 BSC also allows you to send and receive data with trailing blanks removed. Removing trailing blanks is called *truncating* the data.

For output files, data is moved from the logical buffer to the BSC input/output buffer with all trailing blanks removed. After each record, an intermediate record separator character is inserted. The receiving station automatically inserts the same number of trailing blanks where they were removed.

For input files, the data in the BSC input/output buffer is scanned until an intermediate record separator character is found. All data up to that separator character is moved to the logical buffer. The remainder of the logical buffer is blanked.

To remove trailing blanks, specify TRUNCATE on the ALTERCOM procedure before you run the BSC program.

Trailing blanks cannot be removed when you use intermediate block checking. You can specify that trailing blanks should be removed when you use EBCDIC transparency; however, no blanks are removed, because in transparency mode the record length must be equal to the block length.

When you add or remove blanks with blocked records, the number of records per block varies depending on the number of blanks in each record.

Control Breaks

Take care when sending data during total time in any RPG program that both sends and receives. Because of the sequence of total and detail operations in the RPG program cycle, data might not be available for output even though it is read.

Data Formats

System/36 RPG support uses the following data formats for sending data; these formats must be used when sending data to System/36 from a processing unit:

- Nontransparent, non-ITB: STX-data-ETX(ETB)
- Nontransparent, ITB: STX-data-ITB-data-ITB-data-ETX(ETB)
- Transparent, non-ITB: DLE-STX-data-DLE-ETX(ETB)
- Transparent, ITB (receive files only):
  DLE-STX-data-DLE-ITB-DLE-STX-data-DLE-ITB-DLE-STX-data-DLE
  -ETX(ETB)

Data can be fixed-length and either blocked or unblocked.

Errors

If an error occurs at either station, System/36 tries the operation again up to seven times or up to the number of times (1 through 255) specified as the retry count on the ALTERCOM procedure. (See the *System Reference* manual for information on the ALTERCOM procedure.)

RPG Diagnostics

See the *RPG II Messages* manual for a discussion of RPG diagnostics.

Configuring Your System for BSC

Configuring means defining to the system the devices, optional features, and licensed programs installed on the system. When your System/36 is shipped to you, it is configured for BSC.

To display the status of your communications support, use the STATUS COMM command. For information about the STATUS COMM command, see the manual *Operating Your Computer*.

To change the configuration, use the ALTERCOM or the SETCOMM procedure. For information about the ALTERCOM and SETCOMM procedures, see the *System Reference* manual.

Descriptions of BSC Functions

This section describes the functions that System/36 can perform as part of a data communications network. The sample RPG programs later in this chapter illustrate these functions.

Receive-Only Function

The receive-only function allows you to receive input data from another station. The file can be a primary, secondary, table, or demand file. The records can be blocked. Two input/output areas can be used for primary or secondary files but not for demand files.

Code a receive-only file as an input file on the file description specifications (I in column 15) and as a receive file on the telecommunications specifications (R in column 16).

Send-Only Function

The send-only function allows you to send BSC data to a remote location. Two input/output areas can be used to increase processing speed.

Code a send-only file as an output file on the file description specifications (O in column 15) and as a transmit file on the telecommunications specifications (T in column 16).

Send-and-Receive Function

To both send and receive data, use two files. Code one as an output file on the file description specifications and as a transmit file on the telecommunications specifications. Code the other as an input file on the file description specifications and as a receive file on the telecommunications specifications.

In any BSC program that sends and receives, column 15 and columns 17 through 47 of the telecommunications specifications must be identical for the two files.

BSC programs that send and receive can be written in any of these three ways:

- Send a file, then receive a file.
- Receive a file, then send a file.
- Send records interspersed with receive records.

Send a File, Then Receive a File

The receive file must not be defined as the primary input file on the file description specifications. If the receive file is a secondary file, column 60 of the telecommunications specifications must contain an L. The matching-fields and record-available indicators must not be specified for the BSC file.

Receive a File, Then Send a File

The receive file can be defined as a primary, secondary, table, or demand file on the file description specifications. Columns 58 and 59 (record-available indicator) on the telecommunications specifications must be blank.

Send Records Interspersed with Receive Records

A program can send records interspersed with records it receives. Such a program sends records from one file and receives records in another; the two files might not be related. Unlike conversational programs, this kind of program might intersperse several records or several blocks of data at a time.

The receive file must be defined as a demand file on the file description specifications. The record-available indicator must be specified on the telecommunications specifications. System/36 must begin by sending data, then suspend the transmit file to receive data from the other station (see Figure 10-3 for an example of this type of program).

Once BSC begins to process the last record in the transmit file, System/36 ignores the record-available indicator, whether or not the last record was actually sent. When BSC accepts the last record in the file for transmission, RPG completes last-record processing and begins to close the file.

Therefore, if the next-to-last record or block of records intended for transmission prompts the other station to request to send data back to System/36, the request might be ignored. The request is always ignored if it is prompted by the final record or block of records.

You can avoid this problem by adding to the end of the System/36 transmit file a special record that signals that System/36 went to end of job and cannot honor a request to receive, even though that request was just sent. Of course, the programmer of the other system must agree on the meaning of that special record.

Systems That Use BSC

You can use BSC for data communication between your System/36 and any of the following IBM systems and devices:

- Another System/36 with RPG, assembler, or SSP-ICF BSCEL
- System/38 with BSCA using RPG III or COBOL
- System/34 with RPG, assembler, or SSP-ICF BSCEL
- System/32 with RPG or assembler
- System/3 with RPG, CCP, or MLMP
- System/7 with MSP/7
- Operating System or Disk Operating System Basic Telecommunications Access Method (OS, OS/VS, DOS/VS, or DOS BTAM)
- System/360 Model 20 Input/Output Control System for the Binary Synchronous Communications Adapter
- Customer Information Control System (CICS/DOS/VS or CICS/VS)
- Information Management System (IMS/VS)
- 3741 Model 2 Data Station or Model 4 Programmable Work Station
- 3747 Data Converter
- 5231 Data Collection Controller Model 2 (as a 3741 in transmit mode only)
- 3750 Switching System (World Trade only)
- 5110 or 5120 (in 3741 mode)
- Series 1 (in System/3 mode)
- 5260 Point of Sale Terminal (in 3740 mode)
- 5280 Distributed Data System (in 3740 mode)

Device-Dependent Considerations

IBM 3740 Data Entry System

RPG data communications programming supports the IBM 3741 Model 2 Data Station, the IBM 3741 Model 4 Programmable Work Station, or the IBM 3747 Data Converter in communicate mode as a remote device by using the System/36 communications adapter.

Restrictions

The following restrictions apply when a System/36 communicates with a 3740 Data Entry System:

- A 3741 with an Expanded Communications Buffer feature (feature number 1680) has a maximum buffer size of 512 positions.
- A 3747 with the Blocking/Reformatting feature (feature number 1480) has a maximum buffer size of 8050 positions. However, System/36 RPG handles only a maximum of 4075 positions.
- The Operator Identification Card Reader Feature (feature number 5450) and the Expanded Communications/Multipoint Data Link Control Feature (feature number 1685) on the 3741 are not supported by RPG on System/36.
- Through RPG, you can communicate with the 3741 or the 3747 either by using single-file support (one input file, one output file, or one of each) or by using multiple-file support (more than one input file, more than one output file, or more than one of each).
- Through RPG, you can also send blocked records to, and receive blocked records from, a 3741 with the Expanded Communications Buffer feature or a 3747 with the Blocking/Reformatting feature.
- If you send blocked records to a 3741, you must specify a record separator of hexadecimal 1E.
- RPG receives 3741 STATUS messages as data, and these must be handled by the user. Refer to the *IBM 3741 Data Station Reference Manual (GA21-9183)* for details of the possible status messages and the format of the data that will be received.

Single-File Support

If you have single-file support when you communicate with the 3741, a maximum of two BSC files are allowed (one input and one output) per RPG program. If you use two BSC files, you must process the input file completely before you process the output file.

When you communicate with the 3747 Data Converter, only one BSC file is allowed (either input or output).

Multiple-File Support

Through RPG, you can use the multiple-file support of System/36 to communicate with the 3740 Data Entry System. To specify multiple-file support, specify MULTFILE on the ALTERCOM procedure before you run the RPG program (see the *System Reference* manual for information about the ALTERCOM procedure).

When you communicate with the 3741 or with 3741 emulators, multiple files can be received, sent, or received and then sent. All input files from the 3740 must be received before System/36 can begin sending files to the 3740. When you communicate with the 3747, multiple files can be either received or sent. When you communicate with the 5110 or 5120, multiple files can be received, sent, received and then sent, or sent and then received.

Blocked Records

Blocked records can be received from, or sent to, a 3741 with the Expanded Communications Buffer feature for either single or multiple files. If you specify COMPRESS or TRUNCATE on the ALTERCOM procedure but do not specify a record separator on that procedure, the record separator of hex 1E is used.

You can send blocked records to a 3747 with the Blocking/Reformatting feature by using either of the following:

- The ALTERCOM procedure, as explained in the preceding paragraph for the 3741
- Columns 20 through 23 of the file description specifications, which can contain a multiple of the record length

RPG input files can be primary, secondary, or demand files. However, the 3740 files must be processed one file at a time to the end of the file and in the order that the 3740 sends them.

Secondary files are processed in the order listed on the file description specifications in the source program.

Demand files are processed in the order determined by the logic of your calculation specifications.

Output files must be processed one file at a time. That is, all records for one file must be sent before the first record for the next file is sent. When you communicate with a 3741, if multiple files are received and then multiple files are sent in the same program, all input files must be processed before any output files are processed.

RPG Specifications

Use of the 3740 affects RPG file descriptions, telecommunications, and output specifications. Only the entries unique to 3740 are described here.

File Description Specifications

Columns 20-23 (Block Length): Maximum block length is 128 positions without the Expanded Communications Buffer feature on the 3741 or with the Blocking/Reformatting feature on the 3747.

If blocked records are to be sent to a 3741 with the Expanded Communications Buffer feature, the block length can be any multiple of the record length not exceeding 512 positions.

If blocked records are received from a 3741 with the Expanded Communications Buffer feature, the block length *must be* N times the record length, where N is the result (disregarding the remainder) of dividing 512 by the record length plus one. For example, if the record length is 128:

- Record length plus one = 129
- 512 divided by 129 = 3 with a remainder of 125
- N = 3
- Block length = 3 times 128 = 384

When you communicate with a 3747 with the Blocking/Reformatting feature, the block length depends on the use of the data at the 3747 and on the amount of storage available (feature number 7690, 7691, or 7692). Blocking on the 3747 can be identical to that on the 3741 with the Expanded Communications Buffer feature through the use of C3 control records. Blocking can also be handled in a manner that is similar to RPG blocking through use of the C3 control records. For an explanation of the C3 control records format, see the *IBM 3747 Data Converter Reference Manual and Operator's Guide.*

Columns 24-27 (Record Length): Maximum record length is 128 positions when communicating with a 3741. The 3747 maximum record length depends on the use of the data at the 3747 and on whether the Blocking/Reformatting feature is installed.

Telecommunications Specifications

The 3740 files require some restrictions to the telecommunications specifications. Only the columns affected are listed here:

Column 15 (Multipoint network): M must not be specified.

Column 17 (Tributary system on a multipoint network) T must not be specified.

Column 52 (ITB): I must not be specified.

Columns 61 through 74: Must be blank.

Output Specifications

Columns 17-22: Must be blank.

IBM 3750 (World Trade Only)

When a System/36 is connected to an IBM 3750 Switching System, the RPG data communications program must allow message exchanges between the two systems. The RPG program can be written for message exchanges related to the following 3750 functions:

- Recording calls
- Monitoring contact under control of the data processing system
- Inquiry to the data processing system with a recorded answer
- Real-time data collection to the data processing system
- Transfer of recorded data to the data processing system

Communications between the System/36 and the 3750 are binary synchronous, point-to-point operations in transparent mode. Only EBCDIC can be used. The System/36 operates as a send and receive station.

When you receive the end-of-transmission character and the next step in your program is to receive more data, do another read operation to the same BSC input file.

Sample Programs

The following three sample programs are provided as examples of the various types of RPG BSC programs:

- The first example is a send-only program.
- The second is a System/36-to-3740 program.
- The third is a send-interspersed-with-read program.

Send Only

Figure 10-1 shows a program that reads a DISK file and then sends it. The week's data has been sorted by name of salesman. The amount of each sale is written on the disk, and the total sales for each salesman is transmitted to the branch office. After all disk records containing sales information are read, the total of all sales is sent to the branch office.

RPG File Description Specifications

Column	Description
7-14	WKLYSMRY is a BSC file.
15	Since WKLYSMRY is to be transmitted, it is an output file.
19	BSC files always have a fixed-length format.
20-27	Records are blocked.
32	Dual I/O areas are used.
40-46	BSCA is the device name.

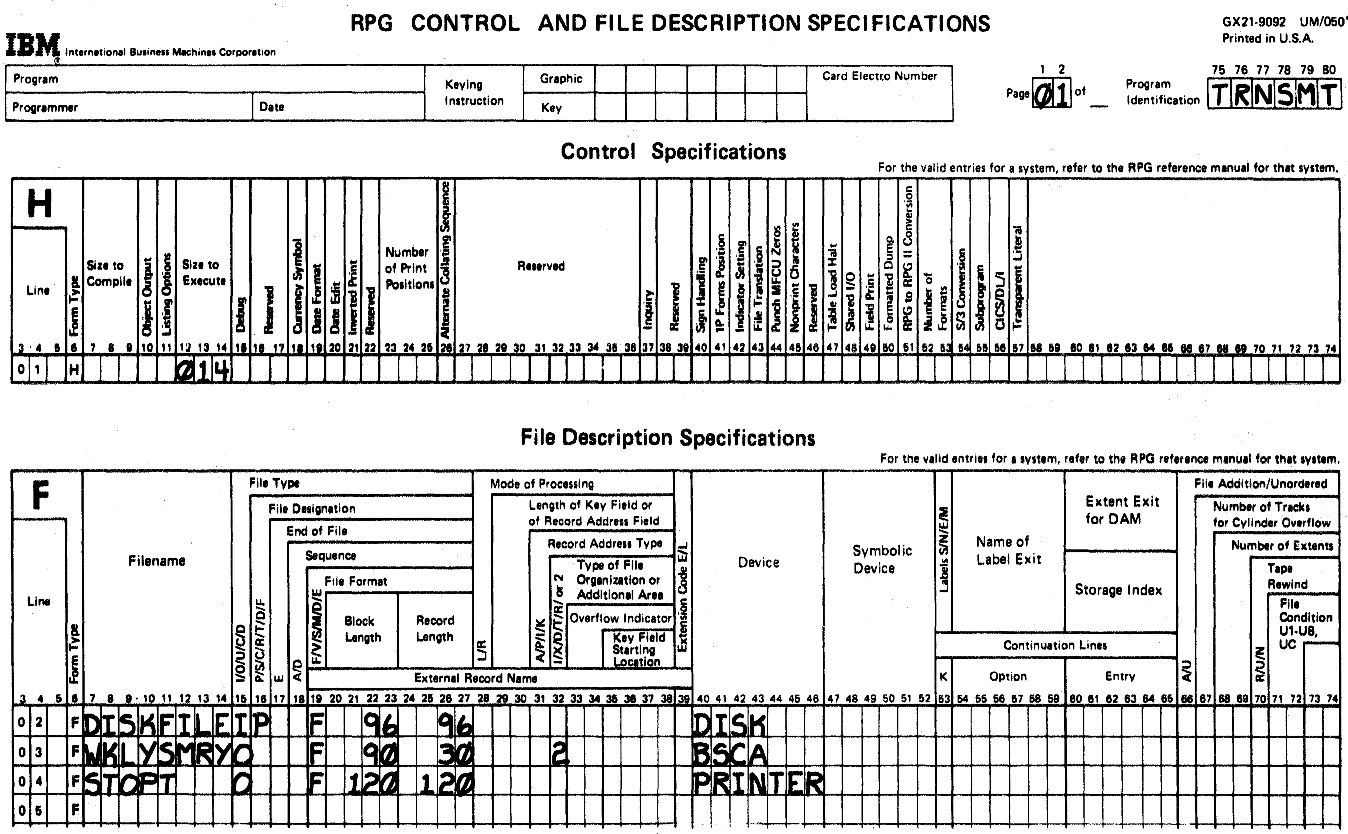


Figure 10-1 (Part 1 of 3). Send-Only Program.

Column	**Description**
7-14	WKLYSMRY is the BSC file for this program.
15-17	This station is part of a switched network (S), and is transmitting (T). Polling is not used, so column 17 is blank.
18-19	EBCDIC (E) and the transparency feature (Y) are both used.
20	Automatic answer (A) is used by this called station.
32-47	Explicit station identification sequences are given for each station. Station IDs help ensure data security on the switched network.
53-54	The permanent-error indicator used is 25.
55-57	The data communications line is kept open for 70 seconds when no messages are being sent or received. After 70 seconds elapse, a permanent error condition results if the line is still not being used.

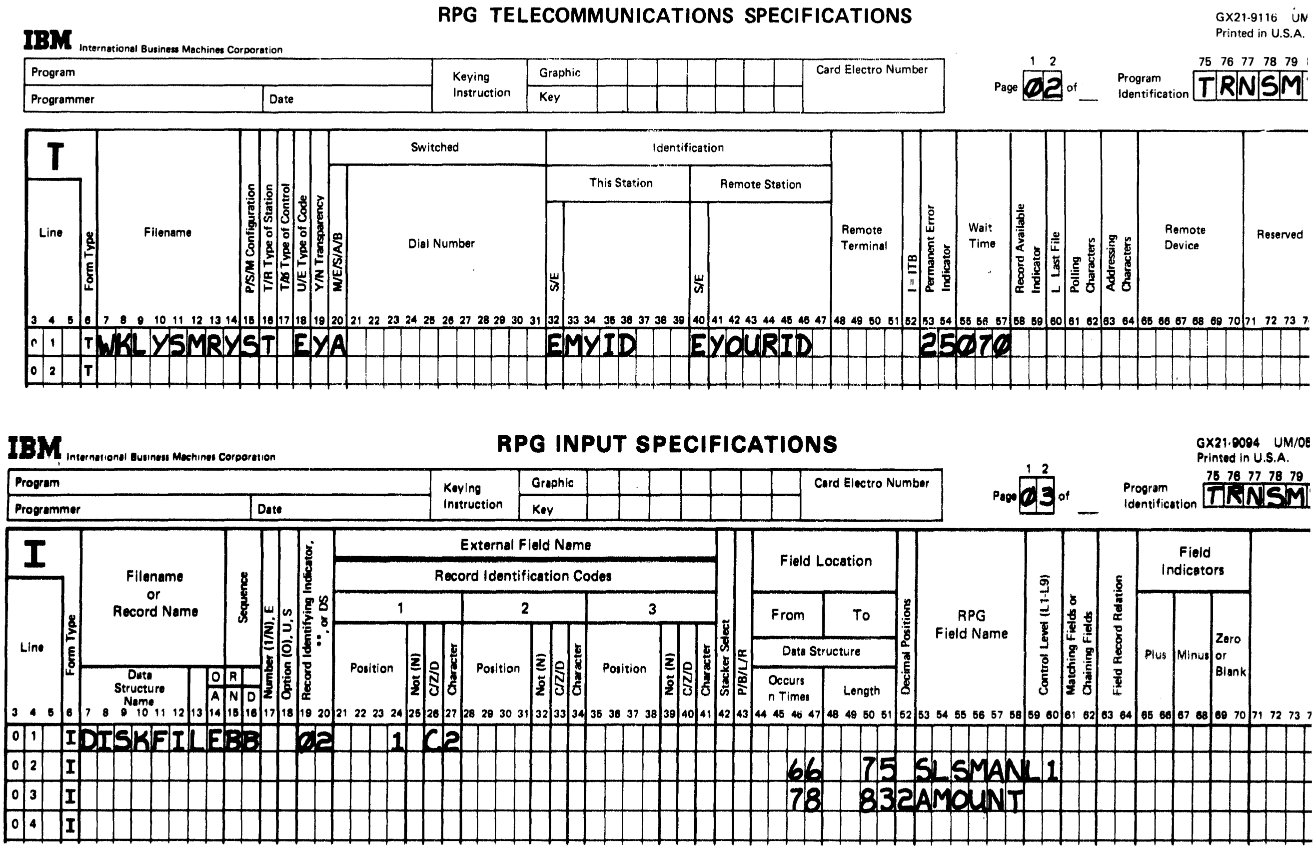


Figure 10-1 (Part 2 of 3). Send-Only Program.

RPG Calculation Specifications

If a permanent error occurs, LR is set on to enter end-of-job processing.

RPG Output Specifications

Lines 01-04: If a permanent error occurs, a message printed at total time identifies the record being processed when the permanent error occurred. However, because of record blocking and the use of dual I/O areas, not all the records preceding the record identified were transmitted.

Lines 05-10: When on, the permanent-error indicator prevents the program from transmitting totals.

IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

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

Program | Programmer | Date | Keying Instruction: Graphic, Key | Card Electro Number | Page 04 of | Program Identification TRNSMT

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Resulting Indicators (Plus/High)	Comments
01	C		25		SETON					LR	
02	C		02 N25	AMOUNT	ADD	MANSUM	MANSUM	6	2		
03	C	L1	N25	MANSUM	ADD	TOTSUM	TOTSUM	6	2		
04	C										

IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

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

Program | Programmer | Date | Keying Instruction: Graphic, Key | Card Electro Number | Page 05 of | Program Identification TRNSMT

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Output Indicators	Field Name or EXCPT Name	Blank After (B)	End Position in Output Record	Constant or Edit Word
01	O	STOPT	T	25				
02	O						19	'PROGRAM INTERRUPTED'
03	O						36	'WHILE PROCESSING'
04	O				SLSMAN		51	
05	O	WKLYSMRY	T	L1N25				
06	O				SLSMAN		10	
07	O				MANSUM	B	30	
08	O		T	LRN25				
09	O						20	'TOTAL ='
10	O				TOTSUM		30	
11	O							

Figure 10-1 (Part 3 of 3). Send-Only Program.

System/36 to 3740

Figure 10-2 shows a program that receives two files from an IBM 3740, then sends two files to the 3740. The first file from the 3740 forms the input file BS1; the second file forms BS2. The data received as input to BS1 and BS2 is written by System/36 to the output file PRINTER. Then the System/36 reads the disk file FILEA. Records that start with a 1 are sent to the 3741 in file BS3. Records that start with a 2 are sent to the 3741 in file BS4. (All records with a 1 must precede any record that begins with a 2.) The job ends when the last record from the disk has been read.

Note: The ALTERCOM procedure must be run with MULTFILE specified.

RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

IBM International Business Machines Corporation

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

Program | Programmer | Date | Keying Instruction: Graphic / Key | Card Electro Number

Page Ø1 of __ Program Identification BSCØ6

Control Specifications

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

Line (3-5)	Form Type (6)	Size to Compile (7-9)	Size to Execute (12-14)
01	H		Ø14

File Description Specifications

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

Line (3-5)	Form Type (6)	Filename (7-14)	File Type / File Designation (15-16)	Block Length (20-23)	Record Length (24-27)	Type of File Organization or Additional Area (32)	Device (40-46)
02	F	BS1	IP	480	80	2	BSCA
03	F	BS2	IS	480	80	2	BSCA
04	F	BS3	O	480	80	2	BSCA
05	F	BS4	O	480	80	2	BSCA
06	F	FILEA	ID	80	80		DISK
07	F	PRINTER	O	132	132		PRINTER
08	F						

Figure 10-2 (Part 1 of 3). System/36-to-3740 Program

RPG TELECOMMUNICATIONS SPECIFICATIONS

IBM International Business Machines Corporation

GX21-9116 UM/050*
Printed in U.S.A.

Program | Keying Instruction | Graphic | Card Electro Number
Programmer | Date | Key

Page Ø2 of ___ Program Identification BSCØ6

Line (3-5)	Form Type (6)	Filename (7-14)	P/S/M Configuration (15)	T/R Type of Station (16)	T/B Type of Control (17)	U/E Type of Code (18)	Y/N Transparency (19)	M/E/S/A/B (20)	Switched: Dial Number (21-31)	Identification: This Station S/E (32)	This Station (33-39)	Remote Station S/E (40)	Remote Station (41-47)	Remote Terminal (48-51)	I = ITB (52)	Permanent Error Indicator (53-54)	Wait Time (55-57)	Record Available Indicator (58-59)	L Last File (60)	Polling Characters (61-62)	Addressing Characters (63-64)	Remote Device (65-70)	Reserved (71-74)
01	T	BS1	P	R																			
02	T	BS2	P	R															L				
03	T	BS3	P	T																			
04	T	BS4	P	T																			
05	T																						

RPG INPUT SPECIFICATIONS

IBM International Business Machines Corporation

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

Program | Keying Instruction | Graphic | Card Electro Number
Programmer | Date | Key

Page Ø3 of ___ Program Identification BSCØ6

Line (3-5)	Form Type (6)	Filename or Record Name / Data Structure Name (7-14)	Sequence (15-16)	Number (1/N), E (17)	Option (O), U, S (18)	Record Identifying Indicator, ** or DS (19-20)	Record Identification Codes 1: Position (21-24)	Not (N) (25)	C/Z/D (26)	Character (27)	2: Position (28-31)	Not (N) (32)	C/Z/D (33)	Character (34)	3: Position (35-38)	Not (N) (39)	C/Z/D (40)	Character (41)	Stacker Select (42)	P/B/L/R (43)	Field Location From / Data Structure Occurs n Times (44-47)	To / Length (48-51)	Decimal Positions (52)	RPG Field Name (53-58)	Control Level (L1-L9) (59-60)	Matching Fields or Chaining Fields (61-62)	Field Record Relation (63-64)	Field Indicators Plus (65-66)	Minus (67-68)	Zero or Blank (69-70)	(71-74)
01	I	BS1	AA			Ø1																									
02	I																				1	8Ø		INDATA							
03	I	BS2	AA			Ø2																									
04	I																				1	8Ø		INDATA							
05	I	FILEA	AA			Ø3	1		C	1																					
06	I																				1	8Ø		RECORD							
07	I		AA			Ø4	1		C	2																					
08	I																				1	8Ø		RECORD							
09	I																														

Figure 10-2 (Part 2 of 3). System/36-to-3740 Program

IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

GX21-9093 UM/0
Printed in U.S.A.

Page 04 of ___ Program Identification BSC06

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators (Plus / Minus / Zero; High / Low / Equal)	Comments
01	C	LR		LOOP	TAG							
02	C	LR			READ	FILEA					44 (Equal)	
03	C	LR	N44		EXCPT							
04	C	LR	N44		GOTO	LOOP						
05	C											

IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21-9090 UM/0
Printed in U.S.A.

Page 05 of ___ Program Identification BSC06

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Output Indicators	Field Name or EXCPT Name	End Position in Output Record	Constant or Edit Word
01	O	PRINTER	D				
02	O						
03	O			01		3	'BS1'
04	O			02		3	'BS2'
05	O				INDATA	90	
06	O	BS3	E	03N04			
07	O				RECORD	80	
08	O	BS4	E	04			
09	O				RECORD	80	
10	O						

Figure 10-2 (Part 3 of 3). System/36-to-3740 Program

Send Interspersed with Receive

Figure 10-3 shows a program that reads a disk file, STUDENT, containing information about a student test, then sends that information in a BSCA file, GRADES, to a remote station. While the System/36 is sending data, the other station might interrupt to send back data in a file called RESULTS. When this occurs, the RPG program turns on record-available indicator 04, reads the input file RESULTS, and prints it as the output file PRINTER. Then System/36 continues sending file GRADES.

RPG File Description Specifications

Column	**Description**
7-14	RESULTS and GRADES are BSC files.
15	RESULTS receives data; therefore, it is an input file. GRADES is transmitted and is an output file.
16	To receive data intermittently, a BSC file must be a demand file.
17	End of file on RESULTS does not determine end-of-job time.
19	BSC files always have a fixed-length format.
20-27	Neither BSC file is blocked.
40-46	BSCA is the device for BSC files.

IBM International Business Machines Corporation

RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

GX21-9092 UM, Printed in U.S.A.

Program | Keying Instruction | Graphic | Card Electro Number
Programmer | Date | Key

Page (1 2) Ø1 of __ Program Identification (75 76 77 78 79) TRTRØ

Control Specifications

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

Line (3-5)	Form Type (6)	Size to Compile (7-9)	Object Output (10)	Listing Options (11)	Size to Execute (12-14)	Remaining columns (15-74)
01	H				Ø14	

File Description Specifications

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

Line (3-5)	Form Type (6)	Filename (7-14)	File Type (15)	File Designation (16)	End of File (17)	Sequence (18)	File Format (19)	Block Length (20-23)	Record Length (24-27)	Type of File Organization or Additional Area (32)	Device (40-46)
02	F	STUDENT	I	P			F	96	96		DISK
03	F	RESULTS	I	D			F	8Ø	8Ø		BSCA
04	F	GRADES	O				F	8Ø	8Ø	2	BSCA
05	F	PRINT	O				F	132	132		PRINTER
06	F										

Figure 10-3 (Part 1 of 5). Send-Interspersed-with-Receive Program.

Column	Description
7-14	RESULTS and GRADES are the BSC files for this program.
15	This station is part of a switched network (S).
16	RESULTS is an input file and receives data (R). GRADES is an output file and is transmitted (T).
17	Polling is not used, so column 17 is blank.
18-19	EBCDIC (E) and the transparency feature (Y) are both used.
10	Automatic answer is used by this station.
32-47	Explicit station identification sequences are given. Station IDs help ensure data security on the switched network.
53-54	The permanent-error indicator used is 44.
55-57	The data communications line is kept open for 70 seconds when no messages are being sent or received. After 70 seconds elapse, a permanent error condition results if the line still is not being used.
58-59	The record available indicator is 04; it is set on when the other system sends an RVI (reverse interrupt). This indicates it is ready to send a record to be received by RESULTS.

IBM International Business Machines Corporation

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

GX21-9091 UM/050* Printed in U.S.A.

Program ______ Programmer ______ Date ______ Keying Instruction: Graphic / Key ______ Card Electro Number ______

Page 02 of __ Program Identification TRTR01

Extension Specifications

Line (3-5)	Form Type (6)	Record Sequence of the Chaining File (7-8)	Number of the Chaining Field (9-10)	From Filename (11-18)	To Filename (19-26)	Table or Array Name (27-32)	Number of Entries Per Record (33-35)	Number of Entries Per Table or Array (36-39)	Length of Entry (40-42)	P/B/L/R (43)	Decimal Positions (44)	Sequence (A/D) (45)	Table or Array Name (Alternating Format) (46-51)	Length of Entry (52-54)	P/B/L/R (55)	Decimal Positions (56)	Sequence (A/D) (57)	Comments (58-74)
01	E					ANSARR		50	1		0							
02	E					STDARR		50	1		0							
03	E																	

IBM International Business Machines Corporation

RPG TELECOMMUNICATIONS SPECIFICATIONS

GX21-9116 UM/050* Printed in U.S.A.

Program ______ Programmer ______ Date ______ Keying Instruction: Graphic / Key ______ Card Electro Number ______

Page 03 of __ Program Identification TRTR01

Line (3-5)	Form Type (6)	Filename (7-14)	P/S/M Configuration (15)	T/R Type of Station (16)	T/B Type of Control (17)	U/E Type of Code (18)	Y/N Transparency (19)	M/E/S/A/B (20)	Switched: Dial Number (21-31)	This Station S/E (32)	This Station (33-39)	Remote Station S/E (40)	Remote Station (41-47)	Remote Terminal (48-51)	I = ITB (52)	Permanent Error Indicator (53-54)	Wait Time (55-57)	Record Available Indicator (58-59)	L Last File (60)	Polling Characters (61-62)	Addressing Characters (63-64)	Remote Device (65-70)	Reserved (71-74)
01	T	RESULTS	S	R		E	Y	A		E	9876	E	5678			44	070	04					
02	T	GRADES	S	T		E	Y	A		E	9876	E	5678			44	070						
03	T																						

Figure 10-3 (Part 2 of 5). Send-Interspersed-with-Receive Program.

RPG Input Specifications

Lines 07-09: RESULTS receives records from the remote station. These records contain a student number and the student's score. Student scores are calculated by the remote station from the data transmitted from GRADES.

IBM International Business Machines Corporation

RPG INPUT SPECIFICATIONS

GX21-9094 UM/0
Printed in U.S.A.

Program		Keying Instruction	Graphic		Card Electro Number	Page (1 2)	of	Program Identification (75 76 77 78 79)
Programmer	Date		Key			Ø4		TRTRØ

Line (3-5)	Form Type (6)	Filename or Record Name / Data Structure Name (7-14)	Sequence (15-16)	Number (1/N), E (17)	Option (O), U, S (18)	Record Identifying Indicator, **, or DS (19-20)	1 Position (21-24)	1 Not (N) (25)	1 C/Z/D (26)	1 Character (27)	2 Position (28-31)	2 Not (N) (32)	2 C/Z/D (33)	2 Character (34)	3 Position (35-38)	3 Not (N) (39)	3 C/Z/D (40)	3 Character (41)	Stacker Select (42)	P/B/L/R (43)	Field Location From / Occurs n Times (44-47)	Field Location To / Length (48-51)	Decimal Positions (52)	RPG Field Name (53-58)	Control Level (L1-L9) (59-60)	Matching Fields or Chaining Fields (61-62)	Field Record Relation (63-64)	Field Indicators Plus (65-66)	Field Indicators Minus (67-68)	Field Indicators Zero or Blank (69-70)	(71-73)
01	I	STUDENT	AA			Ø1	1		C	A	2		C	N	3		C	S													
02	I																				5	7	Ø	#QUES							
03	I																				11	6Ø		ANSARR							
04	I		BB			Ø2	1		C	S	2		C	T	3		C	D													
05	I																				3	5		STUDID							
06	I																				11	6Ø		STDARR							
07	I	RESULTS	CC			Ø3	1		C	R																					
08	I																				3	5		STDNT							
09	I																				8	1Ø		SCORE							
10	I																														

Figure 10-3 (Part 3 of 5). Send-Interspersed-with-Receive Program.

RPG Calculation Specifications

The calculation sequence obtains the scores computed by the remote station from GRADES and makes them available for exception output to PRINT.

Lines 01-06: Loop 1 is performed during detail time; that is, before end of file is reached on STUDENTS. When the remote station causes the record available indicator (04) to be set on, the READ operation accepts a record from the remote station and places it in RESULTS. Indicator 10 is set on and indicator 04 is set off when READ encounters an end-of-file condition. Until indicator 04 is set off and indicator 10 is set on, records are placed in RESULTS and are available for exception output to PRINT. Loop 1 can be entered or reentered anytime 04 is set on, except after end of file is reached on STUDENTS (indicator 10 is set off). Therefore, after one group of records is read by the receiving station, the receiving station must set off the end-of-file indicator (10) so that the next group of records can be read.

Lines 07-12: Loop 2 is performed during total time; that is, after end of file is reached on STUDENTS and the LR indicator is set on. Loop 2 processes records for results in the same way as loop 1 with one difference: Loop 2 always compares the student number received by RESULTS (STDNT) to the last student number transmitted (STUDID). Loop 2 continues until these numbers are equal. This ensures that the results for all student records transmitted are received.

If a permanent error occurs during a transmit or receive operation, LR and 10 are set on to enter end-of-job processing. Indicator 09 is then set off to exit from the total-time loop.

IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

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

Program | Programmer | Date | Keying Instruction: Graphic, Key | Card Electro Number | Page Ø5 of | Program Identification TRTRØ1

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Not	Indicator	Not	Indicator	Not	Indicator	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators High (Plus, 1>2)	Low (Minus, 1<2)	Equal (Zero, 1=2)	Comments
01	C								LOOP1	TAG									
02	C			Ø4						READ	RESULTS							1Ø	
03	C			Ø3	N	1Ø				EXCPT									
04	C			44						SETON						LR	1Ø		
05	C			Ø3	N	1Ø				GOTO	LOOP1								
06	C									SETOF						1Ø			
07	C	LR							LOOP2	TAG									
08	C	LR		Ø4						READ	RESULTS							1Ø	
09	C	LR		Ø3	N	1Ø				EXCPT									
10	C	LR		44						SETOF						Ø9			
11	C	LR	N	44					STDNT	COMP	STUDID					Ø9	Ø9		
12	C	LR		Ø9						GOTO	LOOP2								
13	C																		

Figure 10-3 (Part 4 of 5). Send-Interspersed-with-Receive Program.

RPG Output Specifications

Lines 01-06: GRADES is transmitted to the remote stations.

Lines 07-08: permanent error causes an error message to be printed.

Lines 09-13: Records received by RESULTS are printed.

IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

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

Program | Keying Instruction | Graphic | Card Electro Number | Page Ø6 of | Program Identification TRTRØ1

Programmer | Date | Key

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	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	GRADES	D						Ø1N44						
02	O									#QUES			7		
03	O									ANSARR			6Ø		
04	O		D						Ø2N44						
05	O									STUDID			5		
06	O									STDARR			6Ø		
07	O	PRINT	T						44						
08	O												14		'ERROR OCCURRED'
09	O		E						Ø3N44						
10	O												13		'STUDENT #'
11	O									STDNT			17		
12	O												25		'SCORE ='
13	O									SCORE			29		
14	O														

Figure 10-3 (Part 5 of 5). Send-Interspersed-with-Receive Program.

Chapter 11. Using Primary and Secondary Files

Chapter 11. Using Primary and Secondary Files

Using the RPG program cycle, an RPG program can process more than one input, update, or combined file. How then can the program tell which file to process a record from at any given time? One way is to read one record from every file, compare the contents of a particular field in each of those records, and select a record based on the results of that comparison. The fields that are compared are called *match fields*. Records that contain match fields are called *matching records*.

No Match Fields

If match fields are not used in a program that has more than one input file, the program processes records from one file at a time. After the program processes all the records from one file, it processes all the records from the next file. Files are processed in this order:

1. Primary file, if specified

2. Secondary files in the order in which they are coded in the file description specifications

Match Fields

When match fields are used, the program selects the records for processing based on the contents of the match fields.

When the match field in the primary file record is the same as the match field in one or more of the secondary file records, the matching-record indicator turns on. Your program can use this indicator to control calculation or output operations for the matching records.

If some files contain match fields and other files do not, the program processes all the files without match fields before it processes any files with match fields. The files without match fields are processed in this order:

1. Primary file, if specified

2. Secondary files in the order in which they are coded in the file description specifications

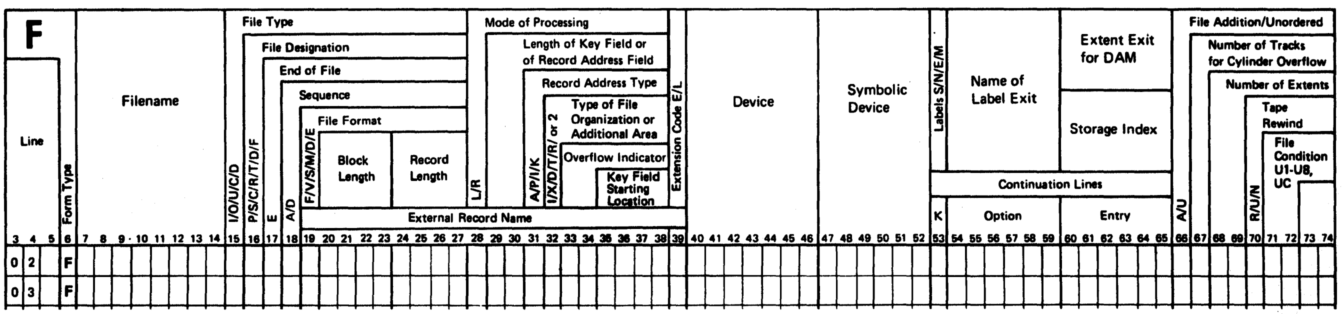
If some records in a file use match fields and other records do not, the records without match fields are processed immediately after the record they follow.

Coding Matching Records

To process matching records, entries are required on both the file description specifications and the input specifications.

File Description Specifications

For matching records, the following entries are required on the file description specifications. For information about columns not mentioned here, see Chapter 5 for DISK files, Chapter 8 for SPECIAL files, or Chapter 9 for CONSOLE files.



Column 15 must contain I, U, or C to indicate that the file is an input, update, or combined file.

Column 16 must contain P or S to indicate that the file is a primary or secondary file.

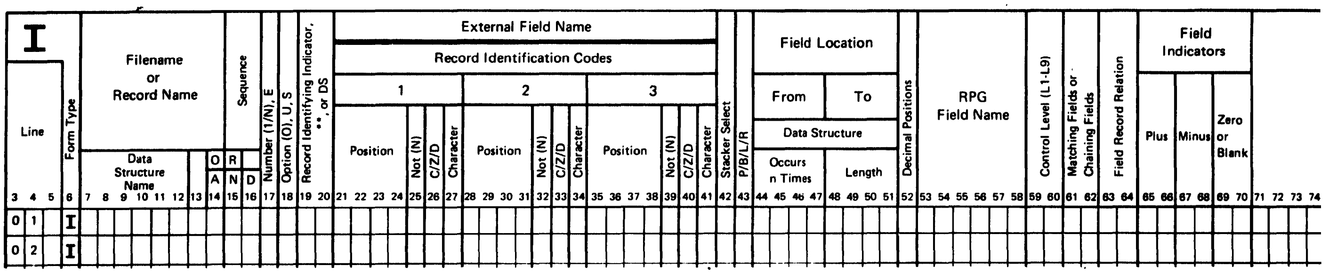
Column 17 must contain E or blank. E indicates that the program must process all records from the file before the program can end. Blank indicates that the program can end before it processes all records from the file. However, if column 17 is blank for all files, the program must process all records from every file before it can end.

Column 18 must contain A, D, or blank. A indicates that the program checks that the records in the file are in ascending sequence. D indicates that the program checks that the records are in descending sequence. Sequence checking is required for all files that use match fields. If column 18 is left blank and match fields are used, ascending sequence is assumed. Column 18 must contain the same entry for all files that specify matching fields on the input specifications.

Columns 40 through 46 must contain DISK, CONSOLE, or SPECIAL.

Input Specifications

For matching records, the following entries are required on the input specifications. For information about columns not mentioned here, see Chapter 25, *Input Specifications.*



Columns 61 and 62 must contain any value from M1 through M9 to indicate that the field named in columns 53 through 58 is a match field.

Rules for Coding Match Fields

You can use one field, many fields, or an entire record to match records.

You can use as many as nine match fields by coding a different value, M1 through M9, for each match field.

M1 through M9 are not indicators. They identify the match fields, and they cause the matching-record indicator (MR) to turn on.

Not all files used by the program must have match fields. Not all record types within one file must have match fields either. However, at least one record type from two files must have match fields in order for the files to match.

The same number of match fields must be coded for all record types that are used in matching. The same matching record values must also be used for all types.

Whenever more than one match-field value is used, all match fields must match before the matching-record indicator (MR) turns on. For example, if you use match-field values M1, M2, and M3, all three fields from one record must match all three fields from the other record. A match on only the M1 and M2 fields does not turn on the matching-record indicator (MR).

If you use more than one match field for a record type, all the fields are combined and treated as one continuous match field (see Figure 11-1). The fields are combined according to descending sequence (M9 to M1) of match-field values.

Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator, **, or DS	Record Identification Codes 1: Position	C/Z/D	Character	Field Location From	Field Location To	RPG Field Name	Matching Fields or Chaining Fields
01	I	MASTER	AA	Ø1	95	C	S				
02	I							1	5	EMPLNO	M1
03	I							1Ø	25	PAYRAT	
04	I							36	4Ø	DIVSON	M3
05	I							31	35	DEPT	M2
06	I		BB	Ø2	95	C	T				
07	I							1	5	EMPLNO	M1
08	I							1Ø	15	HRSWKD	
09	I							2Ø	24	DEPT	M2
10	I							25	29	DIVSON	M3
11	I	WEEKLY	DL	Ø3							
12	I							1	5	EMPLNO	M1
13	I							6	1Ø	DIVSON	M3
14	I							11	15	DEPT	M2
15	I							2Ø	25	HRSWKD	
16	I										

Three record types are used in matching records. All record types have match fields specified, and all use the same values (M1, M2, M3) to indicate which fields must match. The MR indicator turns on only if all three match fields in either of the record types from the MASTER file are the same as all three fields from the record in the WEEKLY file.

The three match fields in each record type are combined and treated as one match field organized as follows:

```
DIVSON     DEPT     EMPLNO

  M3        M2        M1
```

The order in which the fields are specified by the input specifications does not affect the organization of the match fields in the computer.

Figure 11-1. Using Several Fields as One Combined Match Field

Match fields cannot be split. That is, the same match-field value cannot be used twice for one type of record.

All match fields that have the same match-field value (M1 through M9) must be the same length. If the match field contains packed data, the zoned-decimal length, which is (2 x the packed-decimal length) - 1, is used as the length of the match field.

Record positions of different match fields can overlap, but the total length of all fields must not be more than 144 characters.

All match fields that have the same match-field value (M1 through M9) must be the same type (alphameric or numeric). If any of the match fields is described as numeric, all match fields that have the same match-field value are considered numeric.

When numeric fields having decimal positions are matched, they are treated as if they had no decimal position. For example, 3.46 is considered equal to 346.

Only the digit portions of numeric match fields are compared. Even if a field is negative, it is considered to be positive because the sign of the numeric field is ignored. Thus, a -5 matches a +5.

The match field value must be valid alphameric or numeric characters. Other values may cause unpredictable results.

A field coded as binary (B in column 43) cannot have a match-field value. However, a field coded as packed (P in column 43) can have a match-field value.

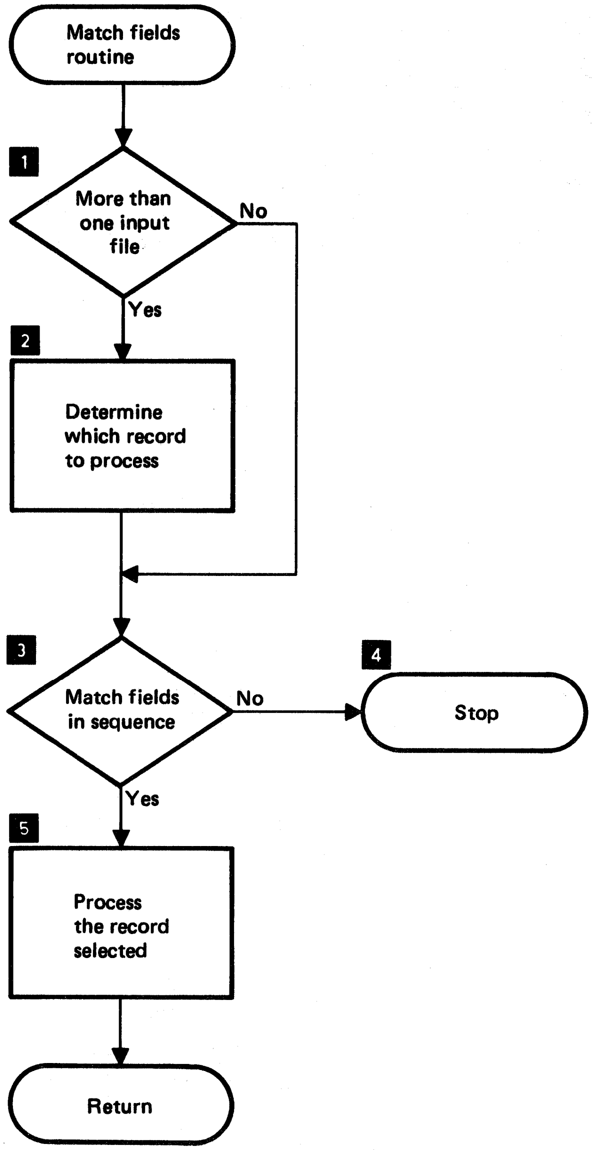
Field names are ignored in match field operations. Therefore, fields from different record types that have the same match-field value can have the same name.

If the program uses an alternative collating sequence, alphameric fields are matched according to the alternative sequence.

Additional rules apply to match fields when entries are coded in columns 63 and 64 of the input specifications (see *Columns 63-64* in Chapter 25, *Input Specifications*).

Processing Matching Records

Figure 11-2 is a flowchart for the processing of matching records.



1 The program checks whether there is more than one input file.

2 If there is more than one input file, the program reads one record from each file and moves the records to a hold area. The program compares the contents of the match fields in these records to determine which record to process next. Records that are not processed stay in the hold area, where they are compared again during the next program cycle. During the next program cycle, the program reads a record from the file containing the record that was processed during the previous program cycle.

3 The program checks whether the match fields are in sequence.

4 If the match fields are not in sequence, the program stops.

5 If the match fields are in sequence, the program processes the record selected.

Figure 11-2. Flowchart for Processing Matching Records

When the match field from one record is the same as the match field from another record, the matching-record indicator (MR) turns on. Your program can use this indicator to control the calculation or output operations you want to do for matching records. For example, when the matching-record indicator (MR) turns on, you can enter data from primary records into their matching secondary records because the program processes the primary record before the matching secondary record. However, you can enter data from the first record of a secondary file to matching primary records only when you use look-ahead fields (see *Look-Ahead* under *Columns 19-20* in Chapter 25, *Input Specifications*).

When a record from the primary file matches a record from the secondary file, the program processes all the matching records from the primary file first. Then it processes all the matching records from the secondary file. The record-identifying indicator that identifies the record type just selected is on when the record is processed. Programs often use record-identifying indicators to control the type of processing.

When records in files that are in ascending order do not match, the program first processes the record whose match field contains the lowest value. When records in files that are in descending order do not match, the program first processes the record whose match field contains the highest value.

If a record type does not use a match field, the program processes it immediately after the record is read. The matching-record indicator is off. If this record type is first in the file, it is processed first even if it is not in the primary file.

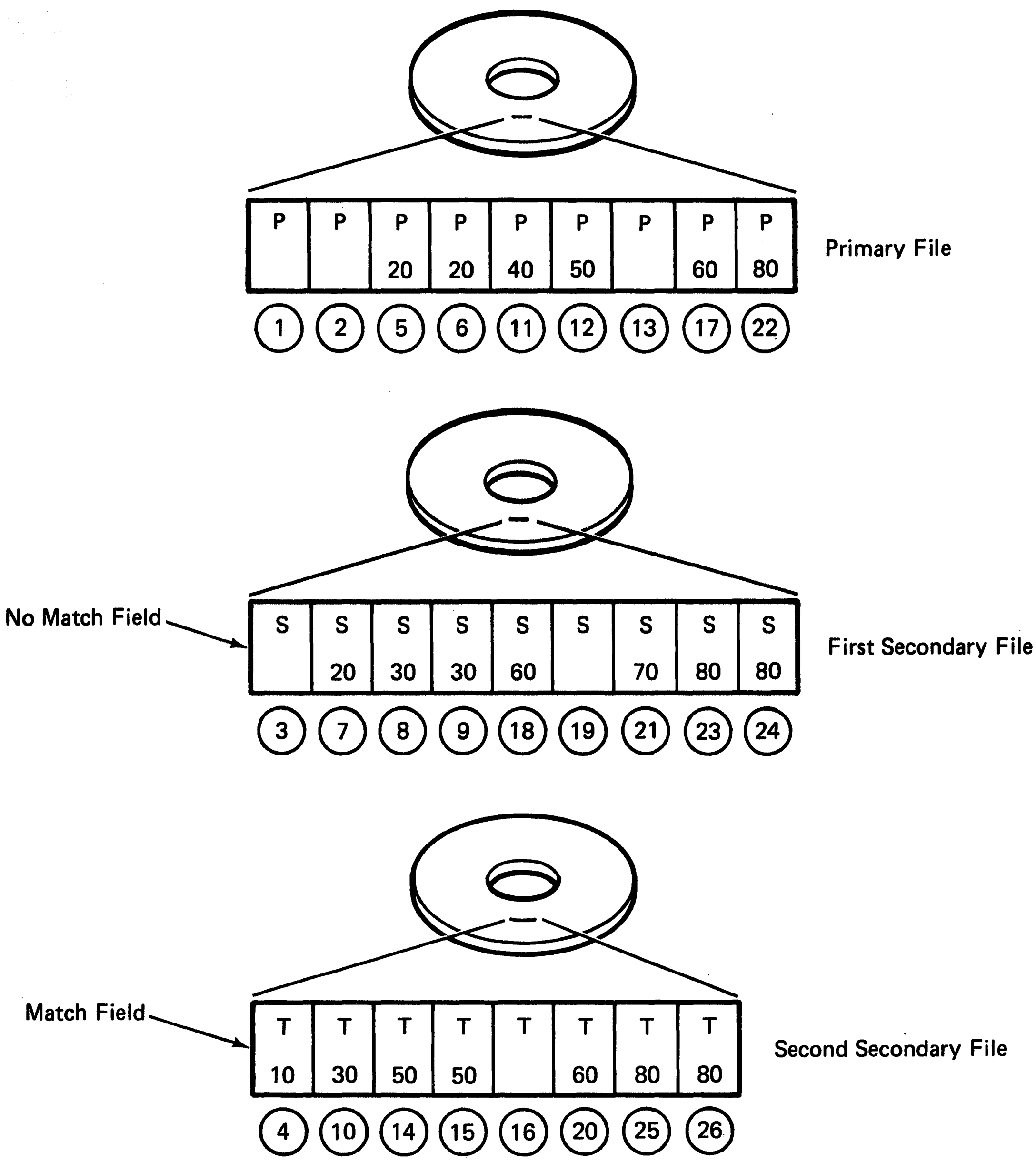
The program checks that the contents of match fields are in the correct sequence. If the contents are not in sequence, error message RPG-9032, FILE CONTAINS A RECORD NOT IN SEQUENCE, is displayed. If you respond by choosing option 1, the program does not process the record that is out of sequence. When the program starts again, it reads the next record from the same file. Therefore, all match fields must be in ascending order, or all must be in descending order (see *Column 18 (Sequence)* in Chapter 21, *File Description Specifications*).

Figure 11-3 shows how to code the file description and input specifications for a program that uses match fields in three DISK files. Figures 11-4 and 11-5 show how the program coded in Figure 11-3 selects records from the three files.

Line	Form Type	Filename	File Type	File Designation	End of File	Sequence	File Format	Block Length	Record Length	Device
02	F	PRIMARY	I	P	E	A	F	64	64	DISK
03	F	FIRSTSEC	I	S		A	F	64	64	DISK
04	F	SECSEC	I	S		A	F	64	64	DISK
05	F									

Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator	Position (1)	Not (N) (1)	C/Z/D (1)	Character (1)	Position (2)	Not (N) (2)	C/Z/D (2)	Character (2)	Field Location From	Field Location To	RPG Field Name	Matching Fields or Chaining Fields
01	I	PRIMARY	AA	Ø1	1		C	P	2	N	C					
02	I												2	3	MATCH	M1
03	I		BB	Ø2	1		C	P	2		C					
04	I												2	3	NOM	
05	I															
06	I	FIRSTSEC	AB	Ø3	1		C	S	2	N	C					
07	I												2	3	MATCH	M1
08	I		BC	Ø4	1		C	S	2		C					
09	I												2	3	NOM	
10	I															
11	I	SECSEC	AC	Ø5	1		C	T	2	N	C					
12	I												2	3	MATCH	M1
13	I		BD	Ø6	1		C	T	2		C					
14	I												2	3	NOM	
15	I															

Figure 11-3. Coding for Match Fields in Three DISK Files

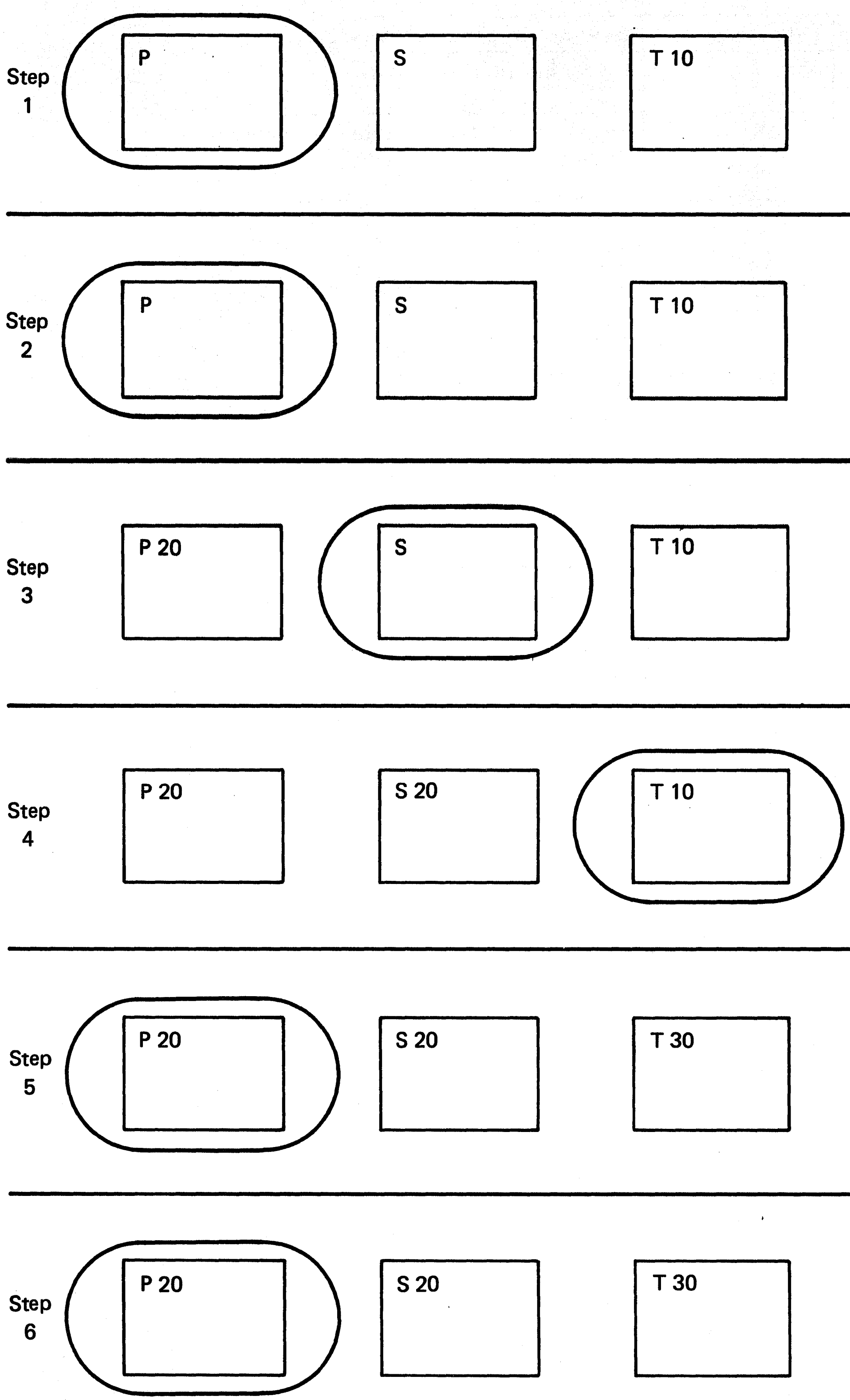


The records from the three DISK files are selected in the order indicated by the circled numbers.

Figure 11-4. Selecting Matching Records from Three DISK Files

Cycle	File Processed	Indicators On	Reason for Record Selection
1.	PRIMARY	02	No match field specified
2.	PRIMARY	02	No match field specified
3.	FIRST SEC	04	No match field specified
4.	SEC SEC	05	Second secondary low No primary match
5.	PRIMARY	01,MR	Primary matches first secondary
6.	PRIMARY	01,MR	Primary matches first secondary
7.	FIRST SEC	03,MR	First secondary matches primary
8.	FIRST SEC	03	First secondary low No primary match
9.	FIRST SEC	03	First secondary low No primary match
10.	SEC SEC	05	Second secondary low No primary match
11.	PRIMARY	01	Primary low No secondary match
12.	PRIMARY	01,MR	Primary matches second secondary
13.	PRIMARY	02	No match field specified
14.	SEC SEC	05,MR	Second secondary matches primary
15.	SEC SEC	05,MR	Second secondary matches primary
16.	SEC SEC	06	No match field specified
17.	PRIMARY	01,MR	Primary matches both secondary files
18.	FIRST SEC	03,MR	First secondary matches primary

Cycle	File Processed	Indicators On	Reason for Record Selection
19.	FIRST SEC	04	No match field specified
20.	SEC SEC	05,MR	Second secondary matches primary
21.	FIRST SEC	03	First secondary low No primary match
22.	PRIMARY	01,MR	Primary matches both secondary files
23.	FIRST SEC	03,MR	First secondary matches primary
24.	FIRST SEC	03,MR	First secondary matches primary
25.	SEC SEC	05,MR	Second secondary matches primary
26.	SEC SEC	05,MR	Second secondary matches primary



Step 1: The first record from each file is read. The P and S records have no match field, so they are processed before the T record that has a match field. Because the P record comes from the primary file, it is selected for processing first.

Step 2: The next P record is read. It contains no match field and comes from the primary file, so the new P record is also selected for processing before the S record.

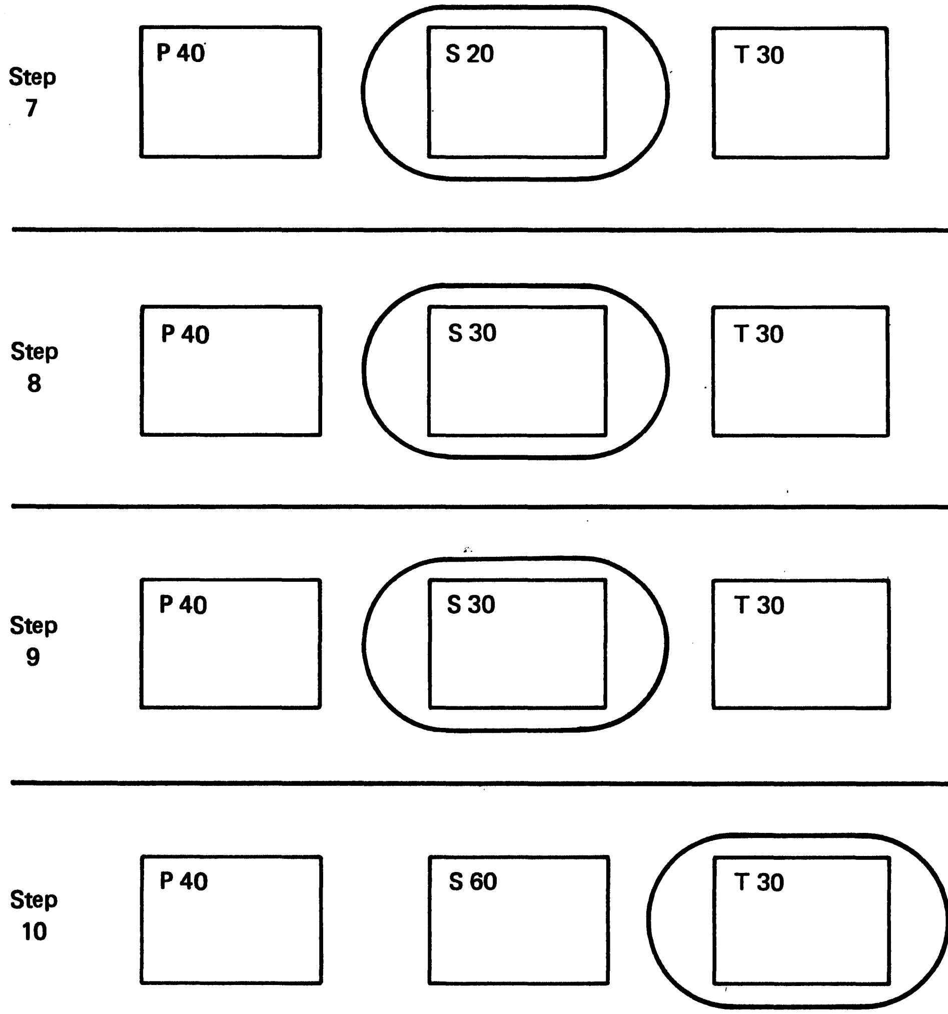
Step 3: The next P record read has a match field. The S record has no match field, so it is selected for processing.

Step 4: The next S record is read. All three records have match fields. Because the value in the match field of the T record is lower than the value in the other two, the T record is selected for processing.

Step 5: The next T record is read. The matching P and S records both have the low match field value, so they are processed before the T record. Because the matching P record comes from the primary file, it is selected for processing first.

Step 6: The next P record is read. Because it contains the same match field and comes from the primary file, the new P record is selected instead of the S record.

Figure 11-5 (Part 1 of 2). Selecting Matching Records from Three DISK Files



Step 7: The next P record is read. The value of the match field in the S record is the lowest of the three, so the S record is selected for processing.

Step 8: The next S record is read. Becau the S and T records match and h the lowest match field, they are selected before the P record. Because the S record comes fron the first secondary file, it is selec for processing before the T reco

Step 9: The next S record is read. Becau it also has the same match field a the S record just selected, it too selected before the T record.

Step 10: The next S record is read. The T record contains the lowest match field value, and is selected for processing.

Figure 11-5 (Part 2 of 2). Selecting Matching Records from Three DISK Files

Chapter 12. Using Indicators

Chapter 12. Using Indicators

The RPG program cycle is built around indicators. To you, an indicator is a 2-character entry on a specification form; the indicator turns on or off as the result of an operation, or it conditions when or if an operation occurs. To RPG, an indicator is an internal switch; the program uses the indicator to determine when or if an operation occurs in the program and what to do when the operation occurs.

Indicators are defined either by an entry on the specifications or by the RPG program itself. The columns on the specifications form in which you define an indicator determine how the indicator is used. An indicator that has been defined can then be used to condition calculation and/or output operations.

Figure 12-1 lists all the 2-character entries that can be used for each type of indicator.

Defined on RPG Specifications	01-99	1P	H1-H9	L1-L9	LR	MR	0A-0G 0V	U1-U8	KA-KN KP-KY	L0
Overflow indicator							X			
Record-identifying indicator[1]	X		X	X	X					
Control-level indicator				X	X					X
Field indicator	X		X							
Resulting indicator	X		X	X	X[2]		X	X	X[3]	
Defined by RPG Program										
External indicator								X		
Internal indicator		X			X	X				
Used to Represent a Condition										
File-conditioning indicator								X[4]		
Field-record-relation indicator[1]	X		X	X	X[5]	X[5]		X		
Level-zero indicator										X
Command-key indicator									X	
Halt indicator			X							
Conditioning indicators on calculation specifications	X		X	X	X	X	X	X	X	
Conditioning indicators on output specifications	X	X[6]	X	X	X	X	X[7]	X	X	X

[1]Not allowed on look-ahead fields.
[2]Not allowed for SETOF operation.
[3]Allowed for SET, KEY, and SETOF operations only.
[4]Not allowed for table input files.
[5]When field named is not a match field or a control field.
[6]Only for detail or heading lines.
[7]Cannot condition an exception line, but can condition fields within the exception record.

Figure 12-1. Entries for Each Type of Indicator

INDICATORS DEFINED ON RPG SPECIFICATIONS

You define the following indicators on the RPG specifications:

- Overflow indicator defined in columns 33 and 34 of the file description specifications
- Record-identifying indicator defined in columns 19 and 20 of the input specifications.
- Control-level indicator defined in columns 59 and 60 of the input specifications.
- Field indicator defined in columns 65 through 70 of the input specifications.
- Resulting indicator defined in columns 54 through 59 of the calculation specifications.

You must define these indicators in the specifications before you can use them on other specifications to condition operations in the program.

Overflow Indicators

An overflow indicator is defined by an entry in columns 33 and 34 of the file description specifications. The purpose of an overflow indicator is to signal when the end of a printed page has been passed. The indicator is assigned to the PRINTER file and turns on when the overflow line on the page is passed. This could occur at exception, detail, or total output time. You use the overflow indicator to condition those lines that you want to print at the end of one page or at the beginning of another.

Figure 12-2 shows the RPG program cycle related to overflow indicators.

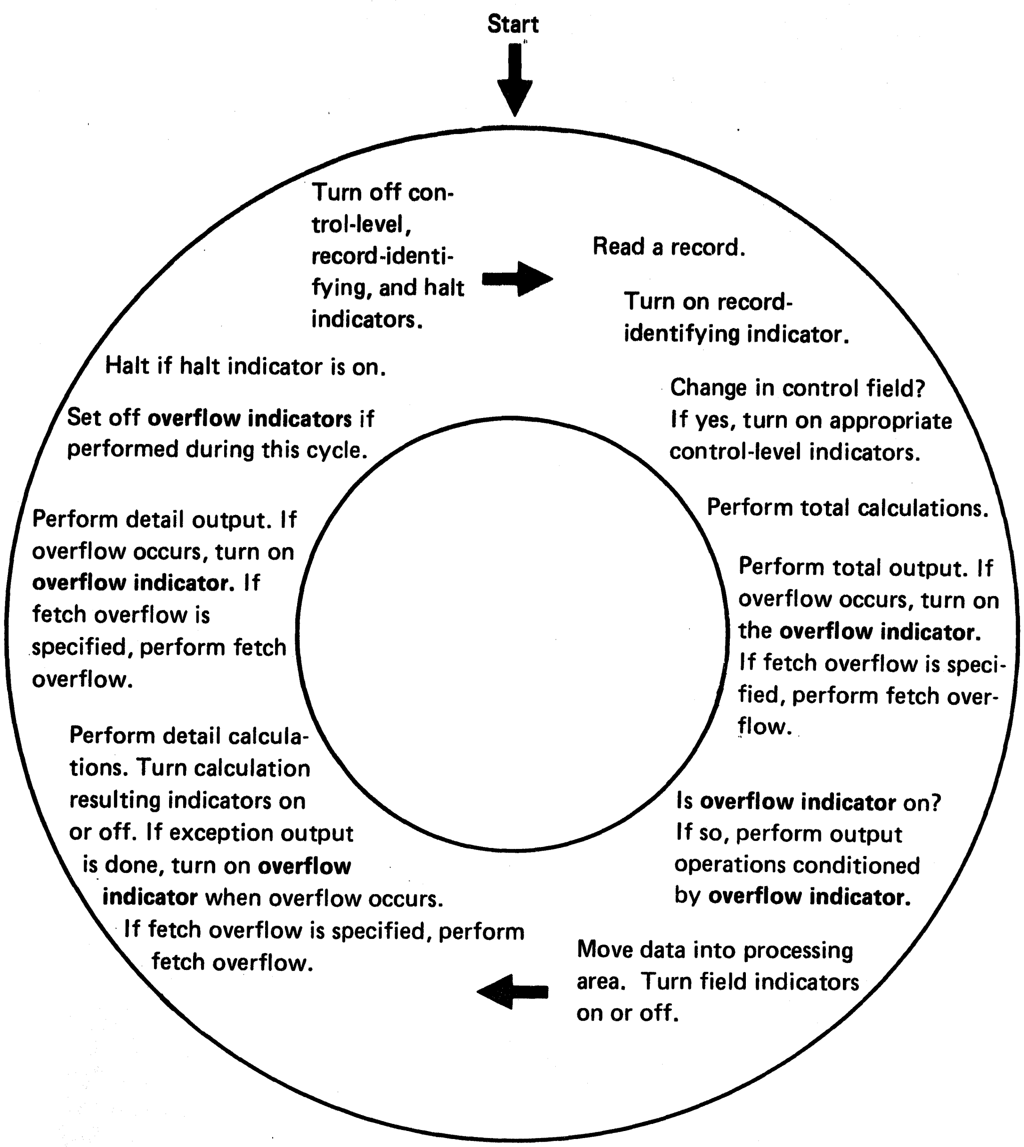


Figure 12-2. RPG Program Cycle for Overflow Indicators

The 2-character entries allowed as overflow indicators are:

OA through OG

OV

On the **file description specifications**:

- Columns 33 and 34 define an overflow indicator for each PRINTER file. If no overflow indicator is defined, the RPG program automatically handles overflow.

On the **calculation specifications**:

- Columns 9 through 17 can contain the overflow indicator defined in columns 33 and 34 of the file description specifications to condition calculations to be performed when overflow occurs.
- Columns 54 through 59 can contain the overflow indicator defined in columns 33 and 34 of the file description specifications to be set on or off as the result of calculations.

On the **output specifications**:

- Columns 23 through 31 must contain the overflow indicator defined in columns 33 and 34 of the file description specifications to condition all lines that are to be written to the associated printer when overflow occurs.

For more information on using overflow indicators, see *Handling Overflow* in Chapter 7, *Using a PRINTER File*.

Record-Identifying Indicators

You assign a record-identifying indicator to each type of record in the input file. You can also use a record-identifying indicator to associate a field with a particular record type by using the record-identifying indicator as a field-record-relation indicator. You do not have to assign the record-identifying indicators in any order. If certain operations in calculations and output are to be done for one record type only, you can condition those operations by the appropriate record-identifying indicator. By this method you can tell the RPG program what operations to do when it processes a specific record type. When several record types are specified in an OR relationship, all fields that do not have a field-record-relation indicator in columns 63 and 64 of the input specifications are associated with all record types in the OR relationship.

After the program selects the next record to process, it turns on the record-identifying indicator that you assigned to that record type. You can use this indicator to condition total and detail operations. This indicator is turned off by RPG before input occurs at input time of the RPG cycle.

Note: If you use a READ, READE, READP, or CHAIN operation in your program, input occurs during calculation time and your record-identifying indicators are not turned off before the READ, READE, READP, or CHAIN operation.

Figure 12-3 shows specific steps in the RPG program cycle related to record-identifying indicators.

The 2-character entries allowed as record-identifying indicators are:

01 through 99

H1 through H9

L1 through L9

LR

On the **input specifications**.

- Columns 19 and 20 define the record-identifying indicator. These columns should contain a different record-identifying indicator for each record type in a file.

- A record-identifying indicator must be assigned to the first input record in a WORKSTN file if this record is blank. The first input record is blank unless:

  - A read under format is performed.
  - PDATA-YES is specified in the procedure that called the program (see the explanation of the $MAINT utility program in the *System Reference* manual or the explanation of end of job in the *Source Entry Utility (SEU) Guide*.
  - Output to the WORKSTN file was performed first.

Note: When you use a control-level indicator (L1 through L9) as a record-identifying indicator and it turns on to indicate the type of record read, only that one control-level indicator turns on. All lower control-level indicators that you used remain unchanged.

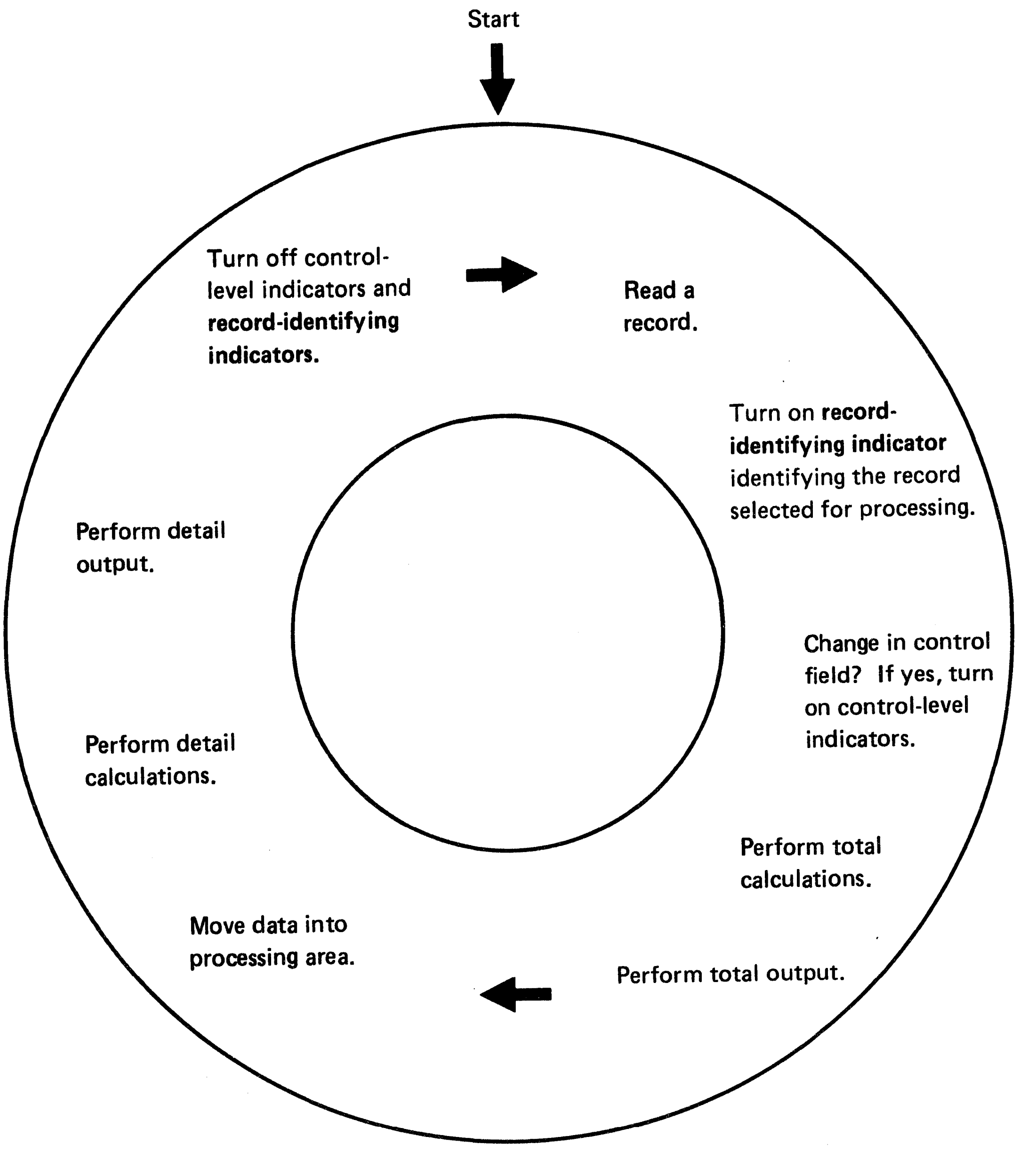


Figure 12-3. RPG Program Cycle for Record-Identifying Indicators

AND Relationship

Each line on the input specifications can contain up to three identifying characters (columns 27, 34, and 41). If the identification code you are using consists of more than three characters, an AND line must be used to describe the additional characters. To specify an AND line, write AND in columns 14 through 16.

You can use any number of AND lines to describe the record-identifying code for a record sequence. If AND lines and OR lines are combined, the total number of OR lines for one record sequence cannot be more than 20 and any number of AND lines can be used. The record must contain all the characters specified as its record identification code before the record-identifying indicator turns on. You cannot use a record-identifying indicator in the AND line of an AND relationship. AND lines are not allowed on CONSOLE files used for interactive data entry.

OR Relationship

If a particular record type can be identified by two different codes, you must use OR lines to specify that either of the codes can be present to identify the record. You can use up to 20 OR lines for each record sequence. If OR lines and AND lines are combined, the total number of OR lines for one record sequence cannot be more than 20 and any number of AND lines can be used. To specify an OR line, write the word OR in columns 14 and 15.

You can use the OR relationship to assign the same record-identifying indicator to two or more different record types if the same operation is to be done on all record types. You can also use record-identifying indicators on OR lines or every record type in the OR relationship that requires special processing.

When several record types are used in an OR relationship, all fields that do not have a field-record-relation indicator are associated with all record types in the OR relationship.

‸ıe of Using Record-Identifying Indicators

You can use record-identifying indicators in a billing program. Suppose that you keep a monthly file that contains records of purchases and payments made by each customer. In addition, the file contains a balance forward record for each customer. Figure 12-4 shows the three input record types used and the output records required.

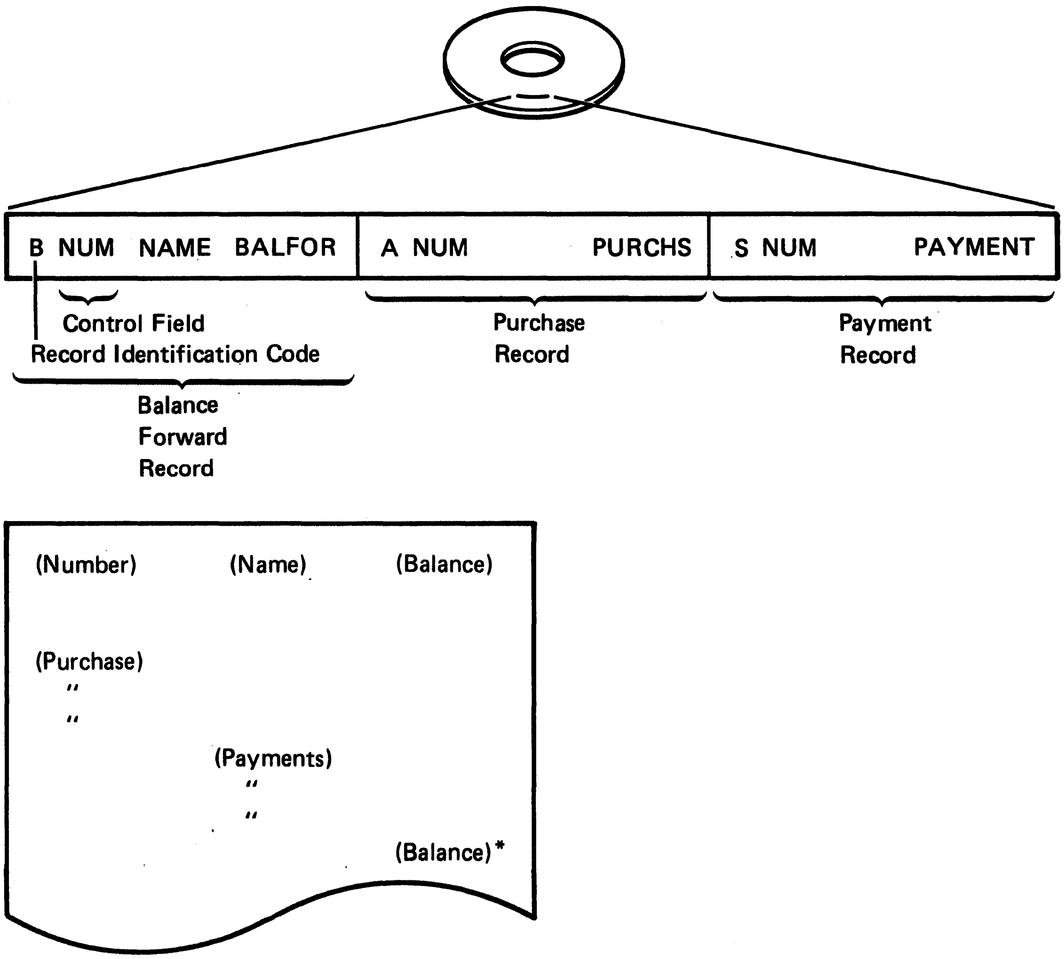


Figure 12-4. Input and Output for a Billing Program

The three record types are defined on the input specifications. Each type has a different record-identifying indicator. The record-identifying indicators are then used to show which operations are to be done for each record type. Figure 12-5 shows the input, calculation, and output specifications for the program. Use these specifications to help you follow, step by step, the operations done in the program cycles shown in Figure 12-6.

IBM International Business Machines Corporation

RPG INPUT SPECIFICATIONS

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

Line	Form Type	Filename or Record Name	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, ** or DS	Position	Not (N)	C/Z/D	Character	Field Location From	Field Location To	Decimal Positions	RPG Field Name	Control Level (L1-L9)
01	I	BILLING	01	1		10	96		C	B					
02	I										1	7		NUM	L1
03	I										8	30		NAME	
04	I										31	38	2	BALFOR	
05	I		02	N	O	20	96		C	A					
06	I										1	7		NUM	
07	I										31	38	2	PURCHS	
08	I		03	N	O	30	96		C	S					
09	I										1	7		NUM	
10	I										31	38	2	PAYMNT	
11	I														

Line	Form Type	Indicators	Factor 1	Operation	Factor 2	Result Field Name
01	C	20		ADD	PURCHS	BALFOR
02	C	30		SUB	PAYMNT	BALFOR
03	C					

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Space Before	Space After	Skip Before	Output Indicators	Field Name or EXCPT Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	Constant or Edit Word
01	O	REPORT	D		3	06	10					
02	O							NUM			15	
03	O							NAME			45	
04	O							BALFOR	A		60	
05	O		D		1		20					
06	O							PURCHS	1		20	
07	O		D		1		30					
08	O							PAYMNT	1		40	
09	O		T	1	2		L1					
10	O							BALFOR	A		60	
11	O										61	'*'
12	O											

Figure 12-5. Specifications Using Record-Identifying Indicators for a Billing Program

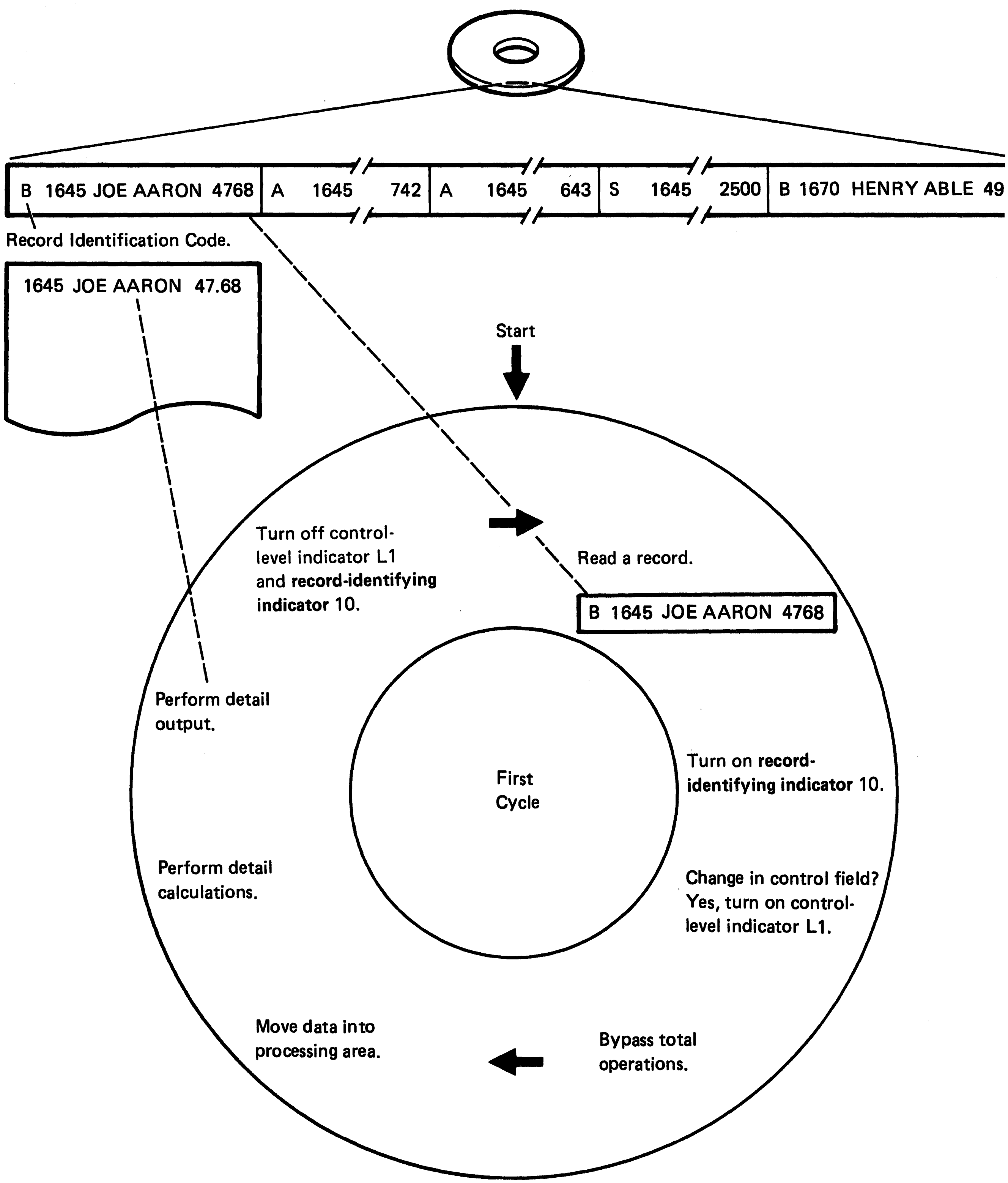


Figure 12-6 (Part 1 of 3). RPG Program Cycle for Record-Identifying Indicators

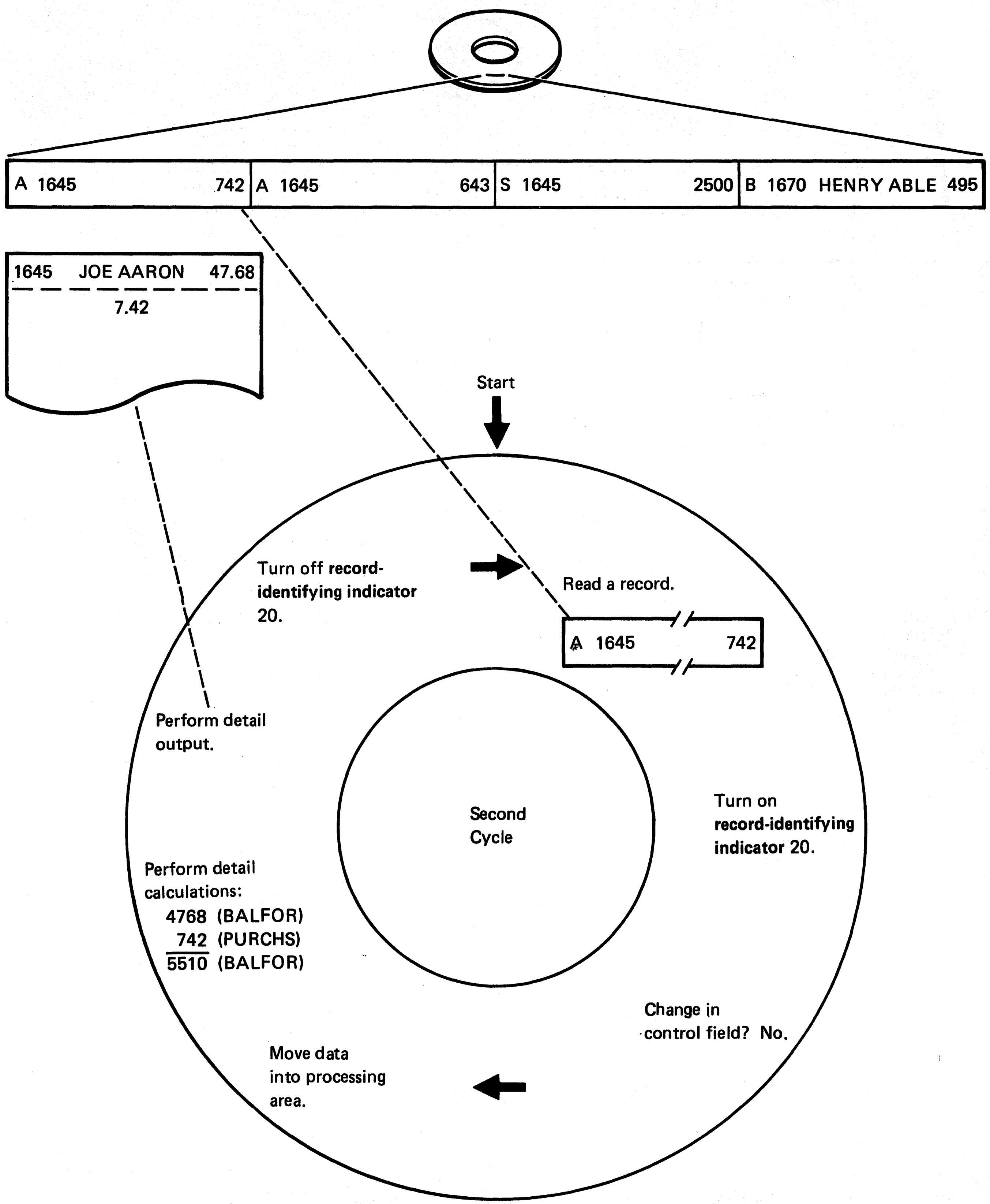


Figure 12-6 (Part 2 of 3). RPG Program Cycle for Record-Identifying Indicators

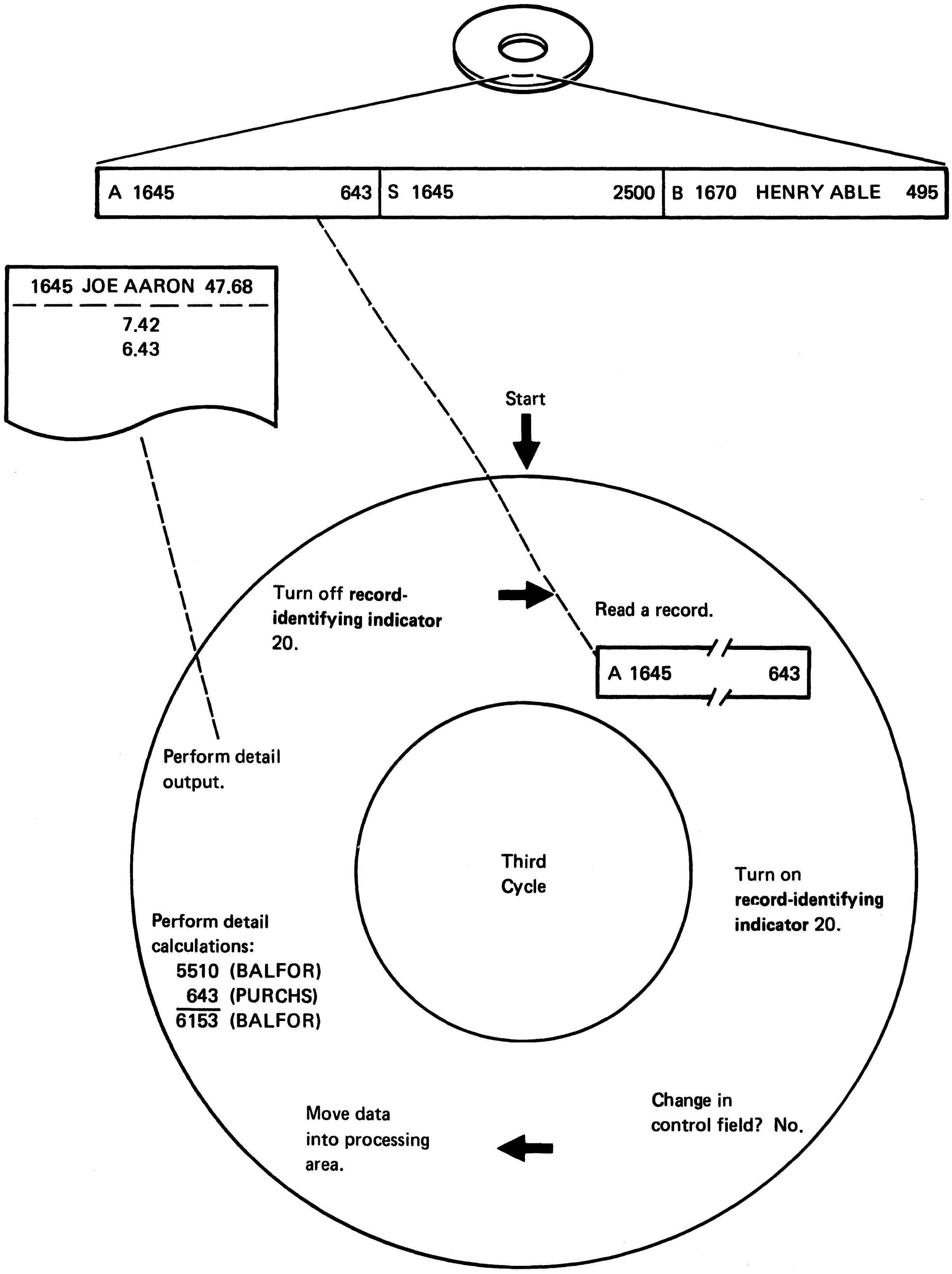


Figure 12-6 (Part 3 of 3). RPG Program Cycle for Record-Identifying Indicators

Control-Level Indicators

A control-level indicator tells the program when calculation or output operations are to be done. You can assign a control-level indicator to any field; this field is then known as a *control field*. The program checks the field for a change in information. When the information changes, a *control break* occurs. All records that have the same information in the control field are known as a *control group*.

Whenever the program reads a record containing a control field, it compares the data in the control field with data in the same control field from the previous record. When a control break occurs, the control-level indicator turns on. Operations conditioned by the control-level indicator are then performed.

There are nine control levels (level 1 through level 9). Each control level has a corresponding control-level indicator (L1 through L9). When a control-level indicator turns on, all control-level indicators with a lower number also turn on. For example, if indicator L3 turns on, indicators L2 and L1 automatically turn on. However, when a control-level indicator used as a record-identifying indicator turns on to indicate the type of record read, or when the SETON operation turns on a control-level indicator, only that one control-level indicator turns on. In that case, all lower control-level indicators remain unchanged.

The 2-character entries allowed as control-level indicators are:

L0

L1 through L9

LR

On the **input specifications**:

- Columns 59 and 60 define the control-level indicator used to condition input fields so that the program can check for any change in the information in this field. L1 through L9 are the only control-level indicators allowed here.

On the *calculation specifications*:

- Columns 7 and 8 must contain L0, LR, or the control-level indicator defined in columns 59 and 60 of the input specifications to specify a calculation operation to be done when the appropriate control break occurs at total calculation time.

- Columns 9 through 17 can contain L0, LR, or the control-level indicator defined in columns 59 and 60 of the input specifications to control the conditions under which a detail calculation operation is done on the record that caused the control break.

On the **output specifications**:

- Columns 23 through 31 can contain L0, LR, or the control-level indicator defined in columns 59 and 60 of the input specifications to tell the program the conditions under which a total record or field is to be written. Also, the control-level indicator can condition detail output operations to be done on the record that caused the control break.

Assigning Control-Level Indicators

The following points apply to control-level indicators:

- If the same control-level indicator is used in more than one record type or in more than one file, the control fields associated with that control-level indicator must be the same length and same type (alphabetic or numeric).

- In the same record type, record positions in control fields assigned different control-level indicators can overlap (see Figure 12-7). However, the total number of positions assigned as control fields must not be greater than 144.

- Field names are ignored in control-level operations. Therefore, fields from different record types that have been assigned the same control-level indicator can have the same name.

- Control levels need not be assigned in any order. For example, you can use indicator L2 before L1. You can also leave gaps in the control levels you assign.

- When numeric control fields with decimal positions are compared to determine whether a control break has occurred, they are always treated as if they have no decimal positions. For instance, 3.46 is considered equal to 346.

- If a field is specified as numeric, only the digit portion determines whether a control break has occurred. This means that a field is always considered to be positive. For instance, -5 is considered equal to +5.

- All control fields given the same control-level indicator are considered numeric if any one of those control fields is described as numeric (that is, if column 52 of the calculation specifications has an entry). Therefore, when numeric control fields are compared to determine whether the information has changed, only the digit portion of each character is compared.

- Control fields are initialized to hexadecimal zeros.

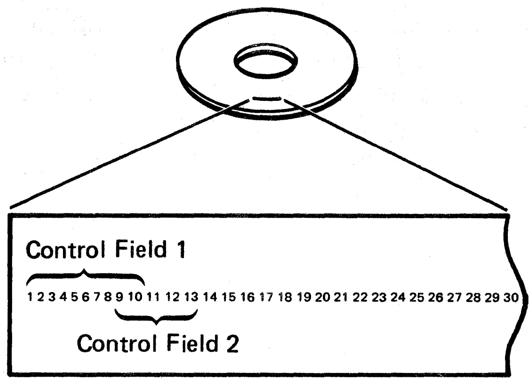


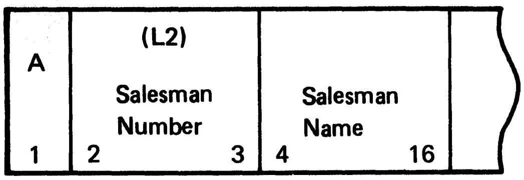
Figure 12-7. Overlapping Control Fields in a Disk Record

- A control break can occur after the first record containing a control field is read. The control fields in this record are compared with an area in storage that contains hexadecimal zeros. Because the fields being compared are not from two different records, total calculations and total output operations are bypassed for this cycle. A control break does occur then, but it is not considered to be a true control break.

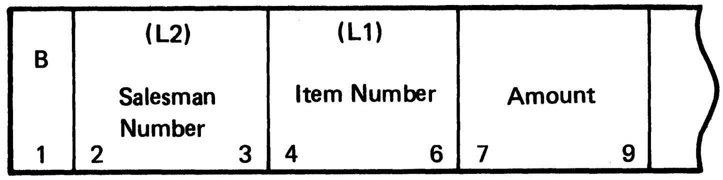
- If different record types in a file do not have the same number of control fields, unwanted control breaks can occur. See Figure 12-8 for an example of how to avoid unwanted control breaks.

- A control field cannot be specified as having a binary format (B in column 43 of the input specifications). However, it can be specified as having a packed-decimal format (P in column 43 of the input specifications).

- A control field can be related to a particular record type in an OR relationship with a field-record-relation indicator. If the control field does not have a field-record-relation indicator, the control field is used with all record types in the OR relationship.



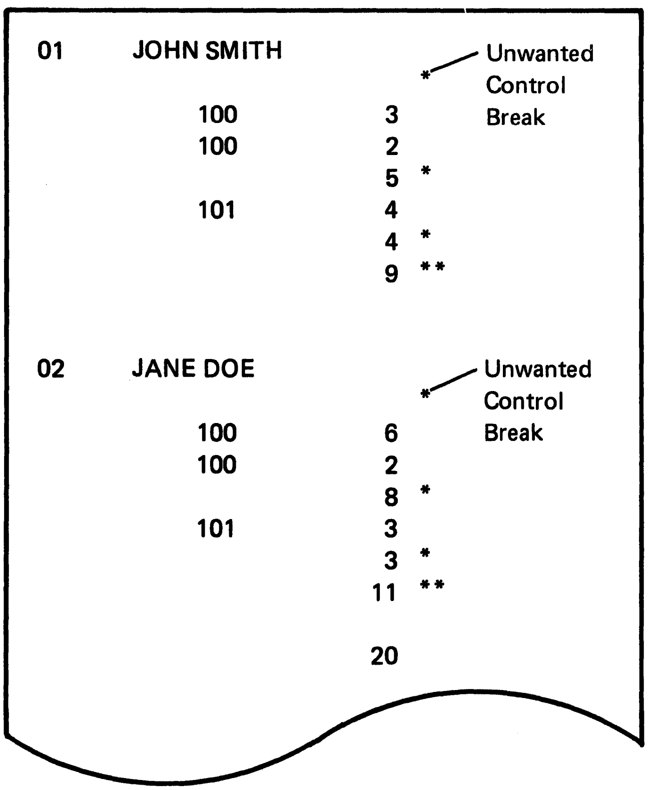
Salesman Record



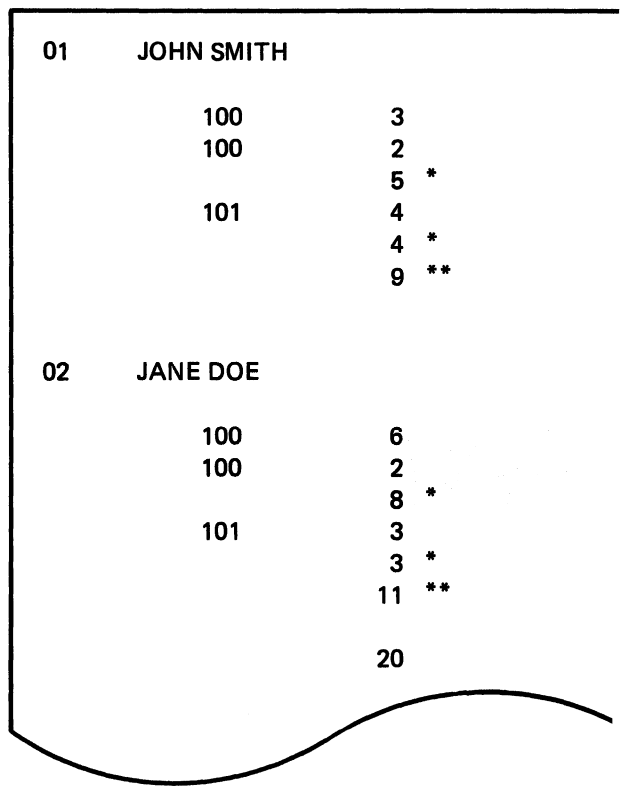
Item Record

Different record types normally contain the same number of control fields. However, some applications require a different number of control fields in some records.

The salesman records contain only the L2 control field. The item records contain both L1 and L2 control fields. With normal RPG coding, an unwanted control break is created by the first item record following the salesman record. This is recognized by an L1 control break immediately following the salesman record and results in an asterisk being printed on the line below the salesman record.



Output Showing Unwanted Control-Level Break



Corrected Output

Figure 12-8 (Part 1 of 3). Unwanted Control Breaks

Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator	Position	C/Z/D	Character	From	To	Decimal Positions	RPG Field Name	Control Level (L1-L9)
01	I	SALES	AA	Ø1	1	C	A					
02	I							2	3		L2FLD	L2
03	I							4	16		NAME	
04	I		BB	Ø2	1	C	B					
05	I							2	3		L2FLD	L2
06	I							4	6		L1FLD	L1
07	I							7	9	Ø	AMT	
08	I											

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Resulting Indicators High (Plus, 1>2)
01	C		Ø1		SETON					11
02	C		Ø2		SETOF					11
03	C		Ø2	AMT	ADD	L1TOT	L1TOT	5	Ø	
04	C	L1		L1TOT	ADD	L2TOT	L2TOT	5	Ø	
05	C	L2		L2TOT	ADD	LRTOT	LRTOT	5	Ø	
06	C									

This coding prevents the unwanted control break. Line 01 of the calculation specifications sets on indicator 11 when the salesman record is read. When the next item record causes an L1 control break, no total output is printed because indicator 11 is on (line 07 of output specifications). Detail calculations are then processed for the item record, and line 02 of the calculation specifications sets indicator 11 off. This allows the normal L1 control break to occur.

Figure 12-8 (Part 2 of 3). Unwanted Control Breaks

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Space Before	Space After	Output Indicators	Field Name or EXCPT Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	Constant or Edit Word
01	O	PRINTER	D	1	1	Ø1					
02	O						L2FLD			5	
03	O						NAME			25	
04	O		D		1	Ø2					
05	O						L1FLD			15	
06	O						AMT	Z		25	
07	O		T		1	L1 N11					
08	O						L1TOT	Z	B	25	
09	O									27	'*'
10	O		T		1	L2					
11	O						L2TOT	Z	B	25	
12	O									28	'**'
13	O		T		1	LR					
14	O						LRTOT	Z	B	25	
15	O										

Figure 12-8 (Part 3 of 3). Unwanted Control Breaks

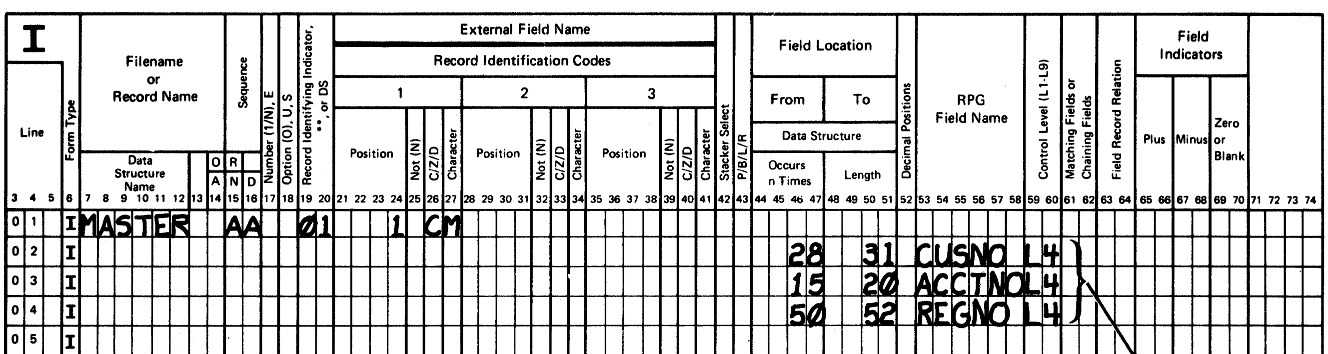
Split Control Fields

If a control field is made up of more than one field of a record, it is known as a *split control field*. A split control field is created when the same indicator is assigned to two or more fields (connected or unconnected) on the same record type.

All fields in one record that have the same control-level indicators are combined by the program in the order specified by the input specifications and are treated as one control field (see Figure 12-9).

The following rules apply to split control fields:

- For one control-level indicator, a field can be split in some record types and not in others if the field names are different. However, the length of the field, whether split or not, must be the same in all record types.
- The length of the portions of a split control field can vary for different record types if the field names are different. However, the total length of the portions must always be the same.
- No other specifications can come between lines that describe split control fields.
- If one section of a split control field is numeric, the whole field is considered numeric.
- A numeric split control field can have more than 15 characters if no portion of the split field has more than 15 characters and if the sum of all control fields is not more than 144 characters.
- A split control field cannot be made up of a packed-decimal field and a zoned-decimal field. Both portions of the control field must be packed decimal, or both must be zoned decimal.



All portions of a split control field must be assigned the same control-level indicator.

Figure 12-9. Split Control Fields

Field Indicators

Field indicators are used to test a field on an input record for a plus, minus, zero, or blank value. You can use the appropriate field indicator to condition operations that are done only when a numeric field is plus, minus, or zero, or when an alphameric field is blank.

Note: A numeric field that is all blanks turns on an indicator used for all zeros. However, an alphameric field that is all zeros does not turn on an indicator used for all blanks. You cannot specify indicators in columns 65 through 68 for an alphameric field.

Field indicators turn on or off after data from the record to be processed moves into the processing area. Figure 12-10 shows the RPG program cycle related to field indicators.

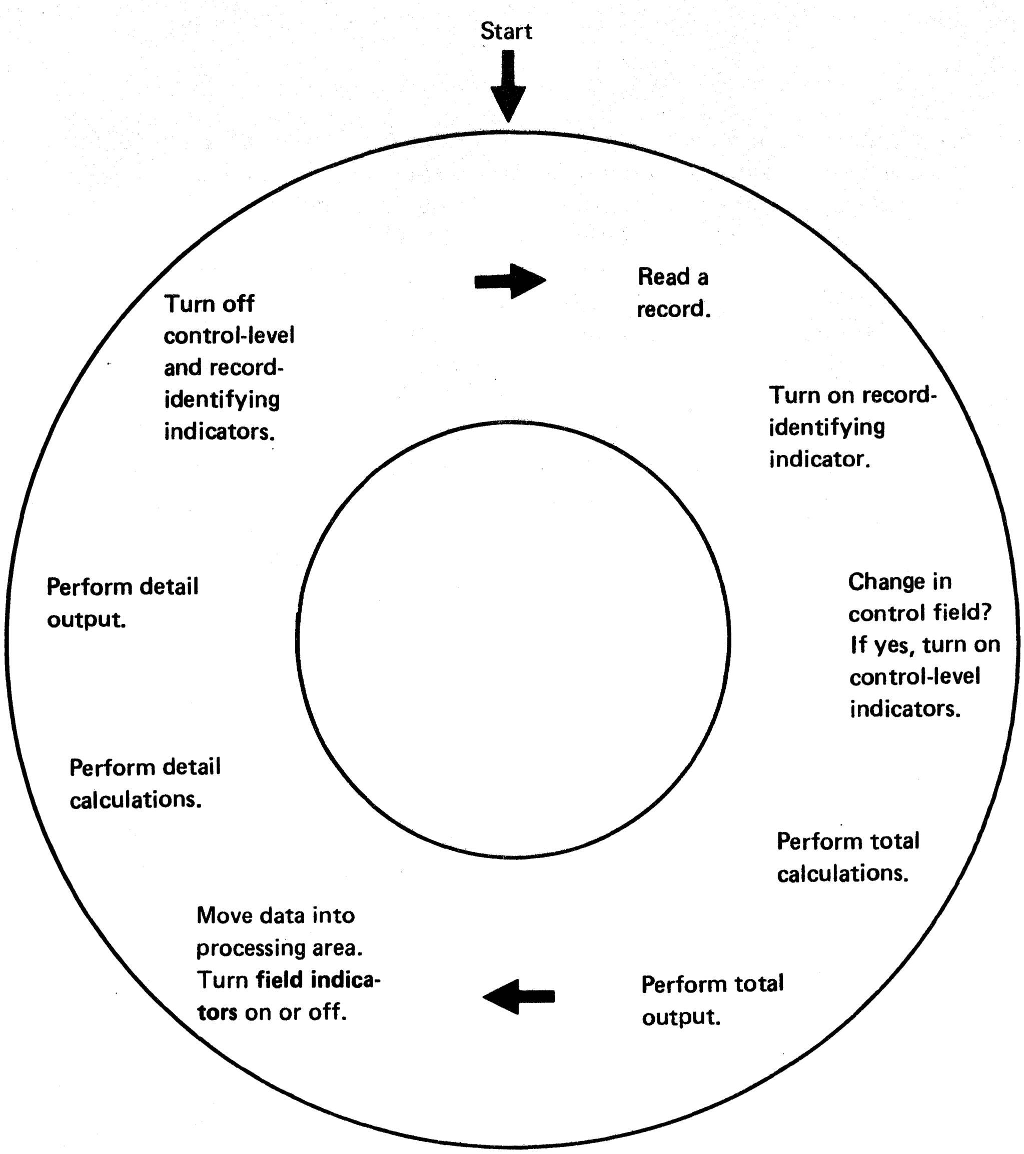


Figure 12-10. RPG Program Cycle for Field Indicators

For each program cycle, field indicators are set to reflect the result of the test on a field. If the condition tested for exists, they turn on; if the condition does not exist, they turn off. After the program tests the result of a field, a field indicator stays on or off until the program uses the same indicator as a resulting indicator.

When the indicator is on, any detail and total operations conditioned by the field indicator can be done before the program resets the indicator by testing a field. However, at total time the field indicator has the setting established in the previous cycle.

The following considerations apply to field indicators:

- A numeric input field can be assigned two or three field indicators. However, only the indicator that signals the result of the test on that field turns on; the others remain off.

- If the same field indicator is assigned to fields in different record types, its status is always based on the last record type selected.

- When different field indicators are assigned to fields in different record types, a field indicator turned on remains on until another record of that type is read. Similarly, a field indicator assigned to more than one field within a single record type always reflects the status of the last field defined.

Field indicators assigned in columns 65 through 70 can also be set on or set off by SETON or SETOF operations in the calculation specifications.

The 2-character entries allowed as field indicators are:

01 through 99

H1 through H9

Use the 2-character entries 01 through 99 to test whether a numeric field is plus, minus, zero, or blank. Use the 2-character entries H1 through H9 to check for an error condition in your data.

On the **input specifications**:

- Columns 65 and 66 define a field indicator to check for a plus condition. The indicator turns on if the numeric field is greater than zero.

- Columns 67 and 68 define a field indicator to check for a minus condition. The indicator turns on if the numeric field is less than zero.

- Columns 69 and 70 define a field indicator to check for zeros or blanks. The indicator turns on if the numeric field is all zeros or if an alphameric field is all blanks.

On the **output specifications**:

- Columns 23 through 31 can contain a 2-character entry H1 through H9 to prevent data that causes an error from being used.

Resulting Indicators

Resulting indicators signal something about the result of a calculation operation. You can use a resulting indicator to condition any operation that depends on the result of the calculation.

You can use a resulting indicator in columns 54 through 59 on the calculation specifications to reflect the result of an operation, or to indicate an end-of-file condition, a no-record-found condition, or an exception/error condition. The indicator specified turns on only if the result field satisfies the condition being tested for. If the condition tested for is not met, the indicator is turned off. This indicator can then be used to condition following calculations or output operations (see Figure 12-11). If you use the same indicator to test the result of more than one operation, the last operation performed determines the setting of the indicator.

In Figure 12-11, indicators 10 and 20 in columns 54 through 57 are used to test for the different conditions in a subtract operation. These indicators are used to condition the calculations that must be done for a payroll job. Indicator 10 turns on if the hours worked (HRSWKD) are greater than 40 and is then used to condition all operations necessary to calculate overtime pay. Indicator 20 turns on if HRSWKD is less than 40. Indicator 20 is also used to condition other operations. In line 03, if indicator 20 is not on (the employee worked 40 or more hours), regular pay is calculated based on a 40-hour week. In line 06, if indicator 20 is on (employee worked less than 40 hours), pay is calculated based on less than a 40-hour week.

The 2-character entries allowed as resulting indicators are:

01 through 99

H1 through H9

KA through KN, and KP through KY

L1 through L9

LR

OA through OG, and OV

U1 through U8

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Plus / High (54-55)	Minus / Low (56-57)	Zero / Equal (58-59)	Comments
01	C													
02	C			HRSWKD	SUB	4Ø	OVERTM	3	Ø		1Ø	2Ø		
03	C		N2Ø	PAYRAT	MULT	4Ø	PAY	6	2	H				
04	C		1Ø	OVERTM	MULT	OVRRAT	OVRPAY	6	2	H				
05	C		1Ø	OVRPAY	ADD	PAY	PAY							
06	C		2Ø	PAYRAT	MULT	HRSWKD	PAY							
07	C													

Figure 12-11. Conditioning Operation (Resulting Indicators)

On the **calculation specifications**:

- Columns 54 through 59 define the resulting indicator to be turned on or off by the SETON or SETOF operation codes. The headings (high, low, and equal) for columns 54 through 59 have no meaning for SETON or SETOF operations.

- Columns 54 and 55 (plus or high) must define a resulting indicator when testing:

  - Whether the result field in an arithmetic operation is positive
  - Whether factor 1 is higher than factor 2 in a compare (COMP) operation
  - Whether factor 2 is higher than factor 1 in an array or table LOKUP operation
  - Whether a CHAIN operation is not successful
  - Whether each bit named in factor 2 is off for a TESTB operation
  - Whether the character tested in a TESTZ operation is one of the following: &, A through I
  - Whether the numeric field entered in a KEY operation is positive
  - Whether the system operator has requested shutdown on a SHTDN operation

- Columns 56 and 57 (minus or low) must define a resulting indicator when testing:

  - Whether the result field in an arithmetic operation is negative
  - Whether factor 1 is lower than factor 2 in a compare (COMP) operation
  - Whether factor 2 is lower than factor 1 in a table or array LOKUP operation
  - Whether the bits named in factor 2 are of mixed status (some bits on, some bits off) for a TESTB operation
  - Whether the character tested in a TESTZ operation is one of the following: - (minus), J through R
  - Whether the numeric field entered in a KEY operation is negative
  - Whether the ACQ, REL, NEXT, READ, or POST operation to a WORKSTN file is not successful

- Columns 58 and 59 (zero or equal) must define a resulting indicator when testing:

  - Whether the result field in an arithmetic operation is zero
  - Whether factor 1 is equal to factor 2 in a compare (COMP) operation
  - Whether factor 2 is equal to factor 1 in a table or array LOKUP operation
  - Whether the program reached the end of a file that is read by a READ operation, the end of an equal key for a READE operation, or the beginning of file condition for a READP operation.
  - Whether each bit named in factor 2 is on for a TESTB operation
  - Whether the character tested in a TESTZ operation is any character other than &, A through I, - (minus), or J through R
  - Whether the numeric field entered in a KEY operation is zero or an alphameric field is blank

Indicators Not Defined on the RPG Specifications

You do not need to define all the indicators in your program to condition operations. External indicators (U1-U8) are defined by a control language statement or by a previous RPG program. The internal indicators first-page (1P) and last-record (LR) are defined for you by the RPG program cycle itself. The matching-record indicator (MR) is defined for you if you use M1 through M9 in columns 61 and 62 of the input specifications.

External Indicators

External indicators are usually set prior to processing by the control language SWITCH statement or by a previous RPG program. External indicators are automatically read into the program when the program begins running and are passed to other programs at the end of the job. Their setting can be changed during processing, allowing the program to change the status of these indicators. However, if an external indicator conditions a file, that indicator must be set on before the program is loaded in order to use the file in the program.

Use external indicators to:

- Determine whether a file is to be used for a program
- Condition calculation operations
- Condition output operations
- Indicate the relation of a field to a record
- Provide communication between programs

Note: You can also use SUBR20 to read and write external indicators. For more information, see Chapter 6, *Using a WORKSTN File.*

The 2-character entries allowed as external indicators are:

U1 through U8

On the **file description specifications**:

- Columns 71 and 72 can contain an external indicator to condition a file. A file conditioned by an external indicator is used only when the indicator is on. When the indicator is off, the file is treated as though the program reached the end of the file; that is, no records can be read from or written to the file.

On the **input specifications**:

- Columns 63 and 64 can contain an external indicator to tell the program to accept and use data from a particular field only when the external indicator is on.

On the **calculation specifications**:

- Columns 9 through 17 can contain external indicators to condition which operations should be done for a specific job. If a file is conditioned by an external indicator, any calculations that are to be performed only on that file should be conditioned by the same external indicator.

On the **output specifications**:

- Columns 23 through 31 can contain an external indicator to condition certain output records on external conditions.

If you want to pass information to other programs, you can use external indicators as resulting indicators.

Internal Indicators

First-Page Indicator

In the first program cycle, the first-page indicator is on during the beginning of the cycle. Any records conditioned by the first-page indicator are printed before the first record is read.

The purpose of the first-page indicator is to condition records that are to be printed on the first page of a report. These records are usually headings used to identify information found on the page, but they can also be detail lines.

The first-page indicator is an internal indicator that is defined by the RPG program cycle itself. It turns on only for the beginning of the first cycle. It turns off before a record is read and is never used again during the program (see Figure 12-12).

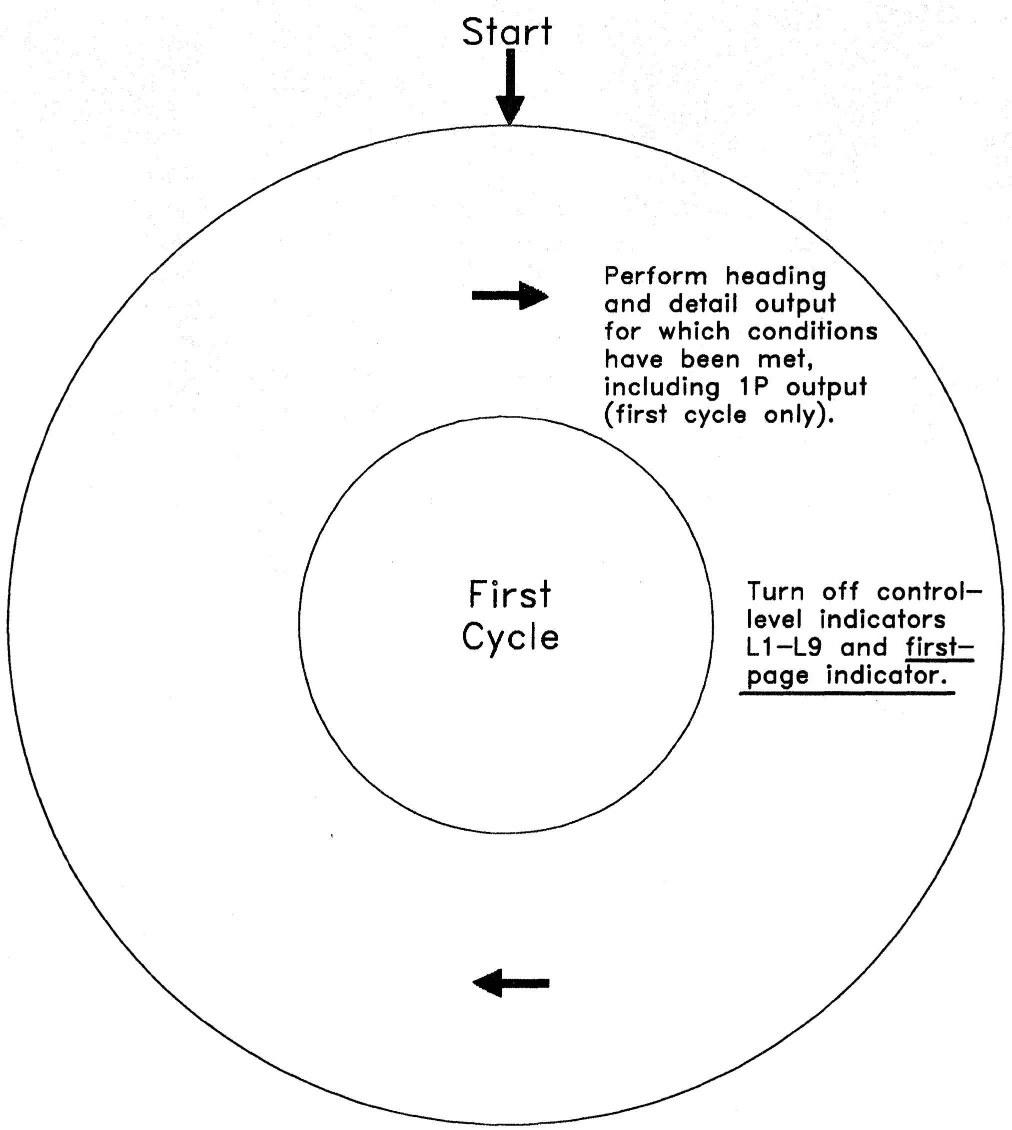


Figure 12-12. RPG Program Cycle for the First-Page Indicator

Notice in Figure 12-12 that the program does first-page output and other heading and detail output first. This happens in every RPG program. The program writes first-page output and any other heading or detail output for which specified conditions are met before the first record is read. After the first cycle, however, it is easier to think of reading a record as the first step in the cycle.

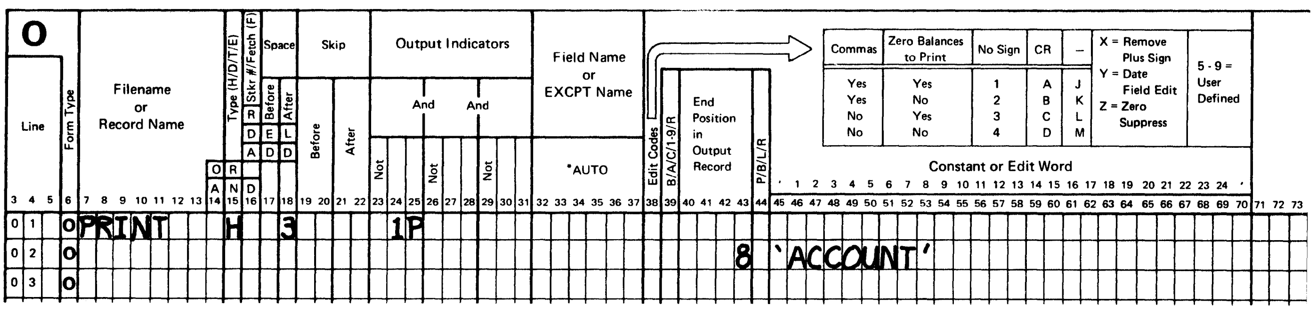
The only 2-character entry allowed as a first-page indicator is:

1P

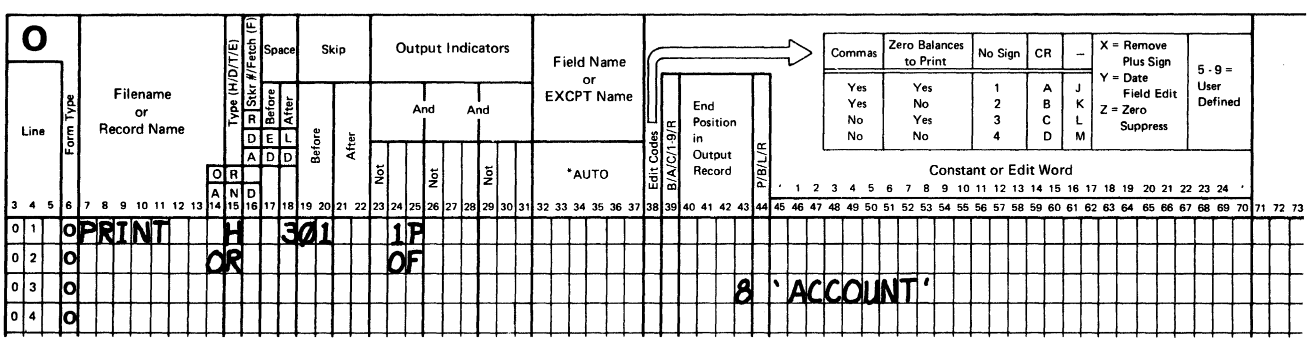
On the **output specifications**:

- Columns 23 through 31 can contain a first-page indicator to condition lines that are to be printed on only the first page.

You can use the first-page indicator only in columns 23 through 31 of heading or detail output lines, not with total or exception output lines. You can use the first-page indicator in an OR relationship with an overflow indicator to allow printing on every page (see Figure 12-13). You cannot use the first-page indicator in an AND relationship with control-level indicators, to condition output for a WORKSTN file, or to condition calculation operations.



The first-page (1P) indicator is used when headings are to be printed on the first page only.



The first-page (1P) indicator and an overflow indicator can be used to print headings on every page.

Figure 12-13. First-Page Indicator

Last-Record Indicator

You use the last-record indicator to condition all operations done at the end of your program. These operations usually include calculating totals for all records or writing summary information. When the last-record indicator turns on, the control-level indicators L1 through L9 also turn on. Thus, all total operations conditioned by L1 through L9 and LR are performed. See Figure 12-14 for the specific steps that occur at the end of a job.

The RPG program cycle sets on the last-record indicator when end of file occurs for a primary file. End of file occurs for a primary DISK file when you read past the last record in the file. End of file occurs for a primary WORKSTN file when:

- All display stations are released (by an R in column 16 of the output specifications or by the REL operation code) if the program is not a never-ending program.
- All display stations are released and the operator entered the STOP SYSTEM command if the program is a never-ending program.

You must set on the last-record indicator if:

- The program contains no primary file.
- KEYBORD is specified as the device for a primary input file.

If you use any of the L0 through L9 indicators in an OR relationship with a last-record (LR) indicator, the specified operation is done twice when LR is on. One operation is done at total time and the other during last-record (LR) processing.

Once the LR indicator has been set on, it must not be set off.

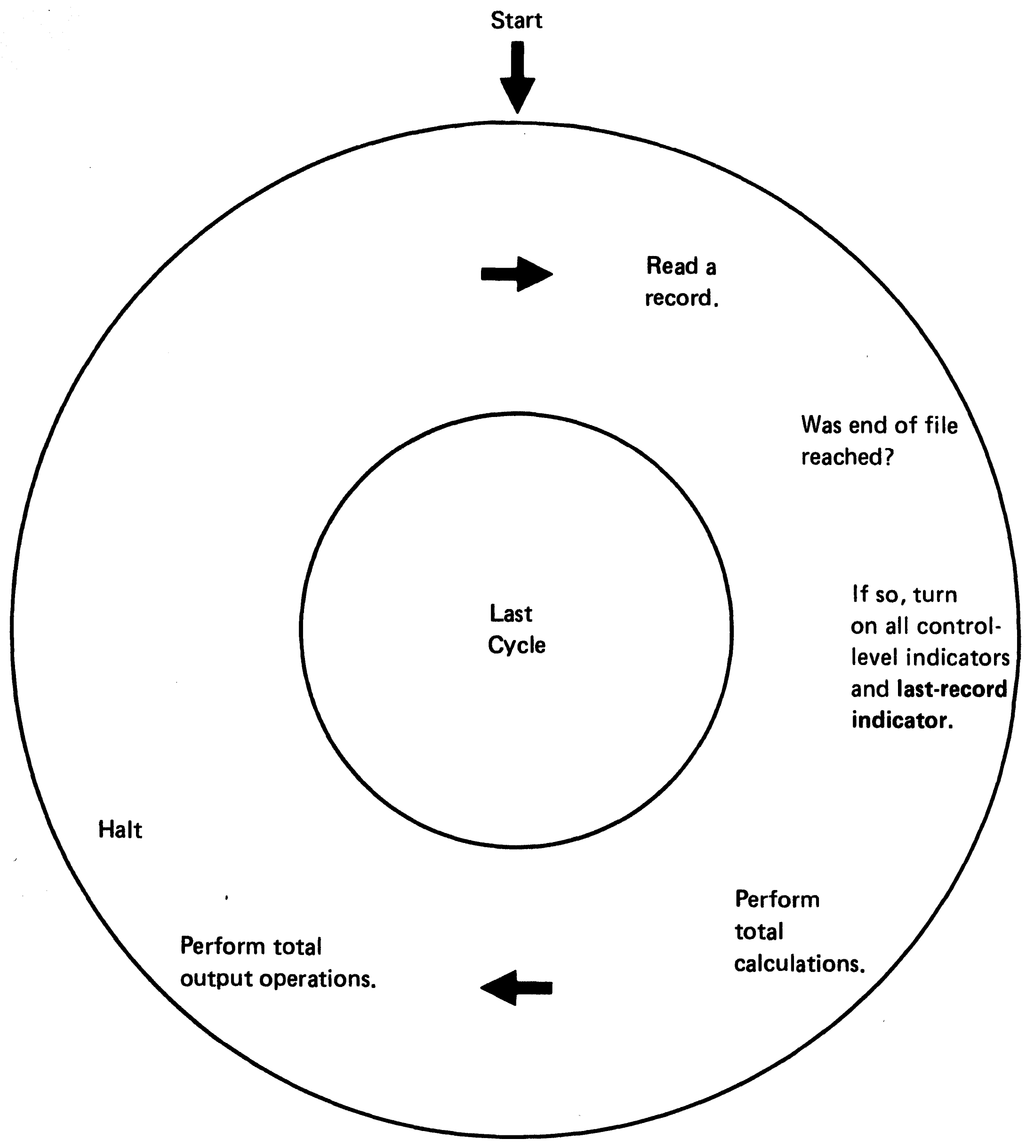


Figure 12-14. RPG Program Cycle for the Last-Record Indicator

The only 2-character entry allowed as a last-record indicator is:

LR

On the **input specifications**:

- Columns 19 and 20 can contain LR as a record-identifying indicator.

On the **calculation specifications**:

- Columns 7 and 8 must contain LR for all operations to be done at the end of the job. When the last-record indicator turns on at the end of the job, the other control-level indicators you specified also turn on.

- Columns 9 through 17 can contain LR to condition operations when the last-record indicator turns on during calculations.

- Columns 54 through 59 can contain LR except for the SETOF operation. When the last-record indicator turns on in calculations, the other control-level indicators you specified do not turn on until the beginning of the next cycle.

On the **output specifications**:

- Columns 23 through 31 can contain LR to condition output after all records are processed.

Matching-Record Indicator

Use the matching-record indicator only when you are processing primary and secondary files. Its purpose is to indicate when fields or records from different files match. The matching-record indicator is set on or off only after total operations are performed. Thus, at detail time, it always signals the matching status of the record just selected for processing; at total time, it reflects the matching status of the previous record.

In processing primary and secondary files, you must specify match fields to compare records from two or more input or update files to determine which record is to be selected for processing. You can use one field, many fields, or an entire record to match records. Whenever the contents of the match field from the primary file record are the same as the contents of the match field from a secondary file record, the matching-record (MR) indicator turns on. The matching-record indicator can then be used to condition those operations that are to be done only when records match.

Note: All match fields that have no field-record-relation indicator should be described before those that do.

For more information on processing primary and secondary files, see Chapter 11.

Figure 12-15 shows the general steps in the RPG program cycle for programs that use more than one input file.

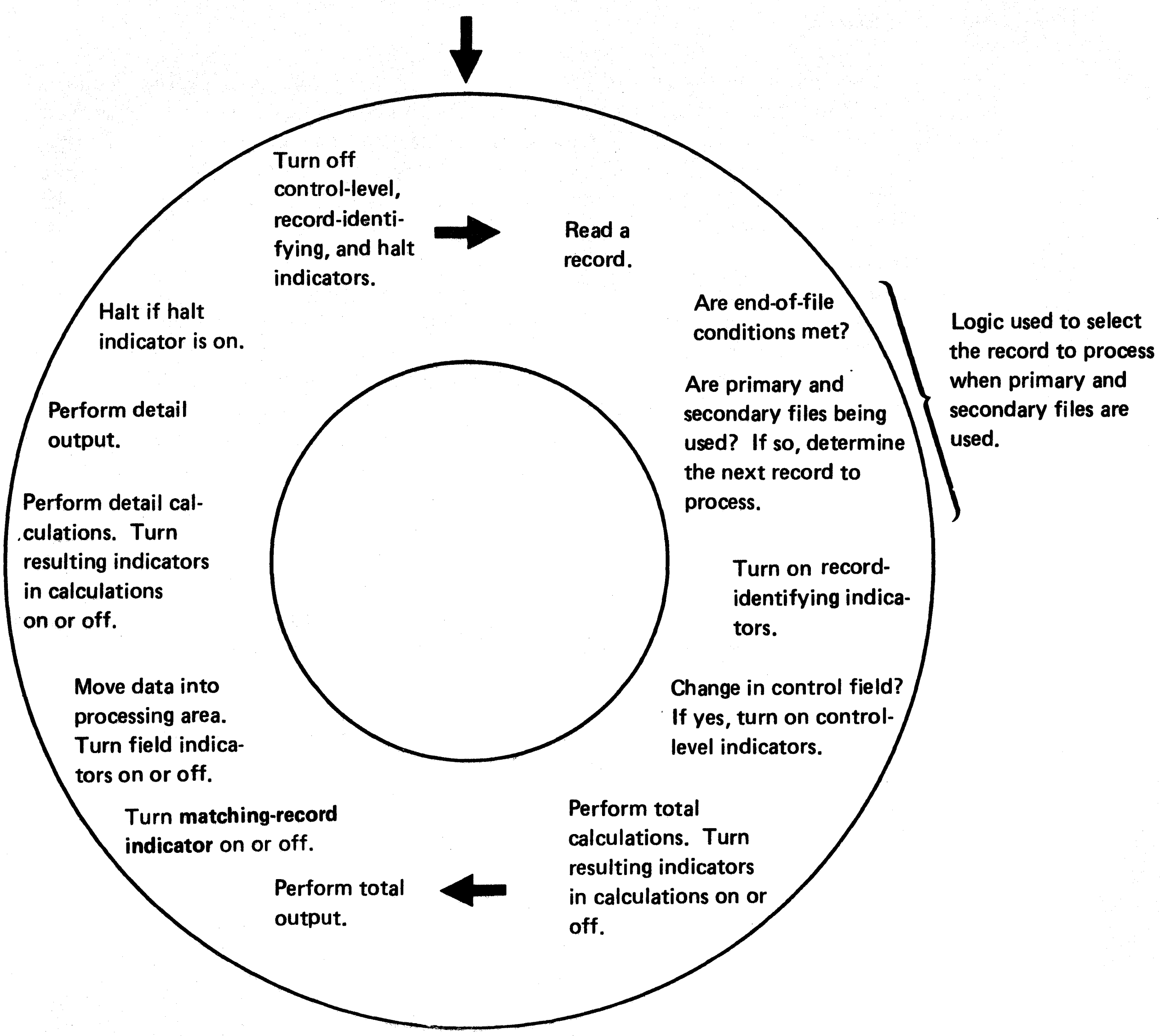


Figure 12-15. RPG Program Cycle for Matching Records

The only 2-character entry allowed as a matching-record indicator is:

MR

On the **input specifications**:

- Columns 61 and 62 must contain the matching-record indicator to tell the program to accept and use data from a particular field when fields or records from different files match.

On the **calculation specifications**:

- Columns 9 through 17 must contain the matching-record indicator to condition an operation that is to be done only when matching records are found.

On the **output specifications**:

- Columns 23 through 31 must contain the matching-record indicator to tell the program to write a line or field when matching records are found.

CONDITIONING INDICATORS

The following indicators are not set on or off when used as conditioning indicators. You can change the status (on or off) only by defining the indicator to represent a certain condition.

Conditioning indicators cannot be used on the END operation of the CAS group or the IF group.

File-Conditioning Indicators

The purpose of the file-conditioning indicator is to condition a file so that the program uses that file only when the file-conditioning indicator is on. When the file-conditioning indicator is off at the beginning of the program, the file is treated as though the end of the file is reached; in other words, no records can be read from or written to the file.

The 2-character entries allowed as file-conditioning indicators are:

U1 through U8

On the **file description specifications**:

- Columns 71 and 72 can contain a file-conditioning indicator to determine whether a file is to be used for a job.

Field-Record-Relation Indicators

Field-record-relation indicators are used to associate fields with a particular record type when that record type is one of several in an OR relationship. The field described on the specification line is available for input only if the indicator specified in the field record relation entry is on or if the entry is blank. If the entry is blank, the field is common to all record types defined by the OR relationship.

An indicator that was previously defined in the program can also be used as a field-record-relation indicator. Control fields (specified by a control-level indicator in columns 59 and 60 on the input specifications) and match fields (specified by a match value in columns 61 and 62 on the input specifications) can also be related to a particular record type in an OR relationship by a field-record-relation indicator. Control fields or match fields in the OR relationship that do not have a field-record-relation indicator are used with all record types in the OR relationship.

When two control fields have the same control-level indicator or two match fields have the same match value, a field-record-relation indicator can be assigned to just one of the control fields or match fields. In this case, the field with the field-record-relation indicator is used only when that indicator is on. If none of the field-record-relation indicators are on for that control field or match field, the field without a field-record-relation indicator is used. Control fields and match fields can only use the 2-character entries 01 through 99 or H1 through H9 in columns 63 and 64.

The 2-character entries allowed as field-record-relation indicators are:

01 through 99

H1 through H9

MR

L1 through L9

U1 through U8

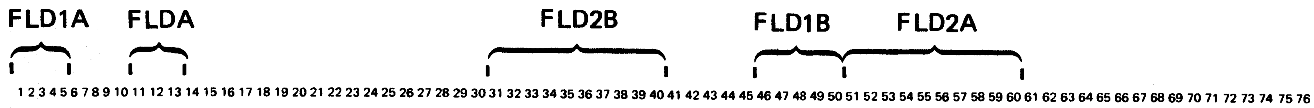
On the **input specifications**:

- Columns 63 and 64 can contain a field-record-relation indicator to associate fields with a particular record type when that record type is one of several in an OR relationship.

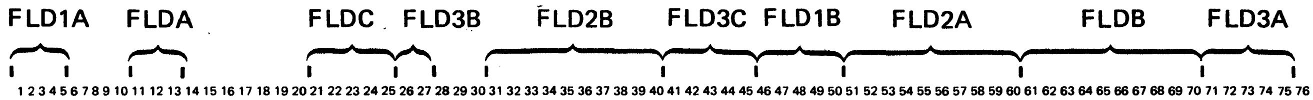
Assigning Field-Record-Relation Indicators

When assigning field-record-relation indicators in the input specifications, consider the following:

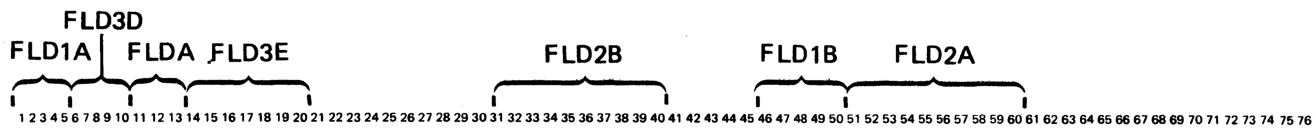
- All fields, including match or control fields, that have no field-record-relation indicator should be described before those that do.
- All fields having the same field-record-relation indicator should be defined on consecutive specification lines for more efficient use of storage. These fields can, however, be entered in any order.
- All portions of a split control field must be assigned the same field-record-relation indicator and must be defined on consecutive specification lines (see Figure 12-16). For more information on split control fields, see *Split Control Fields* earlier in this chapter.
- When the field-record-relation indicator is used with control or match fields, the field-record-relation indicator must match a record-identifying indicator for this file, and the match or control fields must be grouped according to the field-record-relation indicator. The field-record-relation indicator for control or match fields can only be 01 through 99 or H1 through H9.
- When any match value (M1 through M9) is specified for a field without a field-record-relation indicator, all match values used must be specified once without a field-record-relation indicator. If all match fields are not common to all records, a dummy match field should be used (see Figure 12-17).



Record Identification Code = 1



Record Identification Code = 2



Record Identification Code = 3

The record identified by a 1 in position 95 has two split control fields:

FLD1A and FLD1B
FLD2A and FLD2B

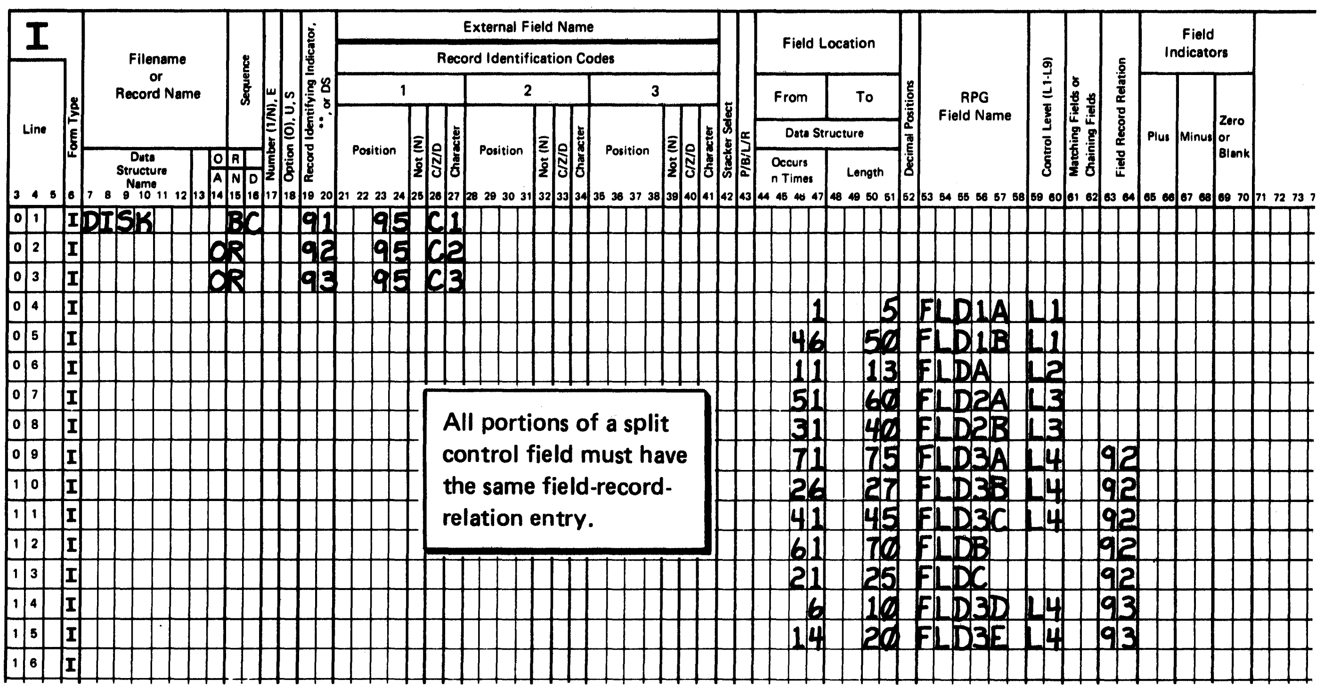
The record with a 2 in position 95 has three split control fields:

FLD1A and FLD1B
FLD2A and FLD2B
FLD3A, FLD3B, and FLD3C

The third record type, identified by a 3 in position 95, also has three split control fields:

FLD1A and FLD1B
FLD2A and FLD2B
FLD3D and FLD3E

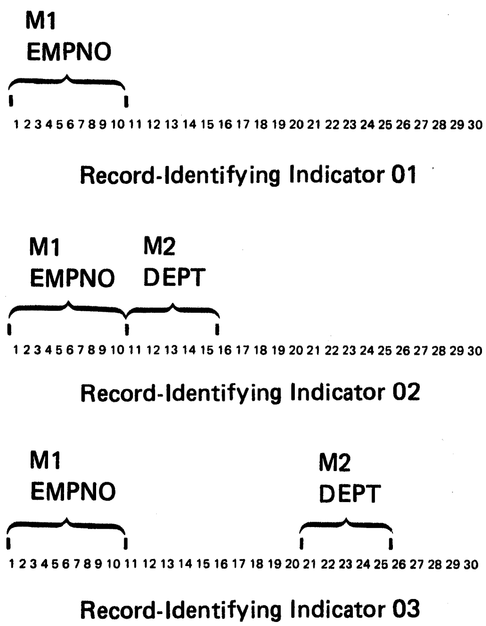
All portions of the split control field must be assigned tl same control-level indicator, and all must have the same field-record-relation entry.



Line	Form Type	Filename or Record Name	Sequence / AND OR	Record Identifying Indicator	Position	C/Z/D	Character	Field Location From	Field Location To	RPG Field Name	Control Level (L1-L9)	Field Record Relation
01	I	DISK	BC	91	95	C	1					
02	I		OR	92	95	C	2					
03	I		OR	93	95	C	3					
04	I							1	5	FLD1A	L1	
05	I							46	50	FLD1B	L1	
06	I							11	13	FLDA	L2	
07	I							51	60	FLD2A	L3	
08	I							31	40	FLD2B	L3	
09	I							71	75	FLD3A	L4	92
10	I							26	27	FLD3B	L4	92
11	I							41	45	FLD3C	L4	92
12	I							61	70	FLDB		92
13	I							21	25	FLDC		92
14	I							6	10	FLD3D	L4	93
15	I							14	20	FLD3E	L4	93
16	I											

Figure 12-16. Field-Record-Relation Indicator (Split Control Field)

Three different record types are found in the input file. All three contain a match field in positions 1 through 10. Two of them have a second match field. Because M1 is found on all record types, it can be specified without an entry in columns 63 and 64. If one match value (M1 through M9) is specified without field-record-relation entries, all match values must be specified once without field-record-relation entries. Because the value M1 is specified without field-record relationship, an M2 value must also be specified once without field-record relationship. The M2 field is not on all record types; thus a dummy M2 field must be specified next. The dummy field can be given any unique name, but its specified length must be equal to the length of the true M2 field. The M2 field is then related to the record types on which it is found by field-record-relation entries (lines 06 and 07).



Line	Form Type	Filename or Record Name	And/Or	Sequence	Record Identifying Indicator	Position (1)	Not (N)	C/Z/D	Character	Field Location From	Field Location To	Decimal Positions	RPG Field Name	Matching Fields or Chaining Fields	Field Record Relation
01	I	DISK		AB	01	1		C	1						
02	I		OR		02	1		C	2						
03	I		OR		03	1		C	3						
04	I									1	10	0	EMPNO	M1	
05	I									11	15	0	DUMMY	M2	
06	I									11	15	0	DEPT	M2	02
07	I									21	25	0	DEPT	M2	03
08	I														

Figure 12-17. Dummy Match Fields

Level-Zero Indicator

The level-zero indicator is always on and cannot be set off with the SETOF operation code. You need never assign this indicator, but you can use it to condition operations, especially when no control fields have been assigned. When a control break occurs, all operations conditioned by control-level indicators (including the level-zero indicator) are done before those that are not conditioned. If no control field is assigned, but total calculations are to be done and total output records are to be written, use the level-zero indicator (L0) to condition those operations (see Figure 12-18).

The only 2-character entry allowed as a level-zero indicator is:

L0

On the **calculation specifications**:

- Columns 7 and 8 can contain the level-zero indicator to specify that the total calculation be done every time.

On the **output specifications**:

- Columns 23 through 31 can contain the level-zero indicator to define total output to be done every cycle.

The program (Figure 12-18) shows how total operations can be performed even though there is no control field (no L1 through L9 indicators).

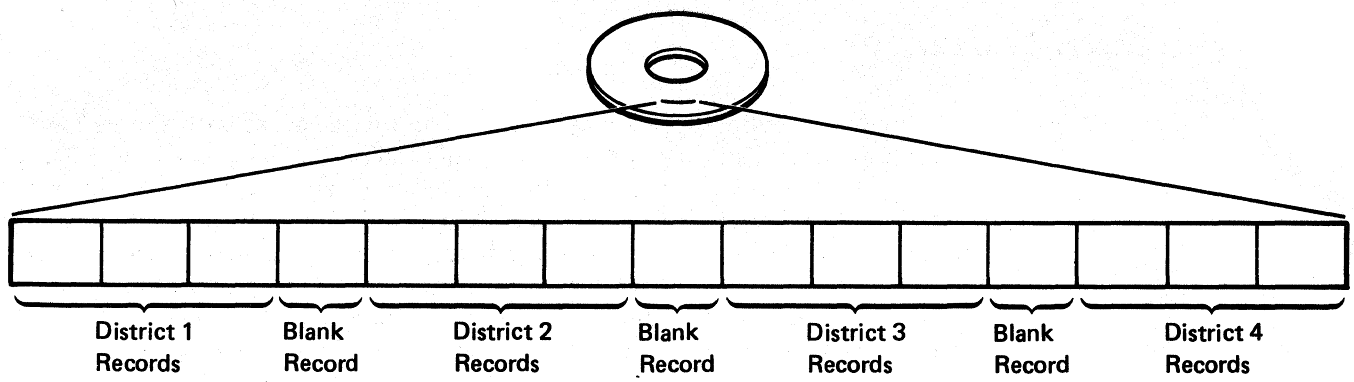
The program requires:

- A list of items sold in each district
- A total of all sales for each district
- A grand total of all sales in all districts

The input records have ITEM and COST fields and a one-position record identification field. The records are grouped in ascending sequence by district. That is, the district 1 records as a group are followed by a blank record, the district 2 records as a group are followed by a blank record, and the district 3 records as a group are followed by a blank record.

No field can serve as a control field because the district number is not on the records. Instead of a control field, the blank record is used to signal a new district. When the blank record is read, indicator 02 turns on. The blank record tells the program that total calculations and total output operations must be done. However, no total operations can be performed unless they are conditioned by some kind of control-level indicator.

Even though L0 is on all the time, it must be used in columns 7 and 8 because some type of control-level indicator must be assigned to all total operations.



Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator	Position (1)	C/Z/D (1)	Character (1)	Field Location From	Field Location To	Decimal Positions	RPG Field Name
01	I	INPUT	AA	Ø1	1	C	1				
02	I							1Ø	15	2	COST
03	I							2Ø	3Ø		ITEM
04	I		BB	Ø2	1	C					
05	I										

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions
01	C		Ø1	COST	ADD	DISTOT	DISTOT	6	2
02	C								
03	C								
04	C								
05	C	LØ	Ø2	DISTOT	ADD	GDTOT	GDTOT	6	2
06	C	LR		DISTOT	ADD	GDTOT	GDTOT		
07	C								

Figure 12-18 (Part 1 of 2). Use of the Level-Zero (L0) Indicator

Line	Form Type	Filename or Record Name	AND/OR	Type (H/D/T/E)	Space Before	Space After	Skip Before	Skip After	Output Indicators	Field Name or EXCPT Name	Edit Codes	Blank After (B)	End Position in Output Record	Constant or Edit Word
01	O	OUT		D		1			Ø1					
02	O									ITEM			3Ø	
03	O									COST	1		5Ø	
04	O			T	2	2			Ø2					
05	O		OR						LR					
06	O									DISTOT	1	B	5Ø	
07	O												51	'*'
08	O			T	2				LR					
09	O									GDTOT	1		5Ø	
10	O												52	'**'
11	O													

Format of the Printed Report

```
J102      4.50
J202      3.75
K450      2.98
B231      9.08

         20.31 *

G10H     92.79
G10K     98.89
A126      4.29

        195.97 *


        216.28 **
```

Figure 12-18 (Part 2 of 2). Use of the Level-Zero (L0) Indicator

Command-Key Indicators

There are 24 command keys. Each one corresponds to a separate command-key indicator:

Command Key	Command-Key Indicator	Keyboard Keys To Press	Command Key	Command-Key Indicator	Keyboard Keys To Press
1	KA	Cmd, 1	13	KM	Cmd, Shift, |
2	KB	Cmd, 2	14	KN	Cmd, Shift, @
3	KC	Cmd, 3	15	KP	Cmd, Shift, #
4	KD	Cmd, 4	16	KQ	Cmd, Shift, $
5	KE	Cmd, 5	17	KR	Cmd, Shift, %
6	KF	Cmd, 6	18	KS	Cmd, Shift, ¬
7	KG	Cmd, 7	19	KT	Cmd, Shift, &
8	KH	Cmd, 8	20	KU	Cmd, Shift, *
9	KI	Cmd, 9	21	KV	Cmd, Shift, (
10	KJ	Cmd, 0	22	KW	Cmd, Shift,)
11	KK	Cmd, -	23	KX	Cmd, Shift, _
12	KL	Cmd, =	24	KY	Cmd, Shift, +

Note: The keyboard keys may vary, depending on what type of keyboard you have.

The purpose of command-key indicators is:

- To condition calculation and output operations for a program with a WORKSTN or KEYBORD file. All 24 command-key indicators are defined for a WORKSTN file.

- To specify what command keys the person using a display station can press for a SET operation with a KEYBORD file. To tell the person how you used the command keys in your program, you can fill out the template assignment form on the *IBM 5251 Display Station Keyboard Template Assignment Sheet and Display Screen Layout Sheet.*

On the **calculation specifications**:

- Columns 9 through 17 can contain command-key indicators that are used with a KEYBORD file or a WORKSTN file to condition calculation operations.

- Columns 54 through 59 can contain a command-key indicator for a SET or SETOF operation.

  When a SET operation occurs, only the command keys in columns 54 through 59 for that SET operation can be pressed at that time. From one to three command keys can be entered for each SET operation. If one or two command keys are specified, they can appear in any of the three sets of columns.

On the **output specifications**:

- Columns 23 through 31 can contain command-key indicators that are used with a KEYBORD file or with a WORKSTN file to condition output operations. If a KEYBORD file is used, any command keys entered in these columns must also be coded in columns 54 through 59 of the calculation specifications for a SET operation.

When the program allows the person using the display station to enter data from the keyboard, all command-key indicators are turned off. If the person presses a command key, the corresponding command-key indicator turns on.

Halt Indicators

The purpose of halt indicators is to stop the program when an unacceptable condition exists. You can use halt indicators as record-identifying, field, or resulting indicators. When you use a halt indicator as a record-identifying indicator, a halt is caused by a specific type of record. When you use a halt indicator as a field indicator, a halt is caused by incorrect input data. When you use a halt indicator as a resulting indicator, a halt is caused by incorrect results from calculations.

A halt indicator can turn on at one of four times in the program cycle, depending on how you use it (see Figure 12-19). The program does not halt immediately when a halt indicator turns on. All total and detail operations remaining in the cycle are performed first; then the program halts. This means that the program completes processing the information from the record that caused the halt. Therefore, you must write specifications that bypass calculation and output operations when an error occurs.

When the halt is issued, you have the following options to choose from:

- *0-Continue*: Control is returned to the program, and processing continues.
- *2-Controlled Cancel*: End-of-job operations specified by the program are done, tables are dumped, and file labels are cataloged.
- *3-Immediate Cancel*: The job is canceled. Any data created or work done by previous programs in this job is saved. Any records added or updates made to existing files by the current program are saved. Records deleted by the current program no longer exist.

If you wish to display your own message text when a halt indicator is on, you must create a message member with message identification code (MIC) numbers of 0101 through 0109. MIC 0100 is issued when all Halt indicators are on at once. MIC 0101 corresponds to H1, 0102 to H2, and so on. You must also have a control language MEMBER statement in the procedure that runs the program. See *Using a Message Member* in Chapter 9.

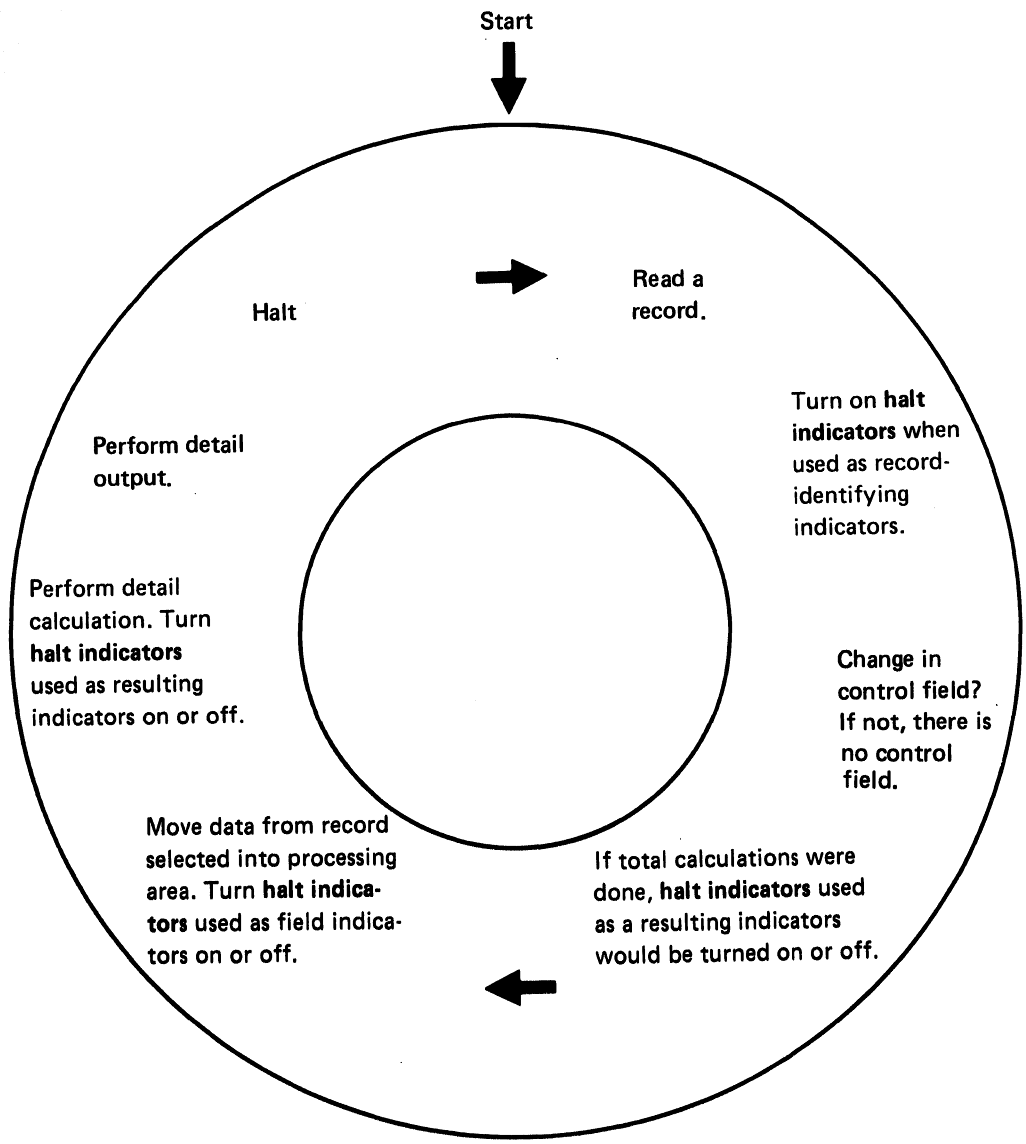


Figure 12-19. RPG Program Cycle for Halt Indicators

The 2-character entries allowed as halt indicators are:

H1 through H9

On the **input specifications**:

- Columns 19 and 20 can contain a halt indicator to be used as a record-identifying indicator.
- Columns 63 and 64 can contain a halt indicator to be used as a field-record-relation indicator.
- Columns 65 through 70 can contain a halt indicator to prevent a calculation or output operation from being done if the program finds a specified error condition in the input data.

On the **calculation specifications**:

- Columns 9 through 17 can contain a halt indicator to condition calculation operations.
- Columns 54 through 59 can contain a halt indicator which is set on or set off as the result of an operation.

On the **output specifications**:

- Columns 23 through 31 can contain a halt indicator to prevent or condition output operations if an error condition occurs.

Indicators Conditioning Calculations

Indicators that you use to specify the conditions under which a calculation is done must be defined elsewhere in the program.

You can use the operation codes SETON or SETOF to turn indicators on or off. See *SETON* or *SETOF* in Chapter 28, *Operation Codes*, for more information on these operations. Any indicators you want turned on or off by the SETON or SETOF operation codes can be specified in any of the three resulting indicator fields (columns 54 through 59 on the calculation specifications). However, you cannot turn on command-key indicators with the SETON operation or turn off the last-record indicator with the SETOF operation. The headings for columns 54 through 59 (high, low, and equal) have no meaning for SETON or SETOF operations.

The 2-character entries in columns 9 through 17 of the calculation specifications must be previously defined as one of the following types of indicators:

- Overflow indicators
- Record-identifying indicators
- Control-level indicators
- Field indicators
- Resulting indicators
- External indicators
- Internal indicators

On the **calculation specifications**:

- Columns 9 through 17 can contain conditioning indicators that control the conditions under which an operation is done.

From one to three indicators (specified in columns 10 and 11, 13 and 14, and 16 and 17) can be used on each line. If the indicator must be off to condition the operation, place an N before the indicator (in column 9, 12, or 15).

Using Indicators in AN/OR Lines on the Calculation Specifications

Use columns 7 and 8 of the calculation specifications to specify that lines of indicators are in an AN/OR relationship. When you use the AN/OR relationship, many lines of indicators can be grouped together to condition an operation. A maximum of seven AN lines, seven OR lines, or seven of any combination of AN and OR lines can condition an operation.

The first line of such a group contains blanks in columns 7 and 8 of the calculation specifications or an L0 through L9, LR, or SR entry if the group of lines is conditioned by a control-level indicator or is part of a subroutine. This entry on the first line applies to all AN/OR lines that follow. All lines after the first line in the group must have an AN or OR entry in columns 7 and 8 of the calculation specifications. The last line of the group contains the operation and the necessary operands. All lines except the last line in the group must contain blanks in columns 18 through 59 of the calculation specifications (see Figure 12-20).

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 Plus / High / 1>2	Minus / Low / 1<2	Zero / Equal / 1=2	Comments
01	C		Ø1	Ø2	Ø3											
02	C	AN	Ø4													
03	C	OR	Ø1	Ø2	Ø3											
04	C	AN	Ø5			FIELDA	SUB	FIELDB	QTY	4	Ø			23		
05	C															

AN and OR entries group lines of indicators. When indicators 01, 02, 03, and 04 are on, or when indicators 01, 02, 03, and 05 are on, the calculation is performed.

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 Plus / High / 1>2	Minus / Low / 1<2	Zero / Equal / 1=2	Comments
01	C	L4	Ø1	Ø2	NØ3											
02	C	OR	Ø1	NØ2	Ø3											
03	C	OR	NØ1	Ø2	Ø3	SUM	ADD	SUMTOT	SUMTOT	8	2	H				
04	C															

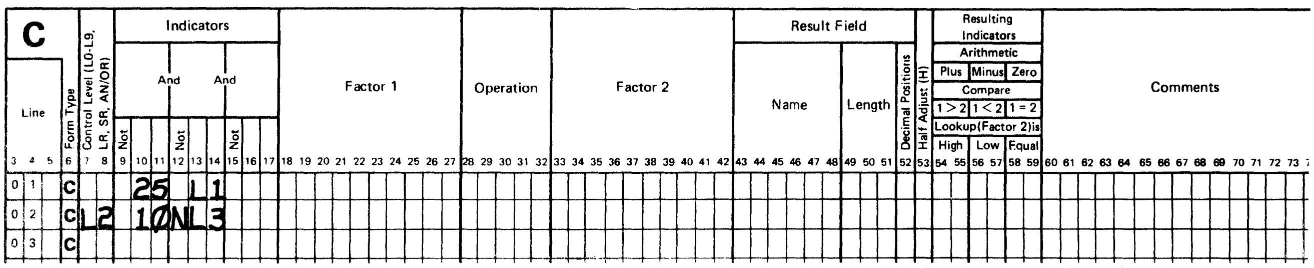
Three conditions cause the L4 total calculations to be performed: 01 and 02 are on, but not 03; or 01 and 03 are on, but not 02; or 02 and 03 are on, but not 01.

Figure 12-20. Use of AN/OR Lines for Indicators

Using Indicators in an AND Relationship on the Calculation Specifications

Indicators specified in columns 9 through 17 of the calculation specifications are in an AND relationship with each other if the indicators are on the same line. The indicators on one line or indicators in grouped lines plus the control-level indicator (if used in columns 7 and 8 of the calculation specifications) must all be exactly as specified before the operation is done (see Figure 12-21).

An indicator that is specified in columns 9 through 17 can also be entered as a resulting indicator on the same line. If the indicator in columns 9 through 17 is on, the calculation is done.



Assume that indicator 25 represents a record type and that a control-level-2 break occurred when record type 25 was read. L1 and L2 are both on. All operations conditioned by the control-level indicators in columns 7 and 8 are performed before operations conditioned by control-level indicators in columns 9 through 17. Thus, the operation in line 02 occurs before the operation in line 01. The operation in line 01 is done on the first record of the new control group indicated by 25, whereas the operation in line 02 is a total operation done for all records of the previous control group.

The operation in line 02 can be done when the L2 indicator is on provided the other conditions are met. Indicator 10 must be on. The L3 indicator must not be on.

The operation conditioned by both L2 and NL3 is done only when a control-level-2 break occurs. These two indicators are used together because this operation is not to be done when a control-level-3 break occurs, even though L2 is also on.

Figure 12-21. Conditioning Operations (Control-Level Indicators)

Indicators Conditioning Output

Indicators that you use to specify the conditions under which an output record or an output field is written must be previously defined in the program.

The 2-character entries in columns 23 through 31 of the output specifications must be previously defined as one of the following types of indicators:

- Overflow indicator
- First-page indicator
- Record-identifying indicator
- Control-level indicator
- Field indicator
- Resulting indicator
- Internal indicators
- External indicators

On the **output specifications**:

- Columns 23 through 31 can contain a conditioning indicator to specify the conditions under which an output record or an output field is written. When the indicator is to condition an entire output line, enter it on the record line (column 15 contains a D, H, T, or E). When an indicator is to condition when a field is to be written, enter it on the same line as the field name.

Using Indicators in an AND/OR Relationship on the Output Specifications

Use an AND line if more than three indicators are needed to condition an output operation. Enter AND in columns 14 through 16 on the output specifications for each additional line. The condition for all indicators in an AND relationship must be satisfied before the output operation is done. Any number of AND lines can be used for an output operation.

Output indicators can also be in an OR relationship. Enter OR in columns 14 and 15 for each OR relationship. If one or the other condition is met, the output operation is done. A maximum of 20 OR lines can be used for an output operation.

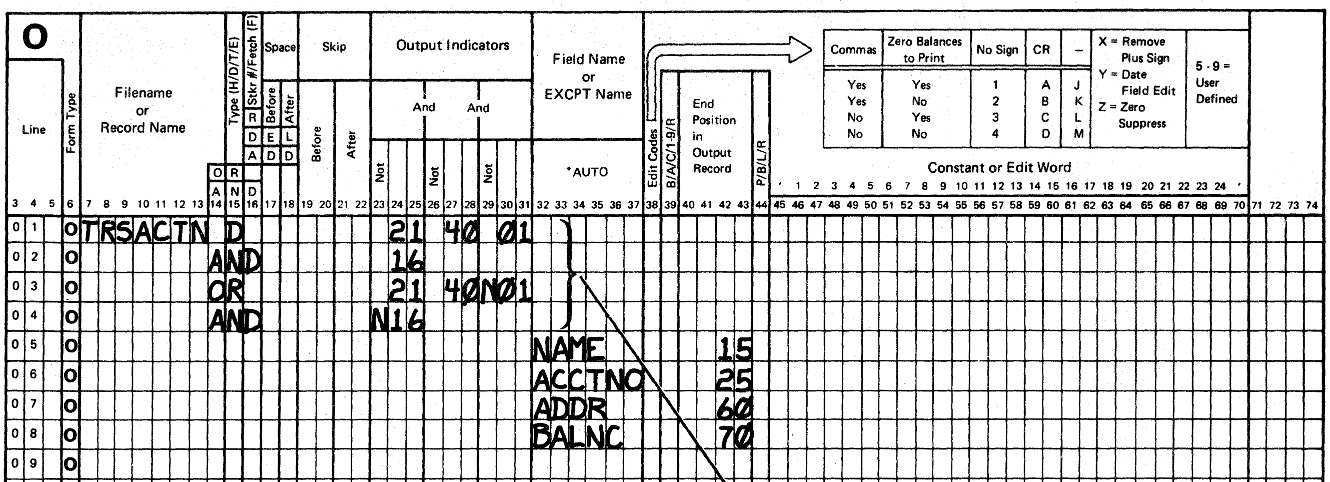
If AND and OR lines are combined, the total number of OR lines for an output operation cannot be more than 20 and any number of AND lines can be used.

A maximum of 255 record line groups can be used in an OR relationship under the following conditions (see Figure 12-23):

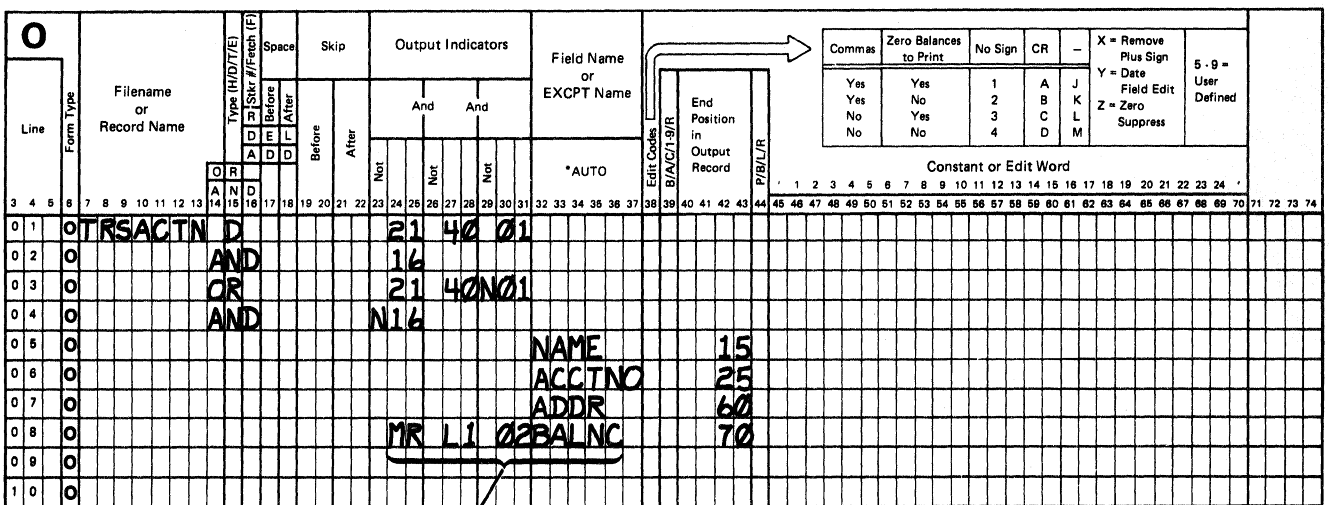
- Column 15 of the record line contains a D,T, or H.
- The first-page indicator or an overflow indicator is used. The first page or overflow indicator may appear on the record line, the OR line, or an AND line.

AND and OR lines can be used to condition entire output lines, but they must not be used to condition fields (see Figure 12-22). However, you can condition an output field with more than three indicators by using the SETON operation in calculations. For example, if indicators 10, 12, 14, 16, and 18 are used to condition an output field named PAY, in the calculations you can set on indicator 20 if indicators 10, 12, and 14 are on. Then condition the output field PAY on indicators 20, 16, and 18 in the output specifications.

The use of any of the L0 through L9 indicators in an OR relationship with an LR indicator can result in the specified operation being done twice when LR is on. One operation is done at total time and the other during LR processing. Figure 12-23 shows how to correctly use the L0 through L9 indicators in an OR relationship.



The detail line is printed if either of two sets of conditions is met. If 21, 40, 01, and 16 are all on, the line is printed; if 21 and 40 are on and 01 and 16 are off, the line is also printed.



A maximum of three indicators can be used to condition a field.

Figure 12-22. Output Indicators in AND and OR Lines

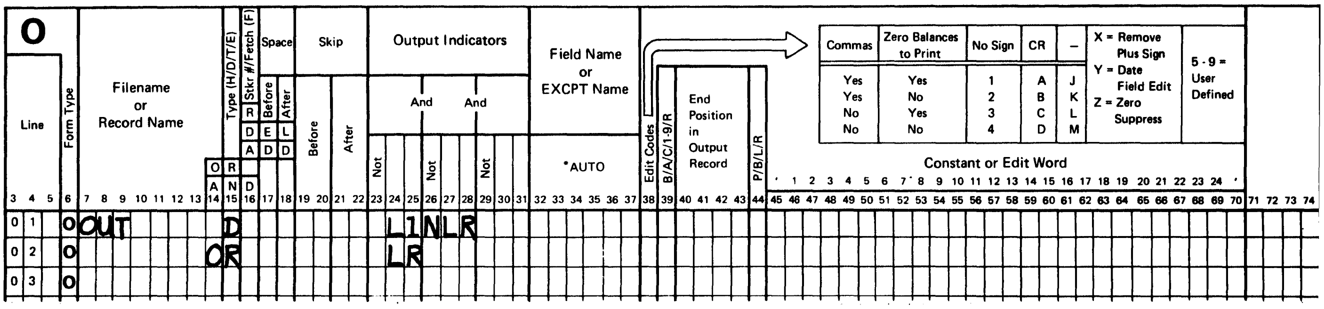


Figure 12-23. Correct Use of Control-Level Indicators (L0-L9) in OR Relationship

Chapter 13. Using Arrays and Tables

Chapter 13. Using Arrays and Tables

Similarities between Arrays and Tables

Both arrays and tables are systematic arrangements of data items, called *elements*. Each element in an array or table has the same field length, the same data type (alphameric or numeric), and the same number of decimal positions if numeric. You can use an array or a table for the same purposes. Both arrays and tables are described on the extension specifications.

Differences between Arrays and Tables

However, arrays and tables differ in two important ways:

- When they can be loaded
- How they can be processed

When Arrays and Tables Can Be Loaded

Loading an array or table means reading it into main storage so that the program can process it.

Arrays can be loaded while the source program is being compiled, before the program is run, or while the program is running.

Tables can be loaded while the source program is being compiled or before the program is run.

How Arrays and Tables Can Be Processed

A program can process an array in either of two ways:

- It can process an entire array at one time. That is, the program does the same operation on every element in the array.

- It can process specific array elements that you refer to by their position relative to other elements. To do this, you must provide an index to specific elements in the array.

The only way that a program can process a table is to perform the operation on only one element. The operation must be specified separately for each additional element you want to process.

Kinds of Arrays and Tables

- *Compile-time arrays and tables* are loaded with the source program and become a permanent part of the load module. The initial content of a compile-time array or table can be changed in the program, or you can recompile the source program with new array or table data (see *Changing the Contents of Arrays and Tables* in this chapter).

- *Preexecution-time arrays and tables* are loaded with the load module before you run the program; that is, before any input records are read, calculations are performed, or output records are written.

- *Execution-time arrays* are loaded or created by input or calculation specifications. The arrays are loaded while the program is running; that is, they are read in as input data or created during calculations in the program. An execution-time array is also described on the extension specifications. Tables cannot be specified for execution-time load.

- *Related arrays and tables* are two arrays or two tables that are read and stored together and that are defined on the same extension specification (see example on line 01 of Figure 13-1 Part 2). Each element in the second array or table gives additional information about its corresponding element in the first array or table. An array can be related to another array, or a table can be related to another table; however, an array cannot be related to a table, or vice versa.

  For example, Figure 13-1 shows related arrays ARR1 and ARR2. An element in array ARR1 provides a part number, and the corresponding element in array ARR2 provides the cost for that part. Although all elements within one table or array must have the same characteristics, corresponding elements of related arrays or tables can have different characteristics. Thus, in Figure 13-1, all elements in array ARR1 are alphameric, and all elements in array ARR2 are numeric.

  Related arrays and tables should have the same number of elements. If the arrays or tables do not have the same number of elements, the program might find the desired element in one array or table but find no corresponding element in the related array or table. In this case, undesirable results can occur.

ARR1 (Part Number)	ARR2 (Unit Cost)
345126	00373
38A473	00498
39K143	01297
40B125	00093
41C023	03998
42D893	00087
43K832	00349
44H111	00679
45P673	00898
46C732	47587

Arrays ARR1 and ARR2 can be described as two separate arrays or as two related arrays.

The following extension specifications show how to describe ARR1 and ARR2 as two separate arrays. The entries in columns 33 through 35 are required for records that are in the source program.

E Line (3-5)	Form Type (6)	Record Sequence of the Chaining File (7-8)	Number of the Chaining Field (9-10)	From Filename (11-18)	To Filename (19-26)	Table or Array Name (27-32)	Number of Entries Per Record (33-35)	Number of Entries Per Table or Array (36-39)	Length of Entry (40-42)	P/B/L/R (43)	Decimal Positions (44)	Sequence (A/D) (45)	Table or Array Name (Alternating Format) (46-51)	Length of Entry (52-54)	P/B/L/R (55)	Decimal Positions (56)	Sequence (A/D) (57)	Comments (58-74)
01	E					ARR1	10	10	6									
02	E					ARR2	10	10	5		0							
03	E																	

When ARR1 and ARR2 are described as two separate arrays, the first record contains elements for ARR1 in positions 1 through 60, and the second record contains elements for ARR2 in positions 1 through 50:

Record 1:

345126	38A473	39K143	40B125	41C023	42D893	43K832	44H111	45P673	46C732

Record 2:

00373	00498	01297	00093	03998	00087	00349	00679	00898	47587

Figure 13-1 (Part 1 of 2). Separate Arrays and Related Arrays

The following extension specifications show how to describe ARR1 and ARR2 as related arrays. The entry in columns 33 through 35 is required for records that are in the source program.

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
3 4 5	6	7 8	9 10	11 12 13 14 15 16 17 18	19 20 21 22 23 24 25 26	27 28 29 30 31 32	33 34 35	36 37 38 39	40 41 42	43	44	45	46 47 48 49 50 51	52 53 54	55	56	57	58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
0 1	E					ARR1	5	1Ø	6				ARR2	5		Ø		
0 2	E																	

When ARR1 and ARR2 are described as two related arrays, the records contain pairs of corresponding elements. Thus, the first record begins with element 1 of ARR1 and the corresponding element of ARR2, followed by element 2 of ARR1 and the corresponding element of ARR2, and so on until the record is full (in this example, the record has 55 positions, or five pairs of corresponding elements). The second record, which also has 55 positions, contains corresponding pairs of elements 6 through 10:

Record 1:

345126	00373	38A473	00498	39K143	01297	40B125	00093	41C023	03998

Record 2:

42D893	00087	43K832	00349	44H111	00679	45P673	00898	46C732	47587

Figure 13-1 (Part 2 of 2). Separate Arrays and Related Arrays

Creating Input Records for Arrays or Tables

Input records for arrays or tables must be formatted according to certain rules:

- The first array or table element for each input record must begin in position 1.
- An entire record need not be filled with array or table elements. If it is not, blanks or comments can be included after the elements (see Figure 13-2). The unused space in numeric arrays or tables is filled with zeros; the unused space in alphameric arrays or tables is filled with blanks.
- Each input record, except the last, must contain the same number of elements. In the last record, unused space must be blank. You can include comments after the blank space. That is, comments in the last record must begin in the same position as comments in preceding records (see Figure 13-2).
- Each element must be contained entirely on one input record; an element cannot be split between two records. Thus, the length of a single element is limited to the maximum record length for the input device. If you use related arrays or tables and describe them in alternating format, corresponding elements must be on the same input record; the combined length of corresponding elements cannot exceed the maximum record length for the device.
- Related arrays or tables can be described separately or in alternating format. Alternating format means that elements of one array or table alternate with elements of the related array or table on the input record (see Figure 13-1).
- The total number of array names, table names, and data structures used in a program cannot exceed 75. The total number of compile-time arrays and tables cannot exceed 70.

```
1 2 3 4 5 1 2 3 4 5 Comments can be
1 2 3 4 5 1 2 3 4 5              anywhere out here
1 2 3 4 5 1 2 3 4 5                             or here
1 2 3 4 5           or here (that is, after the last entry position for the longest record).

1 2 3 4 5 1 2 3 4 5
1 2 3 4 5 1 2 3 4 5
1 2 3 4 5 1 2 3 4 5
1 2 3 4 5  If comments begin here, the compiler cannot tell if you intend them as comments or if you
           provided too much data for the table/array.  Therefore, it prints a warning message.
```

Each of the two tables/arrays contains seven entries, each entry 5 positions long, with two entries per record. The last record contains only one entry. The remaining 5 positions in the last record should be left blank, because using these positions for comments causes warning message RPG-0333, TABLE/ARRAY IS FULL, OR NO TABLES/ARRAYS EXIST FOR FOLLOWING DATA, to be printed during compilation. Therefore, comments should begin after the last entry position for the longest record; that is, (the number of entries per record x the number of positions per entry) + 1.

Figure 13-2. Array or Table Input Record with Comments

Defining Arrays and Tables

All arrays and tables must be defined on extension specifications. Each extension specification defines one set of array or table input records.

If only one array or table is being defined, use columns 11 through 45.

If alternating arrays or tables are being defined, use columns 11 through 57. For compile-time and preexecution-time arrays or tables, the array or table named in columns 46 through 51 is entered in alternating format with the array or table named in columns 27 through 32.

If compile-time arrays or tables are being defined, columns 11 through 26 must be blank.

If preexecution-time arrays or tables are being defined, entries are required in columns 11 through 18 and in columns 27 through 45.

If execution-time arrays are being defined, columns 11 through 26 must be blank.

If the array or table being defined is to be written to a file at the end of the job, enter the name of the file in columns 19 through 26. Execution-time arrays cannot be written to a file at end of job.

If you are defining a table, the name you assign in columns 27 through 32 must begin with TAB. If you are defining an array, it must not begin with TAB.

Arrays and tables can be defined in any sequence on the extension specifications. The sequence in which they are defined determines the order in which they are loaded at the start of the program.

Figure 13-3 shows the extension specifications required for the three types of arrays.

- Line 01 specifies two compile-time arrays, AR1 and AR2, in alternating format. Each array has three elements per record and a total of eight elements in the array. Each element is 12 positions long, including four decimal positions, although the length of elements and the number of decimal positions in AR2 do not have to be the same as those in AR1.

- Line 06 specifies a preexecution-time array, AR3, to be read from file DISKIN. AR3 has 12 elements per record and a total of 250 elements. Each element is 5 positions long. Decimal positions are not specified, so the elements are alphameric data. The elements are arranged in ascending sequence.

- Line 09 specifies an execution-time array, AR4, which contains 10 elements. Each element is 10 positions long. Zero decimal positions are specified, so the elements are numeric data.

For all arrays and tables except execution-time arrays, columns 19 through 26 can also contain the name of a file to which the array or table is to be written at the end of the program, and columns 46 through 57 can also define an array or table that is entered in alternating format with the array or table named in columns 27 through 32.

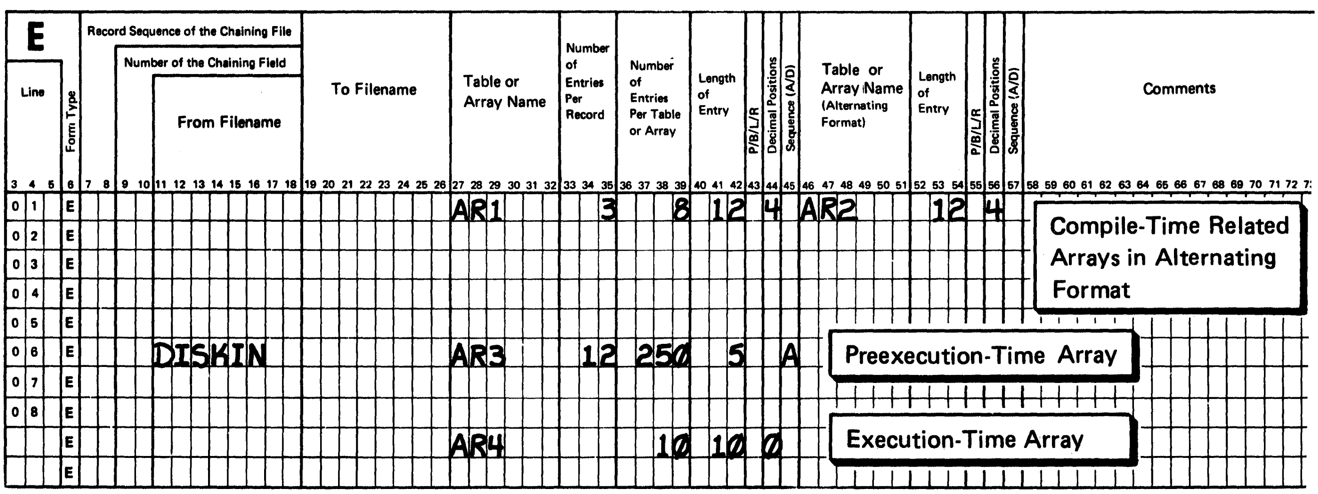


Figure 13-3. Extension Specifications for Three Types of Arrays

Loading Arrays and Tables

Arrays can be loaded at compilation time, preexecution time, or execution time. Tables can be loaded at compilation time or preexecution time.

Loading Compile-Time Arrays and Tables

A compile-time array or table is loaded at compilation time with the data supplied at the end of the source program. Rules for loading arrays and tables at compilation time are as follows:

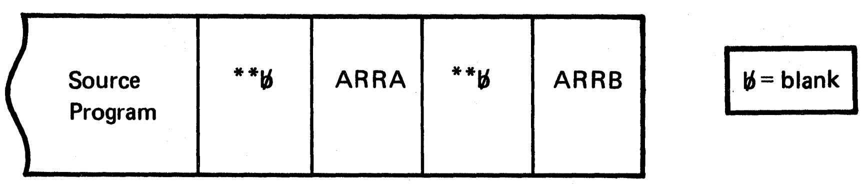
- A compile-time array or table must have entries in columns 33 through 35 of the extension specifications and must not have entries in columns 11 through 18 of the extension specifications.

- Compile-time array or table data must be entered at the end of the source program, after all records for translating files and for changing the collating sequence of characters (see Chapter 17).

- A record with **ƀ in positions 1 through 3 must precede the data for each compile-time array or table (see Figure 13-4).

- Compile-time arrays or tables must be in zoned decimal or alphameric format.

- For compile-time arrays and tables, the maximum length of an alphameric element is 96 because the maximum length of a record in the source program is 96 characters. If a compile-time array or table is not large enough to hold all the data, warning message RPG-0333, TALBE/ARRAY IS FULL, OR NO TABLE/ARRAYS EXIST FOR FOLLOWING DATA, is issued. The extra data is ignored.



ARRA and ARRB are located in the source file with the source program.

Figure 13-4. Arrangement on Disk of the Source Program and Compile-Time Array Data

Loading Preexecution-Time Arrays and Tables

A preexecution-time array or table is loaded by the load program from an input file on disk just before the program runs. The file must be described on the file description specifications as an input table file (IT in columns 15 and 16). A control language FILE statement must also be present for the input file. If two or more arrays or tables are to be loaded, they must be loaded from different disk files, except when the arrays or tables are specified in alternating format. The filename must be specified in columns 11 through 18 of the extension specification that defines the preexecution-time array or table.

Most of the rules that apply to compile-time arrays and tables also apply to preexecution-time arrays and tables except for the following:

- A preexecution-time array or table must have an entry in columns 11 through 18 and in columns 33 through 35 of the extension specifications.

- The file description specification for the file containing the data for a preexecution-time array or table must have an I in column 15, a T in column 16, and an E in column 39.

- For numeric arrays and tables loaded at preexecution time, the data in a DISK file can be in zoned-decimal, packed-decimal, or binary format.

- Preexecution-time array or table records must be in a sequential file, and each record must have the same length.

- If a preexecution-time array or table is not large enough to hold all the data, error message RJPG-9017, OBJECT TABLE DATA EXCEEDS TABLE LENGTH, is displayed. In response to that message, you can ignore the extra data, end the job step, or cancel the job.

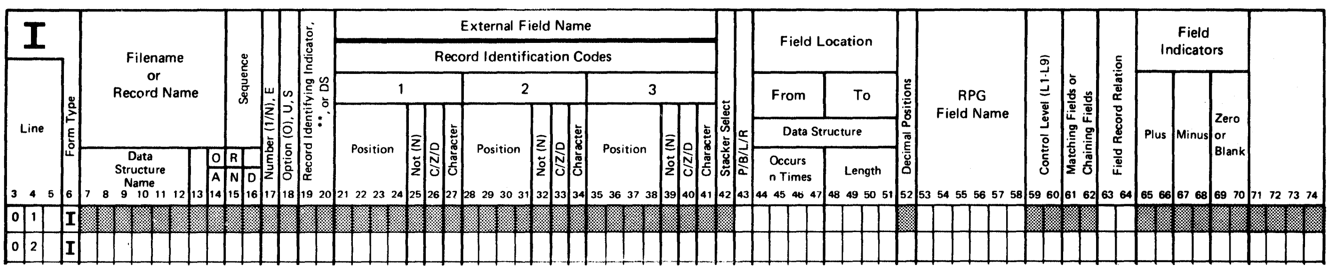
Loading Execution-Time Arrays

To load an array from information in input records, describe that information in the input specifications. The specifications made depend on whether the array information is contained in one record or in more than one record. The input specifications can describe any type of array (compile-time, preexecution-time, or execution-time). When you use input specifications to fill an array with data, the program must read complete data elements.

You can also use an arithmetic or move operation in the calculation specifications to load an execution-time array, either the entire array at one time or one element at a time.

Execution-time arrays are not sequence checked. If you use the SORTA operation, the array is sorted into the sequence (ascending or descending) specified in column 45 of the extension specifications for that array. If no sequence is specified, the array is sorted into ascending sequence. If you use the LOKUP operation with an indicator in columns 54 and 55 or in columns 56 and 57 of the calculation specifications, you must also specify a sequence in column 45 of the extension specifications.

To load an array from a single input record, code entries in the unshaded columns of the input specifications shown below:



Column 43 must contain P (packed decimal), B (binary), or blank (zoned decimal) to indicate the format of the array data.

Columns 44 through 47 must contain the starting position of either the entire array with consecutive elements or an individual element in the array.

Columns 48 through 51 must contain the ending position of either the entire array with consecutive elements or an individual element in the array.

Columns 53 through 58 must contain the name of the array (the same name used on the extension specifications) or the name of an individual array element (array name plus comma and index).

Columns 63 and 64 can contain an indicator to indicate the relation of a field to the record.

Array Information in One Record

If the array information is contained in one record, the information can occupy consecutive positions in the record or it can be scattered throughout the record. If an array is contained in a data structure, all elements in the array are consecutive.

If the array elements are consecutive on the input record, you can load the array with a single input specification. Figure 13-5 shows the extension and input specifications for loading an array, INPARR, that contains six elements (12 characters each) from a single record in the file ARRFILE.

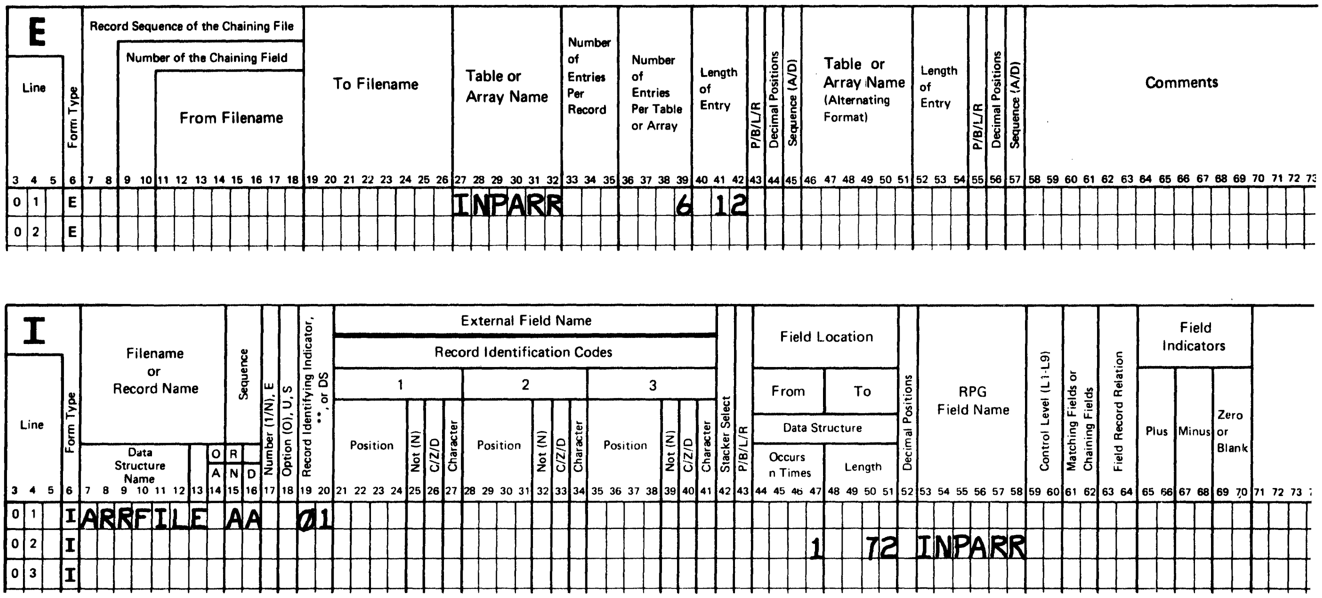


Figure 13-5. Defining an Execution-Time Array with Consecutive Items

If the array elements are scattered throughout the record, they can be defined and loaded one at a time, with each element described on a separate specification line. Figure 13-6 shows the extension and input specifications for loading an array, ARRX, that contains six elements (12 characters each) from a single record in the file ARRFILE. A blank separates each element from the others.

E

Line (3-5)	Form Type (6)	Record Sequence of the Chaining File (7-8)	Number of the Chaining Field (9-10)	From Filename (11-18)	To Filename (19-26)	Table or Array Name (27-32)	Number of Entries Per Record (33-35)	Number of Entries Per Table or Array (36-39)	Length of Entry (40-42)	P/B/L/R (43)	Decimal Positions (44)	Sequence (A/D) (45)	Table or Array Name (Alternating Format) (46-51)	Length of Entry (52-54)	P/B/L/R (55)	Decimal Positions (56)	Sequence (A/D) (57)	Comments (58-74)
01	E					ARRX		6	12									
02	E																	

I

Line (3-5)	Form Type (6)	Filename or Record Name / Data Structure Name (7-14)	Sequence (15-16)	Number (1/N), E (17)	Option (O), U, S (18)	Record Identifying Indicator, **, or DS (19-20)	Record Identification Codes 1: Position (21-24)	Not (N) (25)	C/Z/D (26)	Character (27)	2: Position (28-31)	Not (N) (32)	C/Z/D (33)	Character (34)	3: Position (35-38)	Not (N) (39)	C/Z/D (40)	Character (41)	Stacker Select (42)	P/B/L/R (43)	Field Location From / Data Structure Occurs n Times (44-47)	Field Location To / Data Structure Length (48-51)	Decimal Positions (52)	RPG Field Name (53-58)	Control Level (L1-L9) (59-60)	Matching Fields or Chaining Fields (61-62)	Field Record Relation (63-64)	Field Indicators Plus (65-66)	Minus (67-68)	Zero or Blank (69-70)	(71-74)
01	I	ARRFILE	AA			Ø1																									
02	I																				1	12		ARRX,1							
03	I																				14	25		ARRX,2							
04	I																				27	38		ARRX,3							
05	I																				4Ø	51		ARRX,4							
06	I																				53	64		ARRX,5							
07	I																				66	77		ARRX,6							
08	I																														

Figure 13-6. Defining an Execution-Time Array with Scattered Items

Array Information in More than One Record

If the array information is contained in two or more records, you can use any one of several methods to load the array. Which method to use depends primarily on the size of the array and on whether the array elements are consecutive in the input records.

Figure 13-7 shows an array that is loaded from more than one input record. Records identified by a 1 or 3 in column 1 contain six array elements (4 characters in each element). Records identified by a 2 in column 1 are also in the same input file, but they do not contain array elements. The program processes one record at a time; therefore, the program cannot process the entire array until it reads every record containing the array elements and moves the elements into the array fields. Therefore, be sure that your program reads the entire array before it does any calculation or output operations using the array.

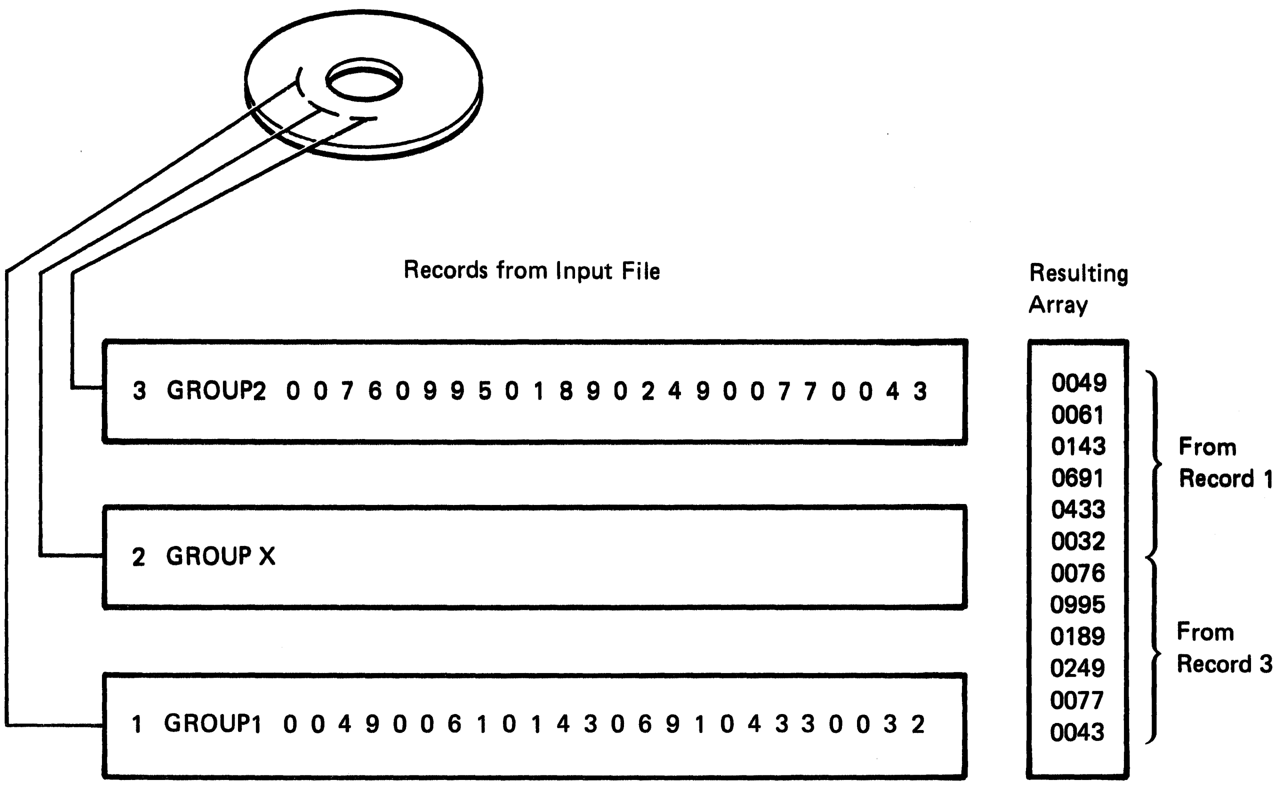


Figure 13-7. Loading an Array from More than One Input Record

Using an Array Name and Index

Your program can refer to an array as a whole or can refer to an individual element in an array. To refer to the entire array, use the array name alone. To refer to a single element of the array, use the array name plus an index. To do so, add a comma and an index after the array name. The index can be either the actual number of the element to be used (for example, AR,1) or the name of a field containing the number of the element to be used (for example, AR,IND).

Remember the following rules when specifying an array name and index:

- The array name must not be the same as the name of a field, data structure, table, another array, or index in your program.
- The array name can be from 1 through 6 characters long.
- The first three characters of the array name cannot be TAB.
- The array name plus comma and index can be from 3 through 6 characters long. An array name plus comma and index should not be longer than 6 characters because the field name on the output specifications and the result field on the calculation specifications contains only six positions. However, if the array name plus comma and index are specified only in factor 1 or factor 2 of the calculation specifications, the array name plus comma and index can be up to 10 characters long.
- The index can be a numeric field with zero decimal positions or a numeric constant with no plus or minus sign.
- The value in the index must not be zero, negative, or more than the number of elements in the array.

The following are examples of valid and invalid array names:

Array Names and Indexes	Explanation
	Valid
ARRAY	
B	
AR,1	This is the first element of array AR.
X,YY2	YY2 is a field name containing the index value.
	Invalid
BALANCE	The array name has more than 6 characters.
6TOTAL	The first character must be alphabetic.
TOTAL-	Characters after the first must be alphameric or numeric, not special characters.
CR TOT	The name cannot contain a blank.
A1,A1	The index cannot be the same as the array name.
BAL,XX1	The array name, including the comma and index, cannot contain more than 6 characters. This name is valid only for factor 1 or factor 2 of the calculation specifications.
AR,+1	The index cannot have a sign.
AR,0	The index cannot be zero.

Searching Arrays and Tables

To search for a particular element in an array or table, use the LOKUP operation with factor 1, factor 2, and at least one resulting indicator (high, low, or equal) specified. The result field can be specified for a table; it must be blank for an array. Searching an array or table is a useful way to find a sequence of characters or multiple occurrences of a character in a record. For example, you can find all the blanks in a record by defining the record as an array with 1-character elements and searching for a blank.

Resulting indicators specify the type of search and reflect the result of the search in the following way:

- A resulting indicator in the equal columns (58 and 59) instructs the program to search the array or table for an element equal to factor 1. The first equal entry found turns on the resulting indicator.

- A resulting indicator in the low columns (56 and 57) instructs the program to search the array or table for an element that is nearest in sequence to, yet lower than, factor 1. The first such element found turns on the indicator.

- A resulting indicator in the high columns (54 and 55) instructs the program to search the array or table for an element that is nearest in sequence to, yet higher than, factor 1. The first such element found turns on the indicator.

At least one resulting indicator must be used, but no more than two can be used (equal and low, or equal and high). If two resulting indicators are used, the program tries to find an equal element before it tries to find the nearest lower or nearest higher element. If resulting indicators are used in both the high and low columns, the indicator in the low columns is ignored.

When you use the LOKUP operation, remember:

- Conditioning indicators can be specified in columns 7 through 17.

- Factor 1 and each array or table element must have the same length and same format (alphameric or numeric).

- The program can search for high, low, high and equal, or low and equal only if the array or table is in sequence. The sequence must be indicated in column 45 of the extension specifications.

- The resulting indicator turns off if the search is not successful.

Searching an Array

To search an array that does not have an index, use a LOKUP operation and specify as factor 1 the data for which you want to find a match in the array to be searched. Factor 1 can be a constant, a field name, an array element, or a table name. In factor 2, specify the name of the array to be searched. The search starts at the first element in the array. Leave the result field blank.

To search an array that has an index, you can begin the search at a particular element in the array. Code the same entries for the LOKUP operation as you would to search an array without an index. However, in factor 2, enter the name of the array to be searched plus a comma and the index for the element at which the search is to begin. If the index is a variable, the index is set to the number of the array element found if the search is successful. If the search is unsuccessful, the index is set to 1.

Figure 13-8 shows an example of a LOKUP for an array with and without an index.

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
01	E			ARRFILE		MANNOS	10	2100	6		0	A						
02	E																	

MANNOS, a 2100-element array of employee numbers, is read in at execution time from the file ARRFILE with ten 6-position elements per record; the array elements are in ascending order.

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators High (1>2, Plus)	Low (1<2, Minus)	Equal (1=2, Zero)	Comments
01	C			100336	LOKUP	MANNOS					20			
02	C		20		GOTO	NEXT								
03	C													
04	C				Z-ADD	1	INX	4	0					
05	C			100336	LOKUP	MANNOS,INX					20			
06	C		N20		GOTO	END								
07	C				MOVE	MANNOS,INX	SAVE	6	0					
08	C			NEXT	TAG									
09	C													
10	C													
11	C					Calculation Operations								
12	C													
13	C													
14	C													
15	C													
16	C													

The first calculation specification is a LOKUP of array MANNOS to find the element nearest to, but higher in sequence than, the constant 100336. If this element is found in the array, indicator 20 turns on and the GOTO in line 02 is performed. Indicator 20 indicates only whether or not the searched-for element exists in the array.

The specification on line 05 shows essentially the same LOKUP operation. Indicator 20 turns on when the first element higher in sequence than 100336 is found. However, in this LOKUP operation, the array MANNOS is indexed by the field INX. This index field was set to 1 in line 04, so the LOKUP begins at the first element of MANNOS. If the searched-for element is found, the number of this element (not its contents) is placed in the field INX. In this way, the actual element that satisfied the LOKUP can be used in subsequent calculation operations, as in line 07. If no element was found to satisfy the LOKUP, the field INX is reset to 1.

Figure 13-8. LOKUP Operation for Arrays with and without an Index

Searching One Table

To search a single table, use the LOKUP operation with factor 1, factor 2, and at least one resulting indicator specified. The result field can be specified for a table (it must be blank for an array).

If the search finds a table element that satisfies the resulting indicator, the program places a copy of that table element in a special area of main storage. Each time a search is successful, the program places the newly found table element in this area, replacing the element that was in the area. If a search is not successful, the contents of the area remain the same as they were before the unsuccessful search. Before the first successful search, the area contains the first element in the table.

If you use a table name as factor 1, the table name actually refers to the table item found in the last successful search. Therefore, the last item found becomes the data for which you are searching in the current search.

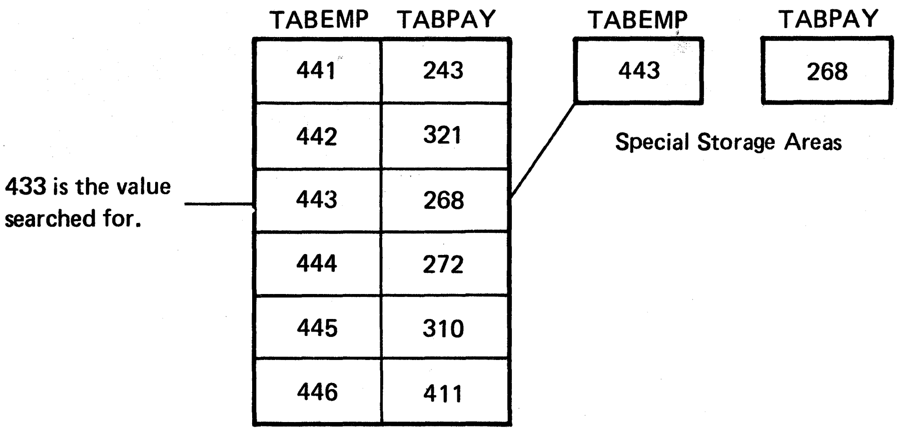
Searching Related Tables

When you use the LOKUP operation to search related tables, the program actually searches only one table (see Figure 13-9). If the search is successful, the corresponding elements from both tables are placed in their respective storage areas.

Note: As used here, the phrase *related tables* means any two tables in the program that use related data, not necessarily tables that are defined as related on the extension specifications.

Factor 1 of the LOKUP operation must contain the data for which you want to search, and factor 2 must contain the name of the table to be searched. The result field must contain the name of the related table. A resulting indicator must be specified.

The two tables should have the same number of elements. If the table that is searched contains more elements than the second table, the program might find the desired element in the first table but find no corresponding element in the second table. In this case, undesirable results can occur.



Related tables TABEMP and TABPAY are read into storage. Assume that an input record is read with 443 in the EMPNUM field. Then the program searches the table TABEMP for an element equal to 443. When the correct entry is found, the table item 443 is moved into the special storage area for TABEMP. At the same time, the corresponding item 268 is moved into the special storage area for TABPAY. The contents of the storage areas can now be specified in subsequent calculation operations by the appropriate table name. The coding needed to perform the LOKUP operation also shows how to specify the contents of the special storage area after a successful LOKUP operation.

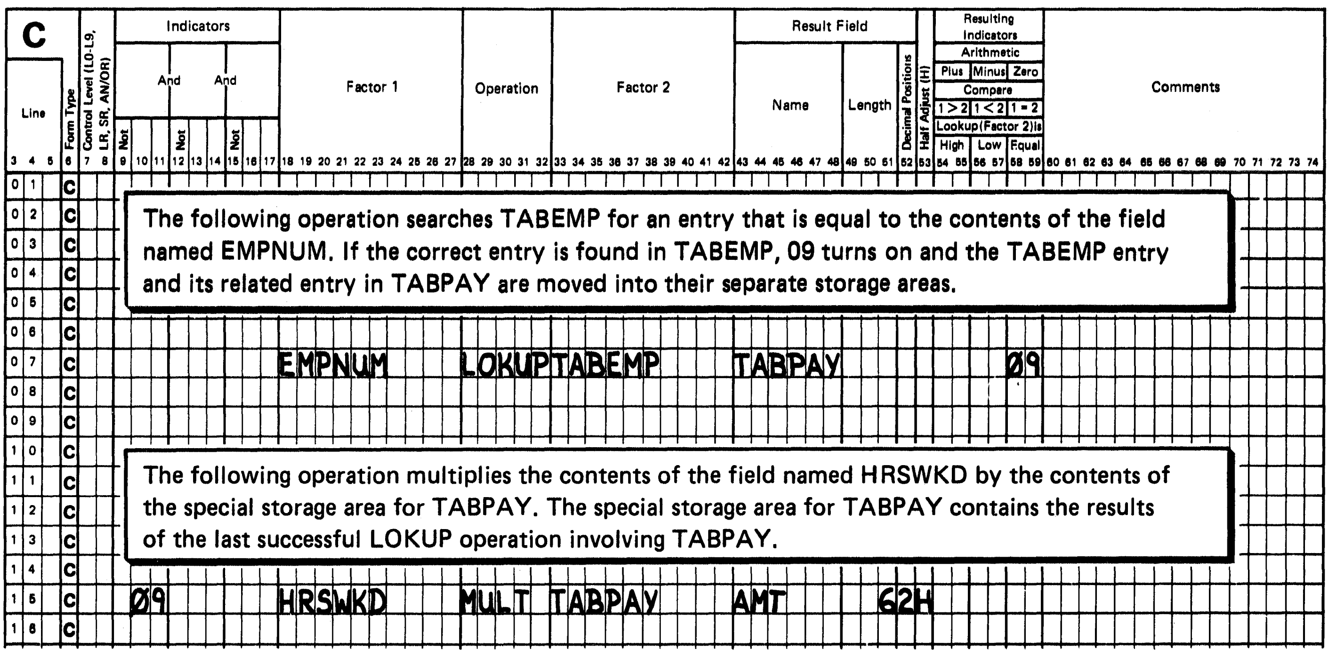


Figure 13-9. LOKUP Operation for Related Tables

Specifying Arrays

You can specify arrays in input, output, or calculation specifications. You can specify individual elements or the array as a whole.

To specify an entire array, use only the array name, which can be used as factor 1, factor 2, or the result field. You can use the following operations with an array name: ADD, Z-ADD, SUB, Z-SUB, MULT, DIV, SQRT, MOVE, MOVEL, MOVEA, MLLZO, MLHZO, MHLZO, MHHZO, DEBUG, XFOOT, SORTA, and LOKUP.

To specify an individual array element, use the array name plus a comma and an index. Process individual elements like fields. Remember, if you use an array element as a result field, the array name with the comma and index cannot exceed 6 characters. Several operations can be used with an individual array element, but not with an entire array. These operations are COMP, TESTZ, TESTB, BITON, BITOF, KEY, SET, and MVR, as well as IF/ELSE, CASxx, DOUxx, and DOWxx.

When specified with an array name, certain operations are repeated for each element in the array. These operations are ADD, Z-ADD, SUB, Z-SUB, MULT, DIV, SQRT, MOVE, MOVEL, MLLZO, MLHZO, MHLZO, and MHHZO. The following rules apply when these operations are specified with an array name:

- If factor 1, factor 2, and the result field are arrays with the same number of elements, the operation uses the first element from every array, then the second element from every array, and so on until all elements in the arrays are processed.

- If factor 1, factor 2, and the result field are arrays that do not have the same number of entries, the operation ends when the last element of the array with the fewest elements has been processed.

- When one of the factors is a field, constant, or figurative constant and the other factor and the result field are arrays, the operation is performed once for every element in the shorter array. The same field, constant, or figurative constant is used in all of the operations.

- The result field must always be an array.

- Resulting indicators (columns 54 through 59) cannot be used because of the number of operations being performed.

- If an operation code uses factor 2 only (such as Z-ADD, Z-SUB, or SQRT) and the result field is an array, the operation is performed once for each element in the array. The same field, constant, or figurative constant is used in all of the operations.

Changing the Contents of Arrays and Tables

Changing the Contents Temporarily

You can change the contents of an array or table in a program, and they remain changed for the duration of the program. However, the next time the program is run, the array or table contains the original contents.

One way to do so is to use the array or table name as the result field in a MOVE operation. Figure 13-10 shows an example of changing the contents of related tables.

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators High (54-55)	Low (56-57)	Equal (58-59)	Comments
01	C													
02	C			25	LOKUP	TABFIL	TABLIT						1Ø	1Ø = FOUND
03	C		1Ø		MOVE	5ØØ	TABLIT							
04	C		1Ø		MOVE	3Ø	TABFIL							
05	C													

The item in TABFIL that contains 25 is to be changed to 30. The corresponding item in TABLIT is to be changed to 500. The search word is the constant 25. When a match is found in the table TABFIL, the item from TABFIL and its corresponding item in TABLIT are placed in their respective storage areas. The number 500 is then moved into the storage area for TABLIT; the number 30 is moved into the storage area for TABFIL. The contents of the appropriate original table entry are now changed to agree with the new entry in the special storage areas.

Figure 13-10. Changing Related Tables by Using MOVE Operations

A second way to change the contents of an array temporarily is to use the SORTA operation. This method cannot be used with tables.

A third way is to use the array or table name as the result field of a calculation. If you use an element in that array as factor 1 or factor 2 in that calculation, your program will use the new value of the element in later calculations.

For example, suppose that you have two numeric arrays with the following values:

```
ARR1,1 = 2     ARR2,1 = 2
ARR1,2 = 4     ARR2,2 = 8
ARR1,3 = 6     ARR2,3 = 1
```

Now suppose that you code the following calculation specification:

C Line (3-5)	Form Type (6)	Control Level (L0-L9, LR, SR, AN/OR) (7-8)	Indicators: Not (9)	Indicators (10-11)	And: Not (12)	And (13-14)	And: Not (15)	And (16-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field: Name (43-48)	Result Field: Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Arithmetic Plus / Compare 1 > 2 / Lookup(Factor 2) is High (54-55)	Arithmetic Minus / Compare 1 < 2 / Lookup(Factor 2) is Low (56-57)	Arithmetic Zero / Compare 1 = 2 / Lookup(Factor 2) is Equal (58-59)	Comments (60-73)
01	C								ARR1	ADD	ARR2,2	ARR2							
02	C																		

This operation adds the three elements in ARR1, one at a time, to the second element in ARR2 and places the result in ARR2. Here is what happens as the three elements are added:

1. The value of ARR1,1 is added to the value of ARR2,2. That is, 2 is added to 8. The result, 10, is placed in ARR2,1. Thus, the value of ARR2,1 changes from 2 to 10.

2. The value of ARR1,2 is added to the value of ARR2,2. That is, 4 is added to 8. The result, 12, is placed in ARR2,2. Thus, the value of ARR2,2 changes from 8 to 12.

3. The value of ARR1,3 is added to the value of ARR2,2. That is, 6 is added to 12 (the new value of ARR2,2), not to 8 (the old value of ARR2,2). The result, 18, is placed in ARR2,3. Thus, the value of ARR2,3 changes from 1 to 18.

Changing the Contents Permanently

One way to change the contents of an array or table permanently is to change the input records for the array or table.

A second way to change an array or table permanently is to use one of the methods to change the array or table temporarily, and then to write the array or table to an output file at the end of the program. To do this, define an output file on the file description specifications, and code the name of that output file in columns 19 through 26 of the extension specifications.

Adding Entries to Arrays and Tables

You can add entries to a short array or table (one in which not all elements are filled) before running the program or while it is running. The simplest way to add entries is to code additional entries on the input records before running the program. While the program is running, you can also add entries that are created by calculation operations or read from an input record.

Figure 13-11 shows an example of adding entries to arrays by using the LOKUP and MOVE operations. These entries are added only temporarily unless the array is written to an output file that is used as input for a preexecution-time array the next time the program is run.

Line	Form Type	Control Level	Indicators	And	And	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust	Resulting Indicators High (1>2)	Low (1<2)	Equal (1=2)	Comments
01	C						⌇									
02	C		Ø1			ØØØ	LOKUP	ARRA,X							35	35 = FOUND
03	C		35	Ø1			MOVE	NEWA	ARRA,X							
04	C		35	Ø1			MOVE	NEWB	ARRB,X							
05	C						⌇									
06	C															

⌇ = Calculations

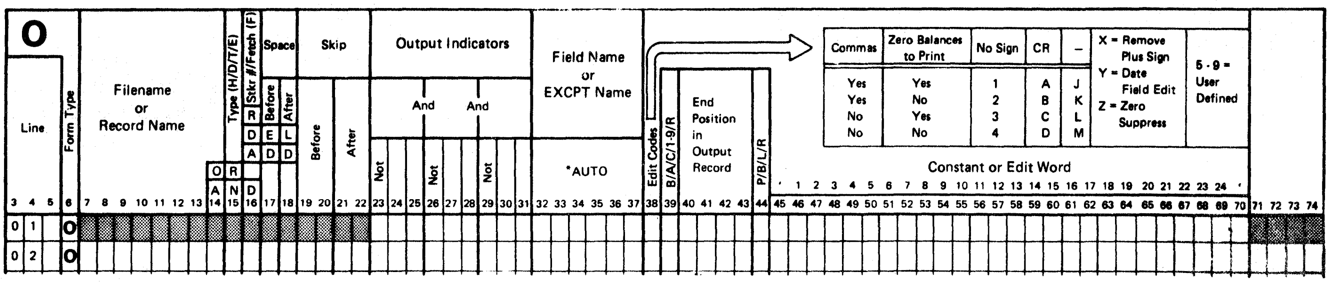
The LOKUP operation is conditioned by indicator 01. Indicator 01 is set on when a record containing information in the fields NEWA and NEWB is read. These fields are to be moved to the arrays ARRA and ARRB, respectively. To get the entry in the correct place in the array, a search is made to find the first empty array element. Unfilled elements in arrays are filled with zeros. Thus, the value searched for is 000. When the first 000 entry is found, indicator 35 is set on, and the NEWA and NEWB fields are moved into the array elements ARRA,X and ARRB,X. These new entries become part of arrays ARRA and ARRB.

Figure 13-11. Adding Entries to Arrays

Writing Arrays and Tables

You can write entire arrays and tables to an output file at total time in the RPG program cycle when the last-record indicator is on. To indicate that an entire array or table is to be written, specify the name of the output file in columns 19 through 26 of the extension specifications.

To write an array to an output record by using output specifications, describe the array in the unshaded columns of the output specifications shown below:



- Columns 23 through 31 can contain output indicators to condition the writing of the array or table.
- Columns 32 through 37 must contain the array name used on the extension specifications.
- Columns 40 through 43 must contain the end position for the last element of the array. If you use an edit code, the end position must account for the skipped positions required by the edit code (see *Editing Arrays*, later in this chapter).
- Entries in columns 38 (edit code), 39 (blank after), and 44 (zoned-decimal, packed-decimal, or binary format) apply to each element in the array.
- Columns 45 through 70 can contain an edit word, which applies to each element in the array.

If an output record is to contain only certain elements from a table or array, describe the elements in the same way as normal fields, using either an array name with an index or a table name.

Editing Arrays

In column 38 of the output specifications, you can specify an edit code for an entire array or for individual elements in an array.

If you specify an edit code for an entire array, all elements of the array are edited. The program skips two positions before each element in the array. The end position specified in columns 40 through 43 must account for these skipped positions (two times the number of elements). If you are overlaying data in a record, these skipped positions are not blanked out.

If different editing is required for various elements, specify them individually.

If you specify an edit word in columns 45 through 70 of the output specifications, two positions are not skipped before each element. The edit word must contain all the blanks that you want inserted. To include a blank in an edit word, use an ampersand (&) in the edit word to represent a blank.

Examples of Using Arrays

Figures 13-12 through 13-18 show examples of the following ways to use arrays:

- Creating an array by using input fields as indexes (Figure 13-12)
- Creating an array by using fixed indexes (Figure 13-13)
- Calculating totals without using arrays (Figure 13-14)
- Calculating totals by using arrays (Figure 13-15)
- Formatting output fields by using arrays (Figure 13-16)
- Printing one array element per line (Figure 13-17)
- Printing more than one array element per line (Figure 13-18)

Line	Form Type	Record Sequence of the Chaining File / Number of the Chaining Field / From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
01	E			AR		12	5									
02	E															

Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator, or **	Record Identification Code 1: Position	C/Z/D	Character	Field Location From	Field Location To	Decimal Positions	RPG Field Name
01	I	FILE1	AA	Ø3	8Ø	C	1				
02	I							2	3	Ø	X1
03	I							4	8		AR,X1
04	I							9	1Ø	Ø	X2
05	I							11	15		AR,X2
06	I							16	17	Ø	X3
07	I							18	22		AR,X3
08	I										
09	I										More Array Elements
10	I										
11	I							54	55	Ø	X1Ø
12	I							56	6Ø		AR,X1Ø
13	I		BB	Ø4	8Ø	C	2				
14	I							2	3	Ø	X1
15	I							4	8		AR,X1
16	I							9	1Ø	Ø	X2
17	I							11	15		AR,X2
18	I										

This figure illustrates a method of loading an array using fields in input records as indexes. The array has 12 elements; each element is 5 positions long. The array could be defined with any number of elements (to a maximum of 99) without additional input specifications. To build an array using field indexes, assign different values to fields X1 through X10 on each input record type 03 and to fields X1 and X2 on each input record type 04. Succeeding type 03 records can then load 10 additional elements in array AR, up to the maximum defined in the array; each type 04 record can load two additional elements.

Blanks and other fields can appear on the input records because the array elements and their indexes are identified by the From and To entries.

To set up the array in this manner requires:

- A minimum of coding
- No calculations

However, extra work is required to set up the indexing scheme for the input records.

Figure 13-12. Creating an Array by Using Input Fields as Indexes

Line	Form Type	From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
01	E			AR1		3Ø	5									
02	E															

Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator	Position (1)	Not (N)	C/Z/D	Character	Field Location From	Field Location To	Decimal Positions	RPG Field Name	Control Level	Matching Fields	Field Record Relation	Plus	Minus	Zero or Blank
01	I	FILE1	AA	Ø3	1ØØ		C	1										
02	I								1	9Ø		AR1						
03	I		BB	Ø4	1ØØ		C	2										
04	I								1	5		AR1,19						
05	I								6	1Ø		AR1,2Ø						
06	I																	
07	I																	
08	I																	
09	I											More Array Elements						
10	I																	
11	I																	
12	I																	
13	I																	
14	I																	

This figure shows how eighteen 5-character elements of array AR1 are loaded with only two specification lines. On succeeding input specifications, the remaining elements of AR1 are loaded one after another until the array is full. Each additional element is coded on a separate line. Each new record requires a separate means of identification. For example, if another 03 record followed the first, the fields on the second record would overlay the fields read in from the first record. This method works well for small arrays.

Figure 13-13. Creating an Array by Using Fixed Indexes

Line	Form Type	Control Level	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Comments
01	C		FIELDA	ADD	L1A	L1A	6	2	ADD TO L1 TOTLS
02	C		FIELDB	ADD	L1B	L1B	6	2	
03	C		FIELDC	ADD	L1C	L1C	6	2	
04	C		FIELDD	ADD	L1D	L1D	6	2	
05	C	L1	L1A	ADD	L2A	L2A	6	2	ADD TO L2 TOTLS
06	C	L1	L1B	ADD	L2B	L2B	6	2	
07	C	L1	L1C	ADD	L2C	L2C	6	2	
08	C	L1	L1D	ADD	L2D	L2D	6	2	
09	C	L2	L2A	ADD	L3A	L3A	6	2	ADD TO L3 TOTLS
10	C	L2	L2B	ADD	L3B	L3B	6	2	
11	C	L2	L2C	ADD	L3C	L3C	6	2	
12	C	L2	L2D	ADD	L3D	L3D	6	2	
13	C								

Line	Form Type	Type (H/D/T/E)	Space Before	Space After	Output Indicators	Field Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record
01	O	T	2	0	L1				
02	O					L1A	K	B	15
03	O					L1B	K	B	30
04	O					L1C	K	B	45
05	O					L1D	K	B	60
06	O	T	2	0	L2				
07	O					L2A	K	B	15
08	O					L2B	K	B	30
09	O					L2C	K	B	45
10	O					L2D	K	B	60
11	O	T	2	0	L3				
12	O					L3A	K	B	15
13	O					L3B	K	B	30
14	O					L3C	K	B	45
15	O					L3D	K	B	60
16	O								

The specifications in this figure tabulate three levels of totals. As they are read from input records, the fields FIELDA, FIELDB, FIELDC, and FIELDD are added to the first-level totals L1a, L1B, L1C, and L1D. These first-level totals are added at the time of an L1 control break to totals L2A, L2B, L2C, and L2D. Similarly, at an L2 control break, the second-level totals are added to third-level totals L3A, L3B, L3C, and L3D. In addition, as control breaks occur, L1, L2, and L3 total output is performed; and total fields are set to zeros after they are written to the output device. Figure 13-15 shows the same tabulations performed on using arrays.

Figure 13-14. Calculating Totals without Using Arrays

E

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
01	E					SL1		4	6		2							L1 TOTALS
02	E					SL2		4	6		2							L2 TOTALS
03	E					SL3		4	6		2							L3 TOTALS
04	E																	

C

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators	Comments
01	C			FIELDA	ADD	SL1,1	SL1,1					ADD FOR L1 TOTL
02	C			FIELDB	ADD	SL1,2	SL1,2					
03	C			FIELDC	ADD	SL1,3	SL1,3					
04	C			FIELDD	ADD	SL1,4	SL1,4					
05	C	L1		SL1	ADD	SL2	SL2					ADD FOR L2 TOTL
06	C	L2		SL2	ADD	SL3	SL3					ADD FOR L3 TOTL
07	C											

O

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Space Before	Space After	Skip Before	Skip After	Output Indicators	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		T	2	0			L1						
02	O								SL1	K	B	60		
03	O		T	2	0			L2						
04	O								SL2	K	B	60		
05	O		T	2	0			L3						
06	O								SL3	K	B	60		
07	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

This figure is similar to Figure 13-14 except that the three levels of totals are tabulated with arrays. Note the reduction in coding required to specify the functions. For example, line 5 of the calculation specifications performs the same function as lines 5 through 8 of the calculation specifications shown in Figure 13-14. Similarly, the output specifications are reduced from 15 lines to 6. The method using arrays results in only two positions between array elements.

Figure 13-15. Calculating Totals by Using Arrays

Line	Form Type	Record Sequence of the Chaining File / Number of the Chaining Field / From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
01	E			ARA		4	5		Ø							
02	E			ARB		5	10									
03	E			ARC		6	4		2							
04	E															

Line	Form Type	Filename or Record Name (Data Structure Name)	AND/OR	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, **, or DS	Position (1)	Not (N)	C/Z/D	Character	Position (2)	Position (3)	Stacker Select	P/B/L/R	Field Location From (Occurs n Times)	Field Location To (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	IN		AA			Ø1	8Ø		C															
02	I		OR				Ø2	8Ø		C	1														
03	I															51	74		ARC						
04	I															1	2Ø		ARA			Ø1			
05	I															1	5Ø		ARB			Ø2			
06	I																								

Line	Form Type	Filename or Record Name	AND/OR	Type (H/D/T/E)	Stkr #/Fetch (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators	Field Name or EXCPT Name (*AUTO)	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O	OUT		D			1			Ø1						
02	O		OR							Ø2						
03	O										ARC			84		'Ø . &CR'
04	O									Ø1	ARA,1	Z		89		
05	O									Ø2	ARB,X1			1ØØ		
06	O															

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

Figure 13-16 (Part 1 of 2). Formatting Output Fields by Using Arrays

This figure illustrates the use of three arrays to format field output. The arrays are defined as follows:

Array Name	Number of Elements	Element Length
ARA	4	5
ARB	5	10
ARC	6	4

Array ARA is contained in the input records with record identifying indicator 01, ARB in the records with record identifying indicator 02, and ARC in both types of records. Array ARC and the first element of array ARA are to be included together in an output record as are array ARC and an element (identified by field X1) of array ARB. Every element in array ARC is edited according to the edit word '0ƀ.ƀƀ&CR' (ƀ = blank).

The contents of the arrays in the first two input records are as follows:

Record	Array	Array Contents
1	ARA	12345678901234567890
	ARC	012345678901234567898 76N (note than N equals minus 5)
2	ARB	JOHNƀDOEƀƀJOEƀSMITHƀ LEEƀMARXƀƀJIMƀKNOTSƀ TIMƀTYLERƀ
	ARC	(the same as record 1)

In the first output record, the location and contents of the arrays are as follows (ƀ = blank):

Array	Location	Contents
ARA (first element)	85-89	12345
ARC	37-84	ƀ1.23ƀƀƀ45.67ƀƀƀ 89.01ƀƀƀ23.45ƀƀƀ 67.89ƀƀƀ87.65ƀCR

For the second output record assume that the content of field X1 is 4; the locations and contents of the arrays are as follows:

Array	Location	Contents
ARB (fourth element)	91-100	JIMƀKNOTSƀ
ARC	37-84	ƀ1.23ƀƀƀ45.67ƀƀƀ 89.01ƀƀƀ23.45ƀƀƀ 67.89ƀƀƀ87.65ƀCR

Figure 13-16 (Part 2 of 2). Formatting Output Fields by Using Arrays

Figure 13-17 shows a method of printing one array element per line. Each time the EXCPT operation on line 03 of the calculation specifications occurs, one element of the 22-element array AR2 is written to the output file ARFILE.

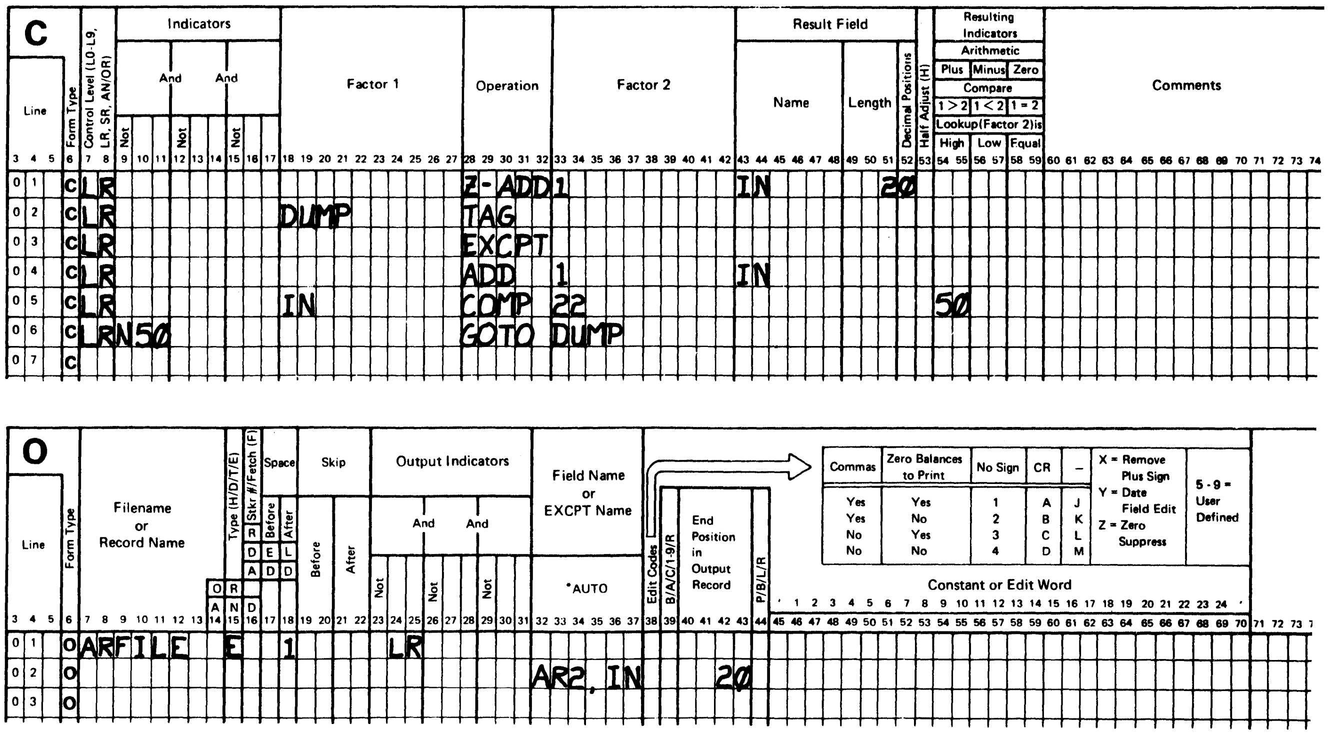


Figure 13-17. Printing One Array Item per Line

Figure 13-18 shows a method of printing more than one array element per line. The number of elements printed on a line depends on the value coded as factor 2 in the COMP operation in line 10 of the calculation specifications. In this example, that value is 10, and the number of elements in array AR2 is 50.

If an edit code is used, each array element is preceded by two spaces. You must take these spaces into account when you compute the end position for the output specifications.

```
C  Line  Form  Control  Indicators     Factor 1   Operation  Factor 2   Result Field  Length  Dec  Resulting Indicators
                Level                                                   Name                     (High/Low/Equal)
   01    C     LR                                 Z-ADD      1          IN            20
   02    C     LR                      DUMP       TAG
   03    C     LR                                 Z-ADD      1          X1            20
   04    C     LR                      UP         TAG
   05    C     LR                                 MOVE       AR2,IN     AR1,X1
   06    C     LR                                 ADD        1          X1
   07    C     LR                                 ADD        1          IN
   08    C     LR                      IN         COMP       50                                      12
   09    C     LR       12                        GOTO       OUT
   10    C     LR                      X1         COMP       10                                      14
   11    C     LR       N14                       GOTO       UP
   12    C     LR                      OUT        TAG
   13    C     LR                                 EXCPT
   14    C     LR       14N12                     GOTO       DUMP
   15    C
```

```
O  Line  Form  Filename   Type  Space  Output       Field Name   Edit   End Position
               or Record        After  Indicators   or EXCPT     Code   in Output
               Name                                 Name                Record
   01    O     ARFILE     E     1      LR
   02    O                                          AR1          B      100
   03    O
```

Figure 13-18. Printing More than One Array Item per Line

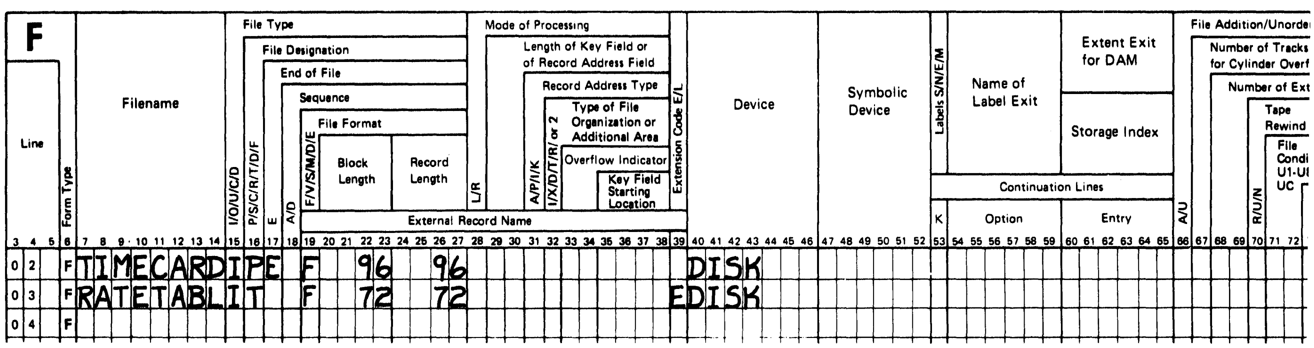
Example of Using Tables

The following payroll program requires two related tables:

TABNUM	TABRAT
12345	407
12346	593
12347	369
12348	390
12349	1379

TABNUM, which contains employee numbers, is the table searched. TABRAT, which contains employee salary rates, is the related table. After the program finds an employee's number and salary rate, it multiplies the rate by the number of hours worked. The result is the amount earned.

File Description Specifications



The input records are contained in the input file TIMECARD, which is designated as the primary file (P in column 16). When the file reaches end of file, processing ends (E in column 17). Each record in this file is 96 positions long. This file is read from disk.

The related tables are contained in the input file RATETABL, which is designated as a table file (T in column 16). This file is read from disk before the program is run. Each record in this file is 72 positions long. The E in column 39 shows that the extension specifications contain additional information about this file.

Extension Specifications

Line (3–5)	Form Type (6)	Record Sequence of the Chaining File (7–8)	Number of the Chaining Field (9–10)	From Filename (11–18)	To Filename (19–26)	Table or Array Name (27–32)	Number of Entries Per Record (33–35)	Number of Entries Per Table or Array (36–39)	Length of Entry (40–42)	P/B/L/R (43)	Decimal Positions (44)	Sequence (A/D) (45)	Table or Array Name (Alternating Format) (46–51)	Length of Entry (52–54)	P/B/L/R (55)	Decimal Positions (56)	Sequence (A/D) (57)	Comments (58–74)
01	E			RATETABL		TABNUM	8	5ØØ	5		Ø	A	TABRAT	4		2		
02	E																	

The extension specifications complete the description of the file RATETABL. The table searched is TABNUM (columns 27 through 32), which has eight elements in each record (columns 33 through 35) and 500 elements in the table (columns 36 through 39). Each element is 5 positions long (columns 40 through 42) with zero decimal positions (column 44). The table is organized in ascending sequence (column 45).

The related table is TABRAT (columns 46 through 51). Each element is 4 positions long (columns 52 through 54) with two decimal positions (column 56).

The table input records are organized in alternating format (although related tables do not have to be in alternating format). That is, the first record begins with the first element of TABNUM, which is followed by the first element of TABRAT, the second element of TABNUM, the second element of TABRAT, and so on in alternating sequence. Each element of TABNUM is 5 positions long, and each element of TABRAT is 4 positions long. Therefore, each pair of related elements is 9 positions long. There are eight elements of each table per record, so each record in RATETABL is 72 positions long.

Each table has 500 elements, so the file requires 63 records (500 elements divided by 8 elements per record = 62.5 records). The first 62 records contain data in positions 1 through 72, but the 63rd record contains data only in positions 1 through 36.

Input Specifications

Line (3-5)	Form Type (6)	Filename or Record Name (7-14)	Sequence (15-16)	Number (1/N), E (17)	Option (O), U, S (18)	Record Identifying Indicator, **, or DS (19-20)	Field Location From (44-47)	Field Location To (48-51)	Decimal Positions (52)	RPG Field Name (53-58)
01	I	TIMECARD	AA			Ø1				
02	I						1	5	Ø	EMPNUM
03	I						42	44	1	HRSWKD
04	I									

The input file TIMECARD is assigned a sequence of AA (columns 15 and 16). Record-identifying indicator 01 turns on whenever an input record is present for processing. No record identification codes are specified in columns 21 through 41 because there is only one record type.

Lines 02 and 03 describe the locations of the two input fields used by the program. The employee number (EMPNUM) is in positions 1 through 5 of the input record. The number of hours worked (HRSWKD) is in positions 42 through 44 of the input record.

Calculation Specifications

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Plus / 1>2 / High	Minus / 1<2 / Low	Zero / 1=2 / Equal	Comments
01	C		Ø1	EMPNUM	LOKUP	TABNUM	TABRAT						Ø3	
02	C		Ø3	TABRAT	MULT	HRSWKD	EARNS	5	2	H				
03	C		NØ3		MOVE	ØØØ.ØØ	EARNS							
04	C													

On line 01, the LOKUP operation instructs the program to search the table TABNUM (factor 2) for an element that matches the value of the field EMPNUM (factor 1). Resulting indicator 03 turns on when an element in TABNUM is found that is equal to the value of EMPNUM. The related table TABRAT is specified as the result field.

When indicator 03 is on, the MULT (multiply) operation in line 02 is performed. The salary rate for the employee, taken from the related table TABRAT (factor 1) is multiplied by the number of hours worked, HRSWKD (factor 2). The result is stored in the result field EARNS, which is 5 positions long with two decimal positions. The result is half-adjusted (H in column 53).

When indicator 03 is not on, the MOVE operation in line 03 occurs. The literal 000.00 (factor 2) is moved into the field EARNS (result field) to indicate that the table does not contain an entry for that employee. The decimal point in the literal is used only to align the data; it is not actually put in the field EARNS.

Chapter 14. Using Data Structures

Chapter 14. Using Data Structures

A *data structure* is an area in storage that is composed of one or more fields, called *subfields*. You can use a data structure to:

- Define that area of storage more than one way
- Subdivide an input field so that your program can refer to either the entire field or its subfields
- Reorganize fields in an input record for easier reference

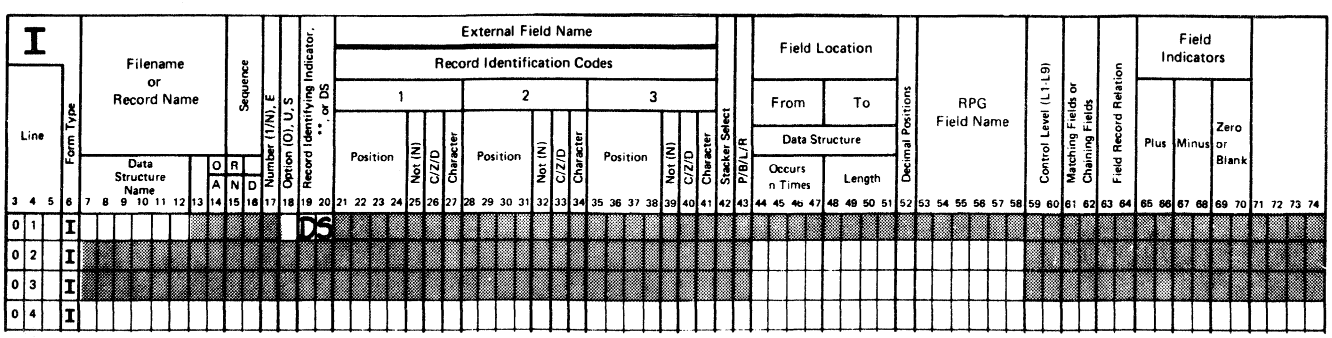
See *Examples of Data Structures*, later in this chapter, for an example of a data structure used for each of these purposes.

Coding a Data Structure

Data structures are coded on input specifications. They must be the last entries on the input specifications. That is, they must follow all specifications for input records.

Specifications for a data structure have two parts: the data structure statement and the subfields. Specifications for the subfields must be coded on the lines immediately below the specification for the data structure statement.

To code a data structure statement and subfields, make entries in the unshaded columns of the input specifications shown below:



Data Structure Statement

Columns 7 through 12 can contain blanks or the name of the data structure. Although columns 7 through 14 are normally used as one entry, the name of the data structure cannot be more than 6 characters long.

Column 18 must contain U if this data structure is to be used as a local data area for a display station (see *Local Data Area for a Display Station* in this chapter). Otherwise, leave this column blank.

Columns 19 through 20 must contain DS, which identifies this statement as a data structure.

Subfields

Columns 44 through 47 must contain the record position in which the subfield begins, relative to the beginning of the data structure, not relative to the beginning of the input record.

Columns 48 through 51 must contain the record position in which the subfield ends, relative to the beginning of the data structure, not relative to the beginning of the input record.

Column 52 must contain the number of decimal positions if the subfield is numeric. It must be blank if the subfield is alphameric.

Positions 53 through 58 must contain the subfield name. The subfield name can be the same as an input field name or a result field name. Subfields can be used as factor 1, factor 2, or the result field of a calculation specification or as output fields. However, the same subfield name cannot be used in more than one data structure, and a data structure name cannot be used as a subfield name in another data structure.

Rules for Coding Data Structures

A data structure is considered alphameric data. Therefore, when a data structure is created, it is set to blanks, except for those subfields that are set by an array or by a local data area for a display station. You must ensure that numeric subfields contain numeric data before you use the subfields in CHAIN, READE, LOKUP, COMP, IFxx, DOUxx, DOWxx, CASxx, or editing operations, or arithmetic operations.

A maximum of 75 data structures can be used in a program.

A data structure can be from 1 through 9999 characters long. However, the maximum length of a data structure used as a local data area for a display station is 512 characters.

The maximum length of an alphameric subfield is 256 characters; the maximum length of a numeric subfield is 15 characters.

If arrays are specified as subfields, the length specified must equal the amount of storage required to store the entire array.

The length of a data structure is one of the following:

- The length specified in the input field specifications if the data structure name is an input field
- The highest entry in columns 48 through 51 of a subfield if the data structure name is not an input field

The length of the data structure is determined by the first specification in the program that defines a length in one of the ways just listed. Conflicting lengths in later specifications are not valid.

The name of an input field or a result field that is being redefined in a data structure must be the data structure name or must be specified in the data structure; however, it does not have to immediately precede the subfields redefining it.

If a field appears as a data structure name or as a data structure subfield name, the physical space reserved for that field is in the data structure, regardless of where the field was defined.

Look-ahead fields cannot appear as a data structure or a subfield.

An RPG reserved word, array item, or table name cannot be specified as a subfield.

A packed-decimal or binary numeric field cannot be specified as a subfield within the data structure. If a field is defined as packed-decimal or binary in a file, the program converts that field to zoned-decimal format when it places the field in the data structure.

Examples of Data Structures

Example 1. Defining One Area of Storage More than One Way

Figure 14-1 shows a 40-position data structure that defines one area of storage in three ways:

- Positions 1 through 32 are defined as the sales record (SREC).
- Positions 1 through 35 are defined as the purchase record (PREC).
- Positions 1 through 40 are defined as the transfer record (TREC).

Figure 14-2 shows the coding for the data structure in Figure 14-1. The DS in columns 19 and 20 of line 08 identifies the following lines as a data structure. The data structure allows the programmer to define 19 subfields (lines 10 through 15, 17 through 23, and 25 through 30) within only 40 positions of storage.

If the programmer defined each subfield as a field in the input record instead of using a data structure, each input field would require a separate area of storage. That is, the sales record would require 32 positions, the purchase record would require an additional 35 positions, and the transfer record would require an additional 40 positions. Together, the three records would require 117 positions of storage, almost three times as much as the 40 positions required for the data structure.

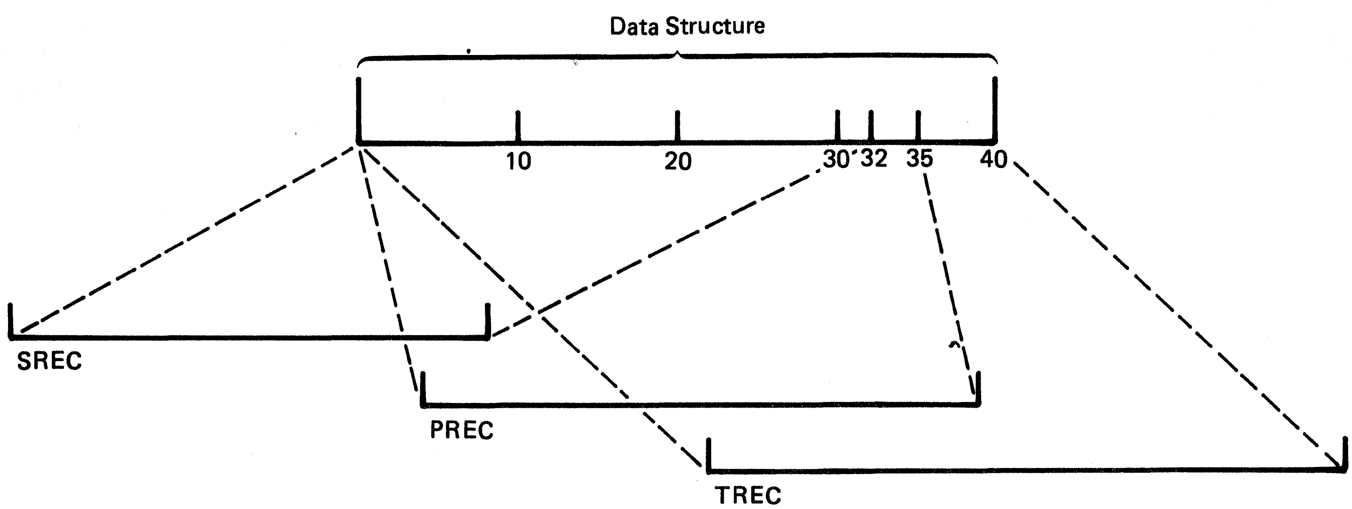


Figure 14-1. Data Structure That Defines One Area of Storage Three Ways

Line	Form Type	Filename or Record Name	Record Identifying Indicator, or DS	Position (1)	C/Z/D (1)	Character (1)	Position (2)	C/Z/D (2)	Character (2)	From	To	Decimal Positions	RPG Field Name
01	I	SALESREC	Ø1	91	C	S	92	C	R				
02	I									1	32		SREC
03	I	PURCHASE	Ø2	91	C	P	92	C	C				
04	I									1	35		PREC
05	I	TRANSFER	Ø3	91	C	T	92	C	F				
06	I									1	4Ø		TREC
07	I	*											
08	I		DS										
09	I	*SALES RECORD FIELDS											
10	I									1	32		SREC
11	I									1	9		ORDNO
12	I									1Ø	16		PART#
13	I									17	2Ø	Ø	QTY1
14	I									21	26	2	UNITCT
15	I									27	32	2	TOTCST
16	I	*PURCHASE RECORD FIELDS											
17	I									1	35		PREC
18	I									1	6		PARTNO
19	I									7	12	Ø	QTYOH
20	I									13	17	Ø	QTYORD
	I									18	23	2	COST
	I									24	29		ORDATE
	I									3Ø	35	2	AMTDUE
	I	*TRANSFER RECORD FIELDS											
	I									1	4Ø		TREC
01	I									1	6		PARTNO
02	I									7	1Ø	Ø	QTY2
03	I									11	2Ø		FRWHSE
04	I									21	3Ø		TOWHSE
05	I									31	4Ø		TIMDAT
06	I												

Figure 14-2. Coding for Data Structure Shown in Figure 14-1

Example 2. Defining Subfields within a Field

Figure 14-3 shows a data structure that subdivides a field in an input record. Input field PARTNO in file FILEIN has 16 positions (from record position 3 through record position 18; see line 02). The data structure defines subfields in field PARTNO.

If a data structure defines subfields within a field, and if that data structure has a name in columns 7 through 12, the data structure name must be the same as the field name (in this case, PARTNO).

Normally, a data structure name cannot be specified as factor 1, factor 2, or the result field of a calculation specification. However, a data structure name can be specified as the result field of an RLABL operation. A data structure subfield name can be specified in a calculation specification. Subfield PARTDS contains all 16 positions in the data structure, so subfield PARTDS can be used to specify the entire data structure in a calculation specification.

Line	Form Type	Filename or Record Name / Data Structure Name	Record Identifying Indicator, **, or DS	Record Identification Code 1 Position	C/Z/D	Character	Record Identification Code 2 Position	C/Z/D	Character	Field Location From	Field Location To	Decimal Positions	RPG Field Name
01	I	FILEIN	Ø1	1	C	A	2	C	B				
02	I									3	18		PARTNO
03	I									19	29		NAME
04	I									3Ø	4Ø		PATNO
05	I									41	61		DR
06	I	PARTNO	DS										
07	I									1	4		MFG
08	I									5	1Ø		DRUG
09	I									11	13		STRNTH
10	I									14	16	Ø	COUNT
11	I									1	16		PARTDS
12	I												

Figure 14-3. Using a Data Structure to Define Subfields within a Field

A data structure can also redefine subfields within a subfield. Figure 14-4 shows that subfields KEY (line 02), ARRFLD (line 07), and ID (line 08) are redefined by subdividing them into even smaller subfields.

Line	Form Type	Filename or Record Name	Record Identifying Indicator, ** or DS	Field Location From	Field Location To	Decimal Positions	RPG Field Name
01	I		DS				
02	I			1	10		KEY
03	I			1	2	0	DIV
04	I			3	3	0	REGION
05	I			4	5	0	BRANCH
06	I			6	10		EMPL
07	I			11	28		ARRFLD
08	I			11	15		ID
09	I			11	11		PLANT
10	I			12	15	0	MANNO
11	I			16	19	3	RATE
12	I			20	26	0	HOURS
13	I			27	28		CODE
14	I						

Figure 14-4. Using a Data Structure to Define Subfields within a Subfield

Example 3. Reorganizing Fields in an Input Record

Figure 14-5 shows a data structure that is used to reorganize fields in an input record. Records in file TRANSACT contain input fields in the following sequence: PARTNO, QTY, TYPE, CODE, and LOCATN (lines 02 through 06). Data structure KEYDS reorganizes this sequence. The sequence of subfields in the data structure is LOCATN, PARTNO, and TYPE (lines 08 through 10). Input fields QTY and CODE are not part of the data structure. Subfield PRTKEY (line 11) includes all 16 positions of the data structure, so it allows you to specify the entire data structure in a calculation specification.

Line	Form Type	Filename or Record Name / Data Structure Name	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, **, or DS	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 / Data Structure Occurs n Times	To / 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	TRANSACT				Ø1	1		C	1	2		C	2																
02	I																				3	1Ø		PARTNO						
03	I																				11	16	Ø	QTY						
04	I																				17	2Ø		TYPE						
05	I																				21	21		CODE						
06	I																				22	25		LOCATN						
07	I	KEYDS				DS																								
08	I																				1	4		LOCATN						
09	I																				5	12		PARTNO						
10	I																				13	16		TYPE						
11	I																				1	16		PRTKEY						
12	I																													

Figure 14-5. Using a Data Structure to Reorganize Fields

Special Data Structures

SAVDS Data Structure

The data structure in Figure 14-6 is used as the SAVDS data structure, which is specified on a continuation line of the file description specification for the WORKSTN file. This data structure contains fields that are to be saved and restored for each display station that uses the WORKSTN file.

For more information about the SAVDS data structure, see *Continuation-Line Options* in Chapter 6, *Using a WORKSTN File.* For an example of a SAVDS data structure, see Figure 6-12, Sample Program AR330R.

Line (3-5)	Form Type (6)	Filename or Record Name / Data Structure Name (7-14)	Record Identifying Indicator, **, or DS (19-20)	Field Location From / Data Structure Occurs n Times (44-47)	Field Location To / Length (48-51)	Decimal Positions (52)	RPG Field Name (53-58)
01	I	SAVDS	DS				
02	I			1	5	Ø	CTR
03	I			6	11	2	TOTAL
04	I			12	15	Ø	LINENO
05	I			16	25		KEYSAV
06	I			26	40		NAME
07	I						

Figure 14-6. SAVDS Data Structure

Local Data Area for a Display Station

Figure 14-7 shows a data structure that is used as the local data area for a display station. A local data area contains 512 positions of storage that is used for passing information between programs and procedures. Coding a local data area data structure requires a U in column 18. The data structure name is optional.

At the beginning of the program, the program reads information from the local data area into the data structure; at the end of the program, the program writes information from the data structure into the local data area. You can also use the control language LOCAL statement or another RPG program to enter information into the local data area.

For a MRT (multiple requester terminals) program, the local data area data structure contains a copy of the local data area for the first display station using the program, and the local data area data structure is not automatically written out at the end of the program. To read and write the local data area for each display station in a MRT, use SUBR21 (see Chapter 6, *Using a WORKSTN File*). For an example of a local data area, see Figure 6-13, Sample Program AR935R.

File Information Data Structure

A file information data structure (INFDS) is used for passing information about an exception or error in a WORKSTN file to the RPG program. This information includes the type of exception or error that occurred, the operation that the program was performing when the exception or error occurred, and the status of various conditions. The INFDS data structure is specified as a continuation-line option on the file description specification for a WORKSTN file.

For more information about the file information data structure, see *Handling Exceptions and Errors* in Chapter 6, *Using a WORKSTN File.*

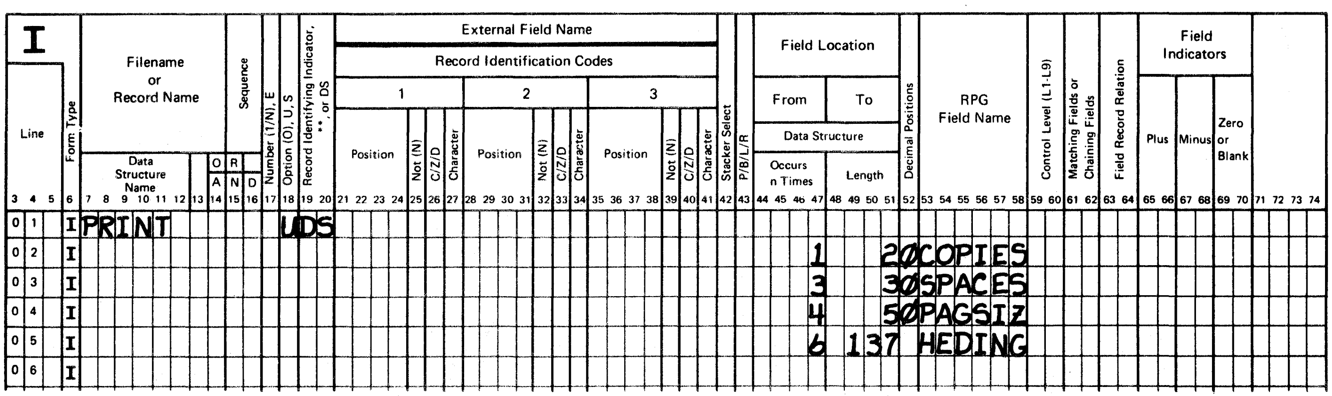


Figure 14-7. Using a Data Structure as a Local Data Area for a Display Station

Chapter 15. Using Auto Report

Chapter 15. Using Auto Report

RPG auto report uses its own specifications and standard RPG specifications to create a complete RPG source program.

Specific auto report statements control the three separate functions of auto report, which can be used in any combination:

- The /COPY statement allows you to copy a source member, containing a group of RPG source specifications, into an RPG source program. Use /COPY so you do not have to repeatedly code identical or nearly identical specifications that are used in several programs. This is the most important benefit of using auto report.
- *AUTO page headings provide a simplified method of coding page headings.
- *AUTO output provides a simplified method of coding output specifications.

Figure 15-1 is an overview of what auto report does.

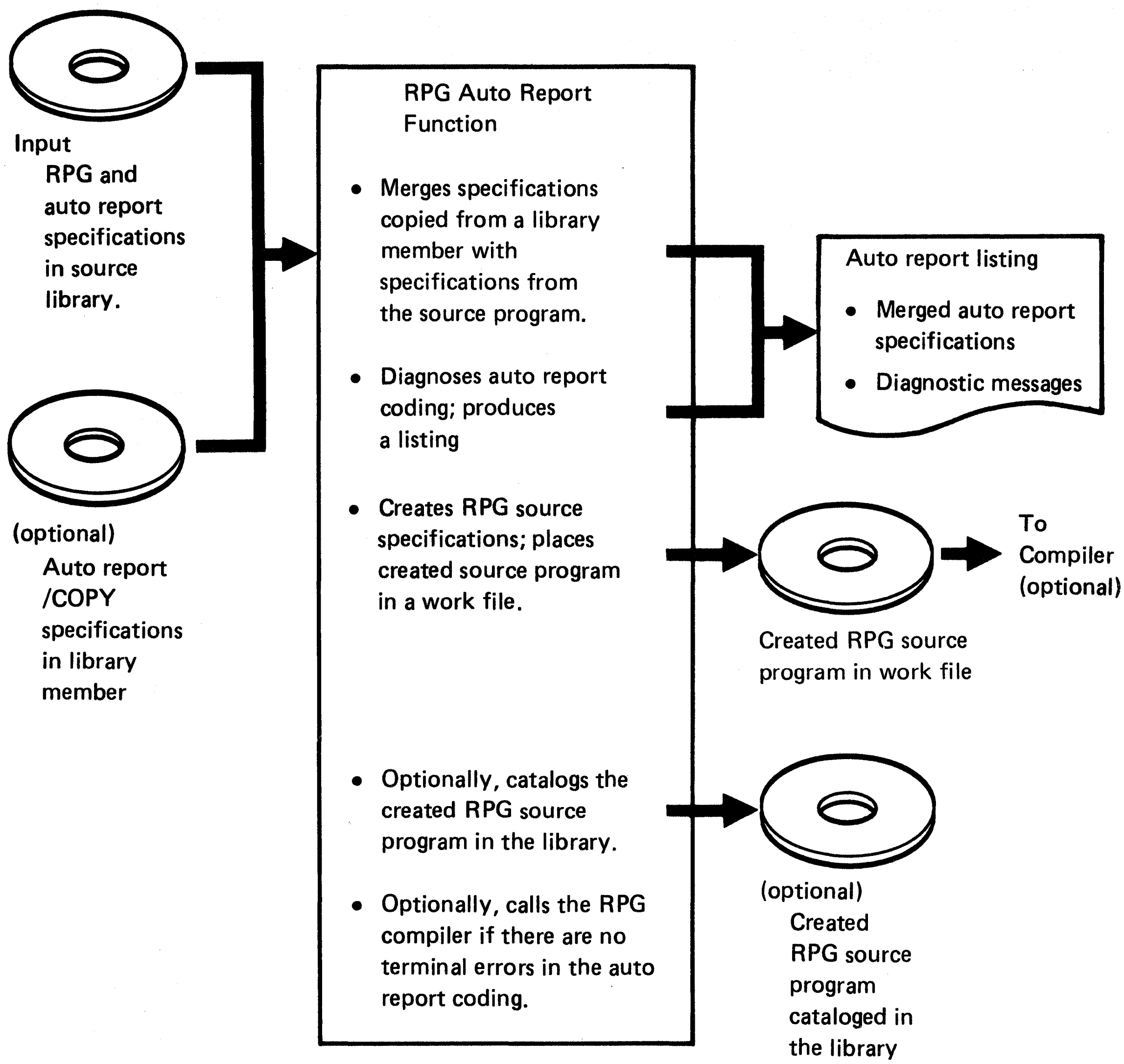


Figure 15-1. Overview of Auto Report

Input for Auto Report

Auto report uses the following input:

- Auto report option specifications
- *AUTO page headings and *AUTO output specifications in the source program
- Standard RPG specifications in the source program
- Auto report /COPY statements in the source program, with or without modifier statements
- Standard RPG specifications, including arrays and tables, and *AUTO specifications that are copied from the library by the auto report copy function

Figure 15-2 shows an example of *AUTO output specifications, and Figure 15-3 shows the calculation and output specifications created by the *AUTO specifications.

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Output Indicators	Field Name or EXCPT Name (*AUTO)	B/A/C/1-9/R	Constant or Edit Word
01	O	PRINTER	H		*AUTO		
02	O						'SALES REPORT'
03	O						'FOR ANY CO.'
04	O		D	Ø1	*AUTO		
05	O			L2	REGION		'REGION'
06	O			L1	BRANCH		'BRANCH'
07	O				ITEMNO		'ITEM'
08	O					C	'NUMBER'
09	O				DESC		'DESCRIPTION'
10	O				SOLDQY		'SALES'
11	O				SOLDVA	A	'AMOUNT'
12	O				ONHAND		'ON-HAND'
13	O				VALUE	A	'VALUE'
14	O					R	'FINAL TOTALS'
15	O						

Figure 15-2. *AUTO Output Specifications That Create the Calculation and Output Specifications Shown in Figure 15-3

```
0140EC    01                 EXSR A$$SUM
0150ECL1         SOLDV2    ADD  SOLDV1    SOLDV2  92
0160ECL1         VALUE2    ADD  VALUE1    VALUE2  92
0170ECL2         SOLDVR    ADD  SOLDV2    SOLDVR  92
0180ECL2         VALUER    ADD  VALUE2    VALUER  92
0190ECSR         A$$SUM    BEGSR
0200ECSR         SOLDV1    ADD  SOLDVA    SOLDV1  92
0210ECSR         VALUE1    ADD  VALUE     VALUE1  92
0220ECSR                   ENDSR
```

Calculations to Roll Totals for SOLDVA and VALUE Fields

```
0230EOPRINTER H 206   1P
0240EO        OR      OA
0250EO                                 45 'SALES REPORT '
0260EO                                 56 'FOR ANY CO.'
0270EO                       UDATE Y    8
0280EO                       PAGE  Z   89
0290EO                                 85 'PAGE '
```

Page Heading (includes date and page number)

```
0300EOPRINTER H 1     1P
0310EO        OR      OA
0320EO                                  6 'REGION'
0330EO                                 14 'BRANCH'
0340EO                                 21 'ITEM'
0350EO                                 36 'DESCRIPTION'
0360EO                                 47 'SALES'
0370EO                                 62 'AMOUNT'
0380EO                                 71 'ON-HAND'
0390EO                                 86 'VALUE'
0400EOPRINTER H 2     1P
0410EO        OR      OA
0420EO                                 22 'NUMBER'
```

Column Headings

```
0430EOPRINTER D 1     01
0440EO                L2     REGION     3

0450EO                L1     BRANCH    12
0460EO                       ITEMNO    23
0470EO                       DESC      40
0480EO                       SOLDQYK   46
0490EO                       SOLDVAKB  62
0500EO                       ONHANDK   69
0510EO                       VALUE KB  86
```

Detail Output Specifications

```
0520EOPRINTER T 12    L1
0530EO                       SOLDV1KB  62
0540EO                       VALUE1KB  86
0550EO                                 87 '*'
0560EOPRINTER T 2     L2
0570EO                       SOLDV2KB  62
0580EO                       VALUE2KB  86
0590EO                                 88 '**'
0600EOPRINTER T 12    LR
0610EO                       SOLDVRKB  62
0620EO                       VALUERKB  86
0630EO                                 47 'FINAL TOTALS'
0640EO                                 89 '***'
```

Total Output Specifications

Figure 15-3. Calculation and Output Specifications Created by the *AUTO Specifications in Figure 15-2

Specifications Created by Auto Report

From input it receives, auto report creates a complete RPG source program that is ready to be compiled.

Format of Created Specifications

The created specifications have the following format:

Column	Contents
1-4	Sequence number of the specification. This number starts at 0010 for the control specification and increases by 0010 for each specification that follows. If the program has more than 999 specifications, the sequence starts again at 0000.
5	Code that identifies the specification as follows: Blank — Standard RPG specification in the auto report program C — Specification copied from the library member specified in the /COPY statement M — Specification copied from the library member specified in the /COPY statement and modified E — Specification created by auto report
6-80	Standard RPG specification

Compile-time arrays and tables are not changed by auto report; they remain in the standard format for an array or table record.

Order of Created Specifications

Auto report creates the specifications in the order required by the RPG compiler. When specifications are included by means of a /COPY statement, those specifications are placed immediately after the /COPY statement. Then, after all specifications are copied but before auto report creates RPG specifications from the H-*AUTO (heading records) and D/T-*AUTO (detail/total records) specifications, the entire auto report source program is sorted into the following order:

1. Control specification
2. File description specifications
3. Extension specifications
4. Line counter specifications
5. Telecommunications specifications
6. Input specifications
7. Calculation specifications (in the following order: detail, L0, L1 through L9, LR, and subroutines)
8. Output specifications
9. Arrays and tables loaded at compilation time, which must be placed last among the input statements to auto report

Calculation Specifications

Auto report places the created calculation specifications in the following order:

1. Detail calculations that you code
2. EXSR statement for the created subroutine
3. Total calculations created by auto report, grouped in order by level (all L0 calculations, then all L1 calculations, and so on)
4. Total calculations that you code
5. Subroutines that you code
6. Created RPG subroutine that accumulates the lowest-level total

Note: If the /COPY statement copies a subroutine that contains a BEGSR operation but no ENDSR operation, incorrect sorting of the created RPG source program can result. Incorrect sorting can also result if invalid characters are coded as control-level indicators in columns 7 and 8.

Output Specifications

Output heading specifications created for H-*AUTO specifications appear in the order in which they are coded on the output specifications in relation to other RPG and *AUTO output specifications for the file.

Normally, RPG output specifications created from a D/T-*AUTO specification are in the following order:

1. Heading specifications created for column headings

2. Detail specifications

3. Total specifications, with the lowest level first and LR last

This group of specifications is placed in the same relative position in the program as the original D/T-*AUTO specification. All other RPG output specifications remain in their original order.

However, if you specify a normal RPG total output specification conditioned by a positive control-level indicator (no N in column 23) in columns 24 and 25 for the file that has a D/T-*AUTO specification, all output specifications in the program are sorted into the following format:

1. All heading, detail, and exception output specifications remain in the order in which they are coded in the created source program. Total specifications that are not conditioned by a positive control-level indicator in columns 24 and 25 remain as they were in the program.

2. Total specifications that are conditioned by a positive control-level indicator in columns 24 and 25 are sorted into ascending order according to the control-level indicator in columns 24 and 25, with LR last.

See *Examples of Using Auto Report* in this chapter for examples of created specifications.

Comment Statements

Comment statements (identified by an asterisk in column 7) are allowed among the statements read by auto report. However, the sorting of RPG specifications is based on the contents of column 6; therefore, comments may be sorted into an unexpected order. To ensure that comments remain with the correct specification, place them after that specification and put the same entry in column 6.

Restriction

The order of arrays and tables does not change when the source specifications are sorted. Therefore, when arrays and tables are included from a library member, they may occur in an incorrect order after the sort. For example, if the auto report source specifications contain a table for translating a file or for changing the collating sequence of characters, then any compile-time arrays or tables included from a library member are out of order. That is, the included arrays or tables are placed ahead of the table for translating files. Compile-time arrays and tables must be loaded in the following order:

1. Tables for translating files

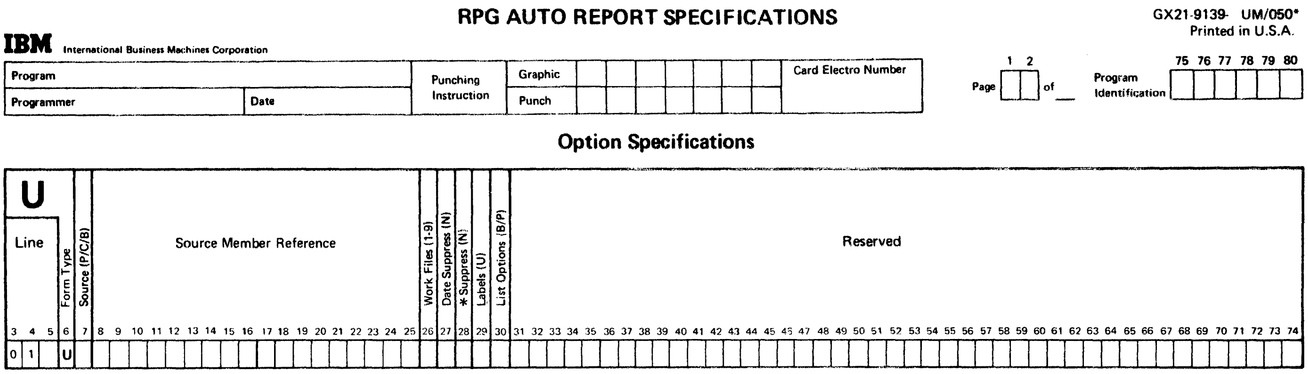
2. Tables for changing the collating sequence of characters

3. Compile-time arrays and tables in the order described on the extension specifications

A solution to this restriction is to place the tables for translating files and for changing the collating sequence of characters in a library member, and then to copy them from the library member before any other compile-time tables and arrays are copied. This procedure ensures that the tables for translating files and for changing the collating sequence of characters are the first compile-time tables in the created RPG source program.

Option Specifications

Specify options for the auto report program on the RPG auto report specifications shown below:


RPG AUTO REPORT SPECIFICATIONS
GX21-9139- UM/050*
Printed in U.S.A.
IBM International Business Machines Corporation
Program
Programmer
Date
Punching Instruction
Graphic
Punch
Card Electro Number
Page 1 2 of
Program Identification 75 76 77 78 79 80
Option Specifications
U
Line
Form Type
Source (P/C/B)
Source Member Reference
Work Files (1-9)
Date Suppress (N)
*Suppress (N)
Labels (U)
List Options (B/P)
Reserved
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
0 1 U

The auto report option specifications are not required in the auto report program. If present, they must appear as the first specifications in the program. If they are not present, auto report assumes the options that correspond to blank entries (see individual entries for the meanings of the blank entries). Option specifications cannot be contained in a library member that is copied by a /COPY statement.

If a control specification (H in column 6) is not present either in the auto report source program or in a copied library member (see */COPY Statement Specifications)*, auto report creates a control specification with blank entries.

The following columns on the auto report specifications are used in the same way as corresponding columns on other RPG specifications:

- Columns 1-2 (page)
- Columns 3-5 (line)
- Columns 75-80 (program identification)

Column 6 (Form Type)

Enter a U in column 6 to identify this line as an auto report option specification.

Column 7 (Source)

Entry	Explanation
Blank	The created source program is not cataloged.
C	The created source program is cataloged in a library on disk.

Use column 7 to specify whether the created source program is to be cataloged in a library. Whether or not the source program is cataloged, the created source program is written to a disk work file from which it is immediately compiled. Created source programs that are cataloged become library source members.

The created source program is not cataloged when terminal errors exist in the auto report specifications.

Columns 8-24 (Source Member Reference)

Entry	Explanation
Library, member	Identifies the library member to be cataloged. The library name can be up to 8 characters long, beginning in column 8. Use a comma after the library name. The member name can also be up to 8 characters long.

Make an entry in columns 8 through 24 if the created source program is to be cataloged in a library (C in column 7). The first character of the library name and of the member name must be alphabetic. The remaining characters can be alphabetic or numeric.

If you enter F1 or blanks for the library name, the library name defaults to the system library. If the member name is not specified or is specified incorrectly, an error results.

If the name used by auto report to catalog the created source program is the same as the name of an existing member in the library, the old member is replaced by the new member.

Columns 25-26

Columns 25 and 26 are not used. Leave them blank.

Column 27 (Date Suppress)

Entry	Explanation
Blank	Date and page number are printed on the first *AUTO page heading line.
N	Date and page number are not printed on the first *AUTO page heading line.

To prevent the date and page number from printing on the first *AUTO heading line, enter N in column 27. When these fields are suppressed, the page title and any other fields specified can occupy the entire line. See **AUTO Page Heading Specifications* for further information on the date and page numbers.

Column 28 (*Suppress)

Entry	Explanation
Blank	Asterisks are printed for total output lines.
N	Asterisks are not printed for total output lines.

To prevent asterisks from printing beside created totals, enter N in column 28. See **AUTO Output Specifications* for rules used in printing asterisks.

Column 29

Column 29 is not used. Leave it blank.

Column 30 (List Options)

Entry	Explanation
Blank	The source program listing, headings, and diagnostics are printed. A source program is also produced if no terminal errors are found.
B	The program listing is not printed; however, a source program is produced.
P	A partial program listing is printed that includes appropriate headings and diagnostics.

Column 30 provides options for printing a source listing when auto report creates RPG source specifications. If any terminal errors are found in auto report specifications, the listing is completed (provided a listing is to be printed).

The auto report source listing consists of the RPG specifications included in the input to auto report, RPG specifications created by auto report, and specifications copied from a library member.

Use the B entry to produce a source program for which you already have a listing.

Use the P entry to determine whether minor changes to a previously tested program created any errors.

Columns 31-74

Columns 31 through 74 are not used. Leave them blank.

/COPY Statement Specifications

The auto report copy function provides a way to include cataloged RPG source specifications in an RPG program. The source specifications that are included must reside as a library member on disk. Use the copy function to include source specifications that are identical or nearly identical in several different programs, thereby reducing the need to repeatedly code specifications that are used in several programs. For example, if file description and input specifications for a particular file are similar in different programs, these specifications can be placed in the library by the source entry utility (SEU) and included in any program by the copy function.

Auto report specifications and any valid RPG specifications, including arrays and tables, can be copied in this manner. The auto report option specifications and other copy statements cannot be copied. See *Examples of Using Auto Report* in this chapter for an example of using the copy function.

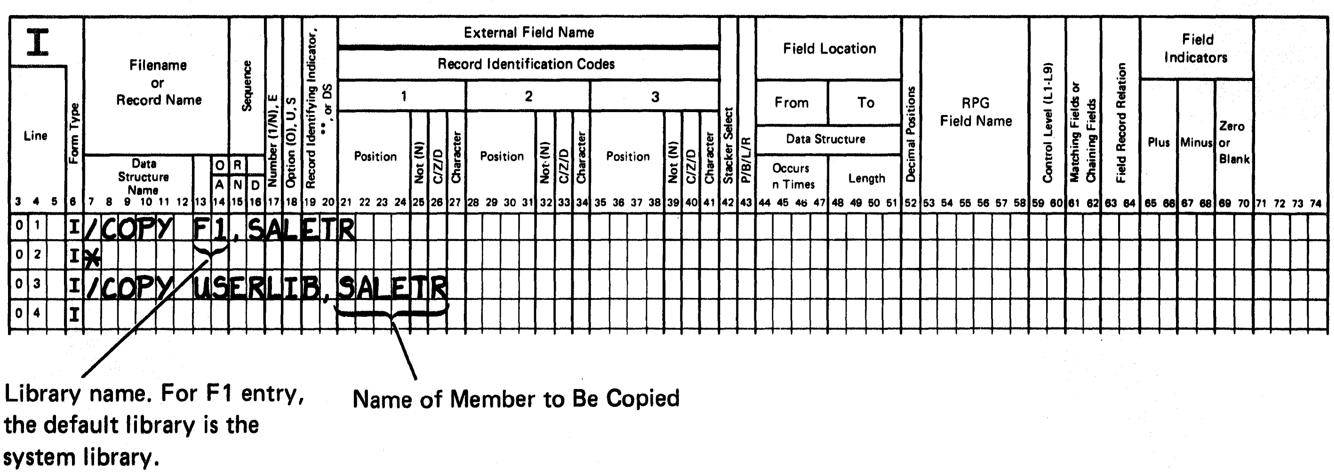
The specifications included in an auto report program by the copy function are first placed in the program immediately following the /COPY statement. After all specifications are copied from the library members, the entire auto report program is sorted into the order required by the RPG compiler (see *Order of Created Specifications* in this chapter). Note that the auto report compiler truncates any record that is longer than 96 characters.

To request the copy function, use the /COPY statement. This statement identifies the library and library member containing the RPG specifications to be included in the source program created by auto report. /COPY statements must follow the auto report option specifications, and they must precede source tables for translating files, tables for changing the collating sequence of characters, and compile-time array and table data.

The format of the /COPY statement is:

Column	Entry
1-5	Page and line number indicating the placement of the statement in the sequence of auto report source specifications.
6	This column can contain any entry except H or U, or can be blank.
7-11	Enter the characters /COPY.
12	Blank.
13-29	Identifies the library and member to be included. Specify the library name, which can be up to 8 characters long, beginning in column 13. Use a comma to separate the library name from the member name, which can also be up to 8 characters long. If you do not enter a library, or if you enter F1, the default is the system library.
30-49	Blank.
50-80	Enter any information or comments. The contents of these columns are not read by auto report.

Figure 15-4 shows an example of the /COPY statement.



Note: It is convenient to code the /COPY statement on the input specifications if the input specifications are to be changed as they are copied.

Figure 15-4. Example of the /COPY Auto Report Statement

Changing Copied Specifications

Statements can be included in the auto report specifications to change file description and input field specifications as they are copied from a library member. No other types of specifications can be changed. /COPY modifier statements from the source program that add, change, or delete entries on cataloged input field specifications are identified by an X in print position 6 of the auto report listing.

Changing File Description Specifications

To change a file description specification that is copied from a library member, enter the filename in columns 7 through 14 of a file description specification. Then make only those entries on the line that are to replace existing entries in the copied specification or that are to be included as new entries. Blank entries in the modifier statement do not affect the copied statement.

For example, the file description specifications for a frequently used file named SALES are to be copied from the system library. The original specification contains I in file type (column 15), defining SALES as an input file (see Figure 15-5). To update the sales file, change column 15 to a U by including in the auto report source program a modifier file description specification that contains the filename, SALES, and the new file type entry, U. As a result of the modifier file description specification, the file type on the copied file description specification is changed from I to U.

```
01 I/COPY F1,SALETR
02 I
```

/COPY Statement to Copy Specifications for SALES File from the System Library Member Named SALETR

```
02 FSALES   IP  F 473  43            DISK
03 F
```

File Description Specification as It Is in the System Library

```
02 FSALES   U
03 F
```

Copy Function Modifier Statement

```
02 FSALES   UP  F 473  43            DISK
03 F
```

Resulting File Description Specification That Is Included in the RPG Source Program

Figure 15-5. Changing a Copied File Description Specification

To set an entry to blanks, enter an ampersand (&) in the first position of that entry on the modifier statement, and leave the remaining positions blank. For example, to remove the block length entry (columns 20 through 23) from the cataloged specification shown in Figure 15-5, add an ampersand to the modifier statement in column 20, as shown in Figure 15-6, and leave columns 21 through 23 blank.

Modifier statements for file description specifications do not have to be in any particular order in the auto report source program, except that they cannot immediately follow the /COPY statement if input field specifications are also being changed.

Only one file description specification with a particular filename is allowed to come from the library entries, and a particular filename can be used only once on a modifier statement.

No changes are allowed to the file description continuation specifications that accompany a copied file description. To add new continuation specifications, place them after a file description modifier statement for the file. A maximum of five continuation specifications are allowed to follow a file description specification (combined total of original and added continuation specifications).

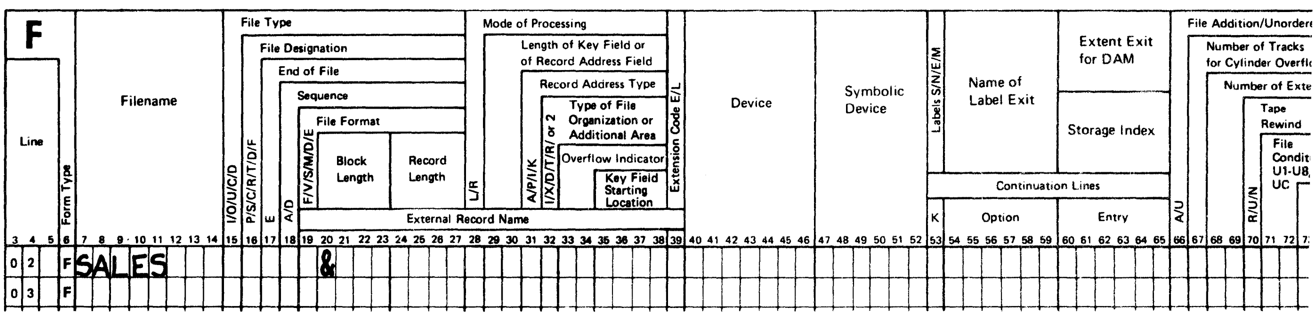


Figure 15-6. Setting a Copied File Description Entry to Blank

Changing Input Field Specifications

Only input field specifications (specifications describing individual fields on the input record) can be changed. To change an input field specification copied from a library member, enter the field name in columns 53 through 58 of an input field modifier statement (I in column 6). Modifier statements for input field specifications must immediately follow the /COPY statement in the auto report program that copies those specifications. The first specification following the /COPY statement that is not an input field specification is considered the end of the input field modifier statements for the /COPY statement. (A comment statement with I in position 6 is not considered the end of the input field modifier statements.)

The fields that can be modified are:

- Column 43 (packed/binary)
- Columns 44-51 (field location)
- Column 52 (decimal positions)
- Columns 59-60 (control levels)
- Columns 61-62 (matching or chaining fields)
- Columns 63-64 (field record relation)
- Columns 65-70 (field indicators)

The method of replacing, adding, or blanking entries is similar to the method used to change file description specifications. To replace or add entries, code the new entry in the proper location in the modifier statement; to set an entry to blank, place an ampersand (&) in the first position of that entry in the modifier statement. Figure 15-7 shows examples of changing input specifications.

The modifier statement changes all copied input field specifications that have the same field name. If there is no input field by the same name, the modifier statement is added to the program as a new input field specification. Modifier statements with duplicate field names are allowed (length and number of decimal positions must also be the same), but only the first is used to change a copied specification. Other field names are added as new input field specifications. Up to 20 input field modifier statements are allowed per /COPY statement.

For best results, those statements that change existing input field specifications should come first; then those that are to be added as new input field specifications. This order is suggested because input field modifier statements that do not fit into the special main storage table for modifier statements are added to the RPG source program as new input field specifications. This order of specifying modifier statements increases the likelihood that excess statements, if any, will be valid field descriptions.

Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator	Field Location From	Field Location To	Decimal Positions	RPG Field Name	Field Indicators Plus	Field Indicators Minus	Field Indicators Zero or Blank
01	I	SALES	AA	Ø1							
02	I				1	7		ITEMNO			
03	I				8	9		BRANCH			
04	I				1Ø	1Ø		REGION			
05	I				11	25		DESC			
06	I				26	27	Ø	SOLDQY			
07	I				28	34	2	SOLDVA		13	
08	I				35	36	Ø	ONHAND			
09	I				37	43	2	VALUE			
10	I										

Input specifications as in the library member.

Line	Form Type	Filename or Record Name	Field Location From	Field Location To	RPG Field Name	Control Level (L1-L9)	Field Record Relation	Field Indicators Minus
01	I	/COPY F1,SALETR					**1**	
02	I				BRANCH	L1		
03	I				SOLDVA		**2**	&
04	I		1	43	RECORD		**3**	
05	I							

/COPY statement and modifier statements:

1 Add an entry to BRANCH field description

2 Blank out minus field indicator on SOLDVA description

3 Add a new file description

Figure 15-7 (Part 1 of 2). Changing Copied Input Field Specifications

Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator	From	To	Decimal Positions	RPG Field Name	Control Level (L1-L9)
01	I	SALES	AA	Ø1					
02	I				1	7		ITEMNO	
03	I				8	9		BRANCH	L1
04	I				1Ø	1Ø		REGION	
05	I				11	25		DESC	
06	I				26	27	Ø	SOLDQY	
07	I				28	34	2	SOLDVA	
08	I				35	36	Ø	ONHAND	
09	I				37	43	2	VALUE	
10	I				1	43		RECORD	
11	I								

Resulting input specifications for SALES file showing:

1 Added L1 indicator

2 Blanks in place of minus field indicator

3 Added field description

Figure 15-7 (Part 2 of 2). Changing Copied Input Field Specifications

*AUTO Specifications

The *AUTO page-heading function and the *AUTO output function provide simplified methods of describing printed output. These functions are requested when the characters *AUTO are present in columns 32 through 36 of a record description specification on the standard RPG output specifications. *AUTO can be entered on a heading, detail, or total specification (H, D, or T in column 15), but not on an exception output specification (E in column 15). Use *AUTO with only one PRINTER file in the program.

Standard RPG output specifications are divided into two general types (see Figure 15-8):

- *Record-description specifications* (columns 7 through 31) describe when and where the output line is to be printed. One record-description specification is required for each type of line to be printed. Only the first record-description for a file need contain a filename in columns 7 through 14.

- *Field-description specifications* (columns 23 through 74) following a record-description specification tell when, where, and how each item of data (field or literal) is to be printed on the output record. There can be several field-description specifications following a record-description specification.

Auto report page headings and auto report output specifications are also divided into the same two types: record-description specifications and field-description specifications. However, the entries on these specifications are used differently from the entries on the standard RPG specifications.

The following output specifications are not changed when they are used with *AUTO:

- Columns 1-2 (page)

- Columns 3-5 (line)

- Column 6 (form type)

- Columns 75-80 (program identification)

Columns 71 through 74 must always be blank on auto report output specifications.

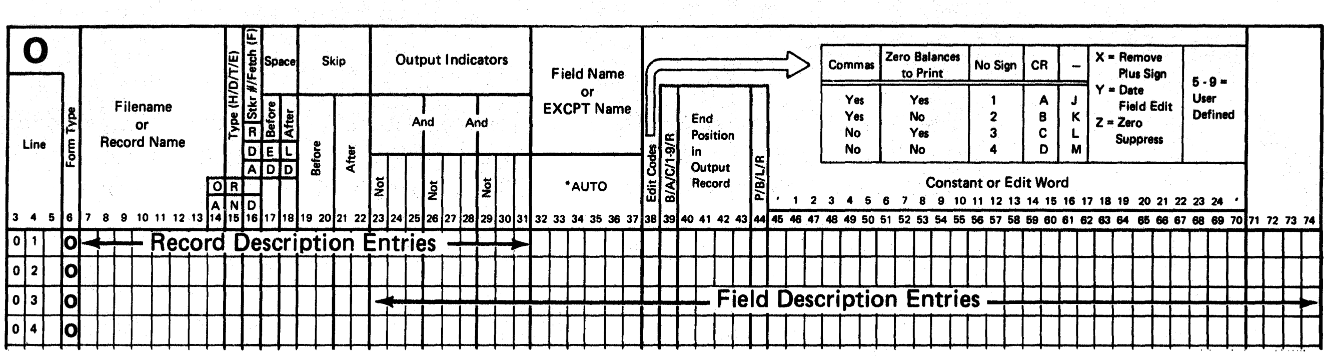


Figure 15-8. Two Types of Output Specifications

*AUTO Page-Heading Specifications

The *AUTO page-heading specifications provide an easy way to produce a page heading at the top of every page of a printed report (see Figure 15-9). Up to five *AUTO page-heading specifications can be used for a page heading. If both standard RPG heading lines and *AUTO page headings are specified in combination for a file, they are printed in the order specified by the output specifications. The *AUTO page headings can be specified for only one PRINTER file per program.

The heading line created by the first *AUTO page heading (H-*AUTO) specification contains a date and page number. The first heading line can also contain a title. (See *Field-Description Specifications* in this section for information on entering a title.)

The created date is printed in the leftmost columns in the format mm/dd/yy unless you change the format by using the date or inverted-print option (columns 19 through 21 of the control specification).

The created page number is printed in the rightmost columns and is preceded by the word PAGE. The page number field is 4 digits long and is zero suppressed. Auto report uses one of the unused PAGE fields (PAGE, PAGE1 through PAGE7) for page numbering. If all PAGE fields are used in the program, auto report does not number pages.

To suppress the date and page number on the first heading line, enter N in column 27 of the auto report option specifications.

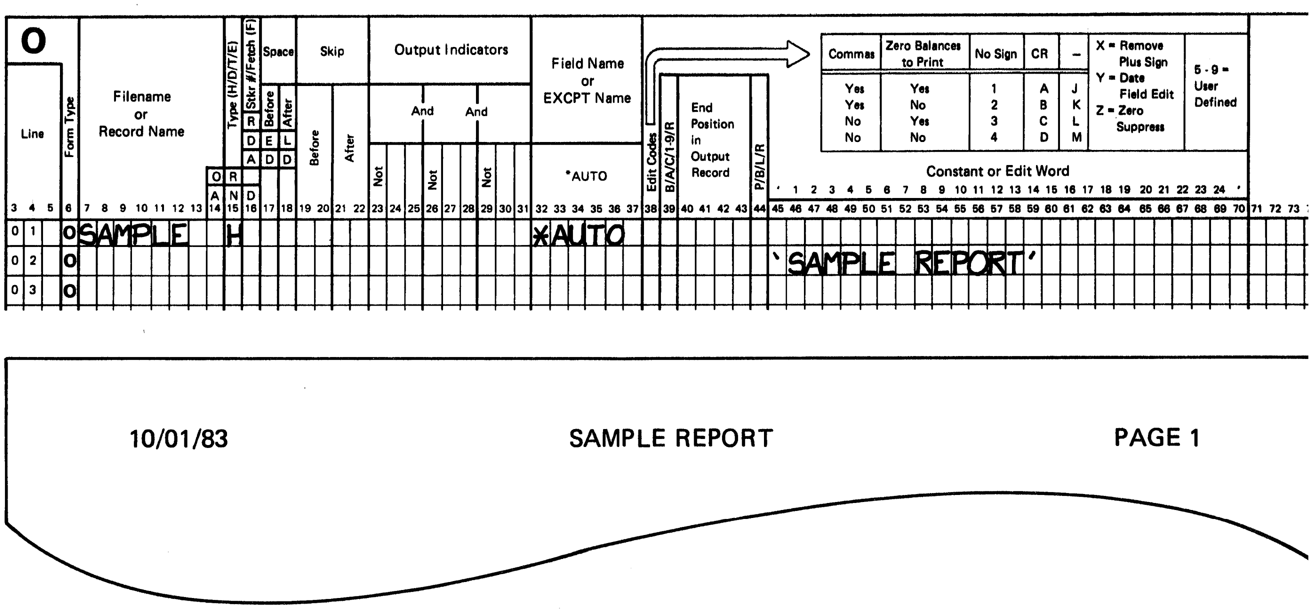


Figure 15-9. *AUTO Specification and the Heading Line That Is Printed

Record-Description Specifications

Each H-*AUTO record description defines a separate heading line. The record-description entries allow you to specify spacing, skipping, and the conditions under which the line is printed.

Columns 7-14 (Filename)

Enter the name of the PRINTER file on which the heading is to be printed.

Column 15 (Type)

Enter H in column 15 on each record-description specification line that defines a page-heading line. The H and the entry *AUTO in columns 32 through 36 define this as an H-*AUTO heading specification (see Figure 15-9). Up to five H-*AUTO specifications are allowed.

Column 16

Column 16 is not used. Leave it blank.

Columns 17-22 (Spacing and Skipping)

Enter spacing and skipping values in these columns according to the rules given under *Columns 17-22 (Spacing and Skipping)* in Chapter 27. If these columns do not contain spacing and skipping values, auto report skips to line 06 before the first line is printed and spaces two after the last H-*AUTO line is printed. If multiple H-*AUTO lines are used, auto report spaces one after each line except the last. For additional information on created spacing and skipping values, see *Report Format* in this chapter.

Columns 23-31 (Output Indicators)

On the first H-*AUTO specification, either leave columns 23 through 31 blank or enter output indicators according to the rules given under *Columns 23-31 (Output Indicators)* in Chapter 27.

If these columns are blank, auto report causes the corresponding output line to be printed at first-page (1P) time in the program cycle and when overflow occurs. Thus, the heading is printed at the top of each page of the printed report. Indicators can be assigned to subsequent H-*AUTO specifications. If columns 23 through 31 are blank on any H-*AUTO specification after the first, that specification is assigned the same indicators as the first.

If an overflow indicator is specified on the file description specifications for the PRINTER file, that indicator conditions the created heading specifications. Otherwise, auto report defines an unused overflow indicator for the PRINTER file and conditions the line with that indicator.

AND and OR lines can be used with H-*AUTO output indicators if an output indicator is used with the first specification. Standard RPG rules for AND and OR lines apply.

Columns 32-37 (*AUTO)

Enter *AUTO in columns 32 through 36. This entry and an H in column 15 of the output specifications (see Figure 15-9) indicate that this is an auto report heading line.

Columns 38-70

Columns 38 through 70 are not used on the record-description line. Leave them blank.

Field-Description Specifications

Each H-*AUTO record-description specification can be followed by one or more field-description specifications. The field-description specifications specify the title to be printed on the heading line and describe any other fields and literals to be printed on the line.

Columns 7-31

Columns 7 through 31 are not used on field-description specifications. Leave them blank. Output indicators in columns 23 through 31 cannot be used to condition a field on an H-*AUTO specification.

Columns 32-37 (Field Name)

Entry	Explanation
Blank	A constant (enclosed in apostrophes) must be entered in columns 45 through 70. The constant is printed on the heading line.
Field name	Field defined in the program is printed on the heading line.
Table name	A table item is printed on the heading line.
Indexed array name	An array item is printed on the heading line.

Use columns 32 through 37 to enter a field name, a table name, or an indexed array name (defined elsewhere in the program) that is to print on the heading line. If a name is entered, an edit word, not a constant, can be entered in columns 45 through 70. A constant must be entered in columns 45 through 70 if columns 32 through 37 are blank.

If output indicators (columns 23 through 31) are left blank on the record-description specification, auto report conditions all fields and all array or table items included on the heading line with N1P in columns 23 through 25. Therefore, the field or the array or table item does not print on the first page. (If printed on the first page, the field might contain meaningless data because the first record is not read.) N1P is not created for the following RPG reserved words: PAGE, PAGE1 through PAGE7, UDATE, UDAY, UMONTH, UYEAR.

For information on formatting and centering *AUTO heading lines, see *Report Format* in this chapter.

Column 38 (Edit Codes)

An edit code can be entered in column 38 if a numeric field, numeric array item, or numeric table element is named in columns 32 through 37. If an edit code is used, columns 45 through 70 must be blank unless asterisk fill or a floating currency symbol is specified. If column 38 is blank, no editing is done by auto report unless an edit word is used.

Column 39 (Blank After)

Enter B in column 39 to reset a numeric field to zeros after it is printed or to reset an alphameric field to blanks after it is printed on the heading line.

Columns 40-44

Columns 40 through 44 are not used with *AUTO heading specifications. Leave them blank.

For information on the positioning of fields and constants in the title line and on the centering of heading lines in relation to the body of the report, see *Report Format* in this chapter.

Columns 45-70 (Constant or Edit Word)

Entry	Explanation
Blank	Columns 32 through 37 contain the name of a field that either is not edited or is edited by an edit code.
Constant	Title or other constant (enclosed in apostrophes) that is to appear on the printed line.
Edit word	The edit pattern used to edit the numeric field named in columns 32 through 37 of the same field-description line.

Use columns 45 through 70 to specify the title and other information that is to appear on the output line and to edit numeric fields that are to appear on the line. Rules for specifying constants and edit words are identical to those given under *Columns 45-70 (Constant or Edit Word)* in Chapter 27, except that no end positions can be specified.

For information on the positioning of fields and constants in the title line and on the centering of heading lines in relation to the body of the report, see *Report Format* in this chapter.

*AUTO Output Specifications

Detail reports (in which a line is printed for each record that is read) and group printed reports (in which only totals are printed) can be specified by the *AUTO output function alone or in combination with standard RPG specifications. The *AUTO output function creates totals and formats columns and column headings.

A single detail or total *AUTO record description (D/T-*AUTO) specification and its associated field-description specifications can specify:

- Up to three lines of column headings to appear above a field
- Accumulation of several levels of totals, including a final total (known as *total rolling*)
- Creation by auto report of end positions for column headings and fields
- Creation by auto report of the K edit code for numeric fields
- Fields or constants to be printed next to created totals

Four types of description specifications can be associated with the *AUTO record-description specification. The four types are distinguished by entries in column 39. The remaining entries on a field-description specification have different meanings, depending on the entry in column 39.

The valid entries in column 39 of the field-description specifications and their meanings are:

- *Blank or B*: Indicates that the associated field or constant appears on the detail line.
- *A*: Indicates that the associated numeric field is printed on the detail line and is accumulated. A total is printed for each control level defined in columns 59 and 60 of the input specifications for the program. A final total is also printed (when the LR indicator is on).
- *C*: Indicates that the associated constant is printed on the second or third line of column headings.
- *1, 2, 3, 4, 5, 6, 7, 8, 9, R*: Indicates that the associated field or constant appears on the total line created for the respective control-level indicator (L1 through L9, LR).

See *Group Printing* in this chapter for the effect of these entries in a group printed report.

See *Examples of Using Auto Report* in this chapter for examples of the four types of field descriptions.

Record-Description Specifications

An auto report record-description specification must contain the entry *AUTO in columns 32 through 36. *AUTO can appear only on a record-description specification. This entry indicates that the record description and the following field descriptions are redefined according to their use by auto report.

Columns 7-14 (Filename)

Enter the name of the PRINTER file on which the report is to be printed. This must be the same file named on H-*AUTO specifications, if any.

Column 15 (Type)

Entry	Explanation
D	The auto report specifications describe a report containing detail lines.
T	The auto report specifications describe a report containing total lines, but no detail lines (group-printed report).

Enter D in column 15 and *AUTO in columns 32 through 36 for auto report to create a report that contains detail lines. The field-description specifications associated with the D-*AUTO record description specify:

- Fields to appear on the detail line
- Column headings
- Total rolling
- Constants to appear on total lines

See *Examples of Using Auto Report* in this chapter for examples of D-*AUTO specifications.

Enter T in column 15 and *AUTO in columns 32 through 36 for auto report to create a group printed report (see *Group Printing* in this chapter).

Only one detail or one total *AUTO (D/T-*AUTO) record description specification can be used in a program.

Column 16 (Fetch Overflow)

Enter F in column 16 to specify fetch overflow. See *Column 16 (Fetch Overflow)* in Chapter 27 for the rules on using fetch overflow.

When used with the *AUTO output function, fetch overflow applies only to the detail line. If group printing is specified (T in column 15), fetch overflow applies to the lowest-level total line to be printed.

Columns 17-22 (Spacing and Skipping)

Enter spacing and skipping values in columns 17 through 22 according to the standard RPG rules. Entries specified apply only to the detail line created by a D-*AUTO specification or to the first total line created by a T-*AUTO specification.

Leave columns 17 through 22 blank to single space after each detail line printed or, if group printing is specified, after the first total line printed. For information on spacing and skipping for created column heading and total lines, see *Report Format* in this chapter.

Columns 23-31 (Output Indicators)

Enter any valid output indicators in columns 23 through 31 to condition the detail or group-print line created by this *AUTO specification. If these columns are left blank on a D-*AUTO specification, the created detail line is conditioned by N1P. Therefore, it is not printed at first-page (1P) time in the RPG program cycle. If these columns are left blank for a T-*AUTO specification, the first created total line is conditioned by the lowest control-level indicator defined in the program. (See *Group Printing* for additional information about the use of this entry with a T-*AUTO specification.)

AND and OR can be used with *AUTO output indicators if an output indicator is specified on the first record-description specification. Standard RPG rules for AND and OR lines apply.

Indicators specified in columns 23 through 31 of the record-description specification (and its associated AND/OR lines) apply only to the detail line created by a D-*AUTO specification or to the group-print line (lowest-level total specification) created by a T-*AUTO specification.

If column headings are specified in the field-description specifications that follow this *AUTO record description, they are conditioned by one of the following:

- The same indicators that are specified for the first H-*AUTO specification.

- The first-page (1P) indicator in an OR relationship with the overflow indicator specified for the file on the file-description specifications. If no overflow indicator is specified, auto report defines an unused overflow indicator and uses it to condition the lines.

Restriction: If N1P is specified on a D-*AUTO record-description specification that is followed by field-description specifications for totaling fields (A in column 39), the calculations created for the totaling fields are also conditioned by N1P. This causes a terminal diagnostic in the RPG compiler.

Columns 32-37 (*AUTO)

To indicate that this is an auto report specification, enter *AUTO in columns 32 through 36 on the record-description line. Column 15 must contain D or T to indicate a detail or total *AUTO specification. Only one D/T-*AUTO specification can be used in a program.

Columns 38-70

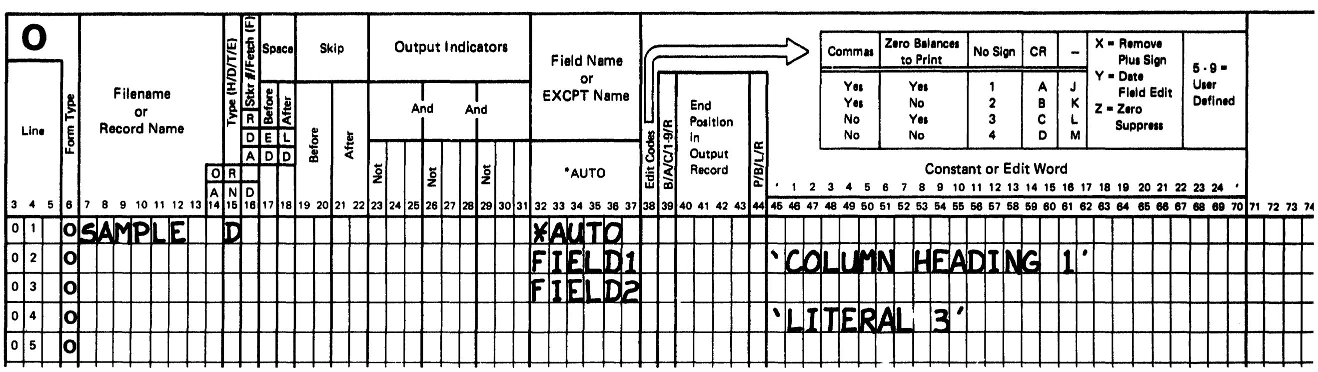
Columns 38 through 70 are not used on a D/T-*AUTO record-description specification. Leave them blank.

Field Description (Blank or B in Column 39)

D-*AUTO and T-*AUTO field-description specifications containing a blank or B in column 39 describe:

- An alphameric field such as an item description
- A numeric field that is not totaled
- A constant
- A field with a literal to be used as a column heading (see Figure 15-10)

A field named on the line (or a constant when no field is named) following a D-*AUTO record-description specification is printed only on the detail report line. If the field (or constant when no field is named) on the line follows a T-*AUTO record description, it appears only on the first total line created.



As a result of these specifications, FIELD1 prints on each detail line under the heading COLUMN HEADING 1. FIELD2 and LITERAL 3 print on each detail line without a column heading.

Figure 15-10. Auto Report Field-Description Specifications (Blank in Column 39)

Columns 7-22

Columns 7 through 22 are not used on the field-description lines. Leave them blank.

Columns 23-31 (Output Indicators)

Enter any valid output indicators in columns 23 through 31, or leave them blank. If these columns are left blank, the field (or constant when no field is named on the line) is printed on each detail line conditioned by the indicators for that record. When group printing is specified (T-*AUTO specification), the field (or constant when no field is named on the line) is printed each time the lowest-level total line is printed. If a column heading is specified in columns 45 through 70 to appear over a field named in columns 32 through 37, the column heading is not affected by output indicators entered in columns 23 through 31.

Columns 32-37 (Field Name)

Enter a field name, data structure name, indexed array name, table name, or blanks in columns 32 through 37. If columns 32 through 37 are blank, a constant must be entered in columns 45 through 70 of the same field-description specification. If a field name, data structure name, indexed array name, or table name is entered, the value of the field or item is printed on the detail line (or on the first total line if group printing is specified).

Column 38 (Edit Codes)

Enter a valid edit code in column 38 if columns 32 through 37 contain the name of a numeric field, a numeric array item, or a numeric table. This column must be blank for alphameric fields, data structures, array items, table items, and literals. If column 38 is left blank on a field-description line for a numeric field, an array item, or a table item, the auto report program provides a K edit code. The K edit code causes a numeric field or item to be printed with commas and a decimal point, such as 3,489.13. It also causes zero suppression, does not print zero balances, and prints a minus sign on the right of negative balances.

Column 39 (Blank After)

Entry	Explanation
Blank	Field is not to be reset to zeros or blanks after printing.
B	Numeric field is reset to zeros after it is printed. Alphameric field is reset to blanks after it is printed.

Enter B in column 39 to reset alphameric fields or data structures to blanks or to reset numeric fields to zeros after they are printed. Blank after cannot be used for constants. This entry applies only to the detail line (or to the first total line if group printing is specified).

Columns 40-43 (End Position in Output Record)

Either leave columns 40 through 43 blank, or enter the print position of the rightmost character of the field (or constant if no field is named in columns 32 through 37) to be printed. If this column is blank, auto report creates end positions for fields, constants, and column headings. See *Report Format* in this chapter for additional information and considerations.

Column 44

Column 44 is not used, because packed-decimal and binary data cannot be specified. Leave this column blank.

Columns 45-70 (Constant)

Enter a constant or blanks in columns 45 through 70 when column 39 contains a blank. Constants are enclosed in apostrophes according to the standard RPG rules for coding constants. If these columns are left blank, a field name, data structure name, indexed array name, or table name must be entered in columns 32 through 37. Column-heading continuation lines can follow this field-description line, but the first line of the printed column heading will be blank. See *Field Description (C in Column 39).*

If a constant is entered in these columns along with a field name in columns 32 through 37, the constant is printed on the first column-heading line over the field value. When a column heading is used, the length used to space the column on the report is the greater of the longest column-heading length or the field length, adjusted for editing. See *Report Format* in this chapter for additional information on how columns and fields are centered and spaced by auto report.

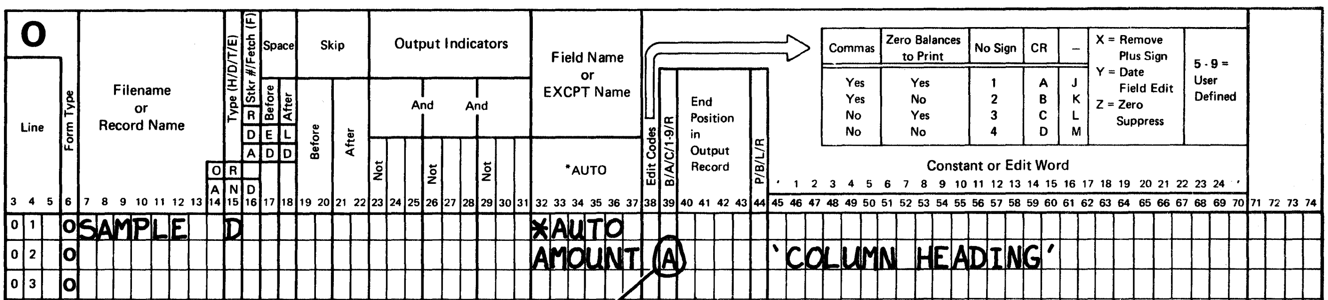
If a constant is entered in columns 45 through 70 and field name (columns 32 through 37) is blank, the constant is printed each time the detail report line is printed. In group printing, the constant is printed each time the first created total line is printed.

Field Description (A in Column 39)

Enter A in column 39 of a field-description specification following a D/T-*AUTO specification to accumulate and print totals for the field named in columns 32 through 37 (see Figure 15-11). The number of levels of totals that are printed is determined by the control-level entry (columns 59 and 60) on input specifications. A final total is also printed when the LR indicator is on. (This process is called *total rolling).*

If group printing is specified and a control-level indicator higher than the lowest-defined control level is specified in columns 23 through 31 on the record-description specification, totals are created for the indicator entered, all higher defined indicators, and LR.

The total output record created by auto report if you entered A in column 39 of a field description specification is conditioned by the associated control-level indicator defined in the input specifications. One total output record is created for each control-level indicator defined in the program.



The A in column 39 causes the AMOUNT field to be accumulated. Totals are printed for each control level and a final total is printed. A column heading is specified in columns 45 through 70.

Figure 15-11. Describing a Field That Is to Be Accumulated

Created Total Fields

When A is specified in column 39 of a detail or total *AUTO field-description specification, auto report creates and names total fields to be used in accumulating the required levels of totals. Auto report creates the field names for the total fields based on the name in columns 32 through 37 of the A-type field description. Names are created in the following way:

- If the specified field name has fewer than 6 characters, 1 character is added to the name to create a name for the total field. The added character is 1 through 9 or R, corresponding to the total indicators L1 through L9 and LR, respectively. For example, if ITEM is the specified field name and all nine control levels are defined, the created field names are ITEM1, ITEM2, . . . ITEM9, and ITEMR.

- If the specified field name has 6 characters, the last character is replaced by one of the characters, 1 through 9, or R. For example, if AMOUNT is the specified field name and all nine control levels are defined, the created field names are AMOUN1, AMOUN2, . . . AMOUN9, and AMOUNR.

Total fields are created and named for all control-level indicators defined in the program and for LR. (For an exception to this rule, see Figure 15-15 under *Group Printing.)* For example, if L1 and L3 are assigned to control fields on the input specifications and the field QTY is specified, three total fields, QTY1, QTY3, and QTYR, are created and named by auto report. All total fields created for the same level, such as QTY1 and AMOUN1, are printed on the same total line, and that line is conditioned by the corresponding control-level indicator.

Created total fields are 2 digits longer than the original field. For example, if the field QTY is defined with a length of 3, QTY1, QTY3, and QTYR all have lengths of 5. The number of decimal positions remains the same in the created fields. If a field previously defined in a program has the same name as a created field name, you can redefine the previous field, giving it whatever length and number of decimal positions you want. If you do this, the created field is assigned the previously defined length and number of decimal positions (if the previous field is numeric).

Considerations

You can specify created field names in RPG specifications that are included in the program. You must be aware, however, that the use of created fields in this way can interfere with the automatic accumulation of totals performed by auto report.

Field names ending in 1 through 9 or R should not be used in an auto report program that accumulates totals, because auto report creates total fields ending in those characters. This is especially important for 6-character field names, because auto report forms total field names by replacing the last character with 1 through 9 or R. No field name can be used more than once with A in column 39. Also, if a 5- or 6-character field name is specified with A in column 39, a second 5- or 6-character field name in which the first 5 characters are identical cannot be specified with A in column 39. For example, if the following four field names are specified with A in column 39 in an auto report specification, all but the first are invalid:

FIELD

FIELDX	Invalid because the first 5 characters duplicate the first 5 characters of the first field.
FIELDY	Invalid for the same reason as for FIELDX.
FIELD	Invalid because it duplicates the first field.

Columns 7-22

Columns 7 through 22 must remain blank on the field-description lines.

Columns 23-31 (Output Indicators)

Enter any valid output indicators in columns 23 through 31, or leave them blank. If these columns are blank, the field described is printed on each detail line. If indicators are entered in columns 23 through 31, the field is printed only when the conditions represented by those indicators are met. Leave these columns blank for group printing.

If a column heading is specified in columns 45 through 70 to appear over a field named in columns 32 through 37, the column heading is not affected by output indicators entered in these columns. Also, output indicators specified when column 39 contains A do not affect the creation of calculations for the field.

Output indicators specified on an A-type field-description specification following a D-*AUTO specification condition the calculations created for the field. If the A-type field description follows a T-*AUTO specification, however, a specified indicator does not condition calculations created for the field.

Columns 32-37 (Field Name)

When column 39 contains A, the name of a numeric field that is to be accumulated must be entered in columns 32 through 37. These columns cannot identify an array, array item, or table. The field named is printed on each detail line of the report. If group printing is specified, the total field for the lowest control-level indicator defined (L1, L2, . . . L9, LR, in that order) is printed on the created total line. (For an exception to this rule, see Figure 15-11 under *Group Printing.)* Totaling for any particular field by means of an A entry in column 39 can be specified only once in each program.

To create calculation and output specifications that accumulate and print the various levels of totals required, auto report creates and names additional totaling fields. Names created for the fields are based on the field name specified in these positions according to a set of rules (see *Created Total Fields).*

Column 38 (Edit Codes)

Enter an edit code in column 38, or leave it blank. If this column is blank, auto report creates a K edit code for the field named in columns 32 through 37. The K edit code causes the field to be edited with commas and a decimal point, such as 1,234,567.89. The field is also zero suppressed. Zero balances are not printed; negative balances are printed with a minus sign on the right. The edit code specified, or the created K edit code, applies to all created total fields as well as to the field named in columns 32 through 37.

Column 39

Enter A in column 39 to indicate that totals are to be accumulated for the field named in columns 32 through 37 of this field description. A total is printed for every control-level indicator defined in the input specifications and for the LR indicator. When column 39 contains A, columns 32 through 37 must contain the name of a numeric field. Columns 45 through 70 can contain a constant to be used as the first line of a column heading. (See *Created Specifications* for additional information.)

When the lowest control-level indicator used for a T-*AUTO specification is higher than the lowest control-level indicator defined in the input specifications, auto report creates only the total lines corresponding to the lowest control-level indicator used for the T-*AUTO specification, the higher defined control levels, and LR (see *Group Printing).*

Resetting Total Fields to Zero When column 39 contains A, the auto report program creates a B (blank after) in column 39 of all the detail and total field-description specifications created from the field name specified. Thus, the value in the specified field and in any created fields is reset to zero after the field value is printed.: If group printing is specified, auto report creates a calculation to reset the specified field to zero on each cycle. This prevents the same value from being accumulated more than once. An unconditioned total-calculation operation (Z-ADD) sets the field value to zero. This calculation is the first total calculation in the created RPG source program.

Asterisk Indication: To indicate that a printed line is a created total line, asterisks are printed on the line to the right of the highest end position created from the D/T-*AUTO specification. One asterisk is printed to the right on the lowest-level total line created. One additional asterisk is printed on each higher level line, including the final total.: For example, if L1 and L3 are defined control-level indicators in a program, one asterisk is printed to the right of the L1 line, two asterisks are printed on the L3 line, and three are printed on the LR line. As many as 10 asterisks are printed on the LR line if all nine control-level indicators are defined in the program.

To suppress the creation of asterisks on total lines, enter N in column 28 of the auto report option specifications.

Columns 40-43 (End Position in Output Record)

Enter the print position of the rightmost character of the field to be printed, or leave these positions blank. If this entry is blank, auto report creates end positions for fields and column headings. See *Report Format* for additional information and considerations.

Column 44

Column 44 is not used with auto report, because packed-decimal and binary data cannot be used. Leave this column blank.

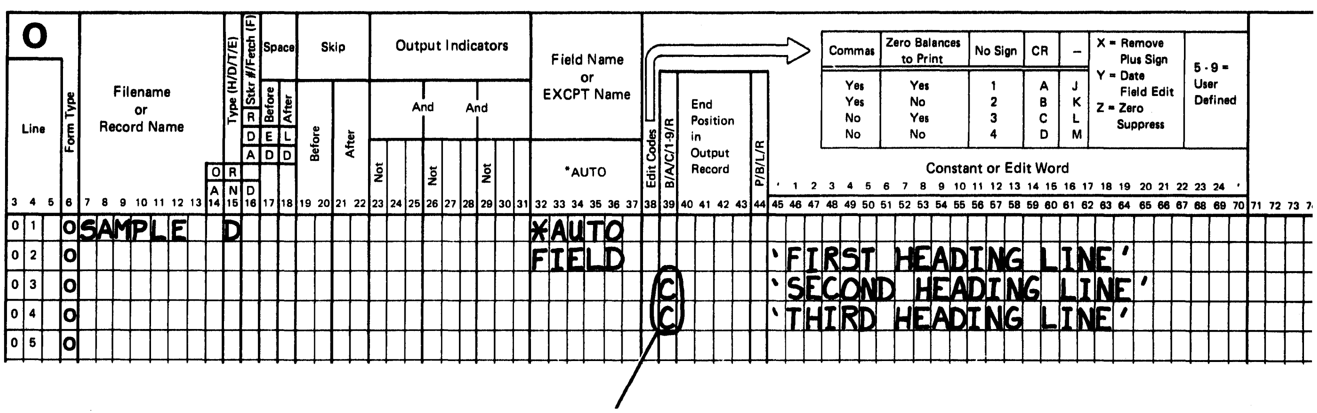
Columns 45-70 (Constant)

Either leave columns 45 through 70 blank, or enter a literal. Do not enter an edit word; editing is done by an edit code. If a literal is entered when column 39 contains A, the literal becomes the first line of the column heading over the accumulated field.

If these columns are left blank, the first line of the column heading is blank, but column-heading continuation lines can specify the second and third lines of the column heading. See *Field Description (C in Column 39).* Also see *Report Format* for information on how column heading and fields are centered and spaced by auto report.

Field Description (C in Column 39)

Enter C in column 39 of the *AUTO field descriptions to specify a second and third column-heading line. At times you may want more information in a column heading than can be contained on one line. Auto report enables you to specify the second and third lines of column headings by simply specifying the literals to appear on those lines. No additional heading output lines need be coded; no end position need be calculated. The special field-description specification that allows you to do this is identified by C in column 39 (see Figure 15-12).



C in column 39 is used to specify second and third column-heading lines. A maximum of three column-heading lines (two C-type field descriptions) can be used.

Figure 15-12. Specifying Second and Third Column-Heading Lines

Columns 7-38

Columns 7 through 38 must be blank on a field description that has C in column 39.

Column 39

Enter C in column 39. One or two C-type specifications can follow a field-description specification that has A, B, or blank in column 39 and an entry in columns 32 through 37. The first C-type specification causes a second column-heading line to be created. The second C-type specification causes a third column-heading line to be created (see Figure 15-12).

Columns 40-44

Columns 40 through 44 must be blank on a C-type field-description specification.

Columns 45-70 (Constant)

Enter a constant, up to 24 positions long including blanks, enclosed in apostrophes. The constant becomes the second or third line of column headings, depending on whether it is on the first or second C-type specification. If two or three column-heading lines are specified, the shorter literals are centered on the longest.

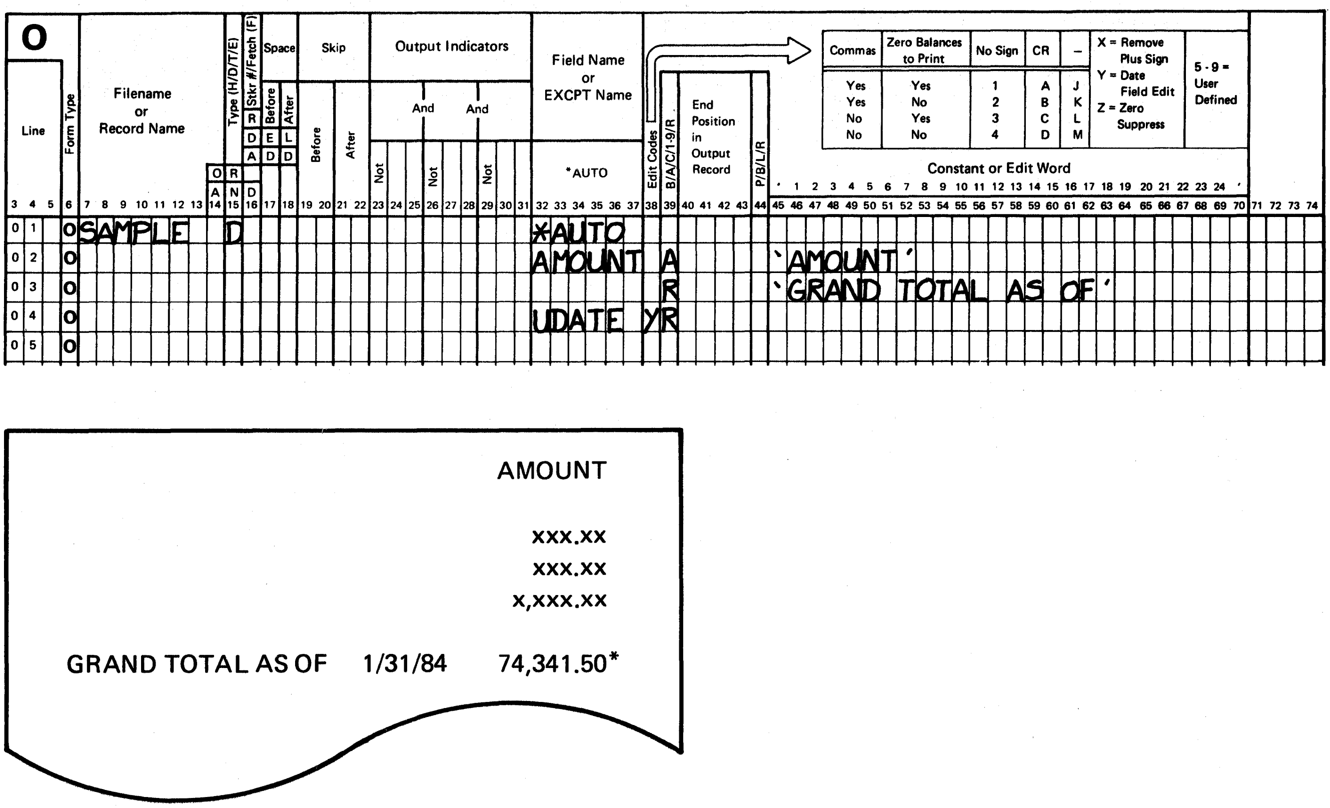
Field Description (1-9 or R in Column 39)

Enter a digit (1 through 9) or R in column 39 of a field description to specify a field or constant to be printed on a specific total line.

Auto report allows you to print other information on created total lines in addition to the created totals resulting from A-type field descriptions. The value entered in column 39 corresponds to the level of the total line on which the information is to be printed (the corresponding control level must be defined in columns 59 and 60 in the input specifications). For example, 3 in column 39 indicates that the information is printed on the L3 total line; R indicates that the information appears on the final total, or LR, line (see Figure 15-13). Fields and constants specified in this way are printed to the left of the leftmost created total on the line. See *Report Format* for exact placement.

This type of field description can print information such as DISTRICT TOTAL, GRAND TOTAL, or other literal information. It can also print a field and specify an edit word, floating currency symbol, or asterisk fill for the field.

If none of the *AUTO output fields is defined with A in column 39, then 1 through 9 or R cannot be used in column 39. In group printing, only specify numbers that are higher than the lowest control-level indicator used to condition the T-*AUTO specification. If the T-*AUTO specification is not conditioned by a control-level indicator, use only numbers that are higher than the lowest control level defined in columns 59 and 60 on the input specifications.



In this example, the literal 'GRAND TOTAL AS OF' followed by the current data prints on the left of the created final total line, as shown below.

Figure 15-13. Specifying a Literal and a Field to Print on a Created Total Line

Columns 7-31

Columns 7 through 31 must be blank on a field-description line with 1 through 9 or R in column 39.

Columns 32-37 (Field Name)

Enter the name of a field, an indexed array name, or a table name. The corresponding field or item value prints on the total line indicated by the entry in column 39. If columns 32 through 37 are blank, a constant must be entered in columns 45 through 70.

Column 38 (Edit Code)

Enter an edit code in column 38 to edit a numeric field named in columns 32 through 37, or leave column 38 blank. If column 38 is left blank, an edit word can be entered in columns 45 through 70. If column 38 is blank, no edit code is assumed by auto report.

Column 39

Enter a digit (1 through 9) or R. These entries correspond to the indicators L1, L2, . . . L9, and LR. The entry identifies a specific total line on which the field or literal described is to be printed. The entry in column 39 must correspond to a control level that is defined by the input specifications. In group printing, the entry in this column must be higher than the control level of the first total line created.

Columns 40-43 (End Position in Output Record)

Do not make an entry in columns 40 through 43 on field-description specifications with 1 through 9 or R in column 39. See *Report Format* for additional information and considerations.

Column 44

Leave column 44 blank.

Columns 45-70 (Constant or Edit Word)

Leave columns 45 through 70 blank, or enter a constant or edit word. If field name (columns 32 through 37) on this specification line contains an entry, then columns 45 through 70 can contain any of the following:

- Blanks, if no editing is needed for the field or if the field is already edited by an edit code in column 38
- Edit word, if special editing is desired
- Floating currency symbol or asterisk-fill entry used with an edit code

Columns 45 through 70 cannot contain a constant when field name contains an entry. However, when field name is blank, columns 45 through 70 must contain a constant.

Group Printing

In group printing, data is summarized for a group of input records, and only totals are printed on the report. Totals can have either subtotals and a final total or only a final total.

Specifications

To specify group printing using auto report, enter T in column 15 and *AUTO in columns 32 through 36. A control-level indicator can be specified in columns 23 through 31. When a T-*AUTO specification is used, a line is not printed for each individual record that is read, but only after a complete control group is read.

Fields and literals defined by field-description specifications that have a blank or B in column 39 and follow a T-*AUTO record description are printed on the lowest-level total line. Fields defined with A in column 39 are not printed on the total lines, but the total fields created by auto report are. Continued column headings (C in column 39) and total-indicated fields (1 through 9 or R in column 39) can also be specified by field descriptions following a T-*AUTO record description.

Output indicators can be entered in columns 23 through 31 of a field-description specification following a T-*AUTO record description if column 39 of the field-description specifications contains a blank or B. If output indicators are used in a field description that has A in column 39 following a T-*AUTO specification, those indicators are ignored by auto report. Output indicators cannot be used in a field description that contains C, 1 through 9, or R in column 39.

Examples

Figure 15-14 shows the file description and input specifications for the group-printed reports shown in Figures 15-15 and 15-16. BRANCH and REGION are defined as control fields.

Figure 15-15 shows the calculation specifications, the output specifications, and the group-printed report showing sales totals for a company. Since the T-*AUTO specification is conditioned by L2, only the totals for REGION (L2) and for the entire company (LR) are printed on the report. The totals for BRANCH (L1) are not printed.

A DISK summary file, DISKSUM, is also produced by this program. The summary file contains a summary record of the sales data for each branch. The output specifications for DISKSUM illustrate the use of standard RPG output specifications in the same program with *AUTO specifications. The output record described is written on the DISK file, DISKSUM, when there is an L1 control break (BRANCH field changes). Since the T-*AUTO specification is conditioned by L2, auto report does not create fields for the L1 control level. Therefore, standard RPG calculation specifications must be used to calculate the L1 totals. The L1 total fields that are written on the DISKSUM file (SOLDQ1, SOLDV1, and VALUE1) must be defined in the calculations.

Figure 15-16 shows a group-printed report similar to the one shown in Figure 15-15. However, the T-*AUTO specifications are not conditioned by a control-level indicator, so totals are printed for all defined control levels and for LR.

Line	Form Type	Filename	File Type (I/O/U/C/D)	File Designation (P/S/C/R/T/D/F)	File Format (F/V/S/M/D/E)	Block Length	Record Length	Device
02	F	SALES	I	P	F	473	43	DISK
03	F	PRINTER	O		F	12Ø	12Ø	PRINTER
04	F	DISKSUM	O		F	25Ø	25	DISK
05	F							

Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator	Field Location From	Field Location To	Decimal Positions	RPG Field Name	Control Level (L1-L9)
01	I	SALES	AA	Ø1					
02	I				1	7		ITEMNO	
03	I				8	9		BRANCH	L1
04	I				1Ø	1Ø		REGION	L2
05	I				11	25		DESC	
06	I				26	27	Ø	SOLDQY	
07	I				28	34	2	SOLDVA	
08	I				35	36	Ø	ONHAND	
09	I				37	43	2	VALUE	
10	I								

L1 and L2 are the defined control levels.

Figure 15-14. File Description and Input Specifications for the Group-Printed Reports Shown in Figures 15-11 and 15-12

```
  01 C   Ø1      SOLDQ1    ADD  SOLDQY    SOLDQ1  40
  02 C   Ø1      SOLDV1    ADD  SOLDVA    SOLDV1  92
  03 C   Ø1      VALUE1    ADD  VALUE     VALUE1  92
  04 C
```

```
  01 OPRINTER H                *AUTO
  02 O                                      'SALES FOR ANY COMPANY'
  03 O                                      ' BY REGION'
  04 O        T        L2      *AUTO
  05 O                         REGION       'REGION'
  06 O                         SOLDQY A     'NUMBER OF SALES'
  07 O                         SOLDVA A     'VALUE'
  08 O                         VALUE  A     'VALUE OF STOCK'
  09 O                                C     ' ON HAND'
  10 O                                R     'COMPANY TOTAL'
  11 ODISKSUM T        L1
  12 O                         REGION     1
  13 O                         BRANCH     3
  14 O                         SOLDQ1 B   7
  15 O                         SOLDV1 B  16
  16 O                         VALUE1 B  25
  17 O
```

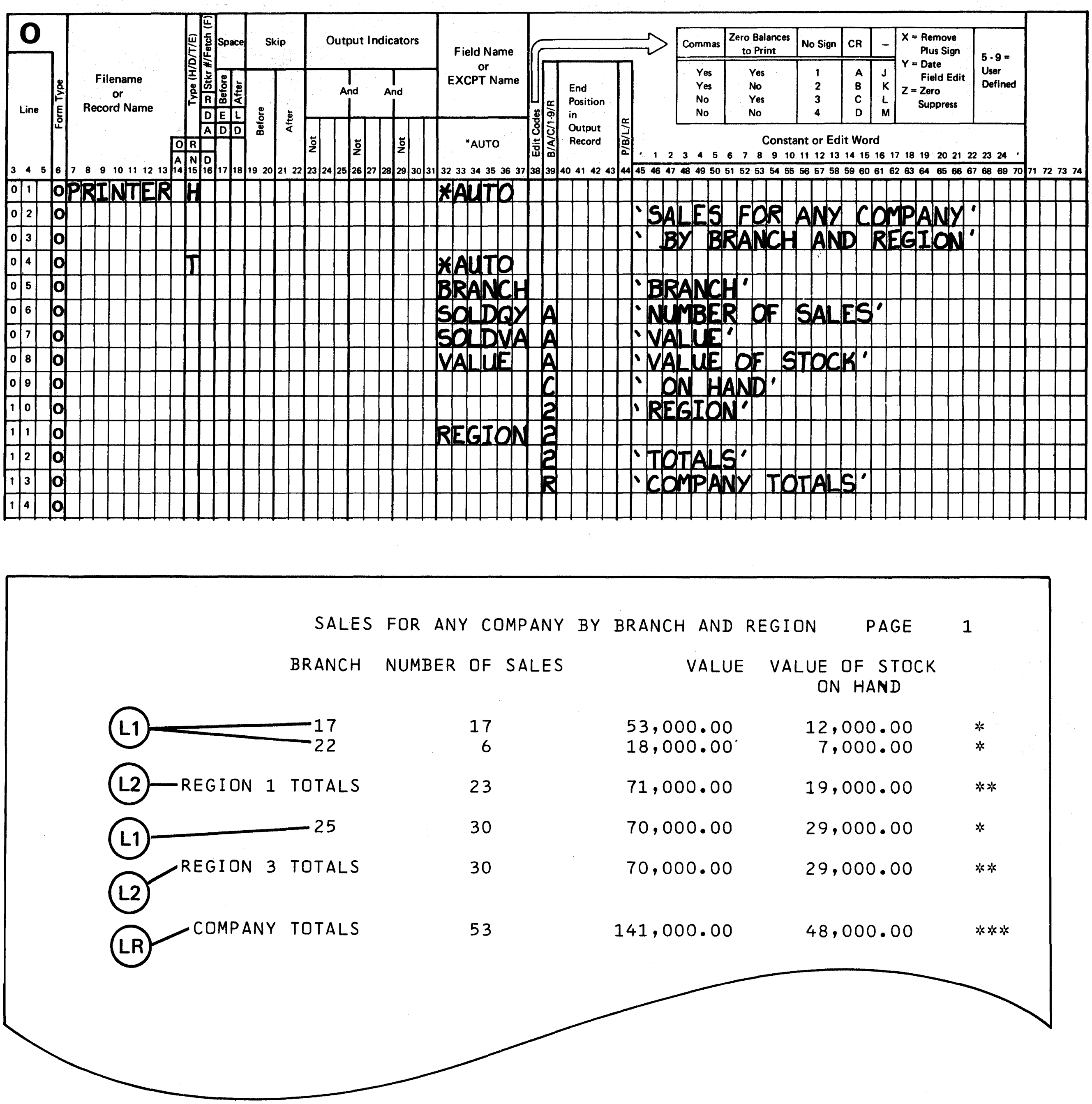
T in column 15 with *AUTO in columns 32 through 37 specifies a group-printed report.

Because L2 is entered under output indicators, total lines are printed only for L2 and LR, although L1 is also a defined control-level indicator.

In group printing, the lowest-level total lines printed (L2, in this case) are single-spaced, like detail lines.

```
11/11/83          SALES FOR ANY COMPANY BY REGION           PAGE    1

              REGION  NUMBER OF SALES          VALUE  VALUE OF STOCK
                                                          ON HAND

                 1              23         71,000.00       19,000.00      *
                 3              30         70,000.00       29,000.00      *

COMPANY TOTAL                   53        141,000.00       48,000.00      **
```

Figure 15-15. Using *AUTO to Produce a Group-Printed Report Showing Region and Final Totals



When no control-level indicators are entered under output indicators, a total line is created for each defined control-level indicator (L1 and L2, in this case) and for LR.

Figure 15-16. Using *AUTO to Produce a Group-Printed Report Showing Branch, Region, and Final Totals

Report Format

One of the advantages of auto report is that it frees you from the task of specifying the format of your report on the output specifications sheet. Auto report can completely format the report by spacing, skipping, centering lines, and calculating end positions for fields and constants.

Spacing and Skipping

You can specify spacing and skipping, or you can leave it to auto report. Figure 15-17 shows spacing and skipping created by auto report. For the specifications used to produce the report, see *Specifications Created by Auto Report* in this chapter. If columns 17 through 22 are left blank on an H-*AUTO specification, auto report skips to line 06 before printing the first heading line, and it spaces two lines after the last heading line. If more than one heading line is specified, auto report spaces one line after the first heading line and after all succeeding heading lines except the last. To specify spacing and skipping, follow the standard RPG rules for spacing and skipping.

Column-heading lines are spaced like page headings. Auto report spaces one line after all column headings except the last. It spaces two lines for a single heading line, or for the last heading line if more than one is specified. Spacing and skipping entries cannot be specified for column headings. If spacing and skipping entries are made on a D-*AUTO record-description specification, the entries apply to the detail line created. The entries do not apply to column headings or to total lines created by auto report from the D-*AUTO specification. Standard RPG rules for spacing and skipping must be followed. If spacing and skipping entries are not made, auto report spaces one line after printing the created detail line.

Two lines are spaced after all total lines produced by auto report from a D-*AUTO specification. In addition, the lowest-level total line and the final total line are also created with one space before.

If spacing and skipping entries are made on a T-*AUTO specification, the entries apply to the lowest-level total line created, but not to column headings or to higher level total lines. If spacing and skipping entries are not made, one line is spaced after the lowest-level total lines; two lines are spaced after all higher levels. One line is always spaced before the next-to-the-lowest-level total and before the final total (see Figure 15-16 for an example).

Skip to line 06 occurs before printing of the first line.

Highest end position in the report.

```
1/15/84                              SALES REPORT FOR ANY CO.                                   PAGE    1

REGION    BRANCH    ITEM       DESCRIPTION       SALES             AMOUNT    ON-HAND              VALUE
                    NUMBER

   1        17      AG7701T    2-TON TRUCK         5            25,000.00       2             10,000.00
                    AG7705S    PICK-UP            10            20,000.00       1              2,000.00
                    AP6545B    CAMPER              2             8,000.00

                                                                53,000.00                     12,000.00 *

            22      AG7701T    2-TON TRUCK         2            10,000.00       1              5,000.00
                    AG7705S    PICK-UP             4             8,000.00       1              2,000.00

                                                                18,000.00                      7,000.00 *

                                                                71,000.00                     19,000.00 **

   3        25      AG6545B    CAMPER             10            40,000.00       5             20,000.00
                    AP6549P    1/4-TON TRUCK      20            30,000.00       6              9,000.00

                                                                70,000.00                     29,000.00 *

                                                                70,000.00                     29,000.00 **

                                          FINAL TOTALS         141,000.00                     48,000.00 ***
```

Auto report creates a blank line (space-two-after) following the last page-heading line (in this case, there is only one page-heading line) and following the last column-heading line.

Auto report creates a blank line before the lowest-level total (in this case, there is only the L1 total) and before the final total (space-one-before).

Auto report creates a blank line following each total line (space-two-after).

Figure 15-17. Report Showing the Format Created by Auto Report

Placement of Headings and Fields

Auto report creates end positions for fields and constants and centers column headings, columns, and report lines (see Figure 15-17 for an example). However, if an end position is specified for a field or constant on a D/T-*AUTO field-description line, that end position is used on all column heading, detail, and total specifications created from the field description. (The specified end position may be changed slightly by auto report when the line is centered or when the column heading and field are positioned in relation to each other.) If the specified end position causes an overlay with a previous field or constant, auto report creates a new end position.

Specify end positions only to eliminate the automatic spacing between fields or to spread out or expand a report on the page.

Page Headings

If the date and page number are printed on the first *AUTO page-heading line (that is, if they are not suppressed by an N in column 27 of the auto report option specifications), the date is always printed in positions 1 through 8. The page number is printed with an end position equal to the highest end position of the longest line in the report. When the first *AUTO page heading (including date, title, and page number) is the longest line in the report, one blank space separates the title from the date and the word PAGE from the title. If the resulting line exceeds the record length of the PRINTER file, the excess information on the right of the line is not printed.

If a line created from a D/T-*AUTO specification is the longest report line, that line is printed starting in print position 1, and the title portion of the first page-heading line is centered in relation to that line. Additional *AUTO page headings are then centered on the first *AUTO page-heading line.

If an *AUTO page heading is the longest line in the report and a D/T-*AUTO specification is present, any other *AUTO page-heading lines and the line created from the D/T-*AUTO specification are centered on the longest page heading.

Fields and constants appear in the order specified in the *AUTO output specifications from left to right. Auto report provides one blank space before and after fields on the heading line. No spacing is provided between constants.

Reformatting *AUTO Page Headings

You can reformat an *AUTO page-heading line if you do not want to use the end positions for fields and constants that are created by auto report. If you want to find what end positions are created for page, date, and title information, see the listing of the created source program that is produced by the RPG compiler (see *Created Specifications).*

Catalog the created RPG source program in a library by specifying the C option in column 7 of the auto report option specifications, and change the end positions on the created source statements by using the source entry utility (SEU).

Body of the Report

Placement of column headings above columns depends on which is longer, the heading or the associated field (including edit characters). If any column heading is longer than the associated field, the field is centered under the longest constant in a column heading. However, if the field is longer than the longest constant in a column heading, the column heading is printed in the leftmost positions over an alphameric field and in the rightmost characters over a numeric field. When more than one column-heading line is specified, shorter column headings are always centered on the longest column heading (see Figure 15-17).

Fields and constants appear from left to right on a line in the order in which they are specified by the output specifications. At least two blank spaces appear before each field on the line. However, no spaces are provided before a constant; you must incorporate blanks within constants if you want to provide additional spacing.

Total indication (fields and constants specified with 1 through 9 or R in column 39 of the output specifications) is placed to the left of the first total field (A in column 39) on the corresponding total line, followed by two spaces. If two or more such fields or constants are specified for a total line, they appear from left to right in the order specified on the left of the first total on the line. Each field is preceded and followed by one space. No spacing is provided for constants.

Overflow of the D/T-*AUTO Print Lines

If the lines created from a D/T-*AUTO specification are longer than the record length specified for the PRINTER file, a second print line (overflow line) is created for each column-heading line, detail (or group-print) line, and total line. (Remember, a second print line is not created for *AUTO page-heading lines.) The excess information is placed in the rightmost positions on the overflow line in the order specified.

Figure 15-18 shows the result of an overflow condition.

In the output specifications for the report shown in Figure 15-18, no spacing or skipping is specified. If spacing and skipping are specified, however, auto report spaces the report as follows:

- Column heading lines and total lines are spaced as shown in Figure 15-18.
- The space-before and skip-before entries specified are for the original detail (or group-print) line. Auto report creates one space after this line.
- The space-after and skip-after entries specified are for the overflow line. Auto report creates blanks for space-before and skip-before for the overflow line.

Auto report prints those columns that cannot be completely contained on the original line on overflow lines.

```
  1/15/84                          CASH RECEIPTS REGISTER                                 PAGE    1

 REGION  ACCOUNT  ACCOUNT NAME           INVOICE  INVOICE  DATE PAID       AMOUNT     DISCOUNT
         NUMBER                          NUMBER    DATE                     OWED       TAKEN

                                                                 AMOUNT       BALANCE     EXCESS
                                                                  PAID          DUE      DISCOUNT

  1      11243    JONES HARDWARE          27541    7/11/71     7/21/1          23.75         .47
                                                                 23.28
  1      11352    NU-STYLE CLOTHIERS      27987    7/14/71     7/26/1          87.07
                                                                 40.00          47.07
  1      11886    MIDI FASHIONS INC       15771    7/04/71     7/14/1         107.22        2.14
                                                                105.08
  1      12874    ULOOK INTERIORS         25622    7/09/71     7/23/1          67.95
                                                                 67.95
  1      18274    STREAMLINE PAPER INC    29703    7/21/71     7/30/1         274.03        2.38
                                                                170.55         101.10

                                                    REGION TOTALS             560.02        4.99
                                                                406.86         148.17              *

  2      23347    RITE-BEST PENS CO       20842    7/18/71     7/20/1          15.80
                                                                 10.00           5.80
  2      25521    IMPORTS OF NM           29273    7/20/71     7/27/1         797.40       11.93
                                                                585.47         200.00
  2      26723    ALRIGHT CLEANERS        19473    7/07/71     7/23/1         462.00
                                                                462.00
  2      28622    NORTH CENTRAL SUPPLY    17816    7/05/71     7/22/1          75.97
                                                                 75.97
  2      29871    FERGUSON DEALERS        27229    7/10/71     7/22/1          61.91
                                                                 61.91

                                                    REGION TOTALS           1,413.08       11.93
                                                              1,195.35         205.80              *

  3      30755    FASTWAY AIRLINES        26158    7/06/71     7/19/1         742.72       16.85
                                                                725.87                      1.90
  3      31275    ENVIRONMENT CONCERNS    20451    7/06/71     7/30/1          29.43
                                                                 15.00          14.43
  3      32457    B SOLE SILOS            27425    7/10/71     7/20/1         110.05
                                                                110.05
  3      37945    HOFFTA BREAKS INC.      18276    7/06/71     7/23/1          47.23
                                                                 47.23

                                                    REGION TOTALS             929.43       16.85
                                                                898.15          14.43       1.90   *

  4      42622    EASTLAKE GRAVEL CO      16429    7/05/71     7/23/1          29.37
                                                                 29.37

                                                    REGION TOTALS              29.37
                                                                 29.37                             *

                                                   COMPANY TOTALS           2,931.90       33.77
                                                              2,529.73         368.40       1.90   **
```

Figure 15-18. Report Showing Overflow of D-*AUTO Print Lines

Created Specifications

Auto report creates standard RPG specifications and combines them with RPG specifications included in the input to auto report and with specifications copied from library members. From them, it produces the final RPG source program. This section describes the created RPG specifications and the order of those specifications in the RPG source program.

Figures 15-19 and 15-20 show auto report specifications for a sales report and the resulting RPG source specifications that are created for the report. Numbers are inserted in the figures to identify the auto report functions and to show the specifications that are created by each function.

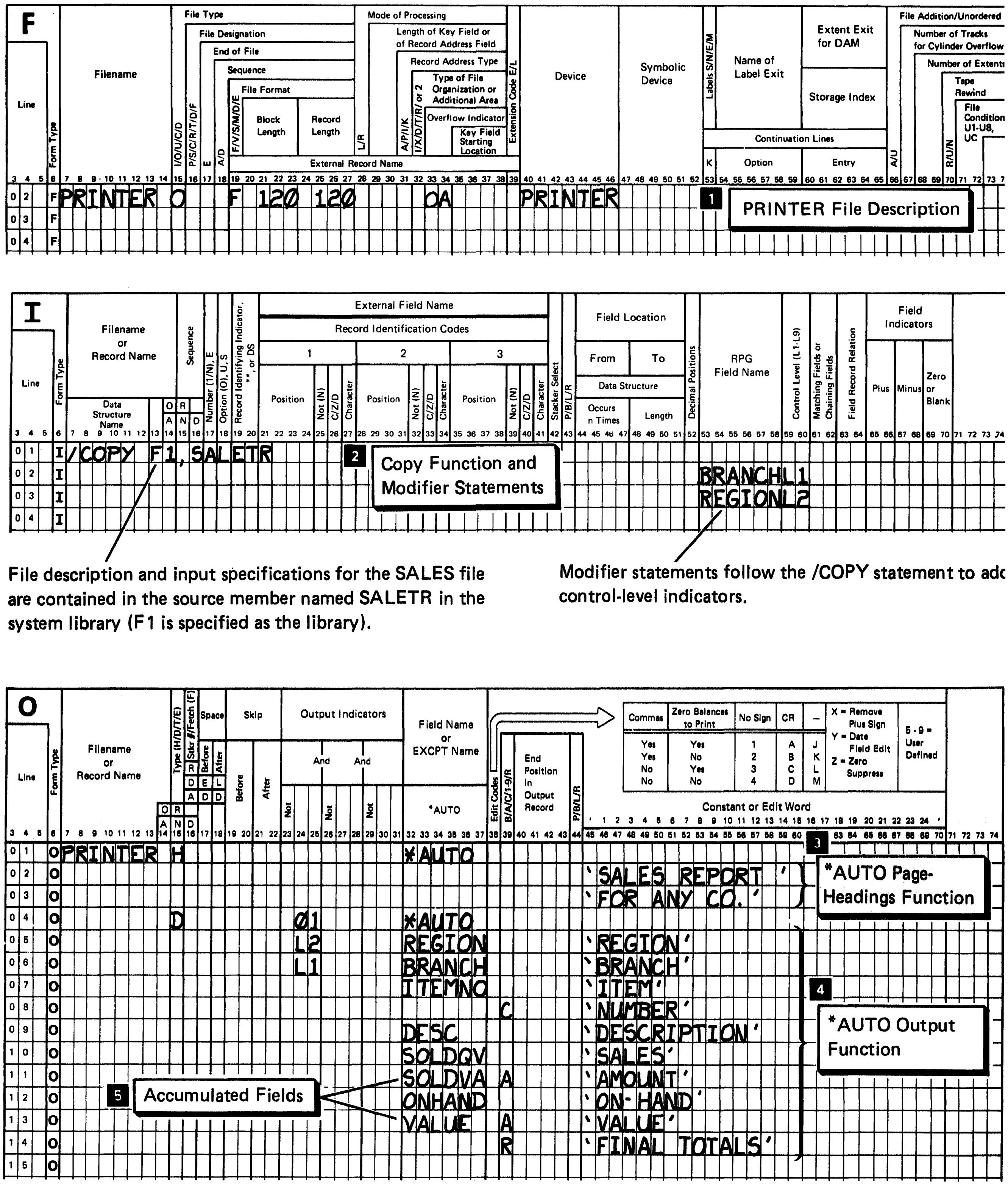


Figure 15-19. Auto Report Specifications for a Sales Transaction Report

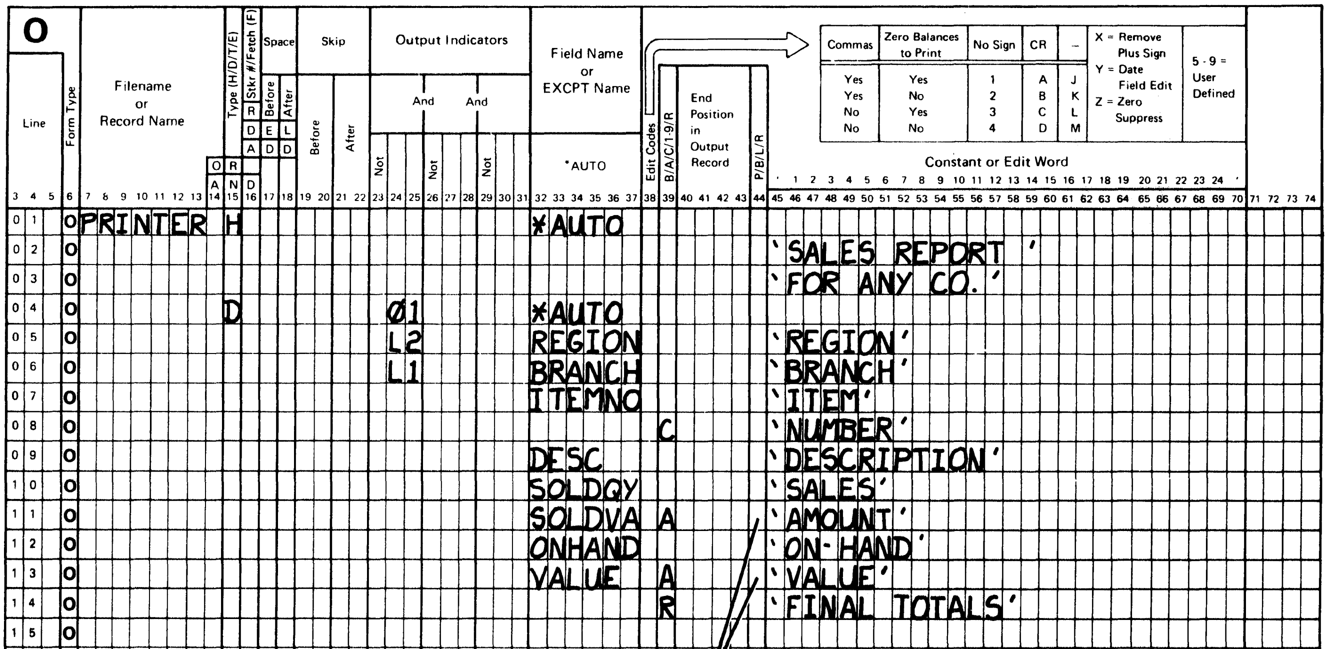
If you do not specify a control specification, auto report creates an all-blank control specification for you.

```
    RG 004      0010 H

1   0001        0020 FPRINTER O    F 120 120        OA     PRINTER
2   0002        0030CFSALES    IP  F 473  43               DISK
                0040 I*/COPY F1,SALETR
    0003        0050CISALES    AA  01
    0004        0060CI                                            1    7 ITEMNO
    0005        0070MI                                            8    9 BRANCHL1
    0006        0080MI                                           10   10 REGIONL2
    0007        0090CI                                           11   25 DESC
    0008        0100CI                                           26   270SOLDQY
    0009        0110CI                                           28   342SOLDVA
    0010        0120CI                                           35   3600NHAND
    0011        0130CI                                           37   432VALUE
5   0012        0140EC    01                  EXSR A$$SUM
    0013        0150ECL1            SOLDV2    ADD  SOLDV1     SOLDV2    92
    0014        0160ECL1            VALUE2    ADD  VALUE1     VALUE2    92
    0015        0170ECL2            SOLDVR    ADD  SOLDV2     SOLDVR    92
    0016        0180ECL2            VALUER    ADD  VALUE2     VALUER    92
    0017        0190ECSR            A$$SUM    BEGSR
    0018        0200ECSR            SOLDV1    ADD  SOLDVA     SOLDV1    92
    0019        0210ECSR            VALUE1    ADD  VALUE      VALUE1    92
    0020        0220ECSR                      ENDSR
3   0021        0230EOPRINTER H     206     1P
    0022        0240EO        OR            OA
    0023        0250EO                                  45 'SALES REPORT '
    0024        0260EO                                  56 'FOR ANY CO.'
    0025        0270EO                          UDATE Y  8
    0026        0280EO                          PAGE  Z 89
    0027        0290EO                                  85 'PAGE '
4   0028        0300EOPRINTER H     1       1P
    0029        0310EO        OR            OA
    0030        0320EO                                   6 'REGION'
    0031        0330EO                                  14 'BRANCH'
    0032        0340EO                                  21 'ITEM'
    0033        0350EO                                  36 'DESCRIPTION'
    0034        0360EO                                  47 'SALES'
    0035        0370EO                                  62 'AMOUNT'
    0036        0380EO                                  71 'ON-HAND'
    0037        0390EO                                  86 'VALUE'
    0038        0400EOPRINTER H     2       1P
    0039        0410EO        OR            OA
    0040        0420EO                                  22 'NUMBER'
    0041        0430EOPRINTER D     1       01
    0042        0440EO                      L2  REGION    3
    0043        0450EO                      L1  BRANCH   12
    0044        0460EO                          ITEMNO   23
    0045        0470EO                          DESC     40
    0046        0480EO                          SOLDQYK  46
    0047        0490EO                          SOLDVAKB 62
    0048        0500EO                          ONHANDK  69
    0049        0510EO                          VALUE KB 86
    0050        0520EOPRINTER T    12       L1
    0051        0530EO                          SOLDV1KB 62
    0052        0540EO                          VALUE1KB 86
    0053        0550EO                                   87 '*'
    0054        0560EOPRINTER T     2       L2
    0055        0570EO                          SOLDV2KB 62
    0056        0580EO                          VALUE2KB 86
    0057        0590EO                                   88 '**'
    0058        0600EOPRINTER T    12       LR
    0059        0610EO                          SOLDVRKB 62
    0060        0620EO                          VALUERKB 86
    0061        0630EO                                   47 'FINAL TOTALS'
    0062        0640EO                                   89 '***'
```

Figure 15-20. RPG Source Program Created from Auto Report Specifications

Created Calculation Specifications

Calculation specifications are created to accumulate totals for fields named on *AUTO field description specifications that have an A in column 39 (see Figure 15-21).



Calculations are created for fields with an A in column 39.

Total calculations roll higher level totals.

```
0012    0140EC     01           EXSR A$$SUM
0013    0150ECL1     SOLDV2     ADD  SOLDV1     SOLDV2     92
0014    0160ECL1     VALUE2     ADD  VALUE1     VALUE2     92
0015    0170ECL2     SOLDVR     ADD  SOLDV2     SOLDVR     92
0016    0180ECL2     VALUER     ADD  VALUE2     VALUER     92
0017    0190ECSR     A$$SUM     BEGSR
0018    0200ECSR     SOLDV1     ADD  SOLDVA     SOLDV1     92
0019    0210ECSR     VALUE1     ADD  VALUE      VALUE1     92
0020    0220ECSR                ENDSR
```

Subroutine accumulates the lowest-level totals (L1, in this example).

Length and decimal position of created total fields.

Note: Placement of the created calculation specifications in the RPG source program is shown in Figure 15-20.

Figure 15-21. Calculation Specifications Created from Auto Report Coding for Sales Transaction Report

An RPG subroutine is created to accumulate the values from these fields into the lowest-level created total fields. The name of the subroutine is always A$$SUM. The subroutine specifications are conditioned differently, depending on whether detail or group printing is specified:

- If detail printing is specified, as in Figure 15-21, the EXSR statement is conditioned by the same indicator(s) that conditions the D-*AUTO specification (01 in this example). Each ADD statement in the subroutine is conditioned by the field indicator(s) specified with the field in its field-description specification (none in this example).

- If group printing is specified, the EXSR statement and all ADD statements in the subroutine are unconditioned.

Total-calculation specifications are created to roll the total from the lowest-level defined total field through the higher level defined total fields and the final total. The total calculation to add the total from one level to that of the next higher level is conditioned by the control-level indicator corresponding to the field name of the lower level. As shown in Figure 15-21, total-calculation specifications to accumulate L2 and LR totals are followed by the subroutine to accumulate the lowest-level total, L1.

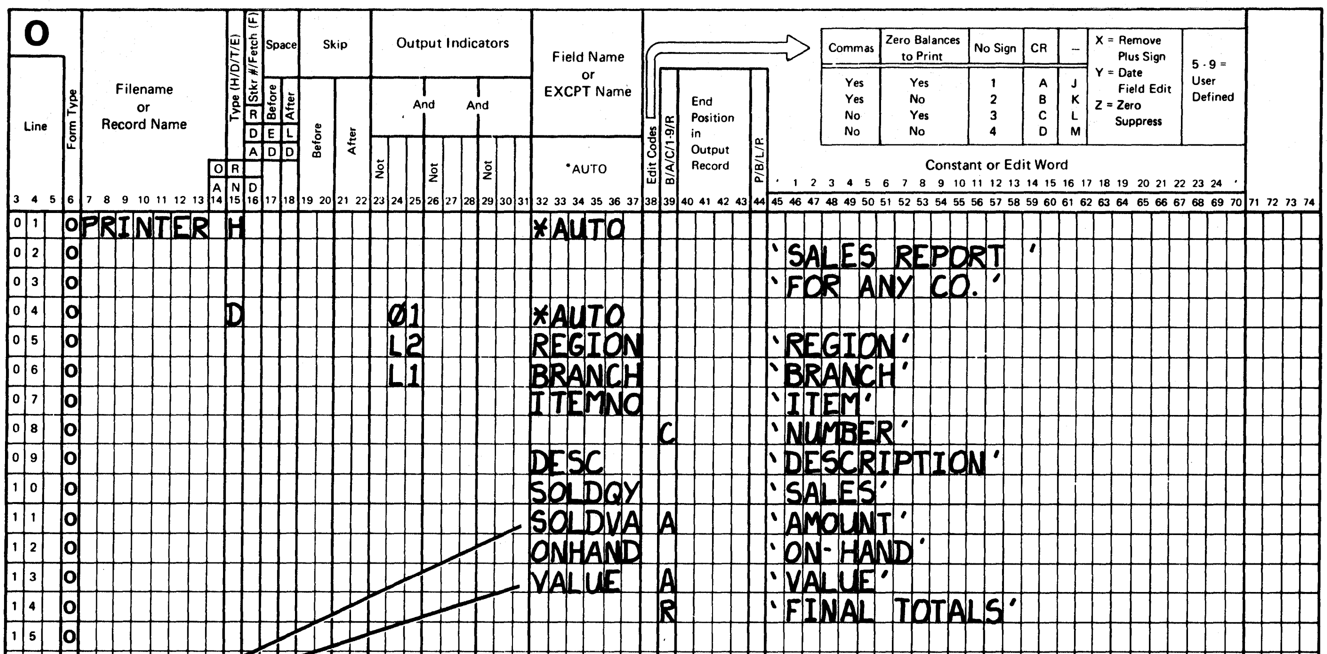
Created total fields are defined (given length and number of decimal positions) when the total field is the result field in a created calculation specification. In the input specifications, SOLDVA and VALUE are numeric fields defined with a length of seven positions with two decimal positions. Figure 15-21 shows that the total fields created from SOLDVA and VALUE are defined as two positions longer than the original fields, with the same number of decimal positions.

When group printing is specified (T-*AUTO specification), auto report creates total-calculation specifications to reset each of the accumulated fields (A in column 39) on the lowest-level total line to zero on each cycle. A Z-ADD calculation, conditioned by L0, is created for each accumulated field. These calculations are the first total-calculation specifications in the created RPG source program.

Created Output Specifications

Figure 15-22 shows the output specifications created by auto report. To identify specifications supplied by auto report (column-heading specifications, total specifications, conditioning indicators, spacing and skipping values, end-position values, blank after) compare the listing with the auto report specifications.

Auto report creates specifications to reset accumulated fields to zero after they are printed. See *Field Description (A in Column 39)* for a discussion of resetting fields to zero. In this example, blank-after is created for accumulated fields.




```
0021   0230EOPRINTER H   206    1P
0022   0240EO          OR       0A
0023   0250EO                                              45 'SALES REPORT '
0024   0260EO                                              56 'FOR ANY CO.'
0025   0270EO                                  UDATE Y      8
0026   0280EO                                  PAGE  Z     89
0027   0290EO                                              85 'PAGE '
0028   0300EOPRINTER H   1      1P
0029   0310EO          OR       0A
0030   0320EO                                               6 'REGION'
0031   0330EO                                              14 'BRANCH'
0032   0340EO                                              21 'ITEM'
0033   0350EO                                              36 'DESCRIPTION'
0034   0360EO                                              47 'SALES'
0035   0370EO                                              62 'AMOUNT'
0036   0380EO                                              71 'ON-HAND'
0037   0390EO                                              86 'VALUE'
0038   0400EOPRINTER H   2      1P
0039   0410EO          OR       0A
0040   0420EO                                              22 'NUMBER'
0041   0430EOPRINTER D   1      01
0042   0440EO                   L2                 REGION       3
0043   0450EO                   L1                 BRANCH      12
0044   0460EO                                      ITEMNO      23
0045   0470EO                                      DESC        40
0046   0480EO                                      SOLDQYK     46
0047   0490EO                                      SOLDVAKB    62
0048   0500EO                                      ONHANDK     69
0049   0510EO                                      VALUE KB    86
0050   0520EOPRINTER T 12       L1
0051   0530EO                                      SOLDV1KH    62
0052   0540EO                                      VALUE1KB    86
0053   0550EO                                                  87 '*'
0054   0560EOPRINTER T  2       L2
0055   0570EO                                      SOLDV2KB    62
0056   0580EO                                      VALUE2KB    86
0057   0590EO                                                  88 '**'
0058   0600EOPRINTER T 12       LR
0059   0610EO                                      SOLDVRKB    62
0060   0620EO                                      VALUERKB    86
0061   0630EO                                                  47 'FINAL TOTALS'
0062   0640EO                                                  89 '***'
```

Two heading specifications are created for column headings because ITEM NUMBER is a two-line heading.

Auto report creates total specifications to print accumulated totals for SOLDVA and VALUE fields.

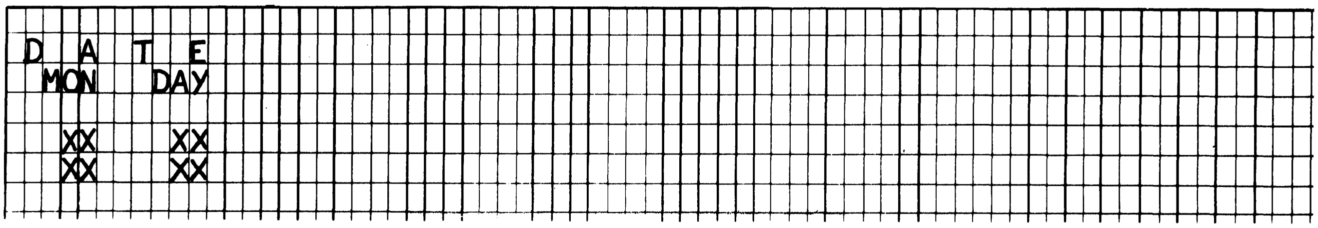
Figure 15-22. Output Specifications Created from Auto Report Coding for Sales Transaction Report

Programming Aids

The chart shown in Figure 15-23 should be helpful in determining valid *AUTO output entries depending on the contents of column 39.

The following programming suggestions may be helpful in specific programming situations.

One column heading can be printed over two or more fields if automatic column spacing is taken into consideration. For example, suppose the heading DATE is to print over a month field and a day field as follows:



39	7-22	22-31	32-37	38	40-43	44	45-70
Blank	Blank	Blank or indicators	Field name	Blank or edit code	Blank or end position	Blank	Blank or column heading
	Blank	Blank or indicators	Blank	Blank	Blank or end position	Blank	Literal
B	Blank	Blank or indicators	Field name	Blank or edit code	Blank or end position	Blank	Blank or column heading
A	Blank	Blank or indicators	Field name	Blank or edit code	Blank or end position	Blank	Blank or column heading
C	Blank	Blank	Blank	Blank	Blank	Blank	Column heading
1-9, R	Blank	Blank	Field name	Blank or edit code	Blank	Blank	Blank or edit word
	Blank	Blank	Blank	Blank	Blank	Blank	Literal

Figure 15-23. Valid *AUTO Entries on the Output Specifications, Depending on the Contents of Column 39

Code the output specifications as follows:

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Field Name or EXCPT Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	Constant or Edit Word
01	O							
02	O			MONTH				'D A'
03	O				C			'MON'
04	O			DAY				'T E'
05	O				C			'DAY'
06	O							

To print a constant on only the first detail line under a column heading, move the constant to a field in calculation specifications and print that field as shown in Figure 15-24.

If group printing is being done and more than one record type is present in the input file, certain precautions must be taken. If a field to be accumulated is not present in all record types, the correct total is not created unless additional coding is used. The specifications shown in Figure 15-25 give incorrect results because the T-*AUTO specification causes an unconditioned ADD subroutine to be created if a field is to be added. Therefore, QTY is added when indicator 10 is on and when indicator 11 is on. Figure 15-26 shows a method of obtaining the correct results.

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Comments
01	C		L1		MOVE	'CONSTANT'	FLDA	8		
02	C									

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Field Name or EXCPT Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	Constant or Edit Word
01	O		D	*AUTO				
02	O			FLDA		B		'COLUMN HEADING'
03	O							

Figure 15-24. Printing a Constant on Only the First Line

Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator	Position	Not (N)	C/Z/D	Character	Field Location From	Field Location To	Decimal Positions	RPG Field Name	Control Level (L1-L9)
01	I	INPUT	AA	1Ø	1		C	A					
02	I								2	27		NAME	L1
03	I												
04	I												
05	I		BB	11	1		C	1					
06	I								2	18		DESC	
07	I								19	21	Ø	QTY	
08	I								22	26	2	SALES	
09	I												

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Output Indicators	Field Name or EXCPT Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	Constant or Edit Word
01	O	PRINT	T	L1	*AUTO				
02	O				DESC				'DESCRIPTION'
03	O				QTY		A		'QUANTITY'
04	O				SALES		A		'AMOUNT'
05	O								

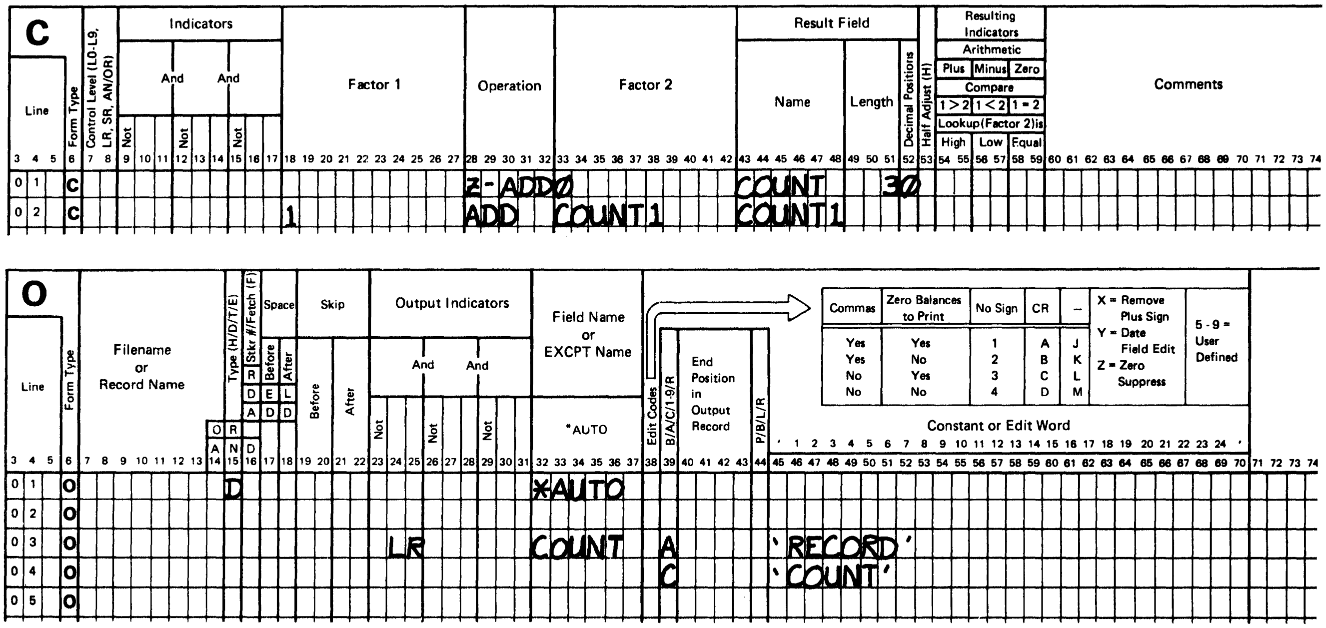
Figure 15-25. Incorrect *AUTO Specifications for More than One Record Type

```
  01 C   11                Z-ADDQTY       QTYA    30
  02 C   11                Z-ADDSALES     SALESA  52
  03 C
```

```
  01 OPRINT   T        L1      *AUTO
  02 O                         DESC         'DESCRIPTION'
  03 O                         QTYA  A      'QUANTITY'
  04 O                         SALESAA      'AMOUNT'
  05 O
```

Figure 15-26. Correct *AUTO Specifications for More than One Record Type

Figure 15-27 shows the specifications for counting records. This method is especially useful when you want to print a detail list, to take totals by control level, or to prevent 1's from being listed down the page.



Calculation Specifications

Line 01 This instruction is needed only to define the field COUNT for accumulation.

Line 02 This instruction accumulates the total for the first control level.

Output Specifications

Line 03 This instruction causes the creation of calculation and output specifications for the detail and total lines. The LR conditioning indicator prevents the created detail calculation from occurring. It also prevents printing at detail time.

Note: If no control levels are specified in the program, a 1 is added to COUNTR rather than to COUNT1 on the calculation specifications.

Figure 15-27. Method of Using *AUTO for Counting Records

Examples of Using Auto Report

Examples 1 through 4 explain how auto report is used to create page headings and such output specifications as column headings, detail lines, and total lines.

Examples 5 and 6 illustrate the use of the auto report copy function to copy specifications from a library member and to change copied specifications for a particular job.

Example 7 prepares a cash receipts register.

Example 1

Problem

Produce the sales report shown below using the *AUTO page headings and *AUTO output functions of auto report.

Procedure

1 Code normal RPG file description and input specifications for the job.

2 Code *AUTO page headings to produce a one-line page heading that includes date and page number.

3 Code *AUTO output to produce one-line column headings, detail report lines, and final totals.

Letters refer to fields on the following page.

```
10/26/78                              SALES REPORT FOR ANY CO.                              PAGE     1
   C        B        A          D                  E            F          G             H
REGION   BRANCH   ITEM      DESCRIPTION        SALES        AMOUNT   ON-HAND          VALUE

   1       17     AG7701T   2-TON TRUCK           5      25,000.00      2         10,000.00
   1       17     AG7705S   PICK-UP              10      20,000.00      1          2,000.00
   1       17     AP6545B   CAMPER                2       8,000.00
   1       22     AG7701T   2-TON TRUCK           2      10,000.00      1          5,000.00
   1       22     AG7705S   PICK-UP               4       8,000.00      1          2,000.00
   3       25     AG6545B   CAMPER               10      40,000.00      5         20,000.00
   3       25     AP6549P   1/4-TON TRUCK        20      30,000.00      6          9,000.00

                                                        141,000.00                48,000.00 *
```

1 Code RPG file description and input specifications.

Line	Form Type	Filename	File Type	File Designation	File Format	Block Length	Record Length	Device
02	F	SALES	I	P	F	473	43	DISK
03	F	PRINTER	O		F	12Ø	12Ø	PRINTER
04	F							

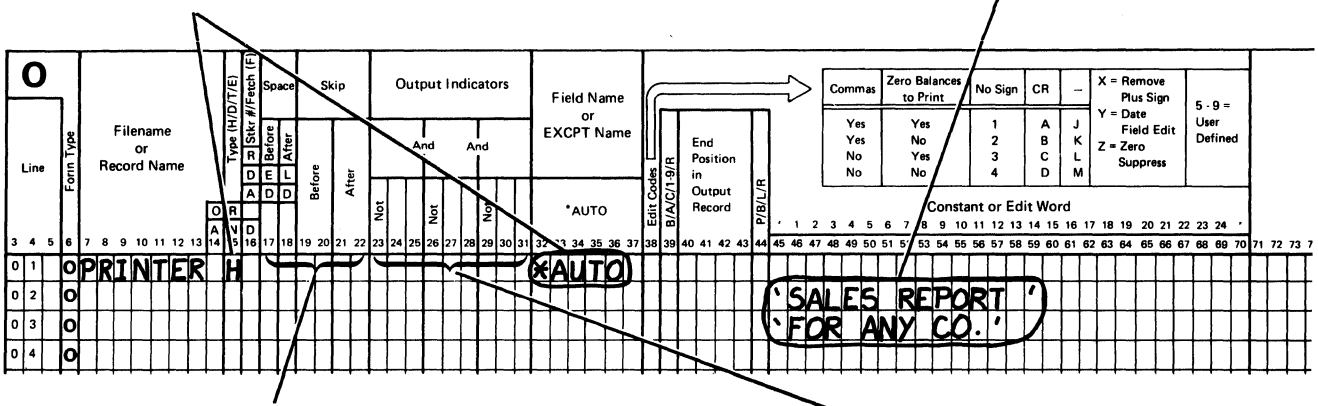
Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator	Field Location From	Field Location To	Decimal Positions	RPG Field Name
01	I	SALES	AA	Ø1				
02	I				1	7		ITEMNO
03	I				8	9		BRANCH
04	I				1Ø	1Ø		REGION
05	I				11	25		DESC
06	I				26	27	Ø	SOLDQY
07	I				28	34	2	SOLDVA
08	I				35	36	Ø	ONHAND
09	I				37	43	2	VALUE
10	I							

Field Name	Contents
A ITEMNO	Item number
B BRANCH	Number of the branch office where the item was sold
C REGION	Sales region in which the branch office is located
D DESC	Description of the sales item
E SOLDQY	Quantity of the item sold
F SOLDVA	Total value of the items sold
G ONHAND	Quantity of the item remaining on hand
H VALUE	Total value of the items remaining on hand.

2 Code *AUTO page-heading specifications.

A Enter an H in column 15 and *AUTO in columns 32 through 36 to request an auto report page heading. Up to five page-heading lines can be described. The system date is printed on the left and the page number on the right of the first heading line on each page. To suppress the date and page, enter an N in column 27 of the auto report option specifications.

B The title information is centered by auto report; do not enter end positions in columns 40 through 43. Fields and table/array elements can also be used.



C When space and skip entries (columns 17 through 22) are left blank, skip to line 06 is assumed for the first heading line; single spacing is done between heading lines, double spacing after the last heading line. (See Example 4 for an example of multiple page-heading lines.)

D When output indicators (columns 23 through 31) are left blank, auto report page headings are printed on each page (conditioned by 1P or overflow). If no overflow indicator is defined for the PRINTER file, auto report assigns an unused overflow indicator to the printer line.

C Line 06

Blank line

A 10/26/83 **B** SALES REPORT FOR ANY CO. **A** PAGE 1

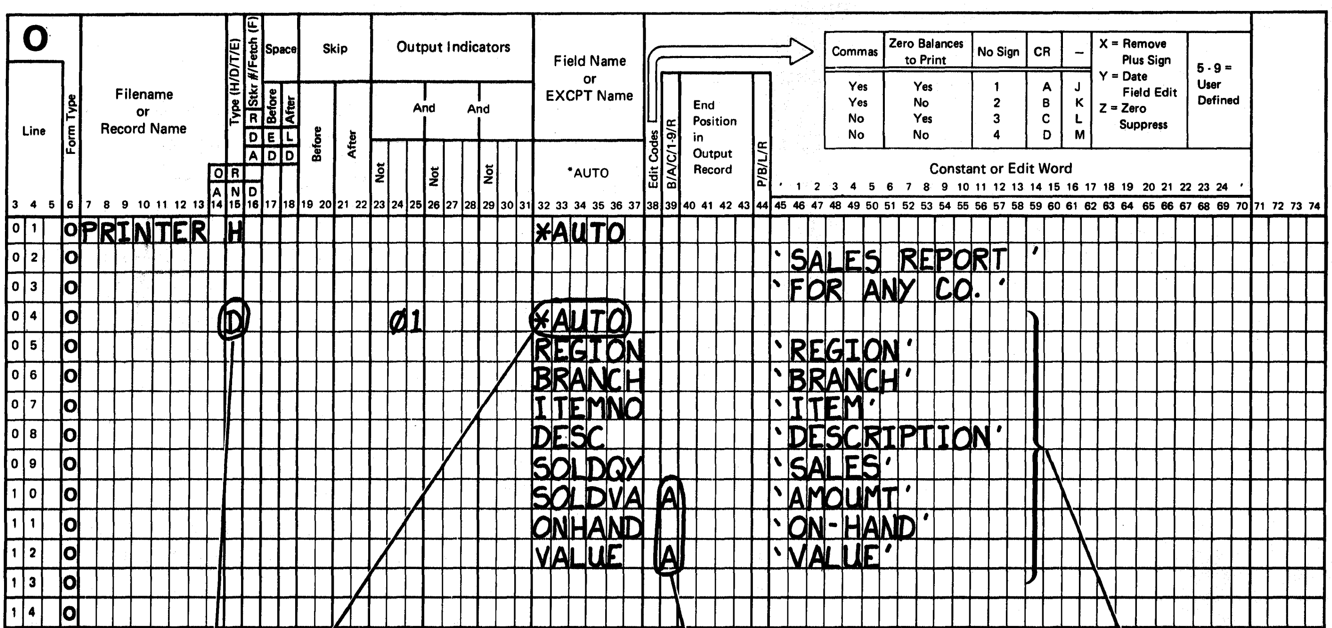
REGION	BRANCH	ITEM	DESCRIPTION	SALES	AMOUNT	ON-HAND	VALUE
1	17	AG7701T	2-TON TRUCK	5	25,000.00	2	10,000.00
1	17	AG7705S	PICK-UP	10	20,000.00	1	2,000.00
1	17	AP6545B	CAMPER	2	8,000.00		
1	22	AG7701T	2-TON TRUCK	2	10,000.00	1	5,000.00
1	22	AG7705S	PICK-UP	4	8,000.00	1	2,000.00
3	25	AG6545B	CAMPER	10	40,000.00	5	20,000.00
3	25	AP6549P	1/4-TON TRUCK	20	30,000.00	6	9,000.00
					141,000.00		48,000.00 *

3 Code *AUTO output specifications to produce:

A Detail report line

B Column headings

C Final totals.



B Column headings are entered on the same lines as the fields over which they appear in the report.

A Enter D in column 15 and *AUTO in columns 32 through 36 to describe an auto report with detail lines. The record-identifying indicator 01 conditions printing of the detail lines.

C Enter an A in column 39 to cause fields to be accumulated. Auto report creates (1) total fields and calculations to accumulate the totals and (2) total output specifications to print the totals.

```
10/26/83                         SALES REPORT FOR ANY CO.                   PAGE      1

REGION  BRANCH  ITEM     DESCRIPTION      SALES      AMOUNT   ON-HAND        VALUE

  1       17    AG7701T  2-TON TRUCK         5    25,000.00      2       10,000.00
  1       17    AG7705S  PICK-UP            10    20,000.00      1        2,000.00
  1       17    AP6545B  CAMPER              2     8,000.00
  1       22    AG7701T  2-TON TRUCK         2    10,000.00      1        5,000.00
  1       22    AG7705S  PICK-UP             4     8,000.00      1        2,000.00
  3       25    AG6545B  CAMPER             10    40,000.00      5       20,000.00
  3       25    AP6549P  1/4-TON TRUCK      20    30,000.00      6        9,000.00

                                                 141,000.00              48,000.00 *
```

Auto report formats the report so that column headings and data are neatly spaced and centered on each other.

All numeric fields for which a blank, B, or A is specified in column 39 are edited by the K edit code unless a different edit code is specified.

Example 2

Problem

Expand sales report from Example 1 to include three levels of totals:

1. Total for each branch
2. Total for each region
3. Final total.

Procedure

1 Code file description and *AUTO speci- fications as in Example 1.

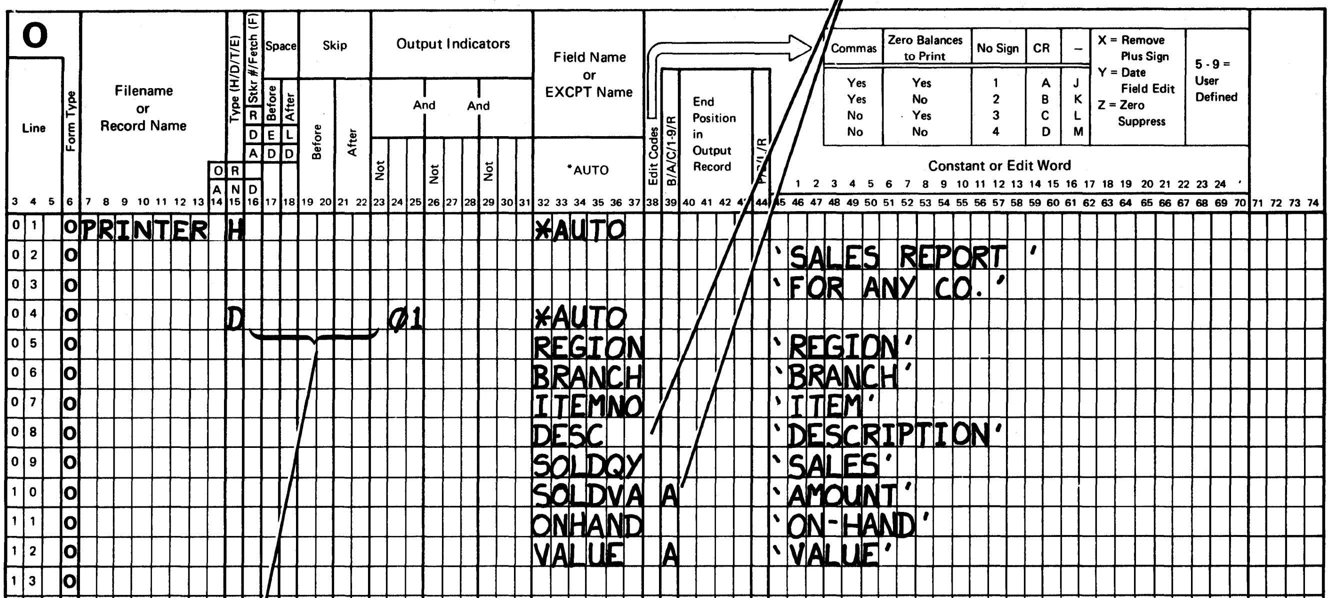
2 Add control-level indicators to the inpu fields BRANCH and REGION.

Note: The *AUTO output function can also be used to produce a group-printed report. See *Group Printing* in this chapter for a discussion and examples of group printing.

Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator	From	To	Decimal Positions	RPG Field Name	Control Level (L1-L9)	Field Record Relation
01	I	SALES	AA	Ø1						
02	I				1	7		ITEMNO		
03	I				8	9		BRANCH	L1	2
04	I				1Ø	1Ø		REGION	L2	
05	I				11	25		DESC		
06	I				26	27	Ø	SOLDQY		
07	I				28	34	2	SOLDVA		
08	I				35	36	Ø	ONHAND		
09	I				37	43	2	VALUE		
10	I									

Because two control-level indicators are defined, the SOLDVA and VALUE fields (see following page) are accumulated to two levels of totals (branch and region) and a final total (LR).

As in Example 1, an A in column 39 of the output specification causes SOLDVA and VLAUE to be accumulated.



Auto report places a blank line after each total line and an additional blank line before the lowest-level total and before and final total. If you enter spacing and skipping values on the D-*AUTO specification, they apply to the detail print line only.

Auto report prints asterisks (*) to the right of created total lines to aid in identifying them. If you want to suppress the asterisks, enter N in column 28 of the auto report option specifications.

```
10/26/83                          SALES REPORT FOR ANY CO.                   PAGE    1

REGION  BRANCH   ITEM     DESCRIPTION      SALES      AMOUNT   ON-HAND        VALUE
                 NUMBER

   1      17     AG7701T  2-TON TRUCK        5     25,000.00       2      10,000.00
   1      17     AG7705S  PICK-UP           10     20,000.00       1       2,000.00
   1      17     AP6545B  CAMPER             2      8,000.00

                                                   53,000.00              12,000.00 *     L1

   1      22     AG7701T  2-TON TRUCK        2     10,000.00       1       5,000.00
   1      22     AG7705S  PICK-UP            4      8,000.00       1       2,000.00

                                                   18,000.00               7,000.00 *     L1

                                                   71,000.00              19,000.00 **    L2

   3      25     AG6545B  CAMPER            10     40,000.00       5      20,000.00
   3      25     AP6549P  1/4-TON TRUCK     20     30,000.00       6       9,000.00

                                                   70,000.00              29,000.00 *     L1

                                                   70,000.00              29,000.00 **    L2

                                                  141,000.00              48,000.00 ***   LR
```

Total fields are always two positions longer than the original fields and have the same number of decimal positions as the original fields.

Example 3

Problem

Expand the sales report from Examples 1 and 2 to contain:

- **A** Group indication for REGION and BRANCH fields
- **B** Second column-heading line
- **C** Literal (constant) on the final total line.

Procedure

1 Code file description and input specifications as for Example 2.

2 Code *AUTO output with:

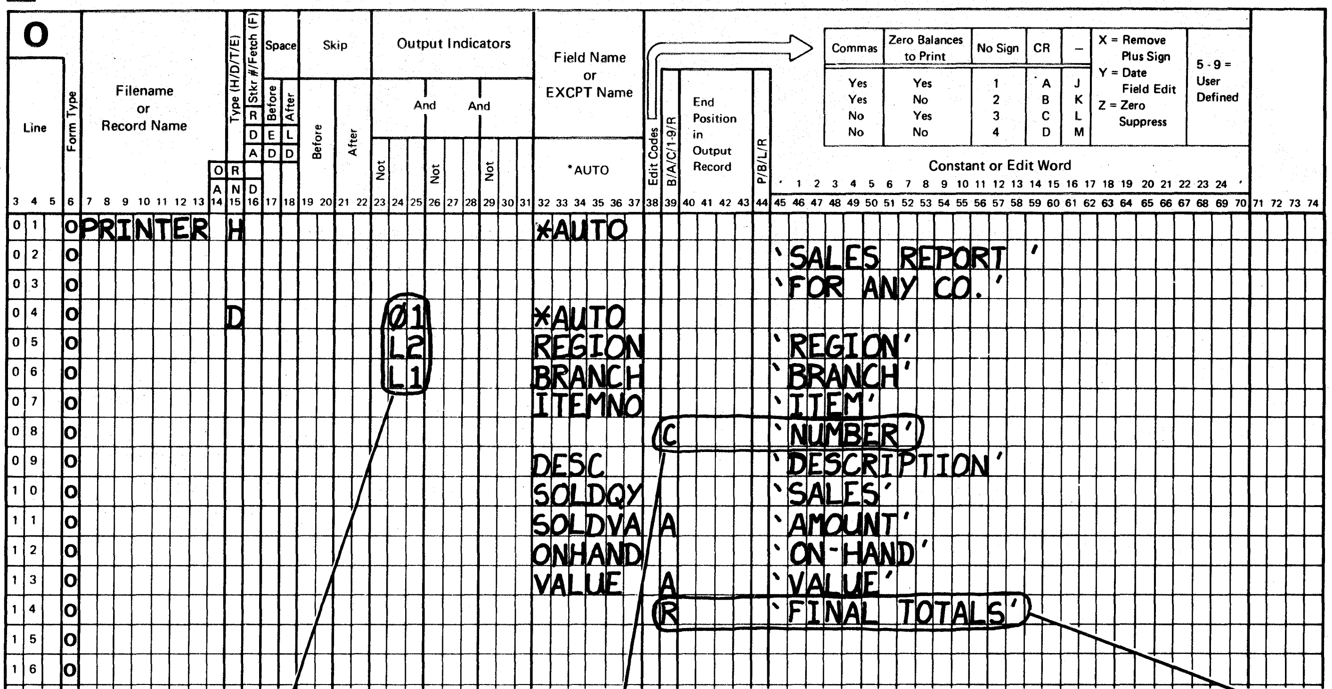
- **A** Output indicator on field description specifications
- **B** C in column 39 and a literal in columns 45 through 70
- **C** R in column 39 and a literal in columns 45 through 70.

1

Line	Form Type	Filename	File Type	File Designation	File Format	Block Length	Record Length	Device
02	F	SALES	I	P	F	473	43	DISK
03	F	PRINTER	O		F	12Ø	12Ø	PRINTER
04	F							

Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator	Field Location From	Field Location To	Decimal Positions	RPG Field Name	Control Level (L1-L9)
01	I	SALES	AA	Ø1					
02	I				1	7		ITEMNO	
03	I				8	9		BRANCH	L1
04	I				1Ø	1Ø		REGION	L2
05	I				11	25		DESC	
06	I				26	27	Ø	SOLDQY	
07	I				28	34	2	SOLDVA	
08	I				35	36	Ø	ONHAND	
09	I				37	43	2	VALUE	
10	I								

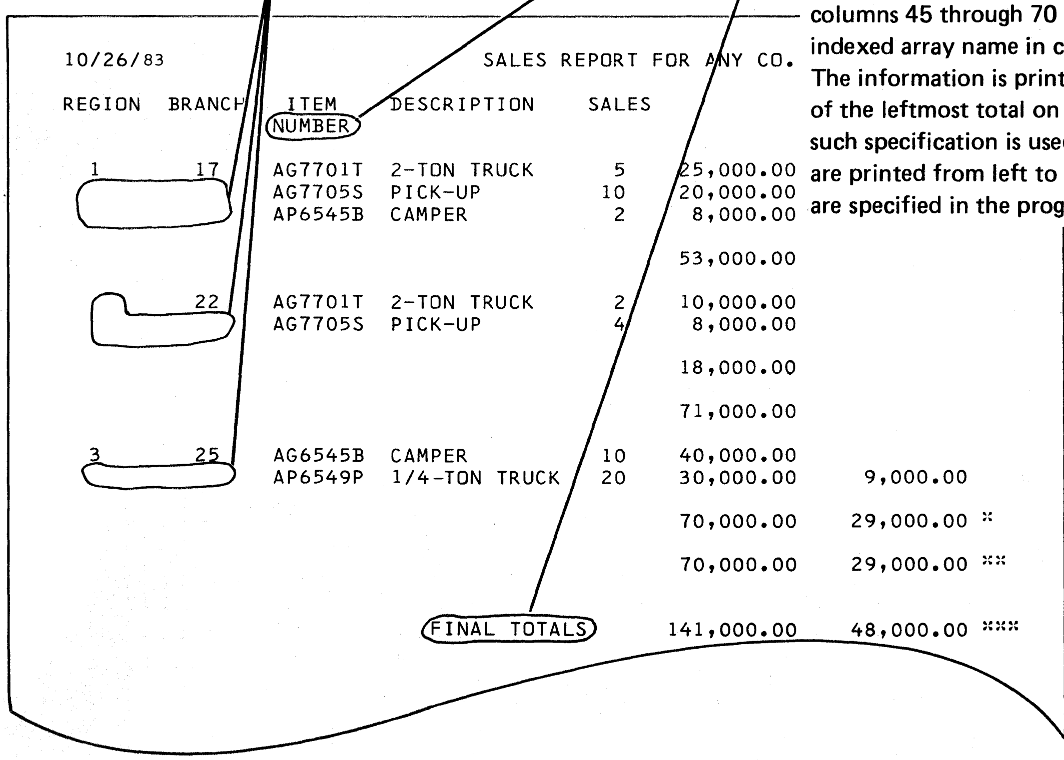
2



A Output indicators can be used on field description specifications. In this example, control-level indicators condition BRANCH and REGION so that they are printed only for the first record of the corresponding control group. This print-suppressing of common fields (group indication) reduces repetitive information.

B One or two additional column-heading lines can be specified by a C entry in column 39 with the heading information in columns 45 through 70.

C The literal FINAL TOTALS makes that line easy to find. To specify information to appear on the final total line, enter R in column 39 with a literal in columns 45 through 70 or a field name/table name/indexed array name in columns 32 through 37. The information is printed two spaces to the left of the leftmost total on the line. If more than one such specification is used, the literals and fields are printed from left to right in the order they are specified in the program.



Example 4

Problem

Expand the sales report from Examples 1, 2, and 3 to include a cross-totals column and:

- A A new report page for each region
- B Two heading lines on each page
- C A field in a page-heading line
- D Identification of branch and region totals.

Procedure

1 Code file description and input specifications as in Example 3; add an overflow indicator to the PRINTER file.

2 Code RPG calculation specifications for cross-total.

3 Code *AUTO specifications:

- A Output indicators on page-heading specifications
- B Two heading lines per page
- C Use of a field in an *AUTO page-heading specification
- D Fields and literals on L1 through L9 total lines (1 through 9 in column 39).

```
 11/18/83                            SALES REPORT FOR ANY CO.                            PAGE    1
                                  B —   REGION 1   — C                                 2
 BRANCH    ITEM    DESCRIPTION      SALES      SALES VALUE   ON   ON-HAND VALUE             TOTAL
          NUMBER                   QUANTITY                  HAND

   17     AG7701T  2-TON TRUCK        5         25,000.00    2       10,000.00          35,000.00
          AG7705S  PICK-UP           10         20,000.00    1        2,000.00          22,000.00
          AP6545B  CAMPER             2          8,000.00                                8,000.00

                         D  BRANCH 17 TOTALS    53,000.00            12,000.00          65,000.00 *

   22     AG7701T  2-TON TRUCK        2         10,000.00    1        5,000.00          15,000.00
          AG7705S  PICK-UP            4          8,000.00    1        2,000.00          10,000.00

                            BRANCH 22 TOTALS    18,000.00             7,000.00          25,000.00 *

                         D  REGION 1 TOTALS     71,000.00            19,000.00          90,000.00 **
```

```
 11/18/78                            SALES REPORT FOR ANY CO.                          A PAGE    2
                                        REGION 3

 BRANCH    ITEM    DESCRIPTION      SALES      SALES VALUE   ON   ON-HAND VALUE             TOTAL
          NUMBER                   QUANTITY                  HAND

   25     AG6545B  CAMPER            10         40,000.00    5       20,000.00          60,000.00
          AP6549P  1/4-TON TRUCK.    20         30,000.00    6        9,000.00          39,000.00

                            BRANCH 25 TOTALS    70,000.00            29,000.00          99,000.00 *

                            REGION 3 TOTALS     70,000.00            29,000.00          99,000.00 **

                            COMPANY TOTALS     141,000.00            48,000.00         189,000.00 ***
```

Note: Compare matching letters (B) on this and the following pages to see the auto report coding to obtain this report

2

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust	Resulting Indicators	Comments
01	C		Ø1	SOLDVA	ADD	VALUE	TOTVAL	8	2			
02	C											

2 RPG calculations can be among the input statements for auto report. This specification calculates a cross-total of the sales and on-hand values. (The placement of the calculation in relation to calculations created by auto report is described under *Created Specifications*.)

3

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Output Indicators	Field Name or EXCPT Name (*AUTO)	Edit Codes	B/A/C/1-9/R	End Position in Output Record	Constant or Edit Word
01	O	PRINTER	H	L2	*AUTO				
02	O	OR		OFNL2					
03	O								'SALES REPORT '
04	O								'FOR ANY CO.'
05	O		H		*AUTO				
06	O								'REGION'
07	O				REGION				
08	O		D	Ø1	*AUTO				
09	O			L1	BRANCH				'BRANCH'
10	O				ITEMNO				'ITEM'
11	O						C		'NUMBER'
12	O				DESC				'DESCRIPTION
13	O				SOLDQY				'SALES'
14	O						C		'QUANTITY'
15	O				SOLDVA		A		'SALES VALUE'
16	O				ONHAND				'ON'
17	O						C		'HAND'
18	O				VALUE		A		'ON-HAND VALUE'
19	O				TOTVAL		A		'TOTAL'
20	O						1		'BRANCH'
	O				BRANCH		1		
	O						1		'TOTALS'
	O						2		'REGION'
	O				REGION		2		
	O						2		'TOTALS'
01	O						R		'COMPANY TOTALS'
02	O								
03	O								
04	O								
05	O								
06	O								
07	O								

A The headings are printed on a new page when the region number changes (L2) or when overflow occurs (OF). (OF must be defined for the PRINTER file in the file description specifications.)

B A second auto report page heading is specified. Because spacing is not specified, space-one is done after the first and space-two after the second. Because no output indicators are specified, the second heading is conditioned like the first.

C The contents of the REGION field are printed on the second page heading.

D Fields and literals can be printed on created total lines if you enter the number of the control level in column 39.

Example 5

Problem

Use the copy function to obtain specifications for the sales report below (same as in Example 1).

Procedure

1 Put the file description and input specifications for the SALES file in a library member.

2 Code the /COPY statement in the specifi cations for auto report.

```
10/26/83                    SALES REPORT FOR ANY CO.                           PAGE     1

REGION   BRANCH   ITEM      DESCRIPTION       SALES        AMOUNT   ON-HAND        VALUE

  1        17     AG7701T   2-TON TRUCK         5       25,000.00      2        10,000.00
  1        17     AG7705S   PICK-UP            10       20,000.00      1         2,000.00
  1        17     AP6545B   CAMPER              2        8,000.00
  1        22     AG7701T   2-TON TRUCK         2       10,000.00      1         5,000.00
  1        22     AG7705S   PICK-UP             4        8,000.00      1         2,000.00
  3        25     AG6545B   CAMPER             10       40,000.00      5        20,000.00
  3        25     AP6549P   1/4-TON TRUCK      20       30,000.00      6         9,000.00

                                                       141,000.00               48,000.00 *
```

1 Put specifications for the SALES file in a library member by using the source entry utility (SEU).

Line	Form Type	Filename	File Type	File Designation	File Format	Block Length	Record Length	Device
02	F	SALES	I	P	F	473	43	DISK
03	F	PRINTER	O		F	120	120	PRINTER
04	F							

These specifications could be replaced by a single statement as shown on the following page if they were put in a library member.

Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator	Field Location From	Field Location To	Decimal Positions	RPG Field Name
01	I	SALES	AA	01				
02	I				1	7		ITEMNO
03	I				8	9		BRANCH
04	I				10	10		REGION
05	I				11	25		DESC
06	I				26	27	0	SOLDQY
07	I				28	34	2	SOLDVA
08	I				35	36	0	ONHAND
09	I				37	43	2	VALUE
10	I							

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Output Indicators	Field Name or EXCPT Name	Edit Codes	Constant or Edit Word
01	O	PRINTER	H		*AUTO		
02	O						'SALES REPORT '
03	O						'FOR ANY CO. '
04	O		D	01	*AUTO		
05	O				REGION		'REGION'
06	O				BRANCH		'BRANCH'
07	O				ITEMNO		'ITEM'
08	O				DESC		'DESCRIPTION'
09	O				SOLDQY		'SALES'
10	O				SOLDVA	A	'AMOUNT'
11	O				ONHAND		'ON-HAND'
12	O				VALUE	A	'VALUE'
13	O						

2 Code the /COPY statement to include the file description and input specifications. (For a detailed description of the copy function, see */COPY Statement Specifications.*)

```
F
Line  Form Type  Filename  File Type  File Designation  End of File  Sequence  File Format  Block Length  Record Length  Device  Symbolic Device
02 FPRINTER O   F 120 120            PRINTER
03 F
```

```
I
Line  Form Type  Filename or Record Name
01 I/COPY F1,SALETR
02 I
```

Column 6 of a /COPY statement must not contain a U or an H.

The source member is in the system library.

The /COPY statement copies file description and input specifications for the SALES file from the library member named SALETR.

```
O
Line  Form Type  Filename or Record Name  Type  Output Indicators  Field Name or EXCPT Name  Edit Codes  End Position in Output Record  Constant or Edit Word
01 OPRINTER H                *AUTO
02 O                                   'SALES REPORT'
03 O                                   'FOR ANY CO.'
04 O       D        01       *AUTO
05 O                         REGION     'REGION'
06 O                         BRANCH     'BRANCH'
07 O                         ITEMNO     'ITEM'
08 O                         DESC       'DESCRIPTION'
09 O                         SOLDQY     'SALES'
10 O                         SOLDVA A   'AMOUNT'
11 O                         ONHAND     'ON-HAND'
12 O                         VALUE  A   'VALUE'
13 O
```

Example 6

Problem

Override copied input specifications to produce a report (below) that includes subtotals for branch and region.

Procedure

1. Put the specifications for the SALES file in a library member, as in Example 5.
2. Code the /COPY statement.
3. Code /COPY modifier statements to add control-level indicators to BRANCH and REGION fields on copied specifications.

```
10/26/83                          SALES REPORT FOR ANY CO.                                 PAGE      1

REGION    BRANCH    ITEM       DESCRIPTION       SALES         AMOUNT    ON-HAND          VALUE
                   NUMBER

  1         17     AG7701T     2-TON TRUCK         5       25,000.00        2         10,000.00
                   AG7705S     PICK-UP            10       20,000.00        1          2,000.00
                   AP6545B     CAMPER              2        8,000.00

                                                           53,000.00                  12,000.00 *

            22     AG7701T     2-TON TRUCK         2       10,000.00        1          5,000.00
                   AG7705S     PICK-UP             4        8,000.00        1          2,000.00

                                                           18,000.00                   7,000.00 *

                                                           71,000.00                  19,000.00 **

  3         25     AG6545B     CAMPER             10       40,000.00        5         20,000.00
                   AG6549P     1/4-TON TRUCK      20       30,000.00        6          9,000.00
```

To produce a report that has subtotals for branch and region, L1 must be assigned to BRANCH and L2 to REGION as the specifications are copied from the library member.

Put input specifications for the SALES file in a source member.

Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator	Field Location From	Field Location To	Decimal Positions	RPG Field Name	Control Level (L1-L9)
01	I	SALES	AA	Ø1					
02	I				1	7		ITEMNO	
03	I				8	9		BRANCH	
04	I				1Ø	1Ø		REGION	
05	I				11	25		DESC	
06	I				26	27	Ø	SOLDQY	
07	I				28	34	2	SOLDVA	
08	I				35	36	Ø	ONHAND	
09	I				37	43	2	VALUE	
10	I								

2 and **3** Code/COPY and modifier statements. As a result of the modifier statements, three levels of totals are accumulated for the SOLDVA and VALUE fields (L1, L2, and LR).

Line (3–5)	Form Type (6)	Filename (7–14)	File Type (15)	File Designation (16)	End of File (17)	Sequence (18)	File Format (19)	Block Length (20–23)	Record Length (24–27)	Device (40–46)
02	F	PRINTER	O				F	120	120	PRINTER
03	F									

Line (3–5)	Form Type (6)	Filename or Record Name (7–14)	RPG Field Name (53–58)	Control Level (L1-L9) (59–60)
01	I	/COPY F1,SALETR		
02	I		BRANCH	L1
03	I		REGION	L2
04	I			

Entries on the modifier statements override the corresponding entries in the copied specifications. The field names, BRANCH and REGION, identify the input field specifications that are to be changed.

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Output Indicators	Field Name or EXCPT Name	B/A/C/1-9/R	Constant or Edit Word
01	O	PRINTER	H		*AUTO		
02	O						'SALES REPORT '
03	O						'FOR ANY CO.'
04	O		D	01	*AUTO		
05	O				REGION		'REGION'
06	O				BRANCH		'BRANCH'
07	O				ITEMNO		'ITEM'
08	O				DESC		'DESCRIPTION'
09	O				SOLDQY		'SALES'
10	O				SOLDVA	A	'AMOUNT'
11	O				ONHAND		'ON-HAND'
12	O				VALUE	A	'VALUE'
13	O						

File description or input specifications in the copied member are overridden as follows (see */COPY Statement Specifications* for examples):

- Entries in a modifier statement override corresponding entries in a copied file description or input field specification.
- Blank entries in a modifier statement remain unchanged in a copied specification.
- Ampersand (&) in the leftmost position of an entry in the modifier statement sets the entry to blanks in the copied specification.
- New fields can be added to input specifications by new input field specifications added as modifier statements.
- Modifier statements do not change the specifications in the copied source member. The modification is only for the program into which the specifications are copied.

Example 7

Example 7 prepares a cash receipts register. The *AUTO page heading function and the *AUTO output function create the RPG output specifications for the report and the calculation specifications to accumulate final totals for several fields on the report. RPG calculation specifications that cannot be created by auto report are included in the auto report program to verify the discount taken by each customer and to calculate the balance due.

The file description specifications for the cash receipts register PRINTER file, CSHRECRG, and the file description and input specifications for the input file, CASHRC, are located in separate members in the library (see Figure 15-28). These specifications are included in the program by the auto report copy function.

The input data for the file CASHRC in EXAUT2 is created by the program EXAUT1 (see Figure 15-29). Figure 15-30 shows the input data.

1 The file description for the PRINTER file, CSHRECRG, is in the library member named EXAUT3.

2 The file description and input specifications for the DISK file, CASHRC, are in the library member named EXAUT4.

Line	Form Type	Filename	File Type	File Designation	End of File	File Format	Block Length	Record Length	Overflow Indicator	Device
1	F	CSHRECRG	O			F	132	132	OA	PRINTER
03	F									
2	F	CASHRC	I	P	E	F	1020	68		DISK
05	F									

Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator	Position	C/Z/D	Character	From	To	Decimal Positions	RPG Field Name	
01	I	CASHRC	AA	01	68	C	9					
02	I											
03	I							1	5		ACCTNO	Account number
04	I							6	25		ACCTNM	Account name
05	I							26	30	0	INVNO	Invoice number
2	I							31	36	0	INVDAT	Invoice date
07	I							37	42	2	AMTOWD	Amount owed
08	I							47	51	2	DISCAL	Discount allowed
09	I							52	56	2	DISTAK	Discount taken
10	I							57	62	2	AMTPD	Amount paid
11	I							63	68	0	DATPD	Date paid
12	I											

Figure 15-28. File Description and Input Specifications That Are in the Library Members EXAUT3 and EXAUT4

```
RG 004    H                                                                R1586B

          FCASHRC  O     1020  68            DISK
          C                        SETON                  LR
          OCASHRC  T          LR
          O                                    24 '11243JONES HARDWARE     '
          O                                    48 ' 27541123199  2375CASH  '
          O                                    68 ' 47    47  2328123199'
          O        T          LR
          O                                    24 '11352NU-STYLE CLOTHIERS '
          O                                    48 ' 27987123199  8707CASH  '
          O                                    68 '174        4000123199'
          O        T          LR
          O                                    24 '11886MIDI FASHIONS INC  '
          O                                    48 ' 15771123199 10722CASH  '
          O                                    68 '214   214 10508123199'
          O        T          LR
          O                                    24 '12874ULOOK INTERIORS    '
          O                                    48 ' 25622123199  6795CASH  '
          O                                    68 '136        6795123199'
          O        T          LR
          O                                    24 '18274STREAMLINE PAPER IN'
          O                                    48 'C29703123199 27403CASH  '
          O                                    68 '548   238 17055123199'
          O        T          LR
          O                                    24 '23347RITE-BEST PENS CO  '
          O                                    48 ' 20842123199  1580      '
          O                                    68 '31         1000123199'
          O        T          LR
          O                                    24 '25521IMPORTS OF NM      '
          O                                    48 ' 29273123199 79740     1'
          O                                    68 '593  1193 58547123199'
          O        T          LR
          O                                    24 '26723ALRIGHT CLEANERS   '
          O                                    48 ' 19473123199 46200CASH  '
          O                                    68 '924       46200123199'
          O        T          LR
          O                                    24 '28622NORTH CENTRAL SUPPL'
          O                                    48 'Y17816123199  7597CASH  '
          O                                    68 '152        7597123199'
          O        T          LR
          O                                    24 '29871FERGUSON DEALERS   '
          O                                    48 ' 27229123199  6191CASH  '
          O                                    68 '124        6191123199'
          O        T          LR
          O                                    24 '30755FASTWAY AIRLINES   '
          O                                    48 ' 26158123199 74272CASH 1'
          O                                    68 '495  1685 72587123199'
          O        T          LR
          O                                    24 '31275ENVIRONMENT CONCERN'
          O                                    48 'S20451123199  2943      '
          O                                    68 ' 59        1500123199'
          O        T          LR
          O                                    24 '32457B SOLE SILOS       '
          O                                    48 ' 27425123199 11005CASH  '
          O                                    68 '220       11005123199'
          O        T          LR
          O                                    24 '37945HOFFTA BREAKS INC  '
          O                                    48 ' 18276123199  4723CASH  '
          O                                    68 ' 94        4723123199'
          O        T          LR
          O                                    24 '42622EASTLAKE GRAVEL CO '
          O                                    48 ' 16429123199  2937CASH  '
          O                                    68 ' 58        2937123199'
```

Figure 15-29. EXAUT1 Program

DATA FOR SAMPLE PROGRAM

```
11243JONES HARDWARE      27541123199  2375CASH   47   47  2328123199
11352NU-STYLE CLOTHIERS  27987123199  8707CASH  174       4000123199
11886MIDI FASHIONS INC   15771123199 10722CASH  214  214 10508123199
12874ULOOK INTERIORS     25622123199  6795CASH  136       6795123199
18274STREAMLINE PAPER INC29703123199 27403      548  238 17055123199
23347RITE-BEST PENS CO   20842123199  1580      31        1000123199
25521IMPORTS OF NM       29273123199 79740     1593 1193 58547123199
26723ALRIGHT CLEANERS    19473123199 46200CASH  924      46200123199
28622NORTH CENTRAL SUPPLY17816123199  7597CASH  152       7597123199
29871FERGUSON DEALERS    27229123199  6191CASH  124       6191123199
30755FASTWAY AIRLINES    26158123199 74272CASH 1495 1685 72587123199
31275ENVIRONMENT CONCERNS20451123199  2943       59       1500123199
32457B SOLE SILOS        27425123199 11005CASH  220      11005123199
37945HOFFTA BREAKS INC   18276123199  4723CASH   94       4723123199
42622EASTLAKE GRAVEL CO  16429123199  2937CASH   58       2937123199
```

Figure 15-30. Input Data Created by EXAUT1 for Auto Report Sample Program EXAUT2

Control Specification

The RPG control specification shown in Figure 15-31 should be included in the auto report program because it is not present in the copied specifications (see Figure 15-28). None of the control specification options are required in this program, so the specification need contain only an H in column 6 and the program identification, EXAUT2, in columns 75 through 80.

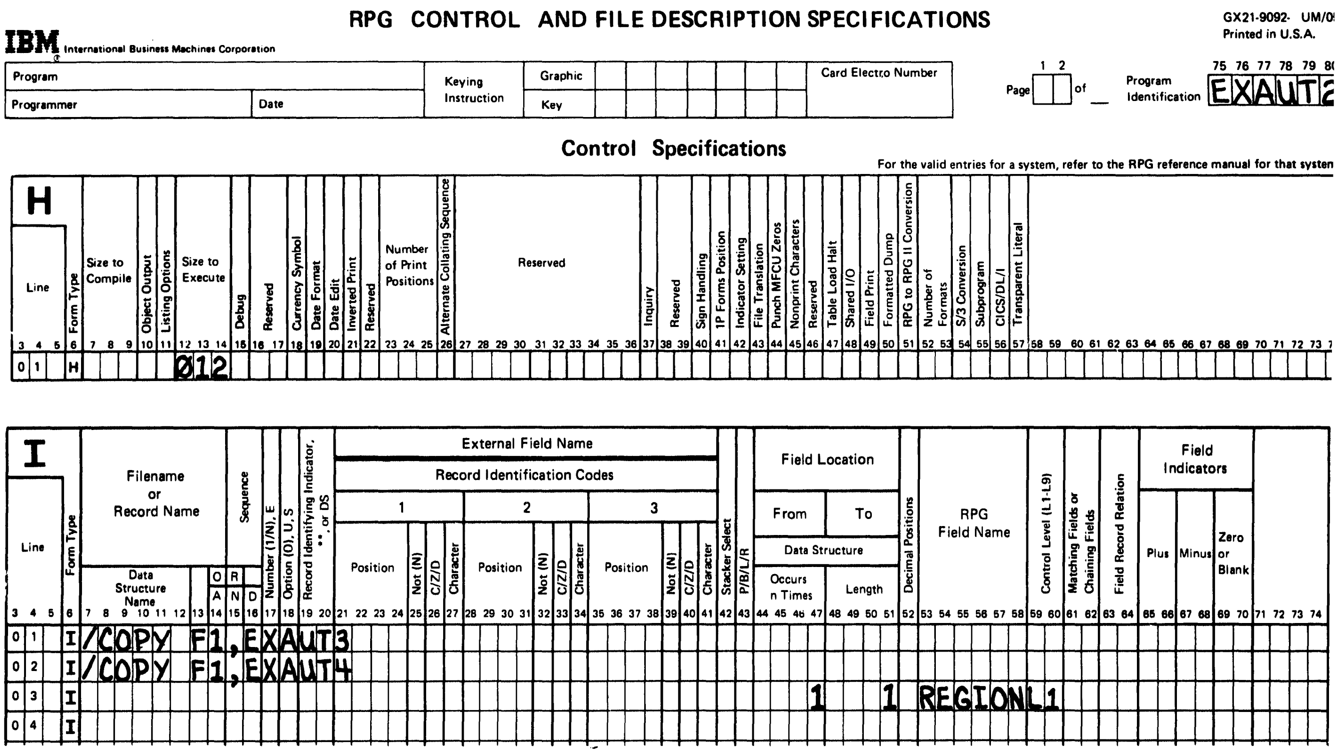


Figure 15-31 (Part 1 of 2). RPG and Auto Report Specifications to Produce the Cash Receipts Register

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust	Resulting Indicators Plus (1>2, High)	Minus (1<2, Low)	Zero (1=2, Equal)	Comments
01	C			DISTAK	SUB	DISCAL	DIFF	4	2					
02	C			DIFF	COMP	1.ØØ					1Ø		1Ø	
03	C			AMTOWD	SUB	DISTAK	NETOWD	6	2					
04	C			NETOWD	SUB	AMTPD	BAL	6	2					
05	C													

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Space/Skip	Output Indicators	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	CSHRECRG	H			*AUTO					
02	O										'CASH RECEIPTS REGISTER'
03	O		D		Ø1	*AUTO					
04	O					REGION					'REGION'
05	O					ACCTNO					'ACCOUNT'
06	O							C			'NUMBER'
07	O					ACCTNM					'ACCOUNT NAME'
08	O					INVNO	3				'INVOICE'
09	O							C			'NUMBER'
10	O					INVDAT	Y				'INVOICE'
11	O							C			'DATE'
12	O					DATPD	Y				'DATE PAID'
13	O					AMTOWD	J	A			'AMOUNT'
14	O							C			'OWED'
15	O					DISTAK		A			'DISCOUNT'
16	O							C			'TAKEN'
17	O					AMTPD		A			'AMOUNT'
18	O							C			'PAID'
19	O					BAL		A			'BALANCE'
20	O							C			'DUE'
	O				1Ø	DIFF		A			'EXCESS'
	O							C			'DISCOUNT'
	O							1			'REGION TOTALS'
	O							R			'COMPANY TOTALS'
	O										

Figure 15-31 (Part 2 of 2). RPG and Auto Report Specifications to Produce the Cash Receipts Register

/COPY Statements

The /COPY statements shown in Figure 15-31 copy the file description and input specifications for the job from the source members in the system library. The first statement copies the file description specifications for the PRINTER file from the library member named EXAUT3. The second statement copies the file description and input specifications for the DISK file named CASHRC from the library member named EXAUT4. A modifier statement adds an input field definition for the REGION field. As a result of these /COPY statements, the file description and input specifications shown in Figure 15-28 are included in the RPG source program created by auto report.

Calculation Specifications

The calculation specifications shown in Figure 15-31 are included in the auto report program to perform special operations that cannot be created by auto report. First, the discount allowed for each customer is subtracted from the discount taken by each customer. Indicator 10 turns on if the difference is $1.00 or more. The remaining calculations subtract the discount taken and the amount paid from the amount owed.

The order in which these calculations are placed in relation to the calculations created by auto report is shown in the auto report listing of the created RPG source program (see Figure 15-32).

R1586

```
H     012
I/COPY F1,EXAUT3
FCSHRECRGO   F 132 132     OA     PRINTER
I/COPY F1,EXAUT4
FCASHRC  IPE F1020  68            DISK
ICASHRC  AA  01  68 C9
I                                        1   5 ACCTNO
I                                        6  25 ACCTNM
I                                       26  300INVNO
I                                       31  360INVDAT
I                                       37  422AMTOWD
I                                       47  512DISCAL
I                                       52  562DISTAK
I                                       57  622AMTPD
I                                       63  680DATPD
I                                        1   1 REGIONL1
C           DISTAK    SUB  DISCAL    DIFF     42
C           DIFF      COMP 1.00                 10  10
C           AMTOWD    SUB  DISTAK    NETOWD   62
C           NETOWD    SUB  AMTPD     BAL      62
OCSHRECRGH                *AUTO
O                                      'CASH RECEIPTS REGISTER'
O        D        01      *AUTO
O                         REGION       'REGION'
O                         ACCTNO       'ACCOUNT'
O                                C     'NUMBER'
O                         ACCTNM       'ACCOUNT NAME'
O                         INVNO 3      'INVOICE'
O                                C     'NUMBER'
O                         INVDATY      'INVOICE'
O                                C     'DATE'
O                         DATPD Y      'DATE PAID'
O                         AMTOWDJA     'AMOUNT'
O                                C     'OWED'
O                         DISTAK A     'DISCOUNT'
O                                C     'TAKEN'
O                         AMTPD  A     'AMOUNT'
O                                C     'PAID'
O                         BAL    A     'BALANCE'
O                                C     'DUE'
O                 10      DIFF   A     'EXCESS'
O                                C     'DISCOUNT'
O                                1     'REGION TOTALS'
O                                R     'COMPANY TOTALS'
```

Figure 15-32 (Part 1 of 3). Auto Report Sample Program EXAUT2

```
     0010 H     012                                                              R1586

0001 0020CFCSHRECRGO   F 132 132      OA      PRINTER
0002 0030CFCASHRC  IPE F1020  68              DISK


     0040 I*/COPY F1,EXAUT3
     0050 I*/COPY F1,EXAUT4
0003 0060CICASHRC  AA  01   68 C9
0004 0070CI                                               1   5 ACCTNO
0005 0080CI                                               6  25 ACCTNM
0006 0090CI                                              26  300INVNO
0007 0100CI                                              31  360INVDAT
0008 0110CI                                              37  422AMTOWD
0009 0120CI                                              47  512DISCAL
0010 0130CI                                              52  562DISTAK
0011 0140CI                                              57  622AMTPD
0012 0150CI                                              63  680DATPD
0013 0160 I                                               1   1 REGIONL1


0014 0170 C            DISTAK    SUB  DISCAL    DIFF     42
0015 0180 C            DIFF      COMP 1.00                  10  10
0016 0190 C            AMTOWD    SUB  DISTAK    NETOWD   62
0017 0200 C            NETOWD    SUB  AMTPD     BAL      62
0018 0210EC    01                EXSR A$$SUM
0019 0220ECL1          AMTOWR    ADD  AMTOW1    AMTOWR   82
0020 0230ECL1          DISTAR    ADD  DISTA1    DISTAR   72
0021 0240ECL1          AMTPDR    ADD  AMTPD1    AMTPDR   82
0022 0250ECL1          BALR      ADD  BAL1      BALR     82
0023 0260ECL1          DIFFR     ADD  DIFF1     DIFFR    62
0024 0270ECSR          A$$SUM    BEGSR
0025 0280ECSR          AMTOW1    ADD  AMTOWD    AMTOW1   82
0026 0290ECSR          DISTA1    ADD  DISTAK    DISTA1   72
0027 0300ECSR          AMTPD1    ADD  AMTPD     AMTPD1   82
0028 0310ECSR          BAL1      ADD  BAL       BAL1     82
0029 0320ECSR 10       DIFF1     ADD  DIFF      DIFF1    62
0030 0330ECSR                    ENDSR
```

Figure 15-32 (Part 2 of 3). Auto Report Sample Program EXAUT2

```
0340EOCSHRECRGH  206    1P
0350EO        OR        OA
0360EO                                    76 'CASH RECEIPTS REGISTER'
0370EO                          UDATE Y    8
0380EO                          PAGE  Z  131
0390EO                                   127 'PAGE '
0400EOCSHRECRGH  1      1P
0410EO        OR        OA
0420EO                                     6 'REGION'
0430EO                                    15 'ACCOUNT'
0440EO                                    29 'ACCOUNT NAME'
0450EO                                    46 'INVOICE'
0460EO                                    56 'INVOICE'
0470EO                                    67 'DATE PAID'
0480EO                                    80 'AMOUNT'
0490EO                                    92 'DISCOUNT'
0500EO                                   105 'AMOUNT'
0510EO                                   118 'BALANCE'
0520EO                                   128 'EXCESS'
0530EOCSHRECRGH  2      1P
0540EO        OR        OA
0550EO                                    14 'NUMBER'
0560EO                                    45 'NUMBER'
0570EO                                    54 'DATE'
0580EO                                    79 'OWED'
0590EO                                    90 'TAKEN'
0600EO                                   104 'PAID'
0610EO                                   116 'DUE'
0620EO                                   129 'DISCOUNT'
0630EOCSHRECRGD  1      01
0640EO                          REGION     3
0650EO                          ACCTNO    14
0660EO                          ACCTNM    37
0670EO                          INVNO 3   45
0680EO                          INVDATY   56
0690EO                          DATPD Y   66
0700EO                          AMTOWDJB  80
0710EO                          DISTAKKB  92
0720EO                          AMTPD KB 105
0730EO                          BAL   KB 118
0740EO                  10      DIFF  KB 129
0750EOCSHRECRGT 12      L1
0760EO                          AMTOW1JB  80
0770EO                          DISTA1KB  92
0780EO                          AMTPD1KB 105
0790EO                          BAL1  KB 118
0800EO                          DIFF1 KB 129
0810EO                                    67 'REGION TOTALS'
0820EO                                   130 '*'
0830EOCSHRECRGT 12      LR
0840EO                          AMTOWRJB  80
0850EO                          DISTARKB  92
0860EO                          AMTPDRKB 105
0870EO                          BALR  KB 118
0880EO                          DIFFR KB 129
0890EO                                    67 'COMPANY TOTALS'
0900EO                                   131 '**'
```

Figure 15-32 (Part 3 of 3). Auto Report Sample Program EXAUT2

*AUTO Specifications

The coding for the *AUTO page heading and the *AUTO output functions is shown in Figure 15-31. Notice that the Y edit code is used for the date fields (lines 10 and 12). Auto report creates a K edit code for numeric fields when an edit code is not specified. No edit code is created for numeric fields when they are described with a digit (1 through 9) or R in column 39. The edit code 3 is specified for the INVNO field to suppress the printing of the comma edit character.

DIFF is printed on the detail line only if it is $1.00 or more. Remember, output indicator 10 conditions only the printing of the field on the detail line; it does not affect the printing of the created field on the total line.

The J edit code allows zero balance to print for the AMTOWD field.

Totals are accumulated and printed by auto report for five fields as indicated by A entries in column 39. Because an L1 control level is defined in the input field specifications for REGION, which is added to the input specifications for CASHRC (see Figure 15-31), regional and final totals are accumulated for each field that has A in column 39. The total lines are identified by the literals shown in lines 23 and 24 of the *AUTO specifications (see Figure 15-31).

Figure 15-33 shows the output data produced by EXAUT2.

11/20/82 CASH RECEIPTS REGISTER PAGE

REGION	ACCOUNT NUMBER	ACCOUNT NAME	INVOICE NUMBER	INVOICE DATE	DATE PAID	AMOUNT OWED	DISCOUNT TAKEN	AMOUNT PAID	BALANCE DUE	EXCESS DISCOUNT
1	11243	JONES HARDWARE	27541	12/31/99	12/31/99	23.75	.47	23.28		
1	11352	NU-STYLE CLOTHIERS	27987	12/31/99	12/31/99	87.07		40.00	47.07	
1	11886	MIDI FASHIONS INC	15771	12/31/99	12/31/99	107.22	2.14	105.08		
1	12874	ULOOK INTERIORS	25622	12/31/99	12/31/99	67.95		67.95		
1	18274	STREAMLINE PAPER INC	29703	12/31/99	12/31/99	274.03	2.38	170.55	101.10	
					REGION TOTALS	560.02	4.99	406.86	148.17	
2	23347	RITE-BEST PENS CO	20842	12/31/99	12/31/99	15.80		10.00	5.80	
2	25521	IMPORTS OF NM	29273	12/31/99	12/31/99	797.40	11.93	585.47	200.00	
2	26723	ALRIGHT CLEANERS	19473	12/31/99	12/31/99	462.00		462.00		
2	28622	NORTH CENTRAL SUPPLY	17816	12/31/99	12/31/99	75.97		75.97		
2	29871	FERGUSON DEALERS	27229	12/31/99	12/31/99	61.91		61.91		
					REGION TOTALS	1,413.08	11.93	1,195.35	205.80	
3	30755	FASTWAY AIRLINES	26158	12/31/99	12/31/99	742.72	16.85	725.87		1.90
3	31275	ENVIRONMENT CONCERNS	20451	12/31/99	12/31/99	29.43		15.00	14.43	
3	32457	B SOLE SILOS	27425	12/31/99	12/31/99	110.05		110.05		
3	37945	HOFFTA BREAKS INC	18276	12/31/99	12/31/99	47.23		47.23		
					REGION TOTALS	929.43	16.85	898.15	14.43	1.90
4	42622	EASTLAKE GRAVEL CO	16429	12/31/99	12/31/99	29.37		29.37		
					REGION TOTALS	29.37		29.37		
					COMPANY TOTALS	2,931.90	33.77	2,529.73	368.40	1.90

Figure 15-33. Output from Auto Report Sample Program EXAUT2

Chapter 16. Editing Numeric Fields

Chapter 16. Editing Numeric Fields

Editing means punctuating numeric data in an output field by adding symbols such as the currency symbol, commas, a decimal point, and a symbol for a negative balance, or by substituting blanks for zeros in unused positions at the left of the field. When you print fields that are not edited, the fields appear exactly as they are represented inside the computer. The following examples show why numeric output fields should be edited:

Type of Field	Field in the Computer	Printing of Unedited Field	Printing of Edited Field
Alphameric	JOHN T SMITH	JOHN T SMITH	JOHN T SMITH
Numeric (positive)	0047652	0047652	47652
Numeric (negative)	004765K	004765K	47652-

The unedited alphameric field and the unedited positive numeric field are easy to read when printed, but the unedited negative numeric field is confusing because it contains a K, which is not numeric. The K is a combination of the digit 2 and the negative sign for the field. They are combined so that one of the positions of the field does not have to be set aside for the sign. The combination is convenient for storing the field in the computer, but it makes the output hard to read. Therefore, numeric fields need to be edited before they are printed.

When you edit fields in a file assigned to any device other than a PRINTER (in columns 40 through 46 of the file description specifications), you must be aware of the contents of the edited field if you want the field to be read back into the program. You must also be aware of the effects of any operations you plan to use on an edited field. For example, if you add an unedited field to an edited field, the results will be wrong.

There are two ways to edit a numeric output field: using an edit code or using an edit word. Edit codes are easier to use, because you merely select the predefined type of editing you want. On the other hand, edit words allow you to do more, because you define exactly the kind of editing you want.

Edit Codes

There are several different edit codes available. Each code edits in a slightly different way according to a set pattern. However, all of them remove the sign of the field so that the rightmost digit always prints as a number.

Figure 16-1 shows the edit pattern for all edit codes. You choose the code that edits a field the way you want it to appear. For example, suppose you want to print blanks instead of zeros, to print decimal points and commas, but not to print the sign of a field. Figure 16-1 shows that edit codes 1 and 2 both do this editing. The difference between edit codes 1 and 2 is that, when the field is zero, edit code 1 prints zeros and edit code 2 prints blanks.

Figure 16-2 shows how various edit codes edit the same data.

Edit Code	Commas	Decimal Point	Sign for Negative Balance	Entry in Column 21 of Control Specification			Zero Suppress
				D or Blank	I	J	
1	Yes	Yes	No sign	.00 or 0	,00 or 0	0,00 or 0	Yes
2	Yes	Yes	No sign	Blanks	Blanks	Blanks	Yes
3		Yes	No sign	.00 or 0	,00 or 0	0,00 or 0	Yes
4		Yes	No sign	Blanks	Blanks	Blanks	Yes
A	Yes	Yes	CR	.00 or 0	,00 or 0	0,00 or 0	Yes
B	Yes	Yes	CR	Blanks	Blanks	Blanks	Yes
C		Yes	CR	.00 or 0	,00 or 0	0,00 or 0	Yes
D		Yes	CR	Blanks	Blanks	Blanks	Yes
J	Yes	Yes	- (minus)	.00 or 0	,00 or 0	0,00 or 0	Yes
K	Yes	Yes	- (minus)	Blanks	Blanks	Blanks	Yes
L		Yes	- (minus)	.00 or 0	,00 or 0	0,00 or 0	Yes
M		Yes	- (minus)	Blanks	Blanks	Blanks	Yes
X[1]							
Y[2]							Yes
Z							Yes

[1]The X code performs no editing.
[2]The Y code suppresses the leftmost zero only. The Y code edits a 3- to 6-digit field according to the following pattern:

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

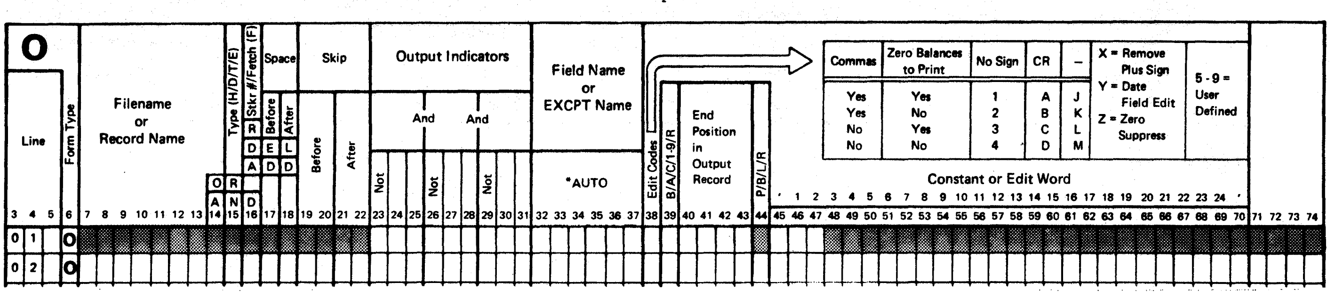
Figure 16-1. Edit Codes

Edit Code	Positive Number, 2 Decimal Positions	Positive Number, 0 Decimal Positions	Negative Number, 3 Decimal Positions	Negative Number, 0 Decimal Positions	Zero Balance, 2 Decimal Positions			Zero Balance, 0 Decimal Positions
					Entry in Column 21 of Control Specification			
					D or Blank	I	J	
Unedited	1234567	1234567	00012[1]	00012[1]	000000	000000	000000	000000
1	12,345.67	1,234,567	.120	120	.00	,00	0,00	0
2	12,345.67	1,234,567	.120	120				
3	12345.67	1234567	.120	120	.00	,00	0,00	0
4	12345.67	1234567	.120	120				
A	12,345.67	1,234,567	.120CR	120CR	.00	,00	0,00	0
B	12,345.67	1,234,567	.120CR	120CR				
C	12345.67	1234567	.120CR	120CR	.00	,00	0,00	0
D	12345.67	1234567	.120CR	120CR				
J	12,345.67	1,234,567	.120-	120-	.00	,00	0,00	0
K	12,345.67	1,234,567	.120-	120-				
L	12345.67	1234567	.120-	120-	.00	,00	0,00	0
M	12345.67	1234567	.120-	120-				
X	1234567	1234567	00012[1]	00012[1]	000000	000000	000000	000000
Y								0/00/00
Z	1234567	1234567	120	120				

[1]The EBCDIC values of negative decimal numbers do not print as numerics. If the negative number has an alphameric equivalent, it is printed. Otherwise, the program halts on an unprintable character unless column 45 of the control specification contains a 1. A minus zero (hex D0) prints as a blank, a minus 1 (hex D1) as J, a minus 2 (hex D2) as K, and so on.

Figure 16-2. Examples of Various Edit Codes

To use an edit code, code the unshaded columns of the output specifications shown below:



Columns 23 through 31 can contain conditioning indicators.

Columns 32 through 37 must contain the name of a numeric field.

Column 38 must contain an edit code.

Column 39 can contain B to indicate that the numeric field is to be set to zero after it is printed.

Columns 40 through 43 can contain the end position of the field in the output record.

Columns 45 through 47 can contain one of the following:

- '*' if you want asterisks to replace the leading zeros of the field.
- The currency symbol enclosed in apostrophes if you want a floating currency symbol. The currency symbol will then appear before the first digit in the field.

  A fixed currency symbol in columns 45 through 47 must be coded on a line before or following the edit code. The currency symbol remains in the end position specified.

  Note: You cannot use the X, Y, or Z edit code if you code an asterisk or the currency symbol in columns 45 through 47 of the output specifications.

When you use an edit code to punctuate an entire array, two spaces are skipped before each edited element.

Examples of Using the Currency Symbol with an Edit Code

Suppose you want to print a currency symbol on a report for a field called AMOUNT. An edit code will not put the currency symbol there. You specify this in addition to the edit code you are using.

When you use a floating currency symbol, the currency symbol changes positions so that it prints immediately in front of the first digit. In this case, the AMOUNT field would look like any of the following (N stands for any number):

```
$NNN.NN
 $NN.NN
  $N.NN
   $.NN
```

Note: If the currency symbol is not the dollar sign ($), the currency symbol must be entered in column 18 of the control specification.

See Figure 16-3 for a coding example of a floating currency symbol.

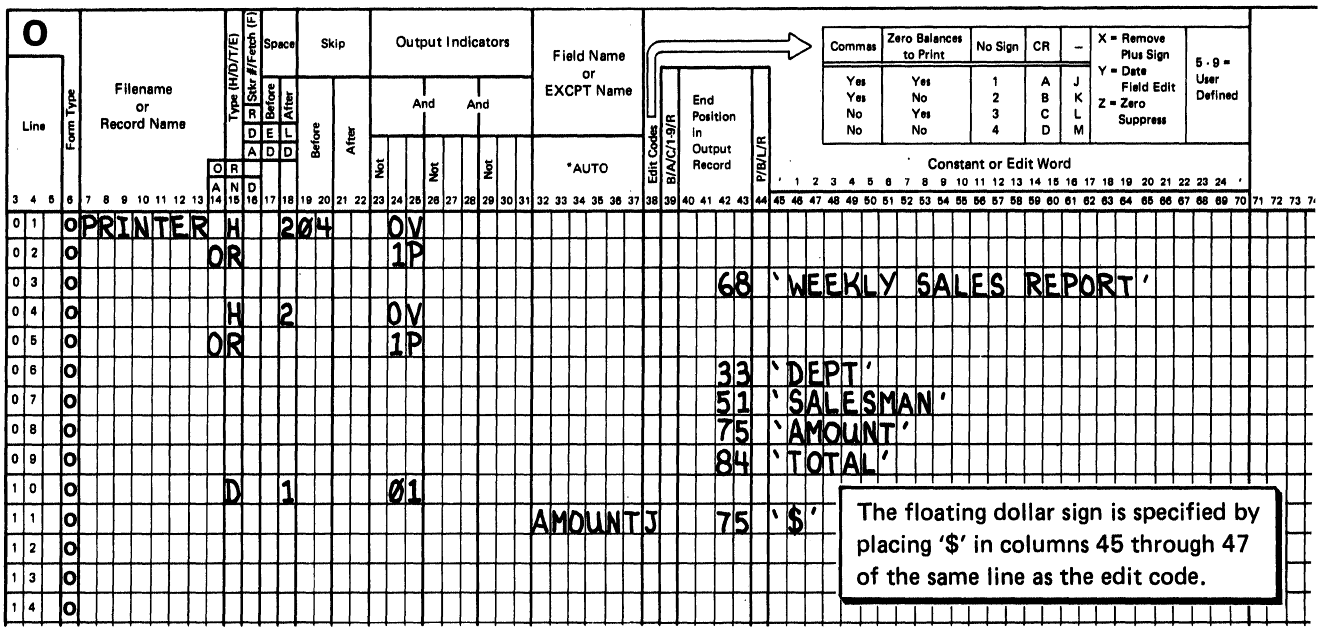


Figure 16-3. Floating Currency Symbol

When you use a fixed currency symbol, the currency symbol remains in the end position specified on the output specifications. In this case, the AMOUNT field would look like any of the following (N stands for any number):

```
$NNN.NN
$ NN.NN
$   N.NN
$     .NN
```

The blanks between the first digit and the currency symbol are the result of zero suppression. See Figure 16-4 for a coding example of a fixed currency symbol.

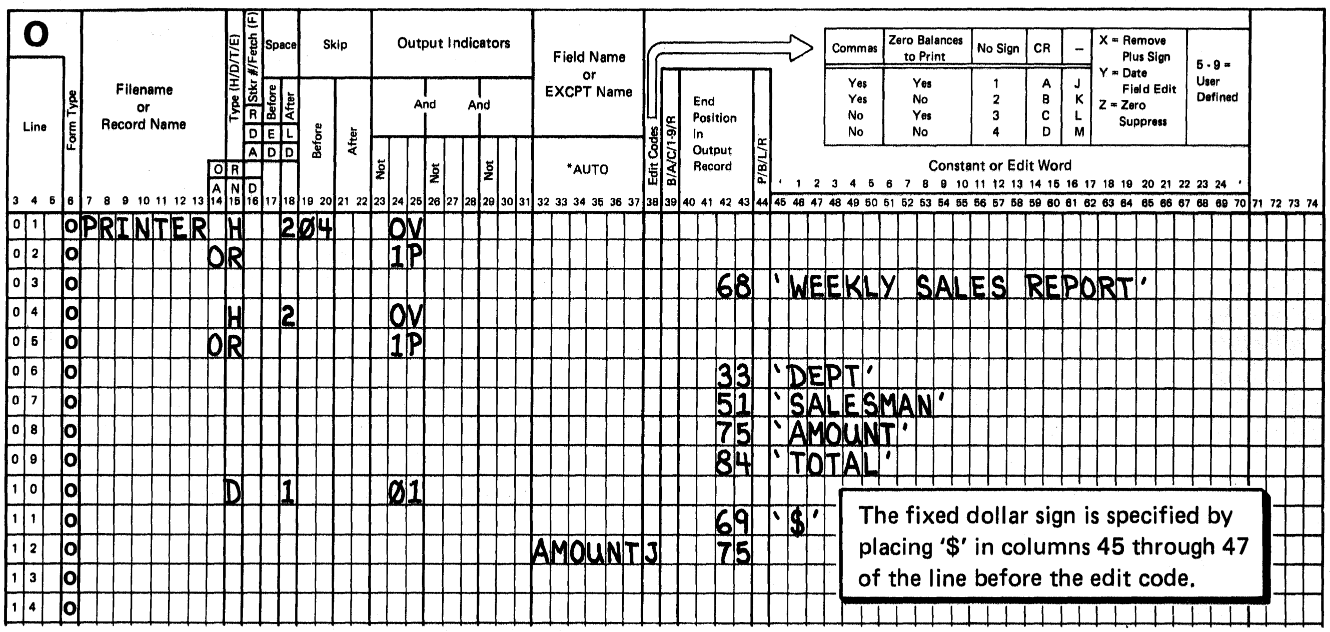


Figure 16-4. Fixed Currency Symbol

Example of Using Asterisks with an Edit Code

When you use asterisks to fill the spaces between the currency symbol and the first digit, the AMOUNT field could look like any of the following (N stands for any number):

```
$NNN.NN
$*NN.NN
$**N.NN
$***.NN
```

See Figure 16-5 for a coding example that uses asterisks to punctuate a field.

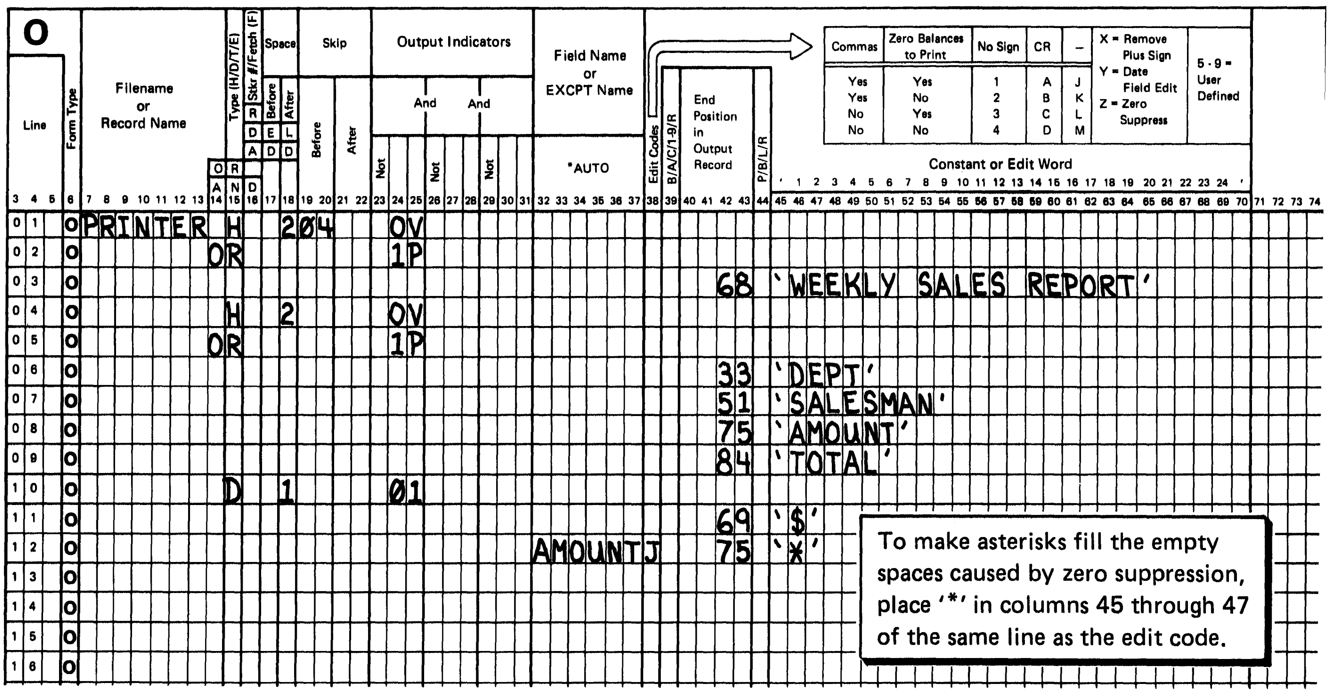


Figure 16-5. Punctuating with Asterisks

Edit codes are also used to edit date fields. The edit code for a date field is Y. See Figure 16-6 for various ways to edit a date field.

		Control Specification					
				Contents of Column 21			
UDATE	Edit Code	Contents of Column 19	Contents of Column 20	Blank	D	I	J
Jan 30, 1984	Y	Blank	Blank	1/30/84	30/01/84	30.01.84	30.01.84
			–	1-30-84	30-01-84	30-01-84	30-01-84
		M	Blank	1/30/84	1/30/84	1/30/84	1/30/84
			–	1-30-84	1-30-84	1-30-84	1-30-84
		D	Blank	30.01.84	30.01.84	30.01.84	30.01.84
			–	30-01-84	30-01-84	30-01-84	30-01-84
		Y	Blank	84.01.30	84.01.30	84.01.30	84.01.30
			–	84-01-30	84-01-30	84-01-30	84-01-30

Figure 16-6. Date Fields

Figure 16-7 shows the effects that the various edit codes have on the same field with a specified end position for output.

	Negative Number, 2 Decimal Positions. End Position Specified as 10.								
	Output Print Positions								
Edit Code	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Unedited				0	0	4	1	K[1]	
1					4	.	1	2	
2					4	.	1	2	
3					4	.	1	2	
4					4	.	1	2	
A			4	.	1	2	C	R	
B			4	.	1	2	C	R	
C			4	.	1	2	C	R	
D			4	.	1	2	C	R	
J				4	.	1	2	-	
K				4	.	1	2	-	
L				4	.	1	2	-	
M				4	.	1	2	-	
X				0	0	4	1	K[1]	
Y[2]			0	/	4	1	/	2	
Z						4	1	2	

[1]K represents a negative 2.
[2]A field edited by the Y edit code must have zero decimal positions.

Figure 16-7. Effect of Edit Codes on End Position

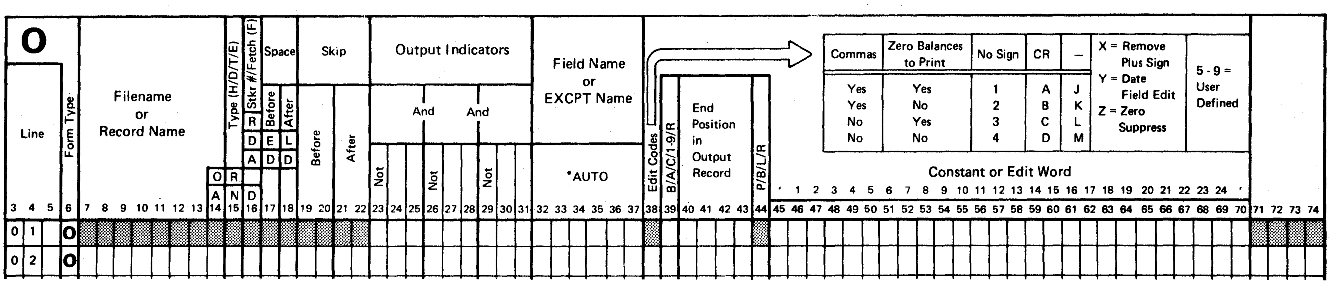
Edit Words

Use edit words when you have unusual (special) editing requirements. An edit word allows you to specify directly:

- If commas, decimal points, and zero suppression are needed
- If the negative sign should print
- If a currency symbol and leading asterisks should be used
- If the constant(s) have to be printed

An edit word gives a pattern for punctuation. When you create an edit word, you are setting up your own editing pattern.

To use an edit word, code the unshaded columns of the output specifications shown below:



Columns 23 through 31 can contain conditioning indicators.

Columns 32 through 37 must contain the name of a numeric field.

Column 38 (Edit Codes) must be blank.

Column 39 can contain b̶ to indicate that the numeric field is to be set to zero after it is printed.

Columns 40 through 43 can contain the end position of the field in the output record.

Columns 45 through 70 must contain the edit word. The edit word can be up to 24 characters long and must be enclosed in apostrophes. Enter the leading apostrophe in column 45. The actual edit word must begin in column 46.

Editing Considerations

When using an edit word, make sure that there is enough space on the printer form for the edited field. If the field you want to edit is 6 characters long, check whether 6 positions allow enough space for it to print on the report: the edited output field might contain more than 6 characters.

When you compute the length of an edited output field, determine how many of the editing characters are replaceable. A replaceable character is one that will be replaced by a digit from the data field. The number of replaceable characters must equal the length of the field to be edited.

The following summary provides more information on creating edit words and describes certain characters that have special meaning when used in an edit word. The *Delta position* is defined as the position in the edit word that corresponds to the leftmost position in the data field. The examples referred to are provided in the section *Examples of Edit Words*, later in this chapter.

ƀ (Blank)

This is always a replaceable character.

Constants

Constants are any character combination, including commas and decimal points, but not including special uses of 0, *, currency symbols, &, -, or CR symbols.

To the right of the Delta position (see examples 11 to 14): A constant imbedded in replaceable characters will print only if a significant digit appears to its left in the edited field. A constant between the last replaceable character and a negative indicator (see - or CR) will print only if the field is negative. Constant(s) at the end of the edit word will always print. Constants are not counted as replaceable characters.

To the left of the Delta position (see example 15): Constants are printed only if preceded by a zero in the edit word.

0 (Zero Suppress)

To the right of the Delta position (see example 2): Leading zeros to the left of and including the position in which the zero appears will be suppressed, but leading zeros to the right will not be suppressed. Any constants to the left of the zero will print only if preceded by a significant digit. The zero is a replaceable character.

In the Delta position (see example 3): Leading zeros and constants to the right of the zero will print. The zero will not print in the first position of the output field. If the field contains leading zeros, a blank will print in the first position; otherwise, a significant digit will print.

To the left of the Delta position (see example 4): The results are described in the preceding paragraph, except that if the field contains leading zeros, a zero will print in the first position of the output field. The zero in the edit word is not counted as a replaceable character and does not print.

NOTE: Any zeros or asterisks to the right of the first zero are treated as constants.

* (Asterisk Fill)

To the right of the Delta position (see example 9): Leading zeros to the left of the asterisk (and the asterisk itself) are replaced by asterisks. Constants will be replaced with asterisks when no significant digits precede the constant(s). Leading zeros to the right of the asterisk will not be suppressed. The asterisk is counted as a replaceable character.

In the Delta position (see example 10): An asterisk will print in the first position of the output field unless there is a significant digit in that position. Leading zeros and constants are not suppressed. The asterisk is counted as a replaceable character.

NOTE: Any asterisks or zeros to the right of the first asterisk are treated as constants.

CR and -

(See examples 5,6,17):

These symbols are used to identify negative fields on printed output and will print only if the field is negative. If the field is positive, they are replaced by blanks. Only the first - or CR to the right of all of the replaceable characters is treated as a negative indicator; all others are treated as constants. Any constants between the last replaceable character and the negative indicator will print only when the field is negative. Any constants following the negative indicator will always print. The - and CR are not counted as replaceable characters.

Currency Symbols

To the right of the Delta position (see example 7):

A. A currency symbol followed directly by a zero is said to float: it will print in the position immediately to the left of the first significant digit. The currency symbol may be replaced by a significant digit, but it should not be counted in the total of replaceable characters.

B. A currency symbol not directly followed by a zero will be treated as a constant. The currency symbol is not counted as a replaceable character.

To the left of the Delta position (see example 8):

The currency symbol will always print to the left of the first position of the output field. Leading zeros will be suppressed along with any constants that are not preceded by a significant digit. The currency symbol is not counted as a replaceable character.

& (Ampersand)

(See example 12):

The ampersand must be coded wherever a blank is to be printed in the output field.

Examples of Edit Words.

1. For all examples, column 38 (edit codes) of the output specification is blank. The symbol ƀ indicates where blank spaces would appear in the output result. The symbol ▼ marks the delta position in each example.

   In the example below all the leading zeros will be suppressed and the decimal point will not print unless there is a significant digit to its left:

Edit Word	Source Data	Appears in Output Record as:
`'       .  '`	000000012	bbbbbbbb12
	000000123	bbbbbb1.23

2. This example causes the decimal point to print even if the field is equal to zero:

Edit Word	Source Data	Appears in Output Record as:
`'      0.  '`	000000001	bbbbbbb.01
	000000000	bbbbbbb.00

3. Leading zeros will print to the left of the first significant digit. Note that a blank, not a zero, prints in the first position:

Edit Word	Source Data	Appears in Output Record as:
`'0      .  '`	000000123	b000001.23

4. If you want a zero to print in the leftmost position of the output field, the zero must be placed to the left of the Delta position in the edit word (see *Editing Considerations*). Note that seven blanks were coded to the left of the decimal point, whereas in example 3, only six blanks were coded. The zero in the edit word will not be printed:

Edit Word	Source Data	Appears in Output Record as:
`'0       .  '`	000000123	0000001.23

5. This example adds a negative value indication. The minus sign will print only when the value in the field is negative. A CR symbol performs the same function as a minus sign:

Edit Word	Source Data	Appears in Output Record as:
`'       .  -'`	000000123-	bbbbbb1.23-
	000000123	bbbbbb1.23b

6. Commas are added to separate thousands, millions, and so on. The comma will print only if preceded by a significant digit:

Edit Word	Source Data	Appears in Output Record as:
`' ,   ,   .  CR'`	123456789	1,234,567.89
	000012345-	bbbbbb123.45CR

7. A floating currency symbol will print in the position immediately to the left of the first significant digit or to the left of the decimal point if the field is zero or less than 100:

Edit Word	Source Data	Appears in Output Record as:
`'  ,   , $0.  '`	000000012	bbbbbbbbb$.12
	000123456	bbbb$1,234.56

8. The currency symbol may also be printed in the position before the first digit of the output field. Constants to the left of the first significant digit are replaced by blanks. See the explanation under the currency symbol in the section *Editing Considerations* for limitations regarding the placement of the currency symbol.

Edit Word	Source Data	Appears in Output Record as:
`'$ ,   ,  0.  '`	000123456	$bbbb1,234.56

9 Blanks and constants to the left of the first significant digit are replaced by asterisks (asterisk fill):

Edit Word	Source Data	Appears in Output Record as:
'$, , *. '	000123456	$****1,234.56

10 An asterisk will be printed in the first position only, unless the field contains a significant digit in the first position. Leading zeros and constants are not suppressed:

Edit Word	Source Data	Appears in Output Record as:
'* , , . '	000123456	*,001,234.56
	123456789	1,234,567.89

11 Constants between the last replaceable character and the '-' or CR symbol will print only if the field is negative; otherwise, blanks will print in these positions. Note the use of ampersands to represent blanks:

Edit Word	Source Data	Appears in Output Record as:
' , , 0. &30&DAY&CR'	000000123-	bbbbbbbb1.23b30bDAYbCR
	000000123	bbbbbbbb1.23bbbbbbbbbb

12 Constants may be added to print on every line:

Edit Word	Source Data	Appears in Output Record as:
' , , 0. &CR&NET'	000000123-	bbbbbbbb1.23bCRbNET
	000000123	bbbbbbbb1.23bbbbNET

13 This edit word could be used to print checks. Note that the second asterisk is treated as a constant:

Edit Word	Source Data	Appears in Output Record as:
`'$      **DOLLARS&  &CTS'`	000012345	$****123*DOLLARSƀ45ƀCTS

14 A date could be printed by using either edit word:

Edit Word	Source Data	Appears in Output Record as:
`'  /  /  '`	010385	ƀ1/03/85
`'0  /  /  '`	010385	01/03/85

15 The example below might be used to edit a telephone number. Note that the zero in the first position is required to print the constant AREA:

Edit Word	Source Data	Appears in Output Record as:
`'0AREA&   &NO.&   -    '`	4165551212	AREAƀ416ƀNO.ƀ555-1212

16 Note that any zeros or asterisks following the first occurrence of either are treated as constants. The same is true for - and CR:

Edit Word	Source Data	Appears in Output Record as:
`'  0.  000'`	01234	ƀ12.34000
`'  *.  000'`	01234	*12.34000
`'   .  -&&NET-'`	01234	ƀ12.34ƀƀƀNET-
`'30'`		30
	The combined output will appear as	ƀ12.34ƀƀƀNET-30

17 If an asterisk or a zero is to appear as a constant and there are no other asterisks or zeros preceding it in the edit word, the asterisk or zero must be defined on a separate specification line as a constant ending in the appropriate print position.

Edit Word	Source Data	Appears in Output Record as:
`'       .  '`	0000000123	bbbbbb1.23
`'*'`		*
	The combined output will appear as	bbbbbb1.23*

18 Note that the CR in the middle of a word may be detected as a negative field value indication. If a word such as SECRET is required, use the coding in the example below, line 3:

Edit Word	Source Data	Appears in Output Record as:
`'  0.  &SECRET'`	12345–	123.45bSECRET
`'  0.  &SECRET'`	12345	123.45bbbbbET
`'  0.  &CR&&SECRET'`	12345	123.45bbbbbSECRET

Creating Edit Words

The printer spacing chart can help you create edit words. Figure 16-8 shows how an output line can be created on this chart. The Xs and zeros show field positions. A zero indicates where zero suppression stops. An X indicates that any number can appear in the position. Use blanks in place of the Xs when writing the edit words.

If it is necessary to show a negative number, you must include a sign in the edit word. Use either the minus sign (-) or the letters CR. These print only for a negative number; however, the character positions they require must be included when you enter the end position of the field on the output specifications.

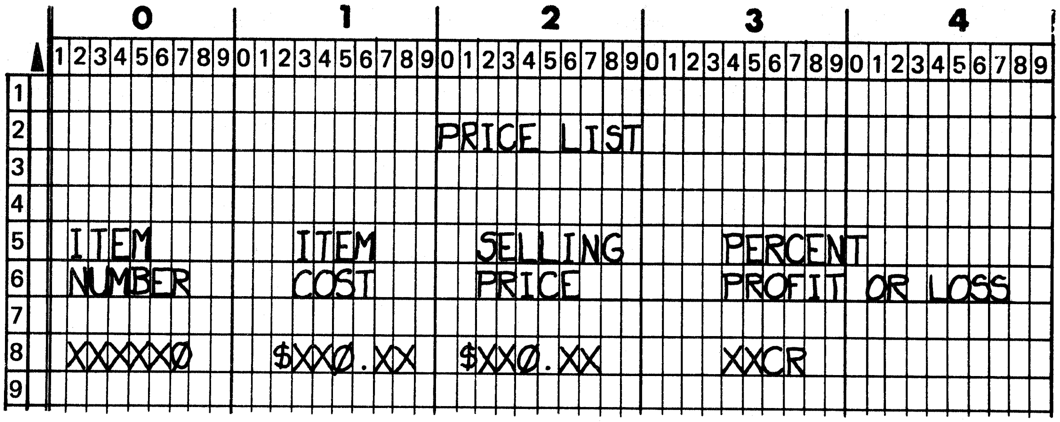
Figure 16-8 shows an edit word (line 08 of the output specifications) that causes CR to print if the field PERCPL has a negative balance. For example, if the field PERCPL contains -25 (which in storage appears as 2N), the printed output is 25CR. If PERCPL is positive, the CR does not print; the printed output is 25ƀƀ.

Another way to indicate a negative number is to use a minus sign. To leave a space between the number and the negative sign, place an ampersand (&) in the edit word before the minus sign. The PERCPL field then prints as 25 -.

Unedited Data

Item number	000241
Item cost	02000
Selling price	02200
% profit or loss	25

Printer Spacing Chart



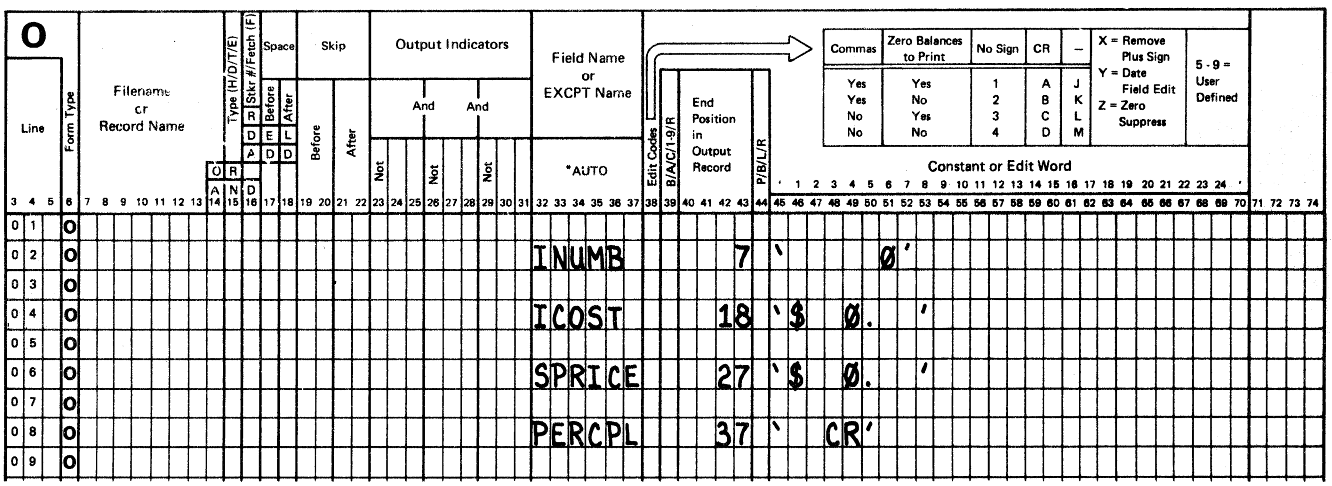


Figure 16-8. Using the Printer Spacing Chart to Create Edit Words

Chapter 17. Changing the Hexadecimal Value of Characters

Chapter 17. Changing the Hexadecimal Value of Characters

Each alphabetic, numeric, and special character is represented in the computer by a separate hexadecimal value. To determine if the value of one character is larger than the value of another character, the computer assigns a sequence to the hexadecimal values of the characters. This sequence is called the *normal collating sequence.* To collate means to place items in proper sequence or to check that items are in proper sequence. Figure 17-1 shows the normal collating sequence and hexadecimal value of each character.

You can change this normal collating sequence in two ways:

- By temporarily using one character in place of another during a comparison but using the original character at all other times during the program. This method is called *changing the collating sequence.*

- By using one character in place of another in one or more files whenever the file is used throughout an entire program. This method is called *translating a file.*

Collating Sequence	Character	Hexadecimal Value	Collating Sequence	Character	Hexadecimal Value
1	Blank	40	49	s	A2
2	¢	4A	50	t	A3
3	.	4B	51	u	A4
4	<	4C	52	v	A5
5	(	4D	53	w	A6
6	+	4E	54	x	A7
7	|	4F	55	y	A8
8	&[1]	50	56	z	A9
9	!	5A	57	{	C0
10	$	5B	58	A	C1
11	*	5C	59	B	C2
12	)	5D	60	C	C3
13	;	5E	61	D	C4
14	¬	5F	62	E	C5
15	- (minus)[1]	60	63	F	C6
16	/	61	64	G	C7
17	¦	6A	65	H	C8
18	,	6B	66	I	C9
19	%	6C	67	}	D0
20	_(underscore)	6D	68	J	D1
21	>	6E	69	K	D2
22	?	6F	70	L	D3
23	`	79	71	M	D4
24	:	7A	72	N	D5
25	#	7B	73	O	D6
26	@	7C	74	P	D7
27	'	7D	75	Q	D8
28	=	7E	76	R	D9
29	"	7F	77	\	E0
30	a	81	78	S	E2
31	b	82	79	T	E3
32	c	83	80	U	E4
33	d	84	81	V	E5
34	e	85	82	W	E6
35	f	86	83	X	E7
36	g	87	84	Y	E8
37	h	88	85	Z	E9
38	i	89	86	0	F0
39	j	91	87	1	F1
40	k	92	88	2	F2
41	l	93	89	3	F3
42	m	94	90	4	F4
43	n	95	91	5	F5
44	o	96	92	6	F6
45	p	97	93	7	F7
46	q	98	94	8	F8
47	r	99	95	9	F9
48	~	A1			

[1]When zones are specified for record identification codes, the & is considered to have a hex C zone, the - (minus sign) is considered to have a hex D zone, and the blank is considered to have a hex F zone, to be consistent with card punches.

Figure 17-1. Normal Collating Sequence and Hexadecimal Value of Characters

Changing the Collating Sequence

There are three reasons why you might want to change the normal collating sequence of characters:

- To compare alphameric characters
- To check the sequence of characters
- To check for match fields

For example, you may want alphabetic characters to follow numeric characters instead of coming before them. Notice in Figure 17-1 that numeric characters come after alphabetic characters in the normal collating sequence. Suppose that a company started with a few departments and assigned each department a number. In their data records, they used only a 2-digit field for the department number. When the company grew and the number of departments got larger than 99, the 2-digit field was no longer long enough. To avoid having to change the department-number field from two to three characters in every record, the manager changed the collating sequence for that one field so that he could use alphabetic characters after numeric characters. That is, after department 99, he named the departments A0, A1, and so on.

Another example is the need in some languages to insert a character such as A or A between A and B in the normal collating sequence.

Coding the Changes

To change the normal collating sequence, you code the control specification and the Translation Table and Alternate Collating Sequence Coding Sheet. Then you use the coding on the Translation Table and Alternate Collating Sequence Coding sheet to create records that actually change the normal collating sequence.

Coding the Control Specification

Column 26 of the control specification must contain S to indicate that you are changing the collating sequence.

Coding the Translation Table and Alternate Collating Sequence Coding Sheet

Figure 17-2 shows the Translation Table and Alternate Collating Sequence Coding Sheet.

TRANSLATION TABLE AND ALTERNATE COLLATING SEQUENCE CODING SHEET

Code	Graphic	Entry	Replaced By/Takes Place Of
00000000		00	
00000001		01	
00000010		02	
00000011		03	
00000100		04	
00000101		05	
00000110		06	
00000111		07	
00001000		08	
00001001		09	
00001010		0A	
00001011		0B	
00001100		0C	
00001101		0D	
00001110		0E	
00001111		0F	
00010000		10	
00010001		11	
00010010		12	
00010011		13	
00010100		14	
00010101		15	
00010110		16	
00010111		17	
00011000		18	
00011001		19	
00011010		1A	
00011011		1B	
00011100		1C	
00011101		1D	
00011110		1E	
00011111		1F	
00100000		20	
00100001		21	
00100010		22	
00100011		23	
00100100		24	
00100101		25	
00100110		26	
00100111		27	
00101000		28	
00101001		29	
00101010		2A	
00101011		2B	
00101100		2C	
00101101		2D	
00101110		2E	
00101111		2F	
00110000		30	
00110001		31	
00110010		32	

Code	Graphic	Entry	Replaced By/Takes Place Of
00110011		33	
00110100		34	
00110101		35	
00110110		36	
00110111		37	
00111000		38	
00111001		39	
00111010		3A	
00111011		3B	
00111100		3C	
00111101		3D	
00111110		3E	
00111111		3F	
01000000	Blank	40	
01000001		41	
01000010		42	
01000011		43	
01000100		44	
01000101		45	
01000110		46	
01000111		47	
01001000		48	
01001001		49	
01001010	¢	4A	
01001011	.	4B	
01001100	<	4C	
01001101	(	4D	
01001110	+	4E	
01001111	|	4F	
01010000	&	50	
01010001		51	
01010010		52	
01010011		53	
01010100		54	
01010101		55	
01010110		56	
01010111		57	
01011000		58	
01011001		59	
01011010	!	5A	
01011011	$	5B	
01011100	*	5C	
01011101	)	5D	
01011110	;	5E	
01011111	¬	5F	
01100000	-	60	
01100001	/	61	
01100010		62	
01100011		63	
01100100		64	
01100101		65	

Code	Graphic	Entry	Replaced By/Takes Place Of
01100110		66	
01100111		67	
01101000		68	
01101001		69	
01101010	¦	6A	
01101011	,	6B	
01101100	%	6C	
01101101	_	6D	
01101110	>	6E	
01101111	?	6F	
01110000		70	
01110001		71	
01110010		72	
01110011		73	
01110100		74	
01110101		75	
01110110		76	
01110111		77	
01111000		78	
01111001	`	79	
01111010	:	7A	
01111011	#	7B	
01111100	@	7C	
01111101	'	7D	
01111110	=	7E	
01111111	"	7F	
10000000		80	
10000001	a	81	
10000010	b	82	
10000011	c	83	
10000100	d	84	
10000101	e	85	
10000110	f	86	
10000111	g	87	
10001000	h	88	
10001001	i	89	
10001010		8A	
10001011		8B	
10001100		8C	
10001101		8D	
10001110		8E	
10001111		8F	
10010000		90	
10010001	j	91	
10010010	k	92	
10010011	l	93	
10010100	m	94	
10010101	n	95	
10010110	o	96	
10010111	p	97	
10011000	q	98	

Code	Graphic	Entry	Replaced By/Takes Place Of
10011001	r	99	
10011010		9A	
10011011		9B	
10011100		9C	
10011101		9D	
10011110		9E	
10011111		9F	
10100000		A0	
10100001	~	A1	
10100010	s	A2	
10100011	t	A3	
10100100	u	A4	
10100101	v	A5	
10100110	w	A6	
10100111	x	A7	
10101000	y	A8	
10101001	z	A9	
10101010		AA	
10101011		AB	
10101100		AC	
10101101		AD	
10101110		AE	
10101111		AF	
10110000		B0	
10110001		B1	
10110010		B2	
10110011		B3	
10110100		B4	
10110101		B5	
10110110		B6	
10110111		B7	
10111000		B8	
10111001		B9	
10111010		BA	
10111011		BB	
10111100		BC	
10111101		BD	
10111110		BE	
10111111		BF	
11000000	{	C0	
11000001	A	C1	
11000010	B	C2	
11000011	C	C3	
11000100	D	C4	
11000101	E	C5	
11000110	F	C6	
11000111	G	C7	
11001000	H	C8	
11001001	I	C9	
11001010		CA	
11001011		CB	

Code	Graphic	Entry	Replaced By/Takes Place Of
11001100		CC	
11001101		CD	
11001110		CE	
11001111		CF	
11010000	}	D0	
11010001	J	D1	
11010010	K	D2	
11010011	L	D3	
11010100	M	D4	
11010101	N	D5	
11010110	O	D6	
11010111	P	D7	
11011000	Q	D8	
11011001	R	D9	
11011010		DA	
11011011		DB	
11011100		DC	
11011101		DD	
11011110		DE	
11011111		DF	
11100000	\	E0	
11100001		E1	
11100010	S	E2	
11100011	T	E3	
11100100	U	E4	
11100101	V	E5	
11100110	W	E6	
11100111	X	E7	
11101000	Y	E8	
11101001	Z	E9	
11101010		EA	
11101011		EB	
11101100		EC	
11101101		ED	
11101110		EE	
11101111		EF	
11110000	0	F0	
11110001	1	F1	
11110010	2	F2	
11110011	3	F3	
11110100	4	F4	
11110101	5	F5	
11110110	6	F6	
11110111	7	F7	
11111000	8	F8	
11111001	9	F9	
11111010		FA	
11111011		FB	
11111100		FC	
11111101		FD	
11111110		FE	
11111111		FF	

Figure 17-2. Translation Table and Alternate Collating Sequence Coding Sheet

To code a change in the normal collating sequence, follow these steps:

1. In the Graphic column, find the character you want to use to replace another character in the collating sequence.

2. Note the hexadecimal value in the Entry column for the replacing character.

3. Code that hexadecimal value in the Replaced By column next to the character being replaced.

For example, if you want to change the normal collating sequence of a blank so that it has the same collating sequence as a zero:

1. Find the zero in the Graphic column.

2. Note that the hexadecimal value in the Entry column for zero is F0.

3. Code F0 in the Replace By column next to the blank.

Figure 17-3 shows this example. The same hexadecimal value is now used for both a blank and zero. Therefore, after you make these changes, the computer treats a blank as equal to zero when it compares alphameric characters, checks the sequence of characters, or checks for matching fields.

If you insert a character between two consecutive characters in the normal collating sequence, you must change the collating sequence for every character that is affected by that change. For example, when you insert the dollar sign ($) between A and B, you must also change the collating sequence for every character from B through I. This example is also shown in Figure 17-3.

TRANSLATION TABLE AND ALTERNATE COLLATING SEQUENCE CODING SHEET

Code	Graphic	Entry	Replaced By/Takes Place Of
00000000		00	
00000001		01	
00000010		02	
00000011		03	
00000100		04	
00000101		05	
00000110		06	
00000111		07	
00001000		08	
00001001		09	
00001010		0A	
00001011		0B	
00001100		0C	
00001101		0D	
00001110		0E	
00001111		0F	
00010000		10	
00010001		11	
00010010		12	
00010011		13	
00010100		14	
00010101		15	
00010110		16	
00010111		17	
00011000		18	
00011001		19	
00011010		1A	
00011011		1B	
00011100		1C	
00011101		1D	
00011110		1E	
00011111		1F	
00100000		20	
00100001		21	
00100010		22	
00100011		23	
00100100		24	
00100101		25	
00100110		26	
00100111		27	
00101000		28	
00101001		29	
00101010		2A	
00101011		2B	
00101100		2C	
00101101		2D	
00101110		2E	
00101111		2F	
00110000		30	
00110001		31	
00110010		32	

Code	Graphic	Entry	Replaced By/Takes Place Of
00110011		33	
00110100		34	
00110101		35	
00110110		36	
00110111		37	
00111000		38	
00111001		39	
00111010		3A	
00111011		3B	
00111100		3C	
00111101		3D	
00111110		3E	
00111111		3F	
01000000	Blank	40	FØ
01000001		41	
01000010		42	
01000011		43	
01000100		44	
01000101		45	
01000110		46	
01000111		47	
01001000		48	
01001001		49	
01001010	¢	4A	
01001011	.	4B	
01001100	<	4C	
01001101	(	4D	
01001110	+	4E	
01001111	|	4F	
01010000	&	50	
01010001		51	
01010010		52	
01010011		53	
01010100		54	
01010101		55	
01010110		56	
01010111		57	
01011000		58	
01011001		59	
01011010	!	5A	
01011011	$	5B	C2 (B)
01011100	*	5C	
01011101	)	5D	
01011110	;	5E	
01011111	¬	5F	
01100000	–	60	
01100001	/	61	
01100010		62	
01100011		63	
01100100		64	
01100101		65	

Code	Graphic	Entry	Replaced By/Takes Place Of
01100110		66	
01100111		67	
01101000		68	
01101001		69	
01101010	¦	6A	
01101011	,	6B	
01101100	%	6C	
01101101	_	6D	
01101110	>	6E	
01101111	?	6F	
01110000		70	
01110001		71	
01110010		72	
01110011		73	
01110100		74	
01110101		75	
01110110		76	
01110111		77	
01111000		78	
01111001	`	79	
01111010	:	7A	
01111011	#	7B	
01111100	@	7C	
01111101	'	7D	
01111110	=	7E	
01111111	"	7F	
10000000		80	
10000001	a	81	
10000010	b	82	
10000011	c	83	
10000100	d	84	
10000101	e	85	
10000110	f	86	
10000111	g	87	
10001000	h	88	
10001001	i	89	
10001010		8A	
10001011		8B	
10001100		8C	
10001101		8D	
10001110		8E	
10001111		8F	
10010000		90	
10010001	j	91	
10010010	k	92	
10010011	l	93	
10010100	m	94	
10010101	n	95	
10010110	o	96	
10010111	p	97	
10011000	q	98	

Code	Graphic	Entry	Replaced By/Takes Place Of
10011001	r	99	
10011010		9A	
10011011		9B	
10011100		9C	
10011101		9D	
10011110		9E	
10011111		9F	
10100000		A0	
10100001	~	A1	
10100010	s	A2	
10100011	t	A3	
10100100	u	A4	
10100101	v	A5	
10100110	w	A6	
10100111	x	A7	
10101000	y	A8	
10101001	z	A9	
10101010		AA	
10101011		AB	
10101100		AC	
10101101		AD	
10101110		AE	
10101111		AF	
10110000		B0	
10110001		B1	
10110010		B2	
10110011		B3	
10110100		B4	
10110101		B5	
10110110		B6	
10110111		B7	
10111000		B8	
10111001		B9	
10111010		BA	
10111011		BB	
10111100		BC	
10111101		BD	
10111110		BE	
10111111		BF	
11000000	{	C0	
11000001	A	C1	
11000010	B	C2	C3 (C)
11000011	C	C3	C4 (D)
11000100	D	C4	C5 (E)
11000101	E	C5	C6 (F)
11000110	F	C6	C7 (G)
11000111	G	C7	C8 (H)
11001000	H	C8	C9 (I)
11001001	I	C9	CA
11001010		CA	
11001011		CB	

Code	Graphic	Entry	Replaced By/Takes Place Of
11001100		CC	
11001101		CD	
11001110		CE	
11001111		CF	
11010000	}	D0	
11010001	J	D1	
11010010	K	D2	
11010011	L	D3	
11010100	M	D4	
11010101	N	D5	
11010110	O	D6	
11010111	P	D7	
11011000	Q	D8	
11011001	R	D9	
11011010		DA	
11011011		DB	
11011100		DC	
11011101		DD	
11011110		DE	
11011111		DF	
11100000	\	E0	
11100001		E1	
11100010	S	E2	
11100011	T	E3	
11100100	U	E4	
11100101	V	E5	
11100110	W	E6	
11100111	X	E7	
11101000	Y	E8	
11101001	Z	E9	
11101010		EA	
11101011		EB	
11101100		EC	
11101101		ED	
11101110		EE	
11101111		EF	
11110000	0	F0	
11110001	1	F1	
11110010	2	F2	
11110011	3	F3	
11110100	4	F4	
11110101	5	F5	
11110110	6	F6	
11110111	7	F7	
11111000	8	F8	
11111001	9	F9	
11111010		FA	
11111011		FB	
11111100		FC	
11111101		FD	
11111110		FE	
11111111		FF	

Blank and zero considered equal.

$ takes B's position.

(no printable character)

B takes C's position.

C takes D's position.

Figure 17-3. Changing the Collating Sequence

Coding the Records That Change the Collating Sequence

The changes to the normal collating sequence must be coded in records that you can enter into the computer after all the RPG specifications in your source program and after the records that translate files. Chapter 3 explains how to enter your specifications.

These records are actually a kind of table. Unlike other tables, however, they do not need to be coded on the file description or extension specifications. Instead, they must be coded as data records.

The first record must contain **ƀ (asterisk asterisk blank) in positions 1 through 3. You can use the remaining positions of this record for comments.

The second record, and any additional records needed to code the translation, must contain specific entries in the following record positions:

Record Position	Entry
1-6	ALTSEQ
7-8	Leave these positions blank.
9-10	Enter the hexadecimal value of the character whose normal collating sequence is being changed. This entry is the same as the value in the Entry column on the translation table and alternate collating coding sheet.
11-12	Enter the hexadecimal value of the character that is replacing another character in the normal collating sequence. This entry is the same as your entry in the Replaced By column on the translation table and alternate collating sequence coding sheet.
13-16, 17-20, 21-24, . . .	Use these positions in the same way as positions 9 through 12. The first two positions contain the hexadecimal value of the character to be replaced. The next two positions contain the hexadecimal value of the replacing character. You can use as many 4-position entries as the record can hold. Do not leave any blank positions between the 4-position entries. The first blank position ends the record.

If you are changing the collating sequence of many characters, you can use more than one record.

A record with **ƀ (asterisk asterisk blank) in positions 1 through 3 must follow the last record that changes the normal collating sequence.

Example of a Record That Changes the Collating Sequence

To change the normal collating sequence by inserting the dollar sign ($) between A and B, as shown in Figure 17-3, code the record as follows:

Record Position	Entry
1-6	ALTSEQ
7-8	Blanks
9-12	5BC2 ($ takes B's position)
13-16	C2C3 (B takes C's position)
17-20	C3C4 (C takes D's position)
21-24	C4C5 (D takes E's position)
25-28	C5C6 (E takes F's position)
29-32	C6C7 (F takes G's position)
33-36	C7C8 (G takes H's position)
37-40	C8C9 (H takes I's position)
41-44	C9CA (I takes the position of an unprintable character)

Translating a File

Translating a file means changing the hexadecimal value of one or more characters throughout an entire program. If the character is in an input file, the computer translates (changes) the hexadecimal value when it reads the file into main storage. If the character is in an output file, the computer translates it before writing the file. If the character is in an update or combined file, the computer translates the character when it reads the file and again before it writes the file.

The usual reason for translating a file is security. You can translate input or output data to protect classified information.

Coding the Translation

To translate a file, you code the control specification and the translation table and alternate collating sequence coding sheet. Then you use the coding on the translation table and alternate collating sequence coding sheet to create records that actually change the characters.

Coding the Control Specification

Column 43 of the control specification must contain F to indicate that you are translating a file.

Coding the Translation Table and Alternate Collating Sequence Coding Sheet

Figure 17-2 shows the translation table and alternate collating sequence coding sheet.

To code a character for translation, follow these steps:

1. In the Graphic column, find the character you want to use as the translation for another character.

2. Note the hexadecimal value in the Entry column for the character used as the translation.

3. Code that hexadecimal value in the Replaced By column next to the character being translated.

Coding the Records That Translate a File

To tell the computer which files to translate, you must code records that you can enter into the computer after all the RPG specifications in your source program but before the records that change the normal collating sequence. Chapter 3 explains how to enter your specifications.

These records for translating a file are actually a kind of table. Unlike other tables, however, they do not need to be coded on the file description or extension specifications. Instead, they must be coded as data records.

The first record must contain **ƀ (asterisk asterisk blank) in positions 1 through 3. You can use the remaining positions of this record for comments.

The second record and any additional records needed to code the translation, must contain specific entries in the following record positions:

Record Position	Entry
1-6 (to translate all files)	Enter *FILES to tell the compiler to translate all input, output, update, and combined files. Leave positions 7 and 8 blank.
1-8 (to translate a specific file)	Enter the name of the specific file to be translated.
9-10	Enter the hexadecimal value of the character that is being translated. This entry is the same as the value in the Entry column on the translation table and alternate collating sequence coding sheet.
11-12	Enter the hexadecimal value of the character that is translating another character. This entry is the same as your entry in the Replaced By column on the translation table and alternate collating sequence coding sheet.
13-16, 17-20, 21-24, . . . , 93-96	Use these positions in the same way as positions 9 through 12. The first two positions contain the hexadecimal value of the character to be translated. The next two positions contain the hexadecimal value of the translating character. You can use as many four-position entries as the record can hold. Do not leave any blank positions between the four-position entries. The first blank position ends the record. If you need more positions to code the translations, you can use more than one record.

Example of File Translation

A department store uses sales slips that contain the wholesale and retail price of each item. To keep the wholesale prices confidential, the store translates the numbers into letters. In output files, it uses the letters in the code name BUCKINGHAM to represent the numbers 1 through 9 and 0. In input files, it translates the letters back into numbers so that the computer can do calculations on the wholesale prices. Figure 17-4 shows how to code the file translation coding sheet for this example.

TRANSLATION TABLE AND ALTERNATE COLLATING SEQUENCE CODING SHEET

Code	Graphic	Entry	Replaced By/Takes Place Of
00000000		00	
00000001		01	
00000010		02	
00000011		03	
00000100		04	
00000101		05	
00000110		06	
00000111		07	
00001000		08	
00001001		09	
00001010		0A	
00001011		0B	
00001100		0C	
00001101		0D	
00001110		0E	
00001111		0F	
00010000		10	
00010001		11	
00010010		12	
00010011		13	
00010100		14	
00010101		15	
00010110		16	
00010111		17	
00011000		18	
00011001		19	
00011010		1A	
00011011		1B	
00011100		1C	
00011101		1D	
00011110		1E	
00011111		1F	
00100000		20	
00100001		21	
00100010		22	
00100011		23	
00100100		24	
00100101		25	
00100110		26	
00100111		27	
00101000		28	
00101001		29	
00101010		2A	
00101011		2B	
00101100		2C	
00101101		2D	
00101110		2E	
00101111		2F	
00110000		30	
00110001		31	
00110010		32	

Code	Graphic	Entry	Replaced By/Takes Place Of
00110011		33	
00110100		34	
00110101		35	
00110110		36	
00110111		37	
00111000		38	
00111001		39	
00111010		3A	
00111011		3B	
00111100		3C	
00111101		3D	
00111110		3E	
00111111		3F	
01000000	Blank	40	
01000001		41	
01000010		42	
01000011		43	
01000100		44	
01000101		45	
01000110		46	
01000111		47	
01001000		48	
01001001		49	
01001010	¢	4A	
01001011	.	4B	
01001100	<	4C	
01001101	(	4D	
01001110	+	4E	
01001111	|	4F	
01010000	&	50	
01010001		51	
01010010		52	
01010011		53	
01010100		54	
01010101		55	
01010110		56	
01010111		57	
01011000		58	
01011001		59	
01011010	!	5A	
01011011	$	5B	
01011100	*	5C	
01011101	)	5D	
01011110	;	5E	
01011111	¬	5F	
01100000	-	60	
01100001	/	61	
01100010		62	
01100011		63	
01100100		64	
01100101		65	

Code	Graphic	Entry	Replaced By/Takes Place Of
01100110		66	
01100111		67	
01101000		68	
01101001		69	
01101010	¦	6A	
01101011	,	6B	
01101100	%	6C	
01101101	_	6D	
01101110	>	6E	
01101111	?	6F	
01110000		70	
01110001		71	
01110010		72	
01110011		73	
01110100		74	
01110101		75	
01110110		76	
01110111		77	
01111000		78	
01111001	`	79	
01111010	:	7A	
01111011	#	7B	
01111100	@	7C	
01111101	'	7D	
01111110	=	7E	
01111111	"	7F	
10000000		80	
10000001	a	81	
10000010	b	82	
10000011	c	83	
10000100	d	84	
10000101	e	85	
10000110	f	86	
10000111	g	87	
10001000	h	88	
10001001	i	89	
10001010		8A	
10001011		8B	
10001100		8C	
10001101		8D	
10001110		8E	
10001111		8F	
10010000		90	
10010001	j	91	
10010010	k	92	
10010011	l	93	
10010100	m	94	
10010101	n	95	
10010110	o	96	
10010111	p	97	
10011000	q	98	

Code	Graphic	Entry	Replaced By/Takes Place Of
10011001	r	99	
10011010		9A	
10011011		9B	
10011100		9C	
10011101		9D	
10011110		9E	
10011111		9F	
10100000		A0	
10100001	~	A1	
10100010	s	A2	
10100011	t	A3	
10100100	u	A4	
10100101	v	A5	
10100110	w	A6	
10100111	x	A7	
10101000	y	A8	
10101001	z	A9	
10101010		AA	
10101011		AB	
10101100		AC	
10101101		AD	
10101110		AE	
10101111		AF	
10110000		B0	
10110001		B1	
10110010		B2	
10110011		B3	
10110100		B4	
10110101		B5	
10110110		B6	
10110111		B7	
10111000		B8	
10111001		B9	
10111010		BA	
10111011		BB	
10111100		BC	
10111101		BD	
10111110		BE	
10111111		BF	
11000000	{	C0	
11000001	A	C1	F9
11000010	B	C2	F1
11000011	C	C3	F3
11000100	D	C4	
11000101	E	C5	
11000110	F	C6	
11000111	G	C7	F7
11001000	H	C8	F8
11001001	I	C9	F5
11001010		CA	
11001011		CB	

Code	Graphic	Entry	Replaced By/Takes Place Of
11001100		CC	
11001101		CD	
11001110		CE	
11001111		CF	
11010000	}	D0	
11010001	J	D1	
11010010	K	D2	F4
11010011	L	D3	
11010100	M	D4	FØ
11010101	N	D5	F6
11010110	O	D6	
11010111	P	D7	
11011000	Q	D8	
11011001	R	D9	
11011010		DA	
11011011		DB	
11011100		DC	
11011101		DD	
11011110		DE	
11011111		DF	
11100000	\	E0	
11100001		E1	
11100010	S	E2	
11100011	T	E3	
11100100	U	E4	F2
11100101	V	E5	
11100110	W	E6	
11100111	X	E7	
11101000	Y	E8	
11101001	Z	E9	
11101010		EA	
11101011		EB	
11101100		EC	
11101101		ED	
11101110		EE	
11101111		EF	
11110000	0	F0	
11110001	1	F1	
11110010	2	F2	
11110011	3	F3	
11110100	4	F4	
11110101	5	F5	
11110110	6	F6	
11110111	7	F7	
11111000	8	F8	
11111001	9	F9	
11111010		FA	
11111011		FB	
11111100		FC	
11111101		FD	
11111110		FE	
11111111		FF	

This is the hexadecimal value of the character to be translated.

This is the hexadecimal value of the charact that will be substituted for the character th is to be translated.

Figure 17-4. Translating a File

The record to translate these files looks like this:

Record Position	Entry
1-6	*FILES (All files are translated.)
7-8	Blanks
9-12	C2F1 (B is translated into 1 at input. 1 is translated into B at output.)
13-16	E4F2 (U is translated into 2 at input. 2 is translated into U at output.)
17-20	C3F3 (C is translated into 3 at input. 3 is translated into C at output.)
21-24	D2F4 (K is translated into 4 at input. 4 is translated into K at output.)
25-28	C9F5 (I is translated into 5 at input. 5 is translated into I at output.)
29-32	D5F6 (N is translated into 6 at input. 6 is translated into N at output.)
33-36	C7F7 (G is translated into 7 at input. 7 is translated into G at output.)
37-40	C8F8 (H is translated into 8 at input. 8 is translated into H at output.)
41-44	C1F9 (A is translated into 9 at input. 9 is translated into A at output.)
45-48	D4F0 (M is translated into 0 at input. 0 is translated into M at output.)

Chapter 18. Techniques for Efficient Coding

Chapter 18. Techniques for Efficient Coding

To create an efficient program and make the best use of the system resources requires careful design and coding. The main purpose of this chapter is to introduce three ways to make your programs more efficient:

- Structured programming
- Overlaying storage
- Specific coding techniques.

Three logical structures used in every computer program are:

- Sequential operation
- Conditional branching
- Repeating an operation based on a certain condition.

These logical structures and how they can be implemented through RPG structured programming operation codes are briefly discussed in this chapter.

See Chapter 28 for detailed information about various groups of operations and individual operation codes.

Sequential Operation

Sequential operation means any series of instructions that actually processes data rather than transfers control to some other part of the program. Figure 18-1 is a flowchart of sequential operations.

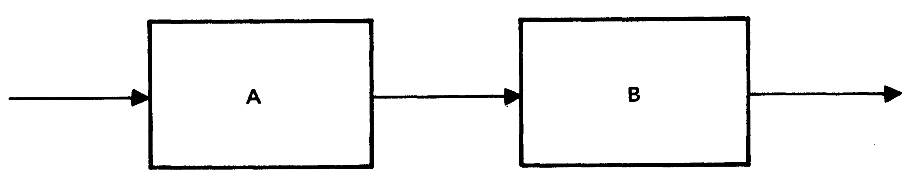


Figure 18-1. Flowchart of Sequential Operations

Conditional Branching

A conditional branch is a change in the sequence of instructions under a certain condition. The program first tests to see if the condition exists. If it does, the program branches to another point. If the condition does not exist, the program continues its sequence of instructions without branching. For this reason, a conditional branch is sometimes called an If-Then-Else structure. An example in simple English is:

IF the weather is cold,

THEN I will wear my coat;

ELSE, I will leave my coat at home.

Figure 18-2 is a flowchart of a conditional branch.

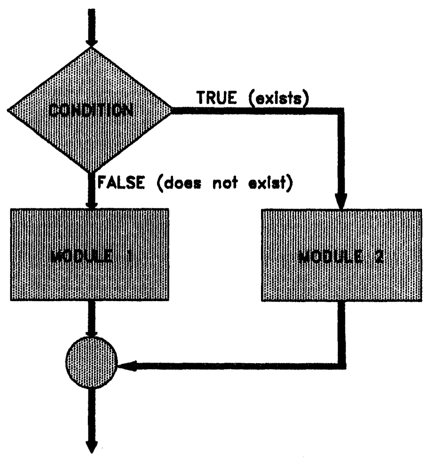


Figure 18-2. Flowchart of a Conditional Branch

In RPG, the If-Then-Else structure is implemented through the operation codes IFxx, ELSE, and END. Figure 18-3 shows a design for a conditional branch using the IFxx, ELSE, and END operations.

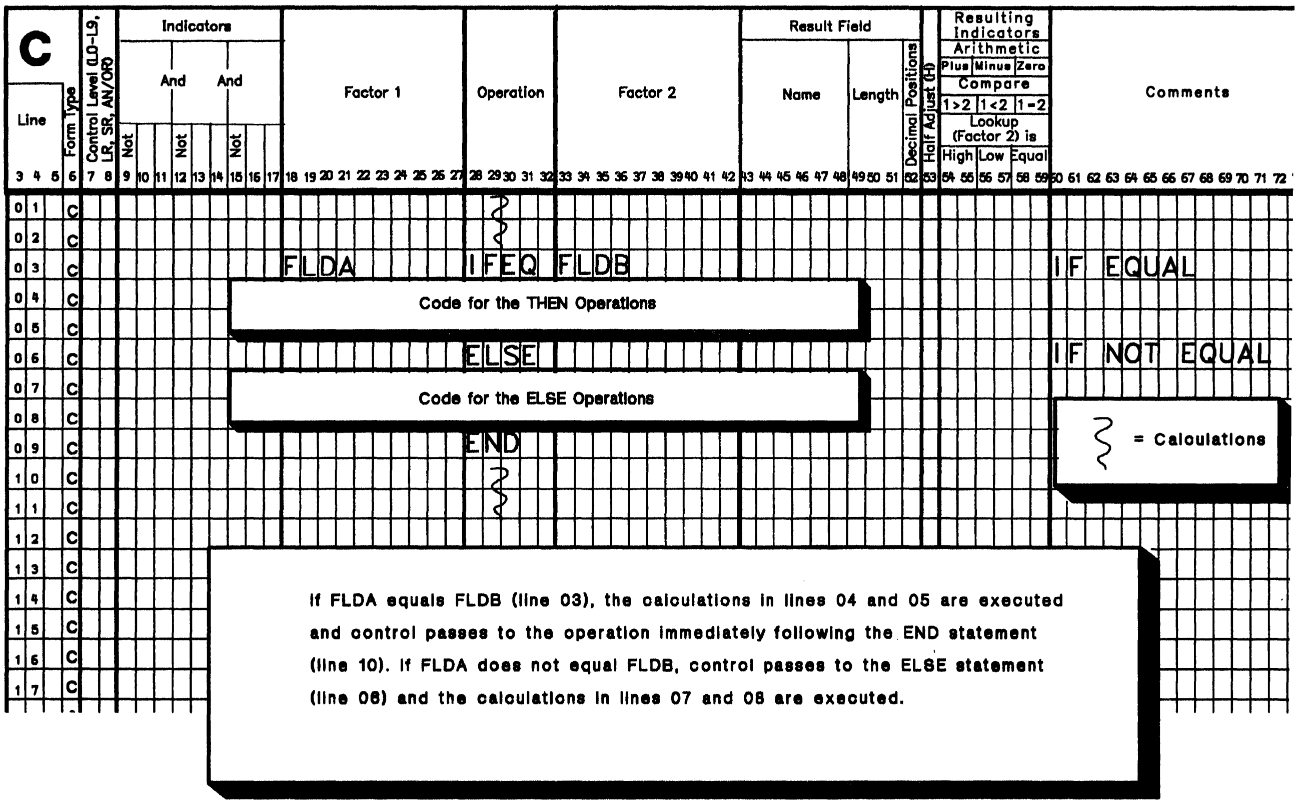


Figure 18-3. Design for a Conditional Branch Using the IF/ELSE/END Operations

There are three other ways to create conditional branches:

- The CASxx operation
- The EXSR operation and conditioning indicators
- The GOTO operation and conditioning indicators.

Repeating an Operation

Repeating an operation or a series of operations based on a certain condition means testing whether a condition exists, performing an operation or a series of operations if this condition exists, and repeating the test and the operation(s) as long as the condition still exists. Three logical structures of the controlled loop - Do While, Do Until and Do - are implemented in RPG through usage of the DOWxx, DOUxx and DO operation codes and the END operation code.

Do While Structure

If you test the condition first and then perform the operation(s), the structure is called a Do While. An example of a Do While is:

1. Compare a sum with 5.

2. If the sum is less than 5, add 1 to the sum.

3. Repeat steps 1 and 2 until the sum is equal to or greater than 5.

Figure 18-4 is a flowchart of a Do While, and Figure 18-5 illustrates coding of a Do While in RPG Calculation Specifications using the DOWxx operation code and the END operation code.

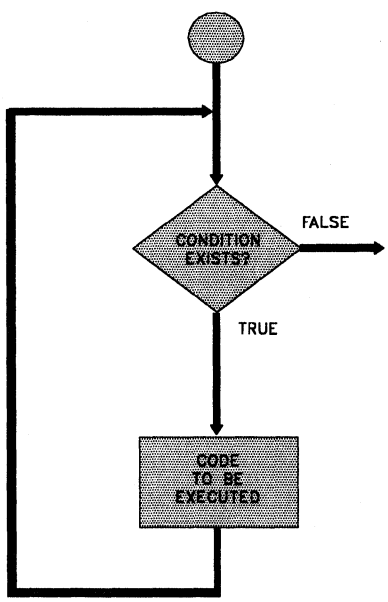


Figure 18-4. Flowchart of a Do While

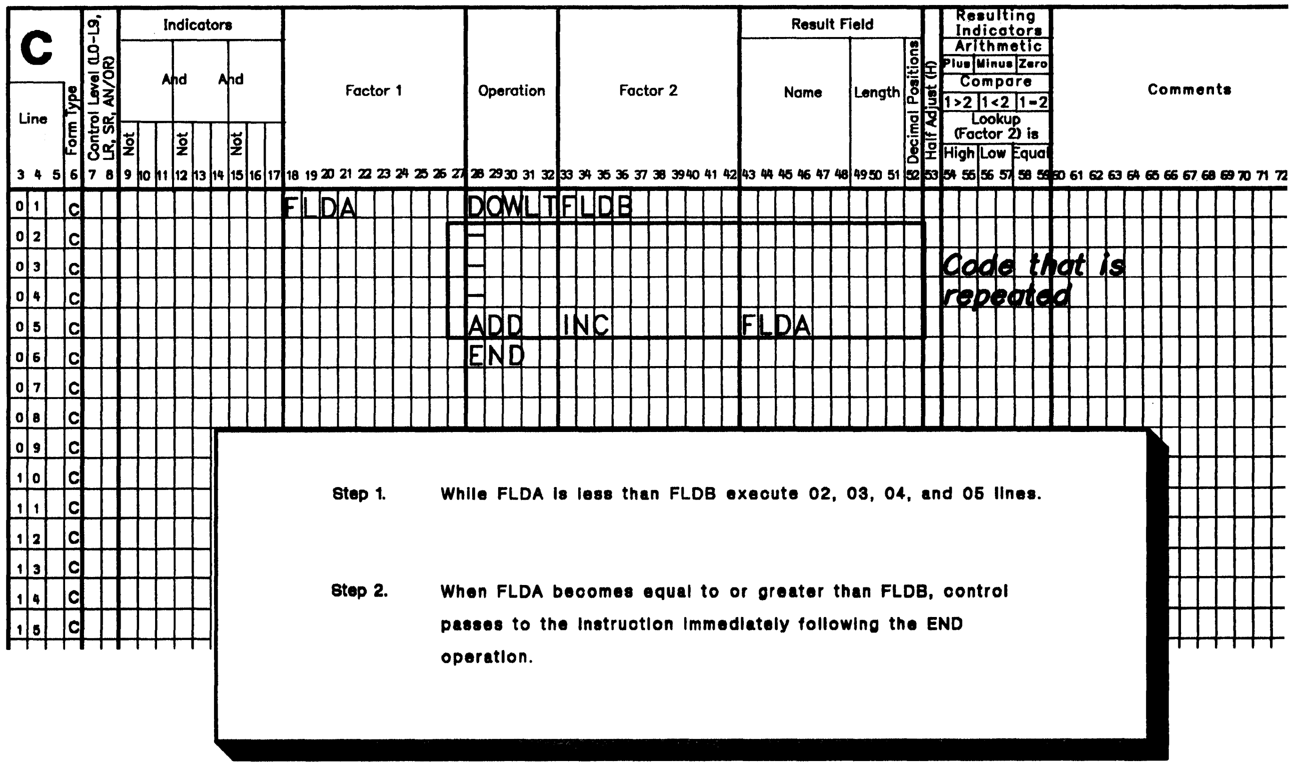


Figure 18-5. Design for a Do While using the DOWxx operation

Notice in Figure 18-5 (the Do While) that the program first tests whether the condition is true (line 01). If it is true, the code between the DOW and the END operations is executed. Then the program goes back to line 1 to test again whether the condition is still true, and the entire cycle is repeated. If the condition is no longer true, control passes to the instruction immediately following the END operation.

Do Until Structure

If you perform the operation(s) first and then test the condition, the structure is called a Do Until. An example of a Do Until is:

1. Add 1 to a sum.

2. Compare the sum with 5.

3. Repeat steps 1 and 2 if the sum is less than 5.

Figure 18-6 is a flowchart of a Do Until, and Figure 18-7 illustrates coding of a Do Until in RPG Calculation Specifications using the DOUxx operation code and the END operation code.

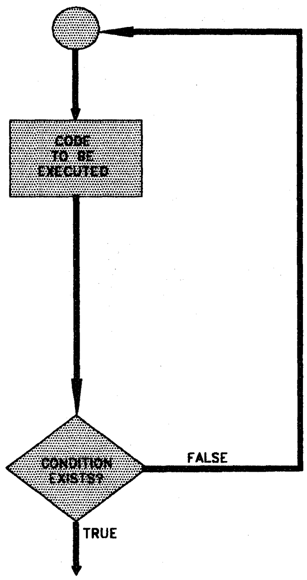


Figure 18-6. Flowchart of a Do Until

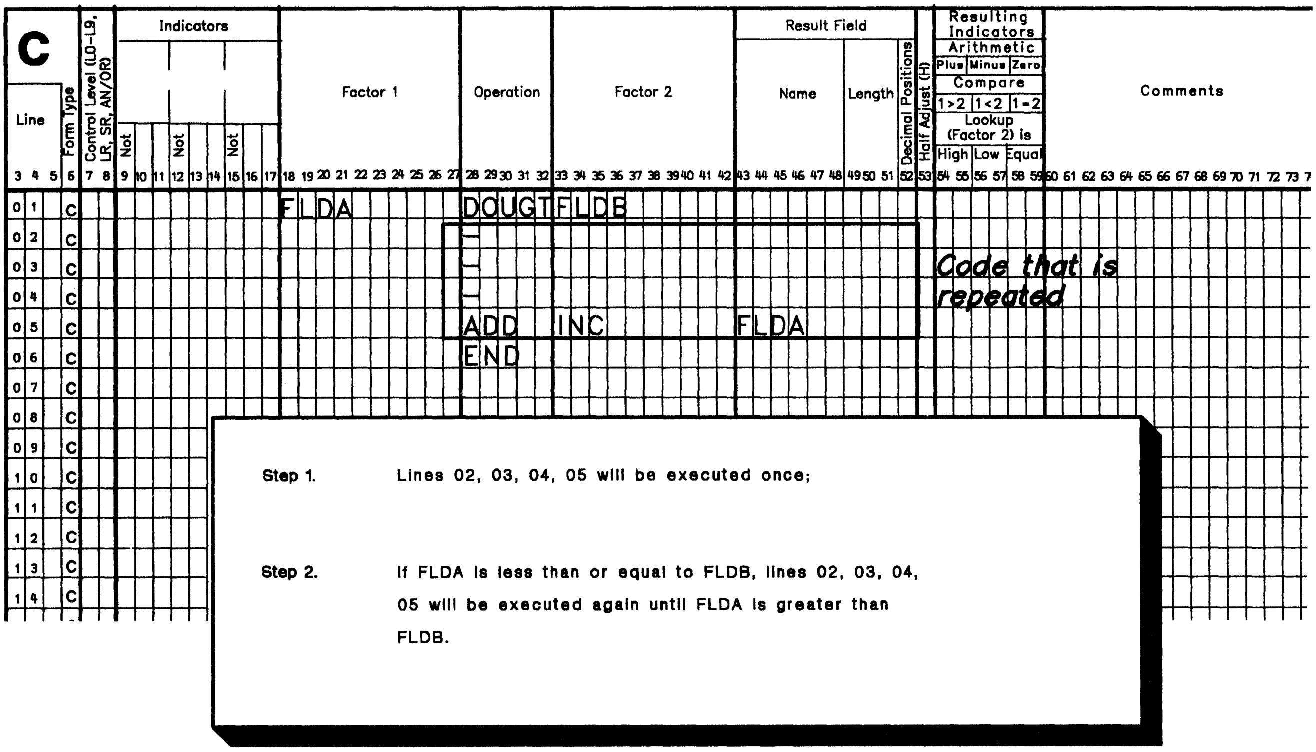


Figure 18-7. Design for a Do Until using the DOUxx Operation

Notice in Figure 18-7 (the Do Until) that the program first executes the operations on lines 02 through 05, and then tests (line 01) to see whether the condition is met. If this condition is not met, lines 02 through 05 are executed again. The program continues looping until the condition becomes true. Then control passes to the line immediately following the END operation.

Do Structure

If an operation or a series of operations has to be performed a fixed number of times, the structure is called a Do. You indicate how many times this operation or series of operations has to be performed by specifying a starting value, a limit value, an index value, and an increment value.

In its simplest form, the DO structure using the DO and the END operation codes is coded in the RPG Calculation Specifications as follows:

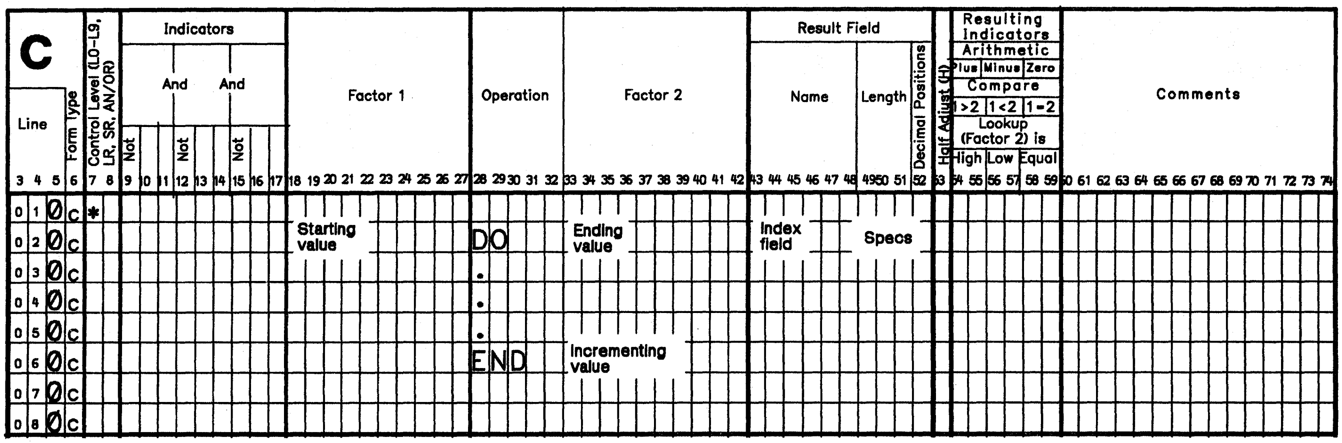


Figure 18-8. Coding of a Do structure on the Calculation Specifications

This is how the Do structure works:

1. Set index field (result field) to starting value (factor 1).
2. Test if the index field value is greater than the ending value (factor 2).

   If the index field value is greater than the ending value, control passes to the statement following the END statement.
3. If the index field value is not greater than the ending value, the operations between the DO statement and the END statement are executed.
4. At END, the index field value is incremented by the increment value specified in factor 2 on the END statement, or by 1 if the increment is not specified.
5. Control passes to Step 2 above.

Figure 18-9 is a flowchart of a Do structure, and Figure 18-10 illustrates coding of a Do structure in RPG Calculation Specifications using the DO operation code and the END operation code.

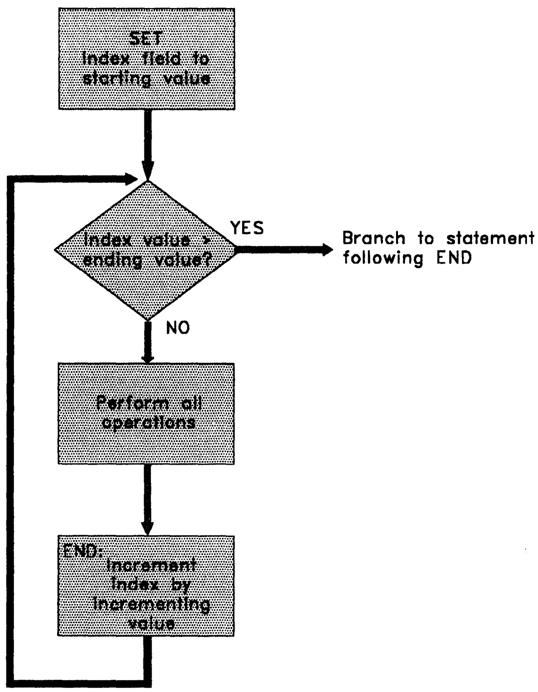


Figure 18-9. Flowchart of a Do structure

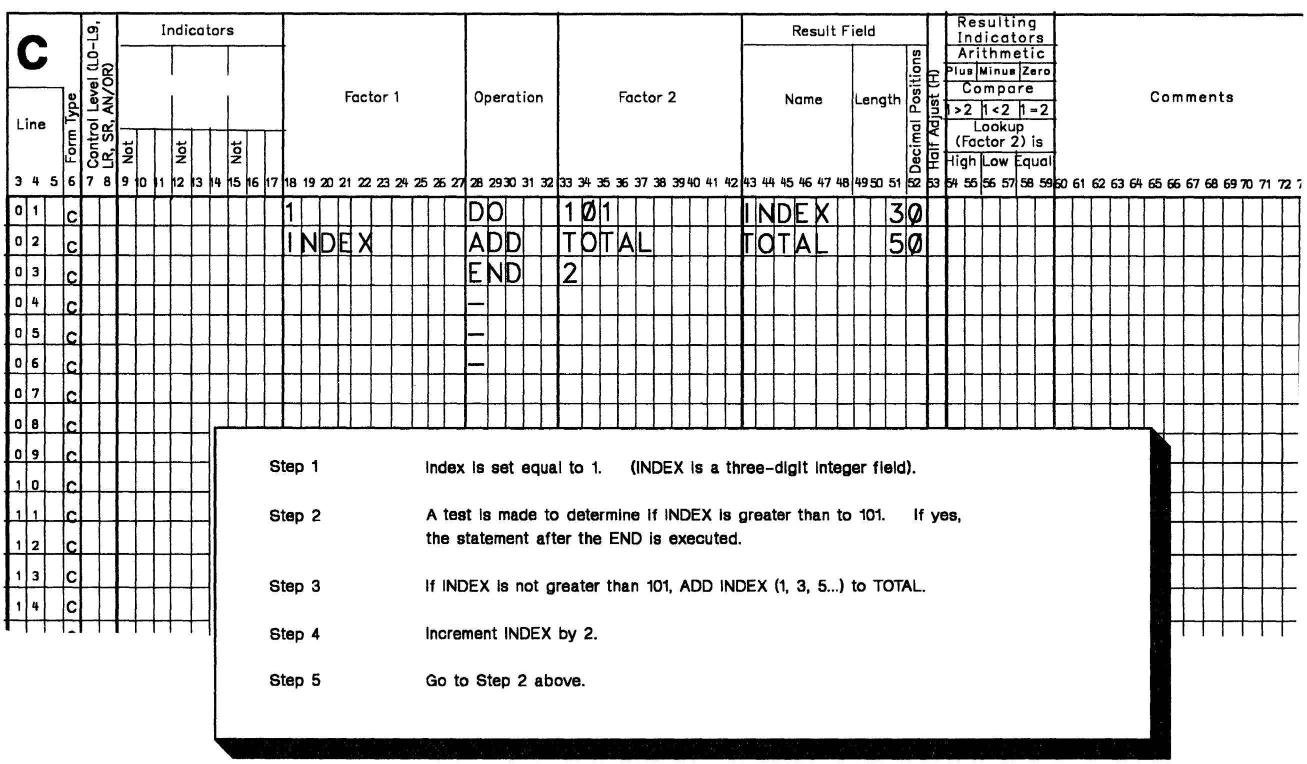


Figure 18-10. Design for a Do structure using the DO and END operations

See *Structured Programming Operation Codes* in this chapter and Chapter 28 for detailed descriptions of the DOWxx, DOUxx, and DO operations.

Structured Programming

Structured programming is an approach to design and coding that makes programs easy to understand, debug, and modify. Ideally, a structured program is a hierarchy of modules that can have a single entry point and a single exit point. Control is passed downward through the structure without unconditional branches to higher levels of the structure.

In RPG, structured programming can be achieved by:

- using conditional branching to subroutines
- using groups of operations controlled by structured programming operation codes, such as DO, DOUxx, DOWxx, and IFxx/ELSE.

Using Subroutines

A subroutine is a set of instructions that contains coding for a single task in a program, and has only one entry and one exit. An RPG subroutine may be used at one or more points in a program.

The calculation specifications of the main program (if structured) consists mostly of EXSR or CASxx operation codes which pass control to certain subroutines under certain conditions. If you give meaningful names to the subroutines and use comments to explain the purpose and step-by-step operation of the subroutines, the calculation specifications of the main program give a clear picture of the overall logic of your program. This clarity is the main reason that structured programs are easy to design, code, debug, and maintain.

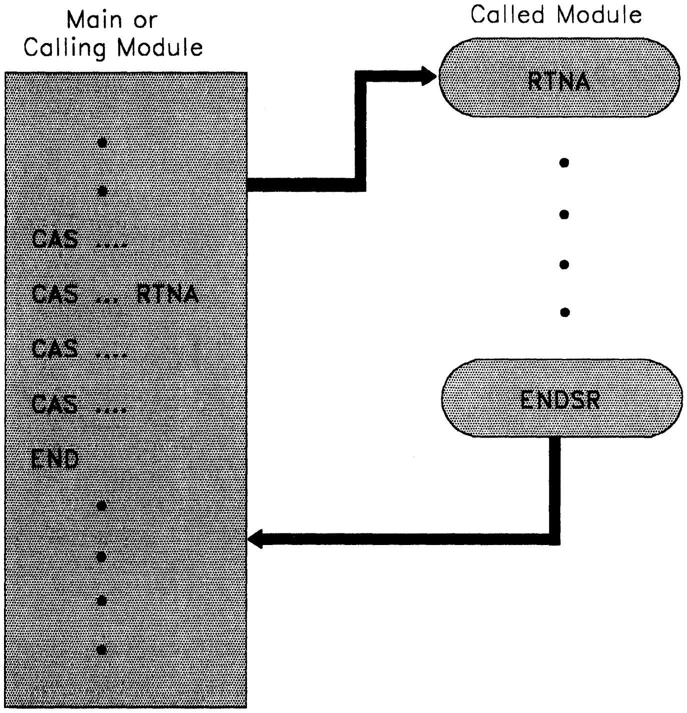


Figure 18-11. Using Subroutines in a Structured Program

Structured Programming Operation Codes

The structured programming operation codes are:

- DO (Do)
- DOWxx (Do While)
- DOUxx (Do Until)
- CASxx (Case)
- IFxx (If/Then)
- ELSE (Else Do)
- END (End)

where xx can be:

GT Factor 1 is greater than factor 2.

LT Factor 1 is less than factor 2.

EQ Factor 1 is equal to factor 2.

NE Factor 1 is not equal to factor 2.

GE Factor 1 is greater than or equal to factor 2.

LE Factor 1 is less than or equal to factor 2.

Blanks Factor 1 is not compared to factor 2 (unconditional execution). This is valid for CASxx operation only.

The IFxx operation allows a group of calculations to be executed based on the results of comparing factor 1 with factor 2.

The CASxx operation causes branching to a subroutine based on the results of comparing factor 1 with factor 2.

The DO operation allows an operation or a group of operations to be performed a fixed number of times. You indicate the number of times you want these operations to be performed by specifying the starting value (factor 1), the limit value (factor 2), the index value (result field), and the increment value (factor 2 of the associated END operation).

The DOWxx and DOUxx operations allow a group of operations to be executed, or repeated one or more times based on the results of comparing factor 1 with factor 2.

The group of operations that begins with a DO, DOUxx, DOWxx, or IFxx operation is called a do group. A CAS group can consist of CASxx operations only. Each do group and CAS group must end with an END operation.

See the *Structured Programming Operations* section and the descriptions of the individual structured programming operation codes in Chapter 28 for more information.

Overlaying Storage

A large program using most or all of the available storage can slow system performance. When this occurs, you can change the program by using the overlay linkage editor (OLE) to overlay storage, thereby freeing storage that can be used by other programs.

There are three ways to access OLE:

- Choose the LINK option on the RPGONL, RPGC, or AUTOC procedure (see Chapter 3 for more information about these procedures).
- Use the OLINK procedure. For information about the OLINK procedure, see the *Overlay Linkage Editor Guide*, SC21-9041, and the *System Reference* manual, SC21-9020.
- Use the OLE control statements. For information about the OLE control statements, see the *Overlay Linkage Editor Guide*, SC21-9041.

To enable OLE to create overlays, you should determine the proper value to be specified for the program size. First, compile your program without specifying the program size (leave columns 12 through 14 of the control specifications or the *Override size-to-execute option in source* parameter on the RPGONL, RPGC, or AUTOC procedure blank). In this case, RPG assumes a program size equal to the size of the region in which you compile. When the compilation is complete, check the OLE storage usage map (Figure 18-13). If storage is not overlaid, recompile the program with a program size smaller than the main storage size shown in the OLE storage usage map.

OLE divides the program into segments. Each segment can run without the entire program being in main storage at the same time. There are two types of segments: the root segment and overlay segments. In some cases OLE will not generate overlay segments, and the entire program will be stored in the root segment.

The root segment contains constants, data, and codes used frequently while the program is running. For this reason, the root segment always remains in main storage. If the overlay segments are generated, the root segment can call routines in the overlay segments, and it can be used by routines in the overlay segments. The overlay segments contain the major routines of the RPG program. Routines in these segments can be called by the root segment or by other routines in the same overlay segment.

Memory Resident Overlays (MRO)

Programs that use particular overlays repeatedly may benefit from the use of memory-resident overlays (MRO). When MRO is *not* used, (NOMRO), only a single overlay can be maintained in memory at any given time. Each call of a new overlay involves a disk load, and if demands on your system have caused disk queueing to occur, program performance will suffer. When MRO *is* used, overlays are maintained in memory for as long as the system has space available, and program performance is less likely to be affected by disk queues. Other factors must be considered, however.

Overlays are linked on 256-byte boundaries when MRO is not used (NOMRO), and on 2K boundaries when MRO is used. Consequently programs that fit into a given region when MRO is not used (NOMRO), may not fit when MRO is used. If this occurs, you should do one of the following:

- Increase the region size
- Rearrange the overlay structure (see *Reducing the Program Size* further in this chapter).
- Not use MRO.

If your system is running higher priority jobs that require memory holding your unused overlays, these will be overwritten.

To make use of the performance advantage possible with MRO, your system should have sufficient memory to contain at least two overlays, and the overlays should be used by the program more than once.

Areas of Main Storage

Main storage is divided into two main parts: the root area and the overlay area. The root area contains the root segment and the overlay fetch routine. The overlay fetch routine controls the loading of the overlays into the overlay area. The overlay area contains the overlay segments currently needed by the root segment.

Some programs using overlays require an additional part of main storage called the system/coresident overlay area. The system overlay area contains system input/output modules. The coresident overlay area contains user modules that do not call modules in the system overlay area.

Figure 18-12 shows how OLE breaks up the area of main storage into root area, user overlay area, and system/coresident overlay area, and the contents of all these areas.

Area of Main Storage	Contains
Root area	Root segment: • Fields (#FLDS) • Constants (#CNST) • Buffers (#BUFF) • DTFs Overlay fetch routine
User overlay area	One overlay segment at any given time
System/coresident overlay area	System input/output modules or User modules that are input/output independent

Figure 18-12. Main Storage Areas

Creating the Overlays

To create overlays, OLE first determines which routines go into the user overlay area and which routines go into the system/coresident overlay area. Then OLE calculates the size of the largest user overlay and the size of the largest system/coresident overlay. OLE rounds off these sizes upward in steps of 2K bytes when memory resident overlays (MRO) are used, and in steps of 256 bytes (1 sector) when MRO is not used (NOMRO). OLE then adds the sizes of the root segment, the largest user overlay, and the largest system/coresident overlay. If the sum is larger than the storage size specified on the control specifications, the program is too large to run in the storage size specified. If you do not want the program to run in a larger storage area, you must use additional storage-saving techniques to reduce the program size.

Reducing the Program Size

To reduce the size of your program, it is sometimes necessary to rearrange the contents of the root segment and/or the overlay segments. This can be done using the OLINK procedure. For information about the OLINK procedure, see the *Overlay Linkage Editor Guide*, SC21-9041.

Another way to reduce the size of your program is by trying to reduce the size of the root overlay segments or other overlays. First, however, you must identify the contents of the root segment and the largest overlay segments. Then you can determine whether the contents of these areas can be changed so that the program will fit into the storage size specified.

Use the Overlay Linkage Editor Storage Usage Map section of the compiler listing to find the contents of the root area, the user overlay area, and the system/coresident overlay area. The OLE storage usage map (see Figure 18-13) shows:

- Overlay number
- Code lengths of the overlay areas
- Start address of the overlay areas

All data and routines on the OLE storage usage map which are not given an overlay number in the Overlay Number Area are in the root segment; the rest are overlays.

User overlays are identified by a U, system overlays are identified by an S, and coresident overlays are identified by a C.

Note that user overlays and system/coresident overlays have different start addresses.

OVERLAY LINKAGE EDITOR STORAGE USAGE MAP

	START ADDRESS	OVERLAY NUMBER	AREA	CATEGORY	NAME AND ENTRY	CODE LENGTH HEXADECIMAL	DECIMAL	
Root	0000			0	RPF552	010E	270	
	010E			0	#FLDS	0196	406	
	02A4			0	#ROOT2	0271	625	
	0515			0	#BUFF	0477	1143	
	098C			0	#MISC	0010	16	
	099C			0	@PGTS	00A8	168	
	0A44			0	#IPCR	0070	112	
	0AB4			0	#OPCR	0040	64	
	0AF4			0	#CNST1	0010	16	
	0B04			0	@PGDM	0019	25	
	0B1D			21	#IHK01	0008	8	
	0B25			126	#IFLD	0039	57	
	0B3E				@0BF3			
	0B47				@0BFC			
	0B50				@0C05			
	0B5E			21	#OHK01	000C	12	
	0B6A			126	#LROF	001D	29	
	0B87			126	#RCDID	0073	115	
	0BAC				@0B66			
	0BFA			126	#CNFLD	0026	38	
	0C20			21	#IHK08	0008	8	
	0C28			50	#CHN00	004B	75	
	0C55				@10CE			
	0C73			21	#IHK0A	0008	8	
	0C7B			26	#SR008	002C	44	
	0CA7			21	#OHK0A	000C	12	
	0CB3			22	#SR002	004A	74	
	0CFD			11	@PGRI	0043	67	
	0D40			21	#IHK09	0008	8	
	0D48			126	#CNST0	005D	93	
	0DA5			21	#OHK09	000C	12	
Overlay Fetch Routine	0DB1				OVLFRTN	01D1	465	
User Overlay 1	1000	1	U	126	#INPUT	00B1	177	
	1079				@0B09			
	1080				@0B10			
	10A7				@0B37			
	1071				@0B01			
	1075				@0B05			
	10B1	1	U	126	#DEOUT	01B1	433	
User Overlay 2	1000	2	U	126	#DETC	009D	157	Largest User Overlay
	109D	2	U	25	#SR005	004C	76	
	10E9	2	U	50	@PGTR	0198	408	
System Overlay 3	1300	3	S	6	@PGTI	06DF	1759	Largest System/ Coresident Overlay
	19DF	3	S	6	@PGTO	02F0	752	
	1CCF	3	S	6	@PGTD	023B	571	
Coresident Overlay 4	1300	4	C	86	#OPEN	00B9	185	
	13B9	4	C	86	#CLOSE	0030	48	
	13C9				@0DE6			
	13E9	4	C	21	#SR001	004A	74	
	1433	4	C	50	#CHN02	005D	93	
	1460				@1173			
	1490	4	C	23	#SR003	0010	16	
	14A0	4	C	24	#SR004	0014	20	
	14B4	4	C	26	#SR006	00BD	189	
	1571	4	C	50	#CHN01	005A	90	
	159E				@1119			
	15CB	4	C	26	#SR007	0037	55	
	1602	4	C	15	@PGAA	00AD	173	
	16AF	4	C	28	@PGMC	006A	106	
	1719	4	C	22	#EXC01	00CD	205	

```
SYS-3130 I RPF552 MODULE'S MAIN STORAGE SIZE IS
           8192 DECIMAL
SYS-3131 I 0000 IS THE START CONTROL ADDRESS OF THIS MODULE
SYS-3132 I THE NONOVERLAY MAIN STORAGE SIZE IS
           9092 DECIMAL
SYS-3134 I RPF552 MODULE IS CATALOGED AS A LOAD MEMBER
           RPGEXLIB IS THE LIBRARY NAME
               40 TOTAL NUMBER OF LIBRARY SECTORS
```

Figure 18-13. Overlay Linkage Editor Storage Usage Map

After identifying the root segment, the largest user overlay, and the largest system/coresident overlay, you can determine whether they contain routines that can be changed to reduce the overlay size.

For each of the following routines:

- #INPUT (input records)
- #DETC (detail calculations)
- #TOTC (total calculations)
- #DEOUT (detail output)
- #TTOUT (total output)

you can use one of the following storage-saving techniques:

To reduce the size of Input Records:

- Process one or more of the input or update files as a demand file using the READ operation code, a full procedural file using the READ, READE, READP, or CHAIN operation code, or chained file using the CHAIN operation code. With a demand, full procedural, or chained file, the instructions to read the file can be moved into the total or detail calculation routines. Remember that total calculations are not done on the first cycle.

To reduce the size of Detail or Total Calculations:

- Use subroutine calculations. As the calculation subroutines are created, the compiler assigns to each subroutine a category for going into the overlay. The first subroutine in the calculations is assigned category 28. The remaining subroutines in the calculations are numbered consecutively 29, 30, 31, and 32. All subroutines after category 32 are also assigned category 32. The subroutines are placed into the overlay, if required, according to the assigned category. Category 32 subroutines go into the overlay first, and the category 28 subroutine goes in last. You should place the most frequently used subroutine first in your calculations, and the least frequently used subroutine last. This may reduce the number of overlay or system/coresident overlay loads.

  In some instances using subroutines can increase, rather than decrease, the storage required because of the nature of the existing calculation routines. If one subroutine calls another subroutine, both subroutines must be in storage at the same time. This can increase the size of the user overlay area or the system/coresident overlay area and thus the total storage required. Therefore, do not call a subroutine from another subroutine if trying to reduce the size of the detail or total calculations.

- Eliminate exception output if possible. This moves the logic for output operations conditioned by exception output to either total or detail output routines.

- Eliminate READ, READE, READP, and/or CHAIN operations by using matching records and consecutive processing. This moves the logic to the input records routine.

- Move part of the detail calculations to total calculations (or total calculation logic to detail calculations). Remember that total calculations are not done on the first cycle.

- Avoid using resulting indicators to reset indicators when a SETOF or SETON will work. Resulting indicators use 7 bytes each while a SETOF or SETON operation code uses 3 bytes.

- Use SETOF and SETON indicators in ascending order.

- Indicator bytes start with indicator numbers that are multiples of 8. For example, SETOF 08, 09, 10 requires one 3-byte instruction, while SETOF 07, 08, 09 requires two 3-byte instructions.

To reduce the size of Detail or Total Output:

- Use exception output. This moves part of the output logic to detail or total calculation routines.

- Move some of the output from total to detail output time, or from detail to total output time. This moves logic to the appropriate output routine.

- Do not specify blank after (column 39 of the output specifications) for fields, but clear them at the beginning of detail or total calculations.

Specific Coding Techniques

Load Module Size Considerations

You should be aware that potential future changes to RPG may increase the size of generated load modules. This may be a concern for programs with load modules approaching the 64K limit. For ways to reduce your storage needs, see the following section.

Storage-Saving Techniques

When OLE finds that a program is too large for the storage size specified, an error message is displayed. If your program is still too large after reducing or changing the overlays, you can use some of the following storage-saving techniques to reduce the main storage needed for your program:

- Divide the program into separate tasks, creating a separate program for each task. For example, if you want to update a file and print a listing of the updated file, you can save storage by updating the file with one program and printing the listing with another program.

- Eliminate unreferenced indicators. Eliminating unreferenced indicators can eliminate the instructions required to set the indicators on and off.

- Eliminate unnecessary conditioning indicators. For example, the following indicator tests are unnecessary:

  - If only one type of input record is to be processed, the indicator associated with that record is always on except during the first detail output time. Therefore, it is not necessary for any calculation to be conditioned with this indicator.
  - When two operations on the result field of a Z-ADD or Z-SUB operation are conditioned on opposite indicator conditions, one of the conditions may not be necessary. For instance, the N09 condition is not required for this example:

Line (3-5)	Form Type (6)	Control Level (L0-L9, LR, SR, AN/OR) (7-8)	Indicators (9-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field Name (43-48)	Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Arithmetic Plus / Compare 1>2 / Lookup(Factor 2)is High (54-55)	Minus / 1<2 / Low (56-57)	Zero / 1=2 / Equal (58-59)	Comments (60-74)
01	C		NØ9		Z-ADD	FLD	FLDB							
02	C		Ø9		Z-ADD	FLDC	FLDB							
03	C													

This technique might not work for certain operations if the same field is used as the result field and as factor 1 or factor 2.

- Reuse calculation work areas and temporary hold areas. Once the data stored in these areas is used for the last time in a given cycle, the area is available. Reusing these areas can eliminate the need for you to define two or more additional areas. However, the areas must be used for the same type of data.

- Reuse input field names. You can reuse input field areas by using the same name for fields in two or more files. This can be done only if the fields have the same attributes (length, alphameric/numeric, packed/binary) and each field is used only in the cycle in which the record is processed. Both files cannot be used in the same cycle.

- Include the necessary intervening blanks when describing alphameric fields and constants for output. This makes the fields adjacent. There is a module in the RPG compiler that optimizes moves so that all adjacent fields and constants can be moved with one instruction instead of using one instruction to move each line:

Not Optimized	**Optimized**
5'DAILY'	18'DAILY TRANSACTIONƀ'
17'TRANSACTION'	26'REGISTER'
26'REGISTER'	

- Use data structures to define the same internal storage area for multiple record types and to reduce the use of MOVE and MOVEL operations.

- Design files so that match fields and control fields are assigned the same position within all record types.

- Group calculation statements that are conditioned by the same indicators. When a large number of indicators are required, try to use GOTO or EXSR to reduce the number of indicator tests required on each statement.

- Use the actual bit numbers in factor 2 when using TESTB, BITON, or BITOF.

- Do not use half adjust unless necessary.

- Try to use either factor 1 or factor 2 as the result field whenever possible.

- Try to use numeric fields of the same length and with the same number of decimal positions. If the fields cannot be the same length, try to have the number of decimal positions the same.

- Do not sequence check your records unless necessary.

- Use OR lines on input specifications rather than multiple record lines because OR lines require less code.

- Specify the fields in a record in ascending order by record position.

- Do not use halt indicators unless necessary.

- Try to eliminate the use of variable indexes with arrays.

- Avoid defining unnecessary tables or arrays.

- Instead of defining all of the fields for an input record, only one large field or array needs to be defined. That field or array can also be defined within a data structure with all of the individual fields. This will save creating the code to move each field to its storage location. Each field will be moved implicitly when the large field or array is moved. This will not work for binary or packed input.

Performance-Improvement Techniques

The following relatively simple program changes can significantly improve a program's performance:

- If the DISP-SHR parameter is not specified on the control language FILE statement, block all sequentially processed indexed files and randomly processed indexed files (especially if the values of the indexes are closely related).

- Multiplication is faster than division and can be used to divide ($800 \div 4 = 800 \times .25$).

- LOKUP is usually much faster than a loop coded in RPG to find data in an array. The use of LOKUP should be balanced by the number of times the function must be performed and whether or not the LOKUP routine is already in the program. If LOKUP is used only once, it may be better to code a loop because you will use less code.

- When blanking out an entire array that is less than 256 bytes, it is faster to define the array as a field within a data structure. The array can then be blanked out by moving *BLANK to the field instead of moving *BLANK to the array.

- Packing and unpacking input/output data is faster than converting decimal to binary or converting binary to decimal.

Storage Requirements

Operation Codes

This section contains the number of bytes of storage required for various RPG operation codes. When used with the preceding information in this chapter, this information helps you determine the amount of storage that you can save by using certain coding practices. For example, one storage-saving technique is to use numeric fields of the same length and with the same number of decimal positions. If the fields cannot be the same length, try to have the number of decimal positions the same.

Figure 18-14 shows that if factor 1, factor 2, and the result field of an ADD operation all have a different number of decimal positions, the operation requires 27 bytes. However, if all the fields have the same number of decimal positions, the same ADD operation requires only 15 bytes. Uniformity of field lengths saves main storage not only for ADD and SUB, but also for most of the other arithmetic operations as well.

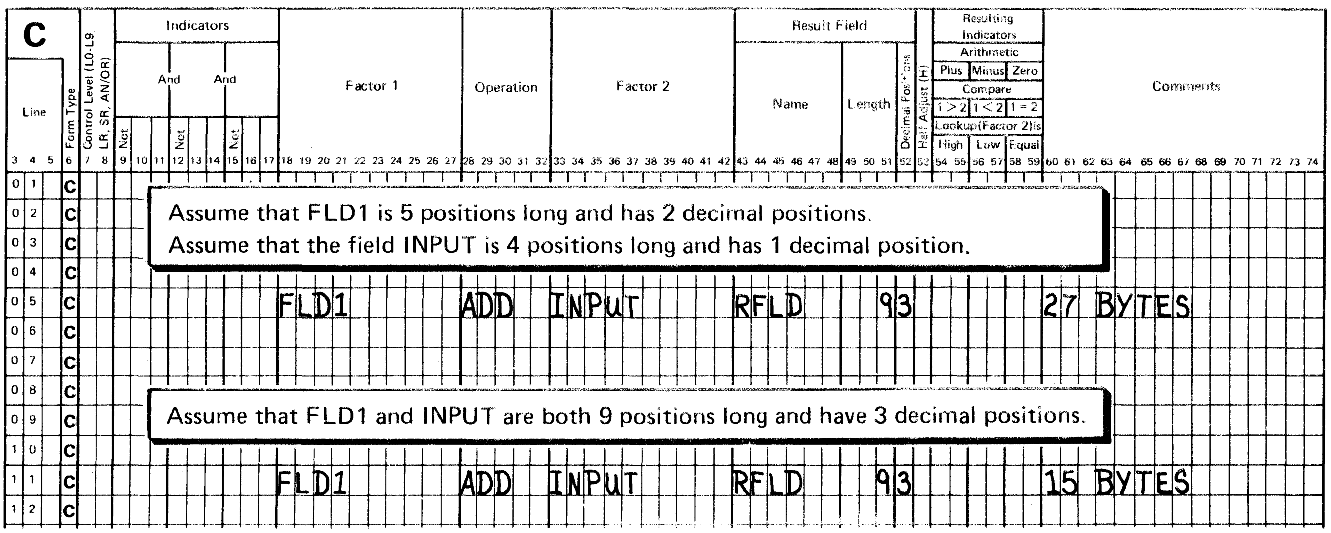


Figure 18-14. Amount of Storage Required for ADD Operation

The table below shows how many bytes of code the compiler generates for each operation. The **base** number refers to the number of bytes generated for an operation code itself, before any additional specific case bytes are generated. Total number of bytes = base bytes + specific case bytes.

Operation	Specific Case	Bytes
ACQ	Inline calculation code	12
	Subroutine	370
ADD	Factor 1 is the same field as the result field. Factor 1, factor 2, and the result field all have the same number of decimal positions.	6
	Factor 2 is the same field as the result field. Factor 1, factor 2, and the result field all have the same number of decimal positions.	6
	Factor 1, factor 2, and the result field all have different lengths. Factor 1, factor 2, and the result field all have the same number of decimal positions.	15
	Factor 1 is the same field as the result field. Factor 2 has more decimal positions than the result field.	27
	Factor 2 is the same field as the result field. Factor 1 has more decimal positions than the result field.	27
	Factor 1 is the same field as the result field. Factor 2 has more decimal positions than the result field. Half-adjust is specified.	18
	Factor 2 is the same field as the result field. Factor 1 has more decimal positions than the result field. Half-adjust is specified.	18
	Factor 1 is the same field as the result field. Factor 2 has fewer decimal positions than the result field.	18
	Factor 2 is the same field as the result field. Factor 1 has fewer decimal positions than the result field.	18
	Factor 1 has the same number of decimal positions as factor 2. Factor 1 and factor 2 have fewer decimal positions than the result field.	23
	Factor 1 is the same field as the result field. The length of factor 2 minus the number of decimal positions for factor 2 is longer than the length of factor 1 minus the number of decimal positions for factor 1.	27
	Factor 2 is the same field as the result field. The length of factor 1 minus the number of decimal positions for factor 1 is longer than the length of factor 2 minus the number of decimal positions for factor 2.	27
	Factor 1 is the same field as the result field. The length of factor 2 minus the number of decimal positions for factor 2 is longer than the length of factor 1 minus the number of decimal positions for factor 1. Half-adjust is specified.	35
	Factor 2 is the same field as the result field. The length of factor 1 minus the number of decimal positions for factor 1 is longer than the length of factor 2 minus the number of decimal positions for factor 2. Half-adjust is specified.	35
	All other combinations without half-adjust specified.	27
	All other combinations with half-adjust specified.	35

Operation	Specific Case	Bytes
BITOF		4
BITON		4
CAS	If there is only one CAS in the group.	4
	First CAS in a multiple CAS group.	16
	Last CAS in a multiple CAS group.	4
	CAS in a multiple CAS group, other than first or last.	11
	Factor 1 and factor 2 are numeric. Factor 1 and factor 2 have the same number of decimal positions.	10
	Factor 1 and factor 2 are numeric. Factor 1 and factor 2 do not have the same number of decimal positions.	18
	Factor 1 and factor 2 are alphameric. Factor 1 and factor 2 are the same length.	6
	Factor 1 and factor 2 are alphameric. Factor 1 and factor 2 are not the same length.	22
	Factor 1 and factor 2 are alphameric. Factor 1 is a table.	26
	Alternative collating sequence (add these bytes to the appropriate CAS listed previously).	10
CHAIN (base = 16)	With external indicator.	6
	When factor 1 has a variable index.	11
	When key is not packed.	9
	When key is packed.	18
	When factor 1 is a table element.	6
	When key is a record number.	5
	When key is a record number and RECNO is specified.	13
	When record-not-found indicator is given.	12
	When record-not-found indicator is not given.	15
	When the file is a full-procedural file.	3
COMP	Factor 1 and factor 2 are numeric. Factor 1 and factor 2 have the same number of decimal positions.	10
	Factor 1 and factor 2 are numeric. Factor 1 and factor 2 do not have the same number decimal positions.	18
	Factor 1 and factor 2 are alphameric. Factor 1 and factor 2 are the same length.	6
	Factor 1 and factor 2 are alphameric. Factor 1 and factor 2 are not the same length.	22
	Factor 1 and factor 2 are alphameric. Factor 1 is a table.	26
	Alternative collating sequence (add these bytes to the appropriate COMP listed previously).	10
DEBUG	Inline code	11-20
	Subroutine	1111

Operation	Specific Case	Bytes
DIV	inline calculation code	
	The number of decimal positions in factor 1 minus the number of decimal positions in factor 2 is the same length as the number of decimal positions in the result field.	23
	The number of decimal positions in factor 1 minus the number of decimal positions in factor 2 is not the same length as the number of decimal positions in the result field.	27
	The number of decimal positions in factor 1 minus the number of decimal positions in factor 2 is the same length as the number of decimal positions in the result field plus 1. Half-adjust is specified.	31
	The number of decimal positions in factor 1 minus the number of decimal positions in factor 2 is not the same length as the number of decimal positions in the result field plus 1. Half-adjust is specified.	35
	Subroutine	92
DO		20
DOU (base = 7)	Factor 1 and factor 2 are numeric. Factor 1 and factor 2 have the same number of decimal positions.	10
	Factor 1 and factor 2 are numeric. Factor 1 and factor 2 do not have the same number of decimal positions.	18
	Factor 1 and factor 2 are alphameric. Factor 1 and factor 2 are the same length.	6
	Factor 1 and factor 2 are alphameric. Factor 1 and factor 2 are not the same length.	22
	Factor 1 and factor 2 are alphameric. Factor 1 is a table.	26
	Alternative collating sequence (add these bytes to the appropriate DOU listed previously).	10
DOW (base = 4)	Factor 1 and factor 2 are numeric. Factor 1 and factor 2 have the same number of decimal positions.	10
	Factor 1 and factor 2 are numeric. Factor 1 and factor 2 do not have the same number of decimal positions.	18
	Factor 1 and factor 2 are alphameric. Factor 1 and factor 2 are the same length.	6
	Factor 1 and factor 2 are alphameric. Factor 1 and factor 2 are not the same length.	22
	Factor 1 and factor 2 are alphameric. Factor 1 is a table.	26
	Alternative collating sequence (add these bytes to the appropriate DOW listed previously).	10
ELSE		4
END (CAS)		0
END (IF/ELSE)		0
END (DO)		10

Operation	Specific Case	Bytes
END (DOU)		4
END (DOW)		4
EXCPT		4
EXSR		4
FORCE	Without an external indicator.	13
	With an external indicator.	13 + 7 = 20
GOTO		4
IF (base = 4)	Factor 1 and factor 2 are numeric. Factor 1 and factor 2 have the same number of decimal positions.	10
	Factor 1 and factor 2 are numeric. Factor 1 and factor 2 do not have the same number of decimal positions.	18
	Factor 1 and factor 2 are alphameric. Factor 1 and factor 2 are the same length.	8
	Factor 1 and factor 2 are alphameric. Factor 1 and factor 2 are not the same length.	22
	Factor 1 and factor 2 are alphameric. Factor 1 is a table.	26
	Alternative collating sequence (add these bytes to the appropriate IF listed previously).	10
KEYnn (base = 27)	When the result field is a variable indexed array.	11
	When the result field is numeric and a table element.	6
	With each resulting indicator.	14
	The result field is alphameric.	0
	The alphameric result field is used with a resulting indicator and the field length is more than 1.	23
	The alphameric result field is used with a resulting indicator and the field length is equal to 1.	7
	Factor 1 is numeric. Factor 1 is used with a resulting indicator and the field length is more than 1.	8
	Factor 1 is numeric. Factor 1 is used with a resulting indicator and the field length is equal to 1.	6
LOKUP (base = 15)	Inline calculation code	
	Factor 1 is a table.	6
	When factor 1 is a variable.	11
	With each resulting indicator.	12
	Subroutine	361
MHHZO		See Figure 18-11
MHLZO		See Figure 18-11

Operation	Specific Case	Bytes
MLHZO		See Figure 18-11
MLLZO		See Figure 18-11
MOVE		See Figure 18-11
MOVEA	Inline calculation code.	14
	Subroutine.	367
MOVEL		See Figure 18-11
MULT	Inline calculation code.	23
	Half-adjust is specified.	27
	Subroutine	106
MVR	Factor 2 and the result field have the same number of decimal positions.	5
	Factor 2 and the result field do not have the same number of decimal positions.	9
NEXT	Inline calculation code.	12
	Subroutine.	253
POST	Inline calculation code.	8
	Subroutine.	448
READ (base = 29)	With an external indicator.	6
	With EOF indicator and BSCA file.	6
	With EOF indicator, but without BSCA file.	12
	With BSCA file, but without EOF indicator.	24
	Without BSCA file and without EOF indicator.	18
	With RAF limits.	6
	With WORKSTN file and with error indicator.	15
	When file is a full-procedural file.	3
	When RECNO is specified.	13
READE (base = 51)	With external indicator	6
	When factor 1 has a variable index	11
	When key is not packed	9
	When key is packed	18
	When factor 1 is a table element	6
READP (base = 47)	With external indicator	6
	When RECNO is specified	13

Operation	Specific Case	Bytes
REL	Inline calculation code.	12
	Subroutine.	408
SETnn (base = 27)	With ERASE function.	4
	Factor 1 is numeric. Factor 1 is used with a resulting indicator and the field length is more than 1.	8
	Factor 1 is numeric. Factor 1 is used with a resulting indicator and the field length is equal to 1.	6
SETnn/KEYnn combination (base = 27)	See KEYnn operation for code in addition to base. If factor 1 code appears on both SET and KEY instructions, both counts should be included.	
SETLL (base = 15)	When key is packed.	12
SETON (each indicator set on)		3
SETOF (each indicator set off)		3
SHTDN		22
SORTA	Inline calculation code.	7
	Subroutine.	464
SQRT	Inline calculation code.	
	Nonarray	12
	Array	46
	Subroutine	371
SUB	Factor 1 is the same field as the result field. Factor 1, factor 2, and the result field all have the same number of decimal positions. The length of factor 1 is greater than or equal to the length of factor 2.	6
	Factor 1 is not the same field as the result field. Factor 1, factor 2, and the result field all have the same number of decimal positions.	15
	Factor 1 is not the same field as the result field. Factor 2 and the result field have the same number of decimal positions.	23
	Factor 1 is not the same field as the result field. Factor 2 and the result field have the same number of decimal positions. Half-adjust is specified.	27
	All other combinations without half-adjust specified.	31
	All other combinations with half-adjust specified.	39

Operation	Specific Case	Bytes
TESTB	Test bit off.	10
	Test bit mixed.	17
	Test bit on.	10
	Test bit off and mixed.	23
	Test bit off and on.	23
	Test bit mixed and on.	23
	Test bit off, mixed, and on.	29
TESTZ	Inline calculation code.	
	The result field is a field.	9
	The result field is a table.	21
	Subroutine	43
TIME	Time only.	21
	Time and system date.	21
XFOOT	Factor 2 and the result field have the same number of decimal positions.	9
	Factor 2 and the result field do not have the same number of decimal positions.	13
Z-ADD	Factor 2 and the result field have the same number of decimal positions.	6
	The number of decimal positions in factor 2 is more than the number of decimal positions in the result field.	14
	The number of decimal positions in factor 2 is more than the number of decimal positions in the result field. Half-adjust is specified.	18
	The number of decimal positions in factor 2 is less than the number of decimal positions in the result field.	18
Z-SUB	Factor 2 and the result field have the same number of decimal positions.	14
	Factor 2 and the result field do not have the same number of decimal positions.	18
	Factor 2 and the result field do not have the same number of decimal positions. Half-adjust is specified.	22

	MOVE Alphameric/Numeric	MOVEL Length of Result Field < Length of Factor 2 Numeric	MOVEL Length of Result Field > Length of Factor 2 Numeric	MOVEL Length of Result Field = Length of Factor 2 Alphameric/Numeric	MOVEL Length of Result Field < Length of Factor 2 Alphameric	MOVEL Length of Result Field > Length of Factor 2 Alphameric	MLHZO	MHLZO	MHHZO	MLLZO
Field to Field	6	26	10	6	6	6	20	20	20	20
Array to Array	42	55	45	42	42	42	42	42	42	42
Field to Array	29	43	32	29	29	29	29	29	29	29
Table to Array	35	53	38	35	40	35	35	41	40	35
Array, Variable Index to Array	40	66	43	40	52	40	40	52	52	40
Array, Variable Index to Array, Variable Index	28	57	38	28	35	35	35	35	47	42
Field to Array, Variable Index	17	34	27	17	17	24	24	31	24	31
Table to Array, Variable Index	20	52	33	20	24	30	30	24	36	20
Array, Variable Index to Table	20	46	27	20	30	24	24	24	36	20
Field to Table	9	23	16	9	9	13	13	9	13	9
Table to Table	15	41	22	15	19	19	19	19	25	15
Array, Variable Index to Field	17	40	21	17	24	17	31	24	36	31
Table to Field	9	29	13	9	13	9	9	13	13	9

Figure 18-15. Amount of Storage Required for MOVE Operations

Indicators

Indicators	Specific Case	Bytes
Conditioning indicators	Each indicator	3
	Each AND type	3
Resulting indicators (does not apply to CHAIN, FORCE, LOKUP, READ, READE, and READP)	First indicator specified	8
	Each additional indicator specified	3

Array Processing

Array control code (initialization and processing) is created for all calculations except LOKUP, CHAIN, READ, and FORCE.

Operation	Specific Case	Bytes
Array initialization	Factor 1 or factor 2 is an array.	6
	Factor 1 or factor 2 is a table.	4
	Factor 1 or factor 2 is an array with a variable index.	11
Array processing	Factor 1, factor 2, and the result field are arrays.	28
	Factor 1 and the result field are arrays. Factor 2 and the result field are arrays.	22
	The result field is an array.	16

If a SUB operation code is specified and has the following conditions:

- Factor 1 is the same field as the result field,
- Factor 1, factor 2, and the result field have the same number of decimal positions,
- Factor 1 and the result field are full arrays,
- Factor 2 is a table,

the length of object code created is as follows:

Operation	Specific Case	Bytes
Array initialization	Factor 1 is an array.	6
	Factor 2 is a table.	4
	The result field is an array.	6
SUB		6
Array processing	Factor 1 and the result field are arrays.	22

Thus, the total bytes of code created for a SUB operation code is 44 bytes.

Whenever an array with a variable index is specified in a program (except with a MOVEA operation), the following are also created:

Inline code	11 bytes
Subroutine	173 bytes

Part 2. Reference

Chapter 19. RPG Program Cycle

Chapter 19. RPG Program Cycle

The RPG program cycle controls certain operations performed on each record, so the program cycle partly determines how you can process your data. The phrase *program cycle* refers to the series of operations that an RPG program performs automatically on each record that it reads.

Overview of RPG Program Cycle

Each RPG program goes through the same general cycle of operations. This cycle of operations has three basic logic steps:

- Reading information (input)
- Doing calculations (processing)
- Writing results (output)

These basic logic steps can be divided into several substeps in which you can assign indicators to control when calculation and output operations occur. For more information on indicators, see Chapter 12, *Using Indicators*.

Calculation and output operations occur at two different times in a program cycle: *detail time* and *total time* (see Figure 19-1). At detail time, the RPG program calculates and writes data for one or more records. For example, if a customer uses his charge account three times and we print a record of each charge, we are printing three detail records. At total time, the program calculates and writes data for a series of related records. If a customer uses his charge account three times and we print only one record that shows the total of the three charges, we are printing one total record.

Generally, totals are calculated and written for data accumulated from a group of related records, called a *control group*. A control group is a set of records all having the same information in a *control field* In an accounts receivable program, for example, you could use the customer account number as a control field; in an inventory program, you could use the part number. Each time a record is read, the program checks the information in the control field to determine whether it differs from the control-field information in the previous record. When the information differs, a *control break* occurs. A control break means that all records from a control group have been read and that a new group is starting. When all records from a control group have been read, the program does the operations coded for that group. Data from the record that starts the new control group is not included in the total operations.

To indicate which field is a control field, you assign one of the control-level indicators (L1 through L9) to that field in columns 59 and 60 of the input specifications. To indicate which calculations are total calculations, you also write this same control-level indicator in columns 7 and 8 of those calculation specifications. Those calculations that do not have a control-level indicator written in columns 7 and 8 are detail calculations. On the output specifications, you do not use control-level indicators to identify detail and total records. Instead, you use a T in column 15 of the output specifications to indicate a total output operation, and you use an H (for heading) or a D (for detail) to indicate a detail output operation.

The program does detail calculations and detail output operations for each record it reads (that is, for each program cycle) if all conditioning indicators are satisfied, regardless of whether it does total calculations or total output. Detail calculations and detail output operations occur in either of the following cases:

- All total calculation and total output operations are complete for a control group, but the last record is not processed.

- No total operations are to be done (the information in the control field has not changed).

Figure 19-1 shows the basic steps in the RPG program cycle. Figure 19-2 is a flowchart for the same steps. A program cycle begins with step 1 and continues through step 11. Then the next cycle begins again with step 1. Steps 7 and 8 are known as total time, and steps 1 and 11 are known as detail time.

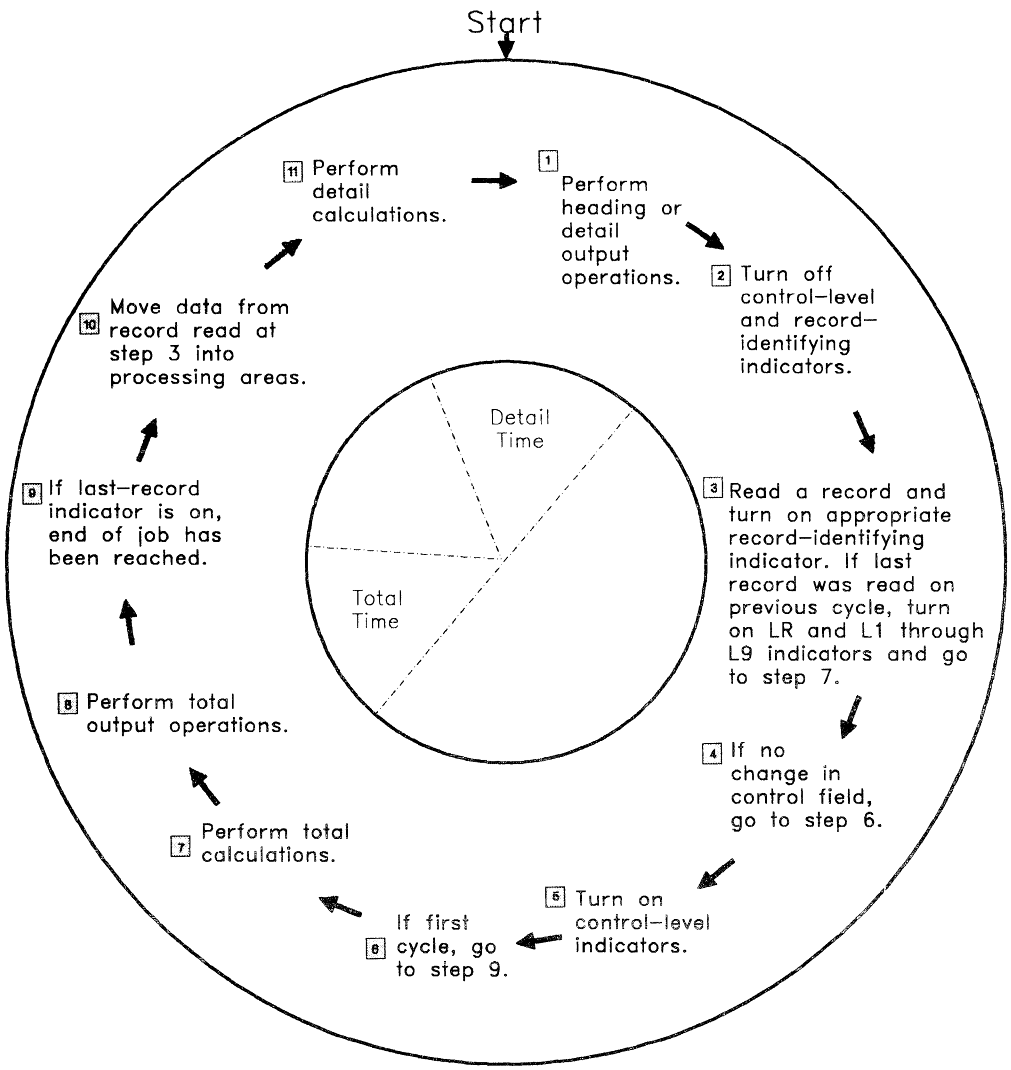


Figure 19-1. Steps in the RPG Program Cycle

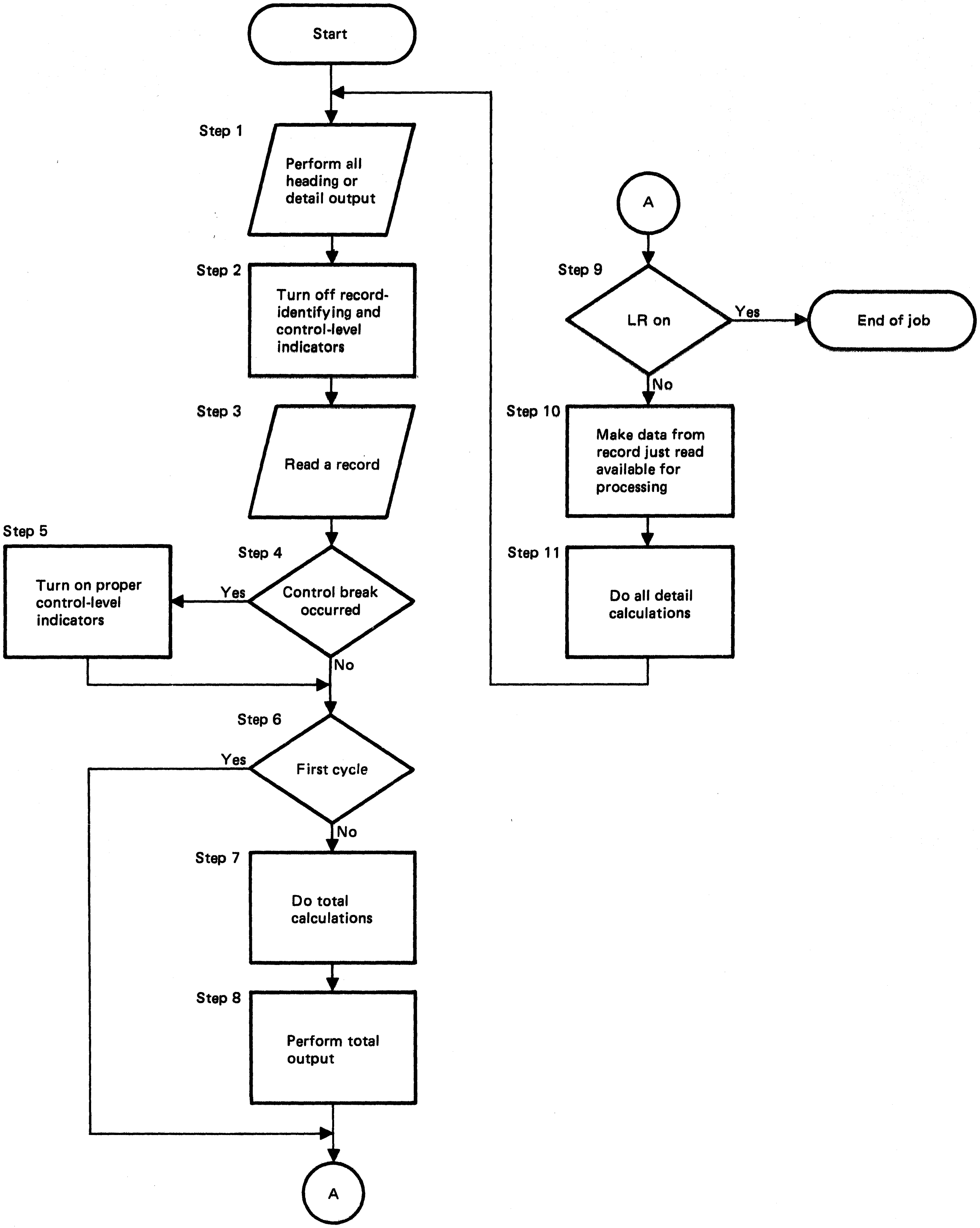


Figure 19-2. Flowchart for the RPG Program Cycle

The following statements describe what the RPG program does at each step in the program cycle. The steps are the same as those shown in Figures 19-1 and 19-2.

1. If the conditioning indicators are satisfied, the program does the heading or detail output (those lines that have H or D in column 15 of the output specifications).

2. The program turns off all control-level and record-identifying indicators.

3. The program reads a record and turns on the appropriate record-identifying indicator.

4. The program determines whether a control break occurred. (A control break occurs when the control field of the record just read differs from the control field of the previous record.)

5. If a control break occurs, the program turns on the proper control-level indicator and all lower control-level indicators except L0, which is always on.

6. If this is the first cycle, the program goes to step 9.

7. The program does total calculations (those conditioned by control-level indicators in columns 7 and 8 of the calculation specifications) if the appropriate control-level indicators are on.

8. The program does total output operations (those lines that have T in column 15 of the output specifications) if the indicators on those lines are on.

9. The program determines whether the last-record indicator is on. If it is, all records have been processed, and the program ends.

10. The program makes data from the record read at the beginning of the cycle (step 3) available for use in detail calculations and output.

11. The program does all detail calculations (those not conditioned by control-level indicators in columns 7 and 8 of the calculation specifications) on the data from the record read at the beginning of the cycle.

The first and last cycles of a program differ somewhat from the other cycles. Before the first record is read in the first cycle, the program prints lines conditioned by the first-page (1P) indicator. The program also performs any heading or detail output operations having no conditioning indicators or all negative conditioning indicators. Heading lines printed before the first record is read might consist of constants, page headings, or fields for reserved words such as PAGE and UDATE. In addition, the program bypasses total calculations and total output steps.

During the last program cycle, when no more records are available, the last-record indicator turns on, automatically causing all control-level indicators to turn on. The program performs the total calculations and total output operations, and the program ends.

Detailed RPG Program Cycle

Figure 19-3 shows the steps in the RPG program cycle in more detail. Steps 1 and 2 are for the first record only. The program cycle, which occurs for each record read, begins with step 3. The program cycle continues through step 26, however, the program may branch to steps out of the actual cycle when specified indicators are set on or off or certain conditions are met.

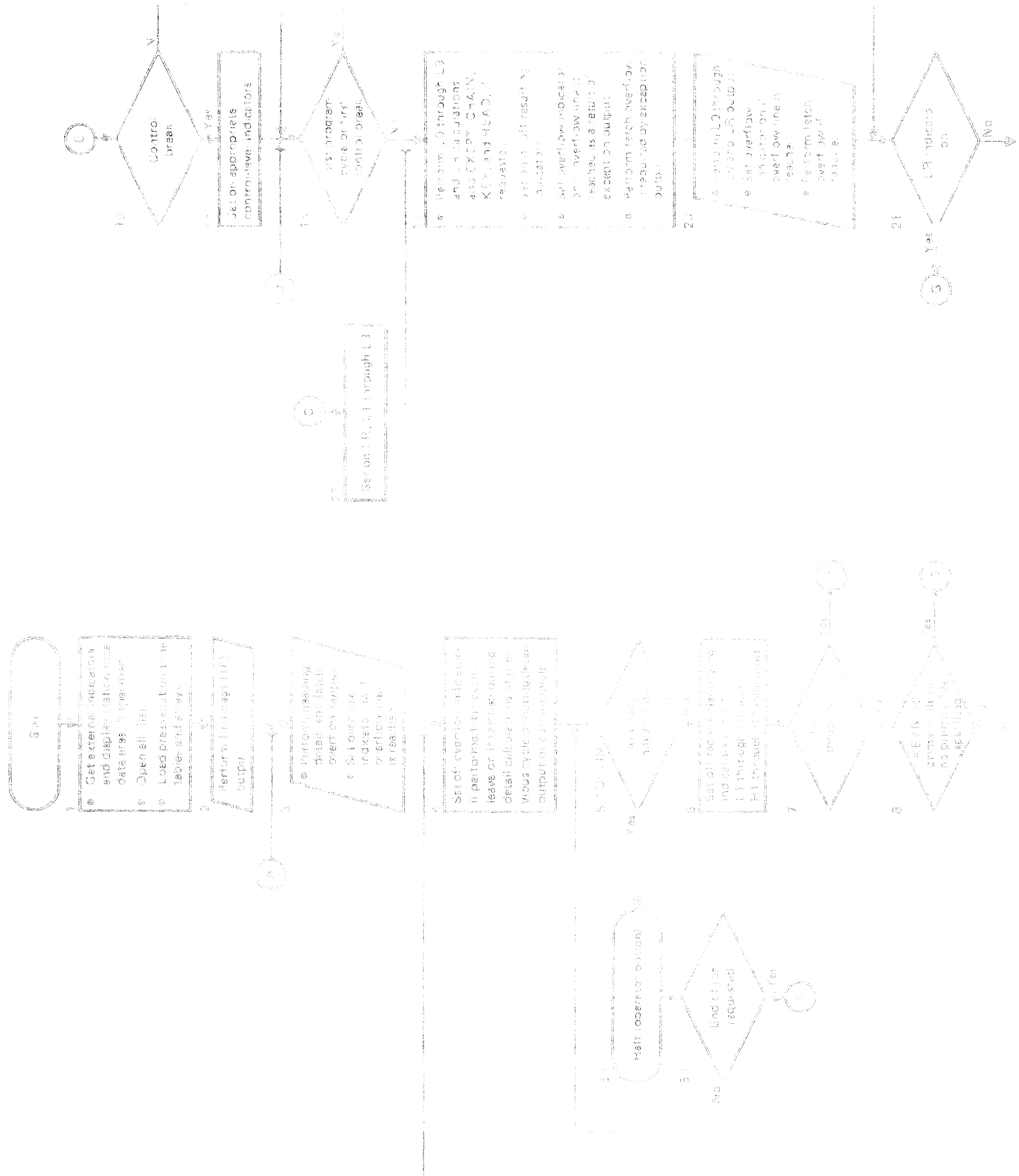


Figure 19-3 (Part 1 of 2). Detailed RPG II Object Program Cycle

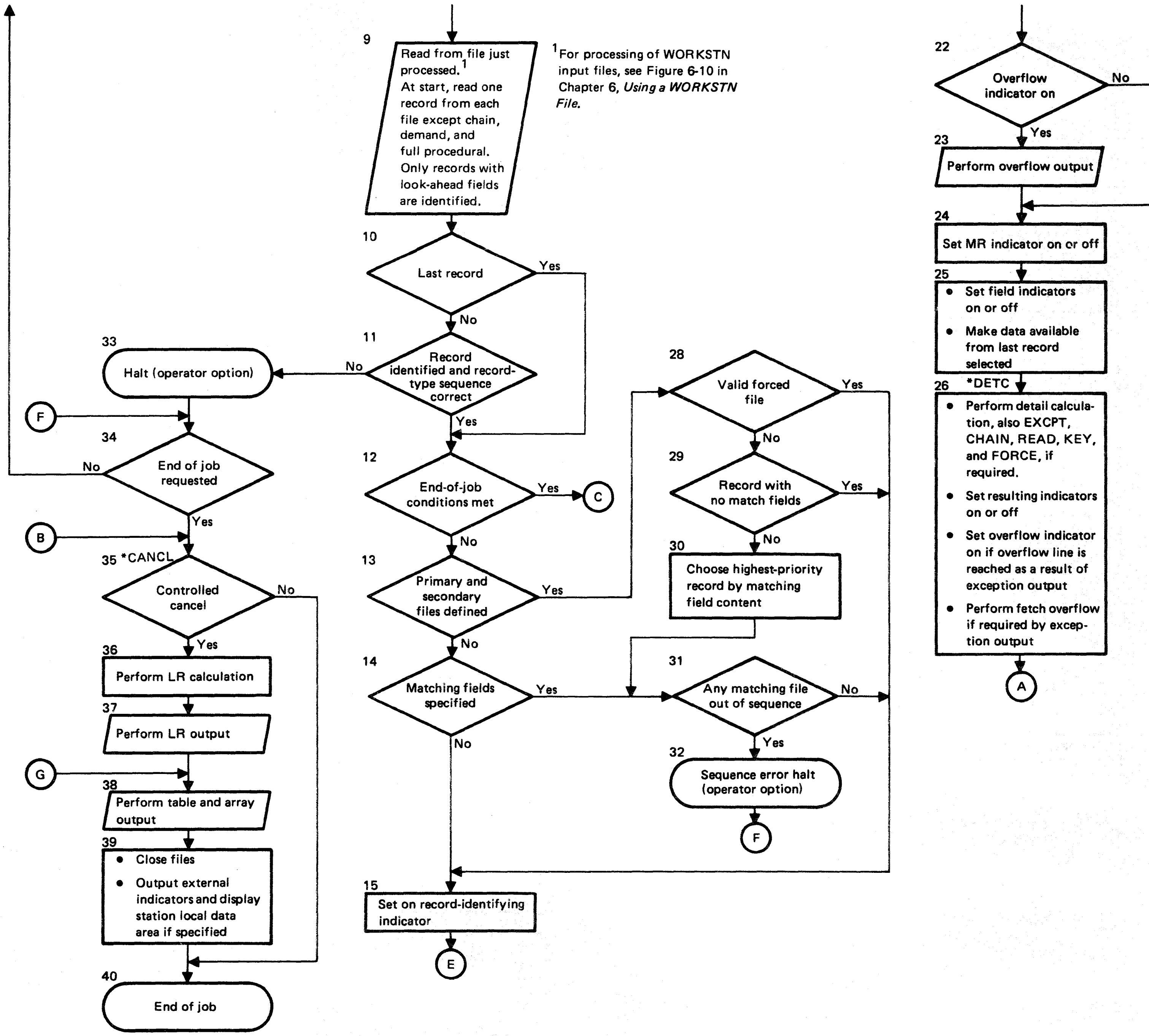


Figure 19-3 (Part 2 of 2). Detailed RPG II Object Program Cycle

The following steps describe in detail what the RPG program does at each step in the detailed program cycle. The steps are the same as those shown in Figure 19-3.

1. The program reads in the external indicators and the display station local data area, if specified, and opens all data files to be used; that is, the files are prepared to be processed. Before the first program cycle, data structures are blanked, and preexecution-time arrays and tables are loaded.

2. The program writes all output conditioned by the first-page indicator (1P). This output is written only once for each job and is not part of the program cycle (steps 3 through 26).

3. The program writes all headings and detail output whose conditions are satisfied. This output includes specifications that are conditioned by the overflow indicator if the overflow routine has been fetched.

4. The program determines whether the overflow line was reached during detail calculations in the previous cycle or when heading and detail records were written in the current cycle. If the printed output from the program reaches the overflow line, the overflow indicator is set on. Otherwise, the indicator is set off unless the overflow routine was fetched in step 3.

5. The program tests the halt indicators. If the halt indicators are off, the program branches to step 6. If the halt indicators are on, the program stops once for each halt indicator that is on. Every time the program stops, you select one of three options:

   a. Continue (the program returns to step 5 to test for other halt indicators)
   b. Controlled cancel (the program branches to step 35)
   c. Immediate cancel (the program branches to step 35)

6. The program sets off all record-identifying indicators and indicators with the 2-character entries 1P, L1 through L9, and H1 through H9.

7. The program determines whether the last-record indicator (LR) is on. If it is on, the program branches to step 27.

8. The program determines whether KEYBORD is specified as the device for the primary file, or whether no primary file is specified. For either condition, the program branches to step 18.

9. The program reads (and translates, if necessary) the next input record. At the beginning of processing, one record from each input file (except forced files, CHAIN files, full-procedural files, and DEMAND files) is read. If the file has look-ahead fields, the file is read only on the first cycle. After that, only records with look-ahead fields are identified. If this is a WORKSTN file and the SAVDS or IND option is specified, the common SAVDS or IND area is moved to the active display station's SAVDS or IND hold area. The next record is accepted, and the current display station's SAVDS or IND area is moved from its hold area to the common SAVDS or IND area.

10. The program tests to determine whether the record is an end-of-file record. If it is an end-of-file record, the program branches to step 12.

11. If the record is not an end-of-file record, the program determines whether the input records are in the order specified on the input specifications sheet. If the order is incorrect, the program branches to step 33. The program also branches to step 33 if input records are not specified in order and the record cannot be identified.

12. If end-of-job conditions have been met, the program branches to step 27. All files for which an E is specified in position 17 of the file description specifications must be at end of file.

13. When more than one input file is used, the program must select the next record to process and branch to step 28.

14. If there is only one input file, no record selection is needed. The program determines whether sequence checking is requested. If so, the program branches to step 31.

15. The program sets on the record-identifying indicator specified for the current record type. Data from the current record type is not available for processing until step 25.

16. If the record contains control fields, the program determines whether a control break has occurred. (A control break occurs when the contents of the control field are not equal to the contents of the previously stored control field.) If a control break has not occurred or if control fields are not specified, the program branches to step 18.

17. If a control break has occurred, the program sets on the control-level indicator showing the condition. All lower control-level indicators are also set on.

18. The program determines whether the total time calculations and total time output should be done. If no control-level indicators are specified on the input specifications, the totals are bypassed only on the first cycle. If control-level indicators are specified on the input specifications, totals are bypassed until after the first record containing control fields is processed. Totals are always processed when the last-record indicator (LR) is on.

19. The program does all calculations conditioned by control-level indicators (in positions 7 and 8 of the calculation specifications) and sets resulting indicators on or off as specified. If the last-record indicator (LR) is on, calculations conditioned by LR are done after other total calculations. File translation, if specified, is done for exception output and for CHAIN, READ, READE, READP and KEY operations. Fetch overflow is done if it is required by exception output. If the overflow line has been reached because of the exception output, the overflow indicator is set on.

20. The program writes all total output that is not conditioned by an overflow indicator. The program determines whether an overflow

condition has occurred. If an overflow condition has occurred at any time during this cycle, the overflow indicator is set on. If the last-record indicator is on, output conditioned by LR is written after other total output. File translation, if specified, is done for total output. Fetch overflow is done if required.

21. The program determines whether the last-record indicator is on. If the indicator is on, the program branches to step 38.

22. The program determines whether any overflow indicators are on. If no overflow indicators are on, the program branches to step 24.

23. The program does all output operations conditioned by a positive overflow indicator (no N before the indicator). File translation, if specified, is done for overflow output.

24. The program sets on the matching-record (MR) indicator if this is a job that processes more than one input file and if the record to be processed is a matching record. Otherwise, the matching-record indicator is set off.

25. The program sets the field indicators on or off as specified. Data from the last record read and from specified look-ahead fields is made available for processing. For a WORKSTN file only, the program sets off command-key indicators (KA through KN, KP through KY). If you press a command key for the WORKSTN file being processed, the program sets that command-key indicator on.

26. The program does any calculations not conditioned by control-level indicators (in columns 7 and 8 of the calculation specifications), and sets resulting indicators on or off as specified. The program translates files for exception output and for CHAIN, READ, READE, READP, FORCE, and KEY operations if specified. Fetch overflow is done if it is required by exception output. If the overflow line is passed because of the exception output, the program sets the overflow indicator on. Processing continues with step 3.

27. The program sets on the last-record (LR) indicator and all control-level indicators (L1 through L9), and processing continues with step 19.

28. If a file was forced or if NEXT was specified, the program selects the next record in that file for processing, and the program branches to step 15.

29. If a record with no match fields is found in a normal input file that is not at end of file, the program selects the record for processing, and the program branches to step 15.

30. When match fields are specified, the program selects the normal file with the highest priority matching record field. If two or more files have equal and highest priority matching record fields, the highest priority file is selected. (The primary file has the highest file priority, the first specified secondary file is next, and so forth.)

31. The program compares the match field value with the match field value of the last record. If it is in sequence, the record is accepted, and processing continues with step 15.

32. The program stops because a file with match fields is out of order. The operator's options, indicated in step 34, are to bypass the record (read the next record from the same file) or to cancel the job.

33. The program stops because a record type was out of order or because a record was unidentified.

34. The program tests the operator's decision either to bypass the record that caused the error condition (branch to step 4) or to cancel the job.

35. If the operator chooses to end the job by a controlled cancel, the program does steps 36 through 40. If the operator chooses an immediate cancel, the job ends.

36. The program does all operations conditioned by the last-record (LR) indicator.

37. Same as 36.

38. The program writes any tables or arrays for which a filename entry is specified on the extension specifications (columns 19 through 26). Output tables or arrays are translated, if necessary.

39. The program closes all the files it used and writes the external indicators and display station local data area, if specified.

40. End of job occurs.

Chapter 20. Control Specification

Chapter 20. Control Specification

The control specification describes your program and the computer system to the RPG compiler. One control specification is required for each source program. The control specification should always be the first specification in the program. Write the control specification on the first line of the RPG Control and File Description Specifications sheet (see Figure 20-1).

If you omit the control specification from the source program, the compiler creates a blank control specification.

See the individual column descriptions for the meaning of blank entries.

IBM International Business Machines Corporation

RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

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

Program		Keying Instruction	Graphic	Card Electro Number
Programmer	Date		Key	

Page (1 2) of ___   Program Identification (75 76 77 78 79 80)

Control Specifications

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

Columns	Field
3–5	Line
6	Form Type
7–9	Size to Compile
10	Object Output
11	Listing Options
12–14	Size to Execute
15	Debug
16–17	Reserved
18	Currency Symbol
19	Date Format
20	Date Edit
21	Inverted Print
22	Reserved
23–25	Number of Print Positions
26	Alternate Collating Sequence
27–36	Reserved
37	Inquiry
38–39	Reserved
40	Sign Handling
41	1P Forms Position
42	Indicator Setting
43	File Translation
44	Punch MFCU Zeros
45	Nonprint Characters
46	Reserved
47	Table Load Halt
48	Shared I/O
49	Field Print
50	Formatted Dump
51	RPG to RPG II Conversion
52–53	Number of Formats
54	S/3 Conversion
55	Subprogram
56	CICS/DL/I
57	Transparent Literal
58–74	

Line 0 1, Form Type H

File Description Specifications

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

Columns	Field
3–5	Line
6	Form Type
7–14	Filename
15	File Type: I/O/U/C/D
16	File Designation: P/S/C/R/T/D/F
17	End of File: E
18	Sequence: A/D
19	File Format: F/V/S/M/D/E
20–23	Block Length
24–27	Record Length
28	Mode of Processing: L/R
29–30	Length of Key Field or of Record Address Field
31	Record Address Type: A/P/I/K
32	Type of File Organization or Additional Area: I/X/D/T/R/ or 2
33–34	Overflow Indicator
35–38	Key Field Starting Location
39	Extension Code E/L
19–28	External Record Name
40–46	Device
47–52	Symbolic Device
53	Labels S/N/E/M
54–59	Name of Label Exit
60–65	Extent Exit for DAM
60–65	Storage Index
53	Continuation Lines: K
54–59	Continuation Lines: Option
60–65	Continuation Lines: Entry
66	File Addition/Unordered: A/U
67	Number of Tracks for Cylinder Overflow
68–69	Number of Extents
70	Tape Rewind: R/U/N
71–72	File Condition U1-U8, UC
73–74	

Lines: 0 2 F, 0 3 F, 0 4 F, 0 5 F, 0 6 F, 0 7 F, 0 8 F, 0 9 F, 1 0 F, F, F

*Number of sheets per pad may vary slightly.

Figure 20-1. RPG Control and File Description Specifications

Columns 1-2 (Page)

Entry	Explanation
Blank	No page number is used.
01-99	Page number.

Use columns 1 and 2 in the upper right corner of each sheet to number the specifications sheets, in ascending order, for your job. You can use more than one of each type of sheet, but keep all sheets of the same type together.

Columns 3-5 (Line)

Entry	Explanation
Blank	No line number is used.
Any numbers	Line numbers.

Use columns 3 through 5 to number the lines on each page. Columns 3 and 4 are preprinted on each sheet so, in most cases, line numbering is already done.

Page and line numbers are optional entries and are not required to successfully run an RPG program. Columns 1 through 5 are checked for ascending order, and RPG prints an S in the left margin of the RPG listing for any statement that is out of order. If you use SEU to enter the source program, you can request that SEU put the statements in order.

The control specification line is always line 01. Any other lines on the sheets can be skipped. The line numbers used need not be consecutive, but should be in ascending order.

Column 6 (Form Type)

An H must appear in column 6 to identify this line as the control (header) specification.

Column 7 (Comments)

Entry	Explanation
*	Comment line

Use an asterisk in column 7 to identify the line as a comment line. Use comments throughout your program to document the purpose of a certain section of coding. You can use any character in a comment line. Comments are not instructions to your program; they only document your program.

Columns 7-9 (Size to Compile)

Columns 7 through 9 are not used to specify the storage size. Leave them blank. Any entry in these columns is ignored by the compiler. The program is compiled in the available storage specified by the control language REGION statement. If no region size is specified, the default region size for the session is used. If the region size is less than 18K, the compiler uses 18K because that is the minimum size required to run the compiler.

Columns 7-12 (/EJECT)

Entry	Explanation
/EJECT	The specifications following this entry are to begin on a new page of the compiler listing.

The /EJECT specification is not printed on the compiler listing.

Columns 7-12 (/TITLE)

Entry	Explanation
/TITLE	The heading information (such as a title or security classification) that follows the /TITLE entry appears at the top of each page of the compiler listing. The heading information is entered in columns 14 through 74.

A program can contain more than one /TITLE statement. Each /TITLE statement provides heading information for the compiler listing until the next /TITLE statement is read. To print on the first page of the compiler listing, a /TITLE statement must be the first statement read. Information specified by the /TITLE statement is printed in addition to compiler heading information.

The /TITLE statement causes an eject to the next page before the title is printed. The /TITLE statement is not printed on the compiler listing.

Columns 7-14 (/SPACE)

Entry	Explanation
/SPACEƀn	Line spacing occurs at this point in the compiler listing. Valid entries for n are 1 to 3. If you do not specify n, 1 is assumed.

One blank (ƀ) must come before the value you specify for n. The value you specify for n indicates the number of blank lines to be spaced before the next specification line is printed. If n is greater than the number of lines remaining on the current page, the next specification line is printed on a new page. If you specify just /SPACE, one line is spaced.

/SPACE is not printed on the compiler listing but is replaced by the actual line spacing. The spacing indicated by /SPACE is in addition to the three blank lines that occur between specification types.

Column 10 (Object Output)

Entry	Explanation
Blank	The system does not halt if warning errors are found.
D	The system halts if warning errors are found. The person using the display station can continue the job after a halt occurs for a warning error.

Use column 10 to indicate whether the system is to halt for warning errors. If you want the system to halt if severe (terminal) errors are found, you must specify HALT on the RPGC or AUTOC procedure.

Column 11 (Listing Options)

Entry	Explanation
Blank	The program listing is printed.
B	No program listing is printed. A prolog is printed, along with information from the overlay linkage editor. Use this entry to produce a program for which you already have a listing.
P	A partial program listing is printed.

Use column 11 to specify the program-listing option to be used when your program is compiled. If any severe errors are found during compilation and if column 11 contains a blank or P, the listing is completed and the system halts.

The listing consists of a prolog, the source program listing, array and table information, indicator-usage information, the relative location of fields and their attributes, unreferenced field names, diagnostics, and a main-storage usage map. The main-storage usage map lists the identification, the start address, and the size of each segment of code in the program; defines the amount of main storage required for execution; and lists the number of library sectors required for the program.

The partial listing includes a prolog, the source program, indicator-usage information, diagnostics, and a main-storage usage map that lists the amount of main storage required for execution and the number of library sectors required for the program. The partial listing does not include array and table information or field information.

If you enter B in column 11, a prolog is printed along with the following information from the overlay linkage editor: the amount of main storage required to run the program, the starting address of the program, and the number of library sectors required for the program.

Columns 12-14 (Size to Execute)

Column 12

Entry	Explanation
Blank or 0	The entry in columns 13 and 14 determines the size required to run the program.
Q, H, or T	The entry in columns 13 and 14 is rounded up to the next even number.

Columns 13 and 14

Entry	Explanation
Blank	The main storage available for the running program defaults to the region size specified. If no region size is specified, the default is the region size in which the compiler is running.
02-64	Enter the main storage available for running the program in a multiple of 2K bytes (K = 1024 bytes).

Use columns 12 through 14 to specify the amount of main storage to be available to run the program. The maximum amount of storage you can specify depends on the system size. If column 12 contains a Q, H, or T, or if columns 13 and 14 contain an odd number, RPG rounds the entry in columns 13 and 14 up to the next even number. For example, an entry of Q04 or 005 is rounded up to 006.

The compiled program can occupy up to the amount of main storage specified in columns 13 and 14. The actual amount of storage the program occupies after it is compiled appears on the overlay linkage editor listing.

Column 15 (Debug)

Entry	Explanation
[illegible]	[illegible]
[illegible]	[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

[illegible]

0 (zero)
[illegible]
[illegible]
[illegible]
. (decimal point)
- (minus sign)
C (letter C)
R (letter R)

Columns 19-20 (Date Option)

Column 19 (Date Format)

Entry	Explanation
Blank	If column 21 is blank, the default is month/day/year. If column 21 contains a D, I, or J, the default is day/month/year.
M	Month/day/year.
D	Day/month/year.
Y	Year/month/day.

Use column 19 to specify the date format for UDATE. The date format specified in column 19 should be the same format as the program date. For example, if columns 19 and 21 are blank, the program date is mm/dd/yy. If column 19 is blank and column 21 contains a D, the program date is dd/mm/yy. If you specify the date in mm/dd/yy format and the program date in the system is in dd/mm/yy format, you will work with the wrong date.

If data containing the UDATE field is sent to, or used by, another system, the UDATE format must be yy/mm/dd.

For a description of the program date, see the *System Reference* manual.

Column 20 (Date Edit)

Entry	Explanation
Blank	A slash (/) is assumed when column 21 contains a blank or D and column 19 contains a blank or M. A period (.) is assumed when column 21 contains I or J and column 19 contains a blank, D, or Y.
&	A blank separates the date field.
Any other character	The character entered separates the edited date field.

Use column 20 to specify the type of edited output that appears for the Y edit code, which is specified on the output specifications. For an example of how the entries in columns 19 through 21 affect the editing of date fields, see *Column 38 (Edit Codes)* in Chapter 27, *Output Specifications*.

Column 21 (Inverted Print)

Entry	Explanation
Blank	Decimal periods are used for numeric literals and editing. UDATE format is mmddyy if column 19 is blank. If columns 19 and 20 are blank, a slash (/) is used for the Y edit code.
I	Decimal commas are used for numeric literals and editing. UDATE format is ddmmyy if column 19 is blank. If columns 19 and 20 are blank, a period (.) is used for the Y edit code.
J	J is the same as I except zero is written to the left of the decimal comma when the field contains a fraction. Nondecimal edited fields print with a zero in the low order (units) position.
D	D is the same as blank except the UDATE format is ddmmyy if column 19 is blank.

Use column 21 to specify the constants to be used with RPG edit codes that are entered on the output specifications. Decimal period means that numbers are edited with a period before the fraction (183.55) and with a comma denoting thousands (1,435). Decimal comma means that numbers are edited with a comma before the fraction (183,55) and with a period denoting thousands (1.435).

For information on how the entries in column 21 are used to format numeric data, see *Column 38 (Edit Codes)* in Chapter 21, *Output Specifications*.

Columns 22-25

Columns 22 through 25 are not used. Leave them blank.

Column 26 (Alternate Collating Sequence)

Entry	Explanation
Blank	Normal collating sequence is used.
S	Alternate collating sequence is used.

Use column 26 only to alter the normal collating sequence for alphameric compare operations, sequence checking, or match fields. For more information, see Chapter 17, *Changing the Hexadecimal Value of Characters*.

Columns 27-36

Columns 27 through 36 are not used. Leave them blank.

Column 37 (Inquiry)

Entry	**Explanation**
Blank or I	The program, when interrupted, will not allow the person using the display station to enter new procedures or commands (does not allow option 1 for an inquiry request).
B	The program, when interrupted, will allow the person using the display station to enter new procedures or commands (does allow option 1 for an inquiry request).

Use column 37 to specify whether a running program can be interrupted to allow another program to run. The person using the display station requests an interruption (called an *inquiry request*) by pressing the Attn key. The procedure or command statements for the interrupting program must be entered from the display station after selecting option 1 (Request COMMAND display).

The program to be loaded following an inquiry request (the interrupting program) can have an I, B, or blank in column 37 of its control specification. However, even if it has a B in column 37, the interrupting program cannot be interrupted to allow another program to run.

If column 37 contains a B, the inquiry function of System/36 allows the person using the display station to interrupt a program that is currently using the display station and to enter new procedures or commands. If column 37 contains any of the valid entries, the person can set the inquiry latch for the inline inquiry subroutine (SUBR95), cancel a single requester terminal (SRT) program that the operator initiated, or release the display station from a multiple requester terminal (MRT) program.

For more information on inquiry, including restrictions on the use of system utilities in inquiry mode, see the *System Reference* manual.

File Sharing

An inquiry program can get active input, update, and add files. However, an inquiry program cannot get indexed sequential add file types or output files. The DISP-SHR parameter must be specified on the control language FILE statement for each file to be shared in both the interrupted and the inquiry programs.

For a description of the valid file-sharing combinations, see the *System Reference* manual.

Inline Inquiry Subroutine (SUBR95)

The IBM-written subroutine SUBR95 can be used to perform an inquiry if the RPG program is not an MRT program (that is, if the MRTMAX parameter on the control language COMPILE statement was 0 when the program was compiled). Column 37 can be blank or contain an I or B. See the *System Reference* manual for restrictions on the inquiry function.

The linkage to SUBR95 must be specified on the calculation specifications at every point in the program where a check is to be made for an inquiry request. The EXIT SUBR95 operation must be followed by **only one** RLABL (see Figure 20-2). The indicator specified in columns 45 and 46 of the RLABL operation must be an RPG indicator. For a detailed discussion of this linkage, see *Linking to External Subroutines* in Chapter 28.

When SUBR95 is called, it checks to determine whether the inquiry latch was set. If it was (that is, the operator selected option 4, *Set inquiry condition for program*, in response to the inquiry display), the indicator specified in the RLABL operation is turned on, and the inquiry request is reset. This indicator can then be used to condition further calculation and output operations.

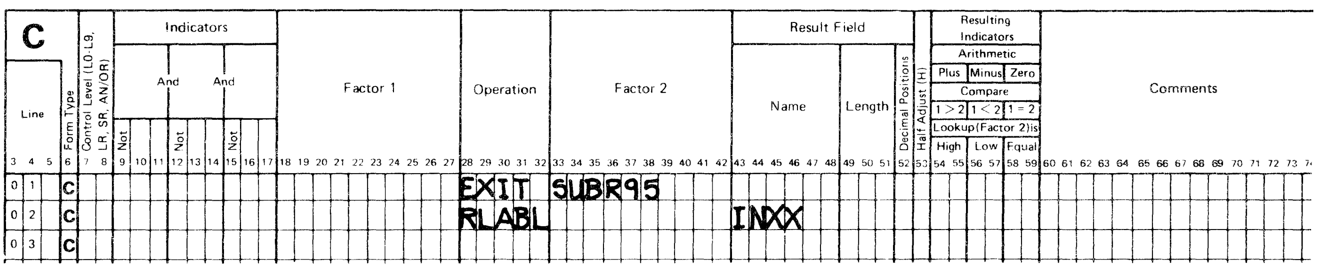


Figure 20-2. Linkage for SUBR95

Columns 38-40

Columns 38 through 40 are not used. Leave them blank.

Column 41 (1P Forms Position)

Entry	**Explanation**
Blank	First line is printed only once.
1	First line can be printed repeatedly.

Use column 41 only when the first output line is written to a PRINTER file. If the program contains more than one PRINTER file, the first-page indicator (1P) entry in column 41 applies to each PRINTER file that has first-page (1P) output.

When forms are first put in the printer, they may not be in the right position. Sometimes several lines must be printed to determine the correct position of the form. If 1P forms position is specified, the system prints the first line of output and issues a message. The person using the display station can then line up the forms and select the option to try printing the line again or to continue printing. The 1P forms specification is also valid if the output is spooled. The page counter is not increased until the forms have been positioned correctly.

The 1P forms position specification can be overridden on the control language PRINTER statement, or forms alignment can be specified on the PRINTER statement.

Column 42

Column 42 is not used. Leave it blank.

Column 43 (File Translation)

Entry	Explanation
Blank	No file translation is needed.
F	Input, output, update, or combined files are to be translated.

Use column 43 only when information contained in an input, output, update, or combined file is in a character code different from the character code used by System/36.

For more information, see *Translating a File* in Chapter 17, *Changing the Hexadecimal Value of Characters.*

Column 44

Column 44 is not used. Leave it blank.

Column 45 (Nonprint Characters)

Entry	Explanation
Blank	The program halts if the last line printed contained an unprintable character.
1	The program does not halt for unprintable characters.

Use column 45 to bypass halts for unprintable characters. This column applies only to PRINTER files.

All characters are represented in the system by a hexadecimal value, which is a numeric code. If a hexadecimal value is formed during a calculation that is not in the printer character set and that character is to be printed, the program halts after printing the line. In the printed line, the unprintable characters are replaced with blanks.

To bypass this halt, enter a 1 in column 45. An unprintable character is then replaced with a blank, and no halt occurs. Note, however, that your output is not correct, and, by bypassing the halt, the incorrect output may not become known (for example, when a packed key field is printed or when a nonprintable field is built by calculation specifications).

Columns 46-56

Columns 46 through 56 are not used. Leave them blank.

Column 57 (Transparent Literal)

Entry	Explanation
Blank	No transparent literals or constants are present in the program.
1	Transparent literals or constants can be present in the program.

The transparent literal option must be specified if your program uses ideographic data. A transparent literal or constant is one that begins with an apostrophe followed immediately by the shift-out (S/O) control character (hex 0E), and ends with the shift-in (S/I) control character (hex 0F) followed immediately by an apostrophe.

If the transparent literal option is specified and a literal or constant is found that begins with an apostrophe immediately followed by the S/O control character, the RPG compiler checks for a valid transparent literal or constant. The following conditions cause a literal or constant to be diagnosed as an invalid transparent literal or constant:

A second S/O control character is found before the S/I control character.

An odd number of 1-byte characters are found between the S/O and S/I control characters.

The S/I control character is not immediately followed by the ending apostrophe.

If a literal or constant is found to be an invalid transparent literal or constant, it is rechecked as an alphameric literal or constant.

Transparent literals and constants are not checked for embedded apostrophes.

For more information about ideographic data, see Chapter 31.

Columns 58-74

Columns 58 through 74 are not used. Leave them blank.

Columns 75-80 (Program Identification)

Entry	**Explanation**
Blank	Program identification defaults to RPGOBJ.
Any valid program name	The first character of the program identification must be alphabetic and cannot be #, $, or @. The remaining characters must be alphameric; however, no special characters can be used, and blanks must not appear between characters.

Use columns 75 through 80 to assign a name to your program. The compiler uses the program name to catalog the program in the library directory.

If the program contains a CONSOLE device, the compiler also uses this program identification to name the display format load member for the program. The display format load member is created by RPG only for CONSOLE files; however, the name is created for both CONSOLE and WORKSTN files. This name is used by RPG; therefore, you *must* create your own load member with this or an alternative name for WORKSTN files. For the display format load member name, the compiler uses the name specified as the value of the FMTS continuation-line option. If the FMTS continuation-line option is not specified, the compiler uses the characters specified in columns 75 through 80 of the control specification (the program identification) and adds the characters FM to the end of the program identification. FM is added to the end of the program identification regardless of its length, and the resulting name contains no blanks.

If a cross-reference listing is to be created for the program, this program identification is also used to identify the listing.

Chapter 21. File Description Specifications

Chapter 21. File Description Specifications

File description specifications describe each file used by a program. One file description specification is required for each file, and a maximum of 20 files can be described for each program.

Write the file description specifications on the Control and File Description Specifications sheet (see Figure 21-1).

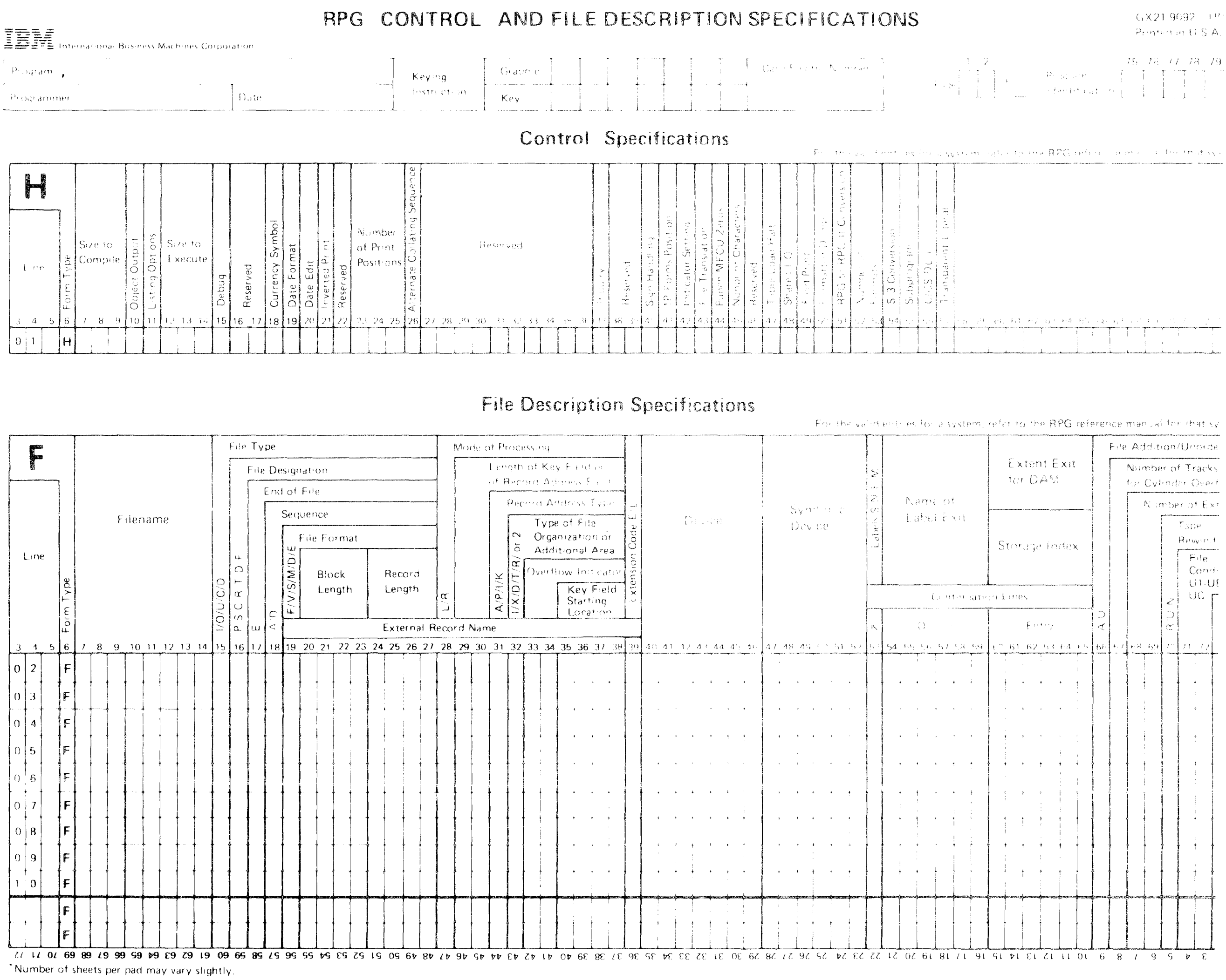


Figure 21-1. RPG Control and File Description Specifications

File Description Charts

Figures 21-2 through 21-13 show the file description specification entries for DISK files (which are presented by file organization and processing method), WORKSTN files, PRINTER files, SPECIAL files, CONSOLE files, KEYBORD files, CRT files, and BSCA files. When you use the charts, keep the following in mind:

- The entries in the chart must be made for the processing method and type of file described on that line.
- The shaded columns must be blank for the file described on that line.
- The unshaded columns with no entries may be required or optional, but cannot be indicated on the chart because the entries represent information that changes from program to program.

How to Use the Charts

As an example, if you are updating an indexed DISK file using the CHAIN operation code, see Figure 21-4, for indexed DISK files, and refer to random processing by CHAIN operation code. Then choose the chained or full-procedural update file with or without record addition.

In this example, the following columns are required but may change from one program to another: filename, record length, length of key field, and key field starting location. Optional entries are line, block length, and file condition.

Type of Processing

Consecutive — The entire file is read from beginning to end.

Random
- by CHAIN
- by CHAIN (delete-capable)
- by CHAIN
- by CHAIN (delete-capable)
- by Addrout
- by Addrout
- by Addrout
- by Addrout
- by Addrout
- by Addrout

Consecutive and/or Random
- by READ, READP and/or CHAIN
- by READ, READP and/or CHAIN (delete-capable)
- by READ, READP and/or CHAIN
- by READ, READP and/or CHAIN (delete-capable)

Load — The file is written on disk as entered.

Add records only

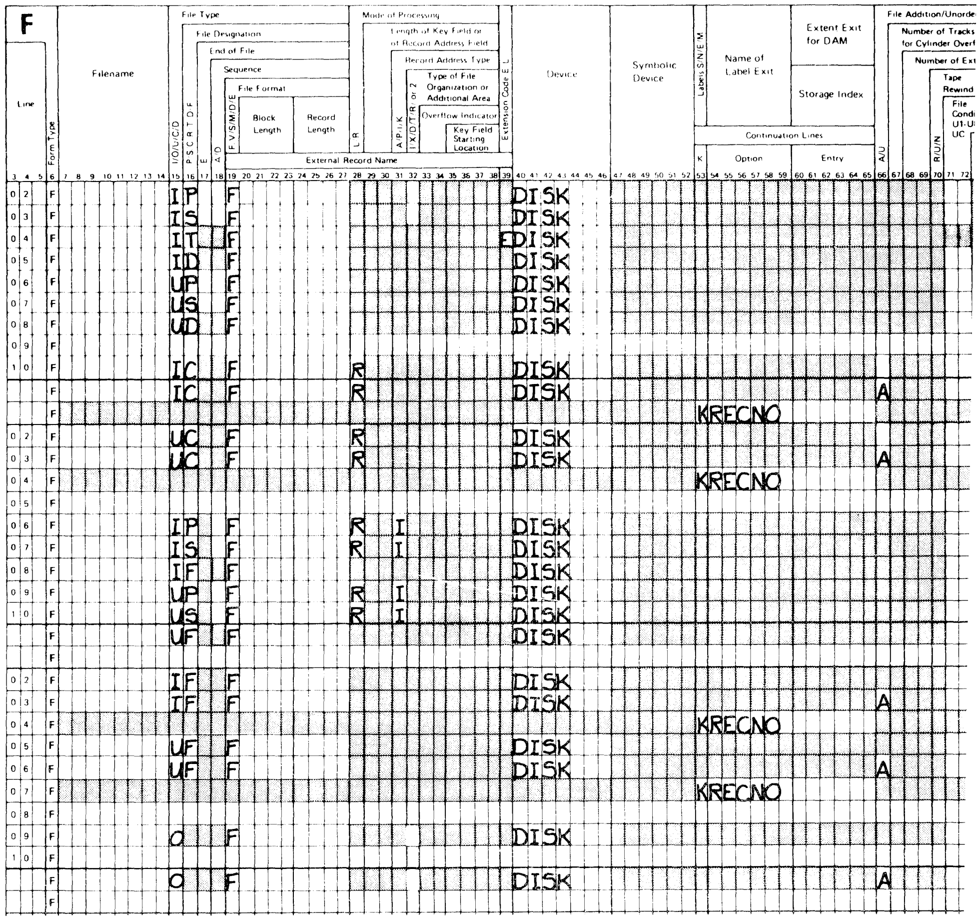


Figure 21-2. Processing Methods for Sequential DISK Files

Type of Processing[1]		Line	Form Type	File Type / Designation (15–16)	File Format (19)	Mode of Processing (28)	Type of File Organization (31)	Device (40–46)	Continuation Option (54–59)	A/U (66)
Consecutive	The entire file is read from beginning to end.	02	F	IP	F			DISK		
		03	F	IS	F			DISK		
		04	F	ID	F			DISK		
		05	F	UP	F			DISK		
		06	F	US	F			DISK		
		07	F	UD	F			DISK		
		08	F							
Random	by CHAIN	09	F	IC	F	R		DISK		
	by CHAIN (delete-capable)	10	F	IC	F	R		DISK		A
			F						KRECNO	
	by CHAIN		F	UC	F	R		DISK		
	by CHAIN (delete-capable)	02	F	UC	F	R		DISK		A
		03	F						KRECNO	
	by Addrout	04	F	IP	F	R	I	DISK		
	by Addrout	05	F	IS	F	R	I	DISK		
	by Addrout	06	F	IF	F			DISK		
	by Addrout	07	F	UP	F	R	I	DISK		
	by Addrout	08	F	US	F	R	I	DISK		
	by Addrout	09	F	UF	F			DISK		
		10	F							
Consecutive and/or Random	by READ, READP and/or CHAIN		F	IF	F			DISK		
	by READ, READP and/or CHAIN (delete-capable)		F	IF	F			DISK		A
		02	F						KRECNO	
	by READ, READP and/or CHAIN	03	F	UF	F			DISK		
	by READ, READP and/or CHAIN (delete-capable)	04	F	UF	F			DISK		A
		05	F						KRECNO	
Load	Disk addresses are developed for each record entered.	06	F							
		07	F	OC	F	R		DISK		
		08	F							
		09	F							
Load	Delete-capable file	10	F	O	F	R		DISK		
			F						KRECNO	
			F							

[1] To insert or change records in a direct file, define the file as an update file processed consecutively or as an update file processed randomly by the CHAIN operation code.

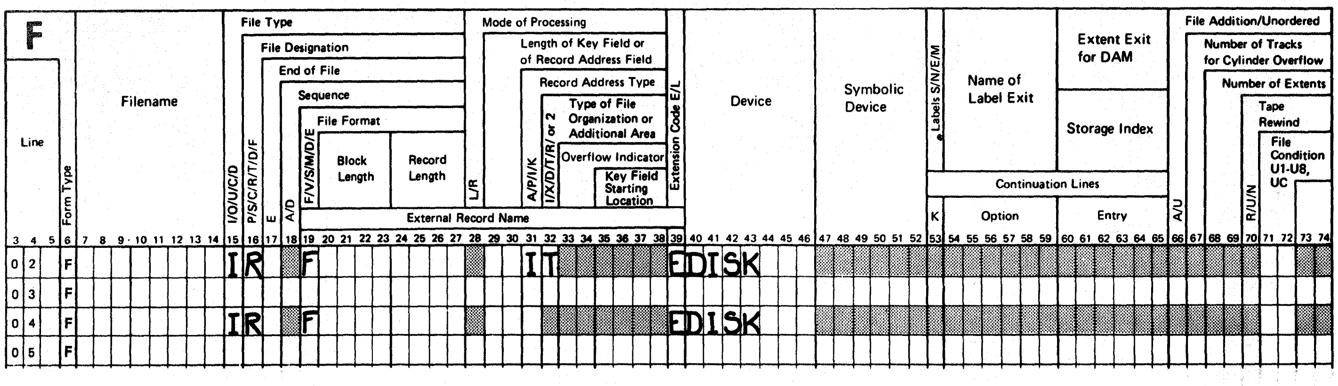
Figure 21-3. Processing Methods for Direct DISK Files

Type of Processing		Line (3-5)	Form Type (6)	File Type / File Designation (15-16)	File Format (19)	L/R (28)	A/P/I/K (31)	I/X/D/T/R or 2 (32)	Device (40-46)	A/U (66)
Sequential[1]	by Key, no Add	02	F	IP	F			I	DISK	
	by Key, no Add	03	F	IS	F			I	DISK	
	by Key, no Add	04	F	ID	F			I	DISK	
	by Key, with Add	05	F	IP	F			I	DISK	A
	by Key, with Add	06	F	IS	F			I	DISK	A
	by Key, with Add	07	F	ID	F			I	DISK	A
	by Key, no Add	08	F	UP	F			I	DISK	
	by Key, no Add	09	F	US	F			I	DISK	
	by Key, no Add	10	F	UD	F			I	DISK	
	by Key, with Add		F	UP	F			I	DISK	A
	by Key, with Add		F	US	F			I	DISK	A
	by Key, with Add	02	F	UD	F			I	DISK	A
	by Limits	03	F	IP	F	L		I	DISK	
	by Limits	04	F	IS	F	L		I	DISK	
	by Limits	05	F	IF	F	L		I	DISK	
	by Limits	06	F	ID	F	L		I	DISK	
	by Limits	07	F	UP	F	L		I	DISK	
	by Limits	08	F	US	F	L		I	DISK	
	by Limits	09	F	UF	F	L		I	DISK	
	by Limits	10	F	UD	F	L		I	DISK	
	by Limits, with Add		F	IF	F	L		I	DISK	A
	by Limits, with Add		F	UF	F	L		I	DISK	A
		02	F							
Random	by CHAIN, no Add[2]	03	F	IC	F	R		I	DISK	
	by CHAIN, with Add[2]	04	F	IC	F	R		I	DISK	A
	by CHAIN, no Add[2]	05	F	UC	F	R		I	DISK	
	by CHAIN, with Add[2]	06	F	UC	F	R		I	DISK	A
		07	F							
	by Addrout	08	F	IP	F	R	I	I	DISK	
	by Addrout	09	F	IS	F	R	I	I	DISK	
	by Addrout	10	F	UP	F	R	I	I	DISK	
	by Addrout		F	US	F	R	I	I	DISK	
			F							
Sequential and/or Random	by Key, no Add	02	F	IF	F			I	DISK	
	by Key, with Add	03	F	IF	F			I	DISK	A
	by Key, no Add	04	F	UF	F			I	DISK	
	by Key, with Add	05	F	UF	F			I	DISK	A
		06	F							
Load	Unordered	07	F	O	F			I	DISK	U
	Ordered	08	F	O	F			I	DISK	
		09	F							
Add records only	Add only	10	F	O	F			I	DISK	A
			F							

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

[2] If chained files are processed by key, column 31 should contain an A; however, if chained files are processed by relative record number, columns 31 and 32 must be blank.

Figure 21-4. Processing Methods for Indexed DISK Files (Using the Index)



[1] Record address files containing relative record numbers can be associated with indexed, sequential, or direct DISK files.

Record address files containing key-field limits can be associated only with indexed DISK files, but can be a DISK or CONSOLE file. (See chart for CONSOLE files.)

Figure 21-5. Record Address Files Located on Disk

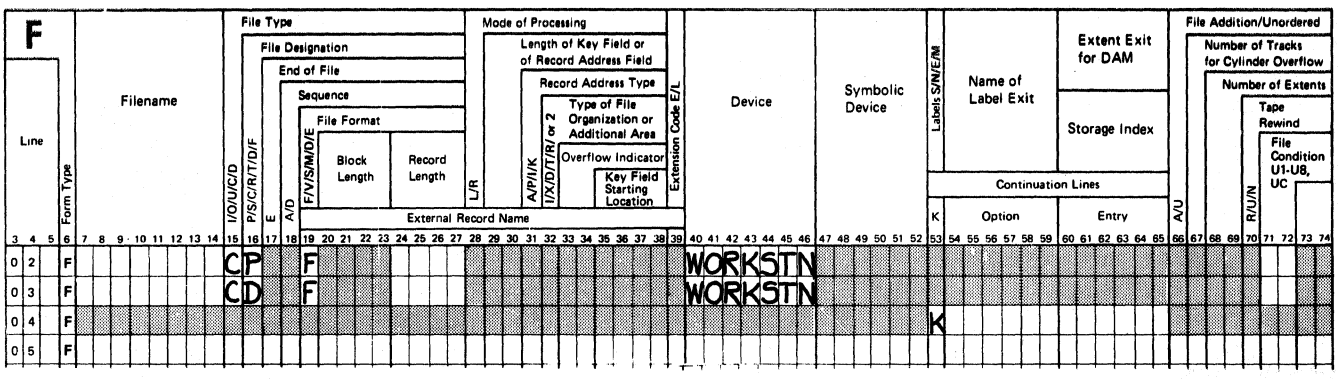


Figure 21-6. WORKSTN Files

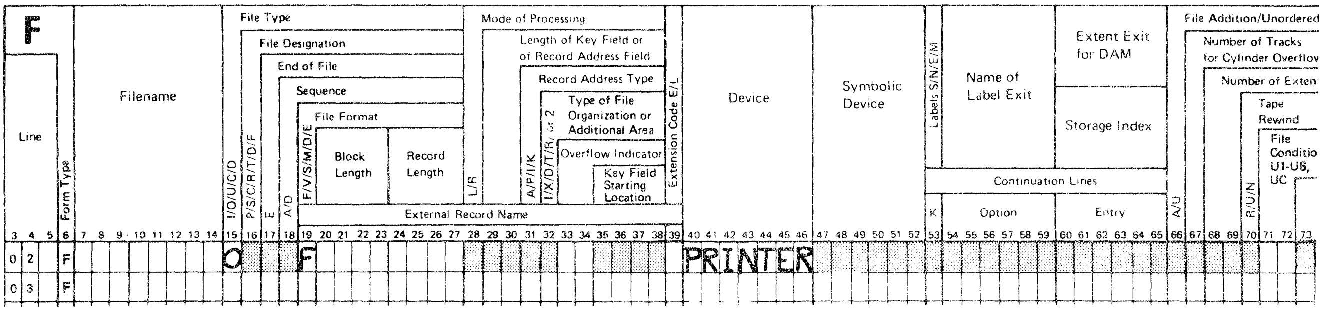


Figure 21-7. PRINTER Files

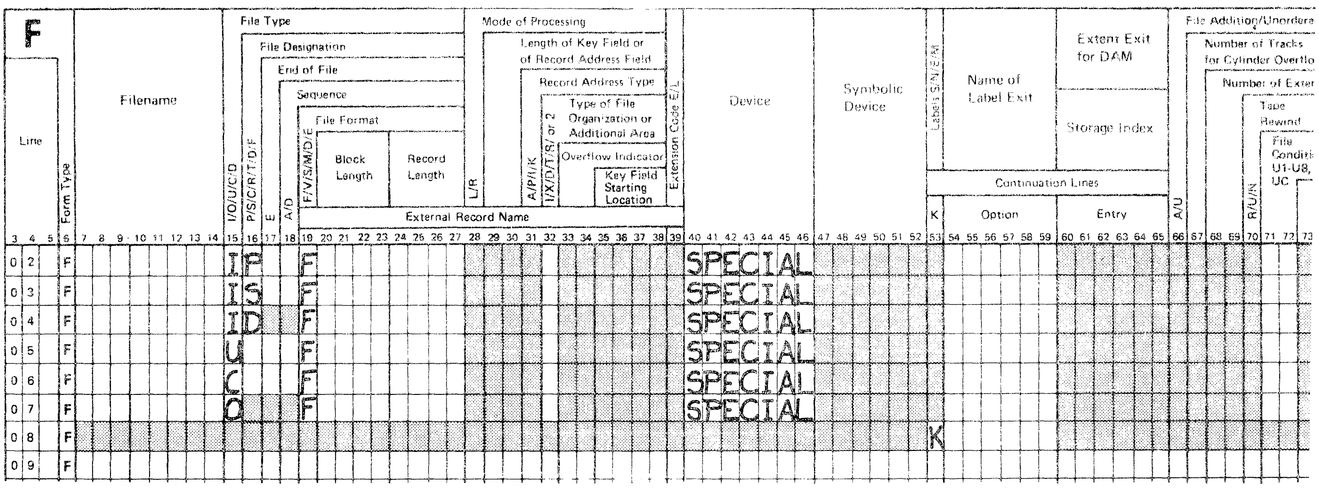
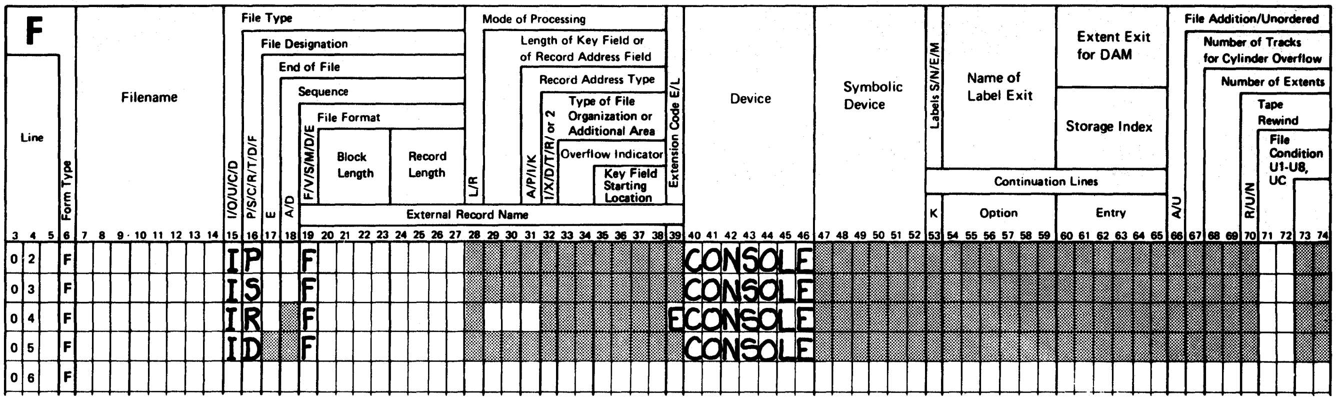


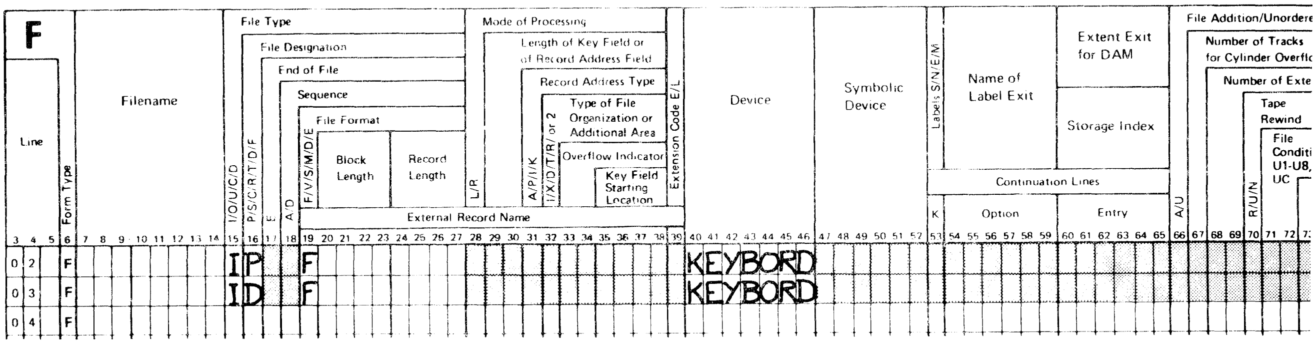
Figure 21-8. SPECIAL Files



Line	Form Type	Filename (7-14)	File Type (15)	File Designation (16)	End of File (17)	Sequence (18)	File Format (19)	Extension Code E/L (39)	Device (40-46)
02	F		I	P			F		CONSOLE
03	F		I	S			F		CONSOLE
04	F		I	R			F	E	CONSOLE
05	F		I	D			F		CONSOLE
06	F								

Input file records are displayed on the display screen when keyed into the program.

Figure 21-9. CONSOLE Files



If a KEYBORD file is specified as a primary input file, no other input files in the program can be specified as primary or secondary files.

Input data entered from the KEYBORD device must be defined in calculation specifications for a KEY operation.

No input specifications can be used for KEYBORD files.

Figure 21-10. KEYBORD Files

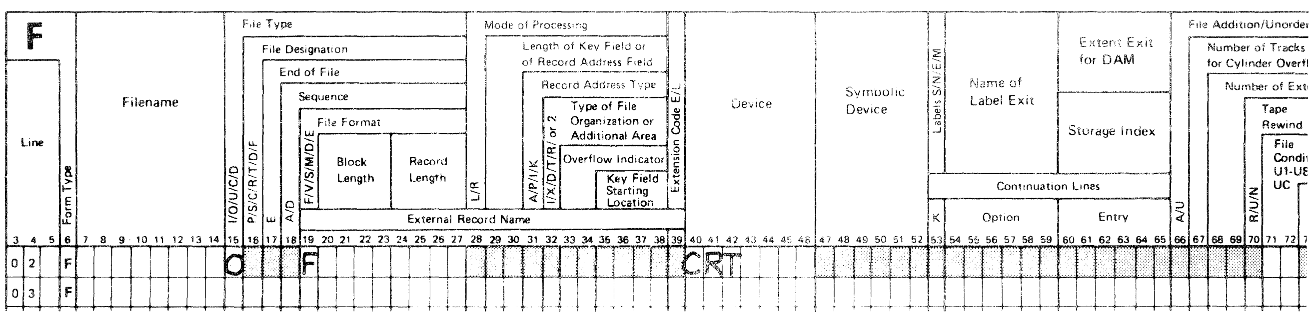


Figure 21-11. CRT Files

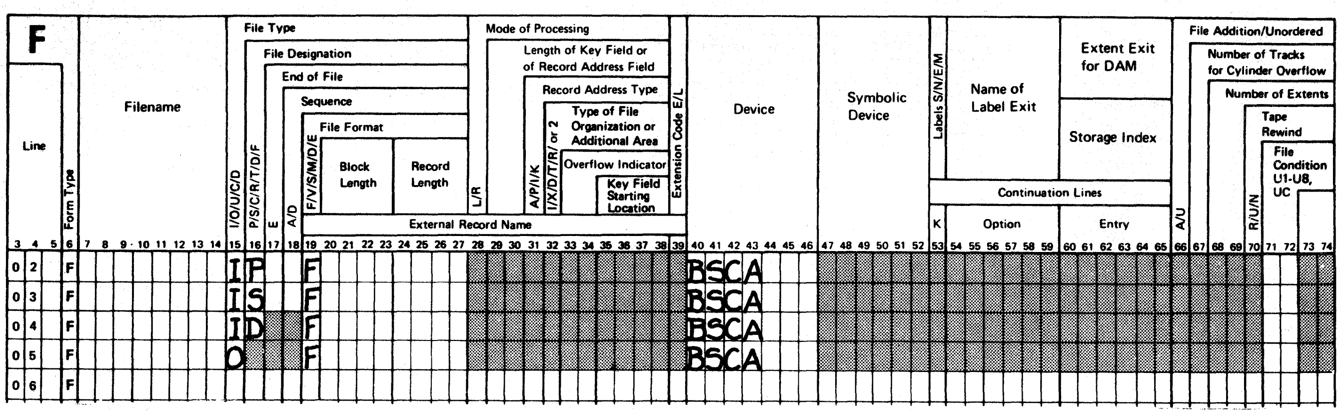


Figure 21-12. BSCA Files

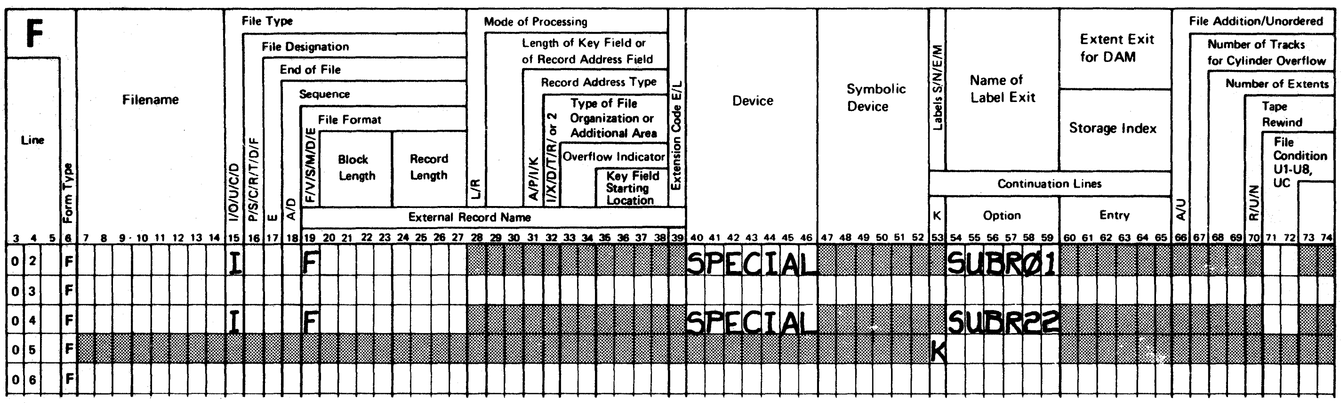


Figure 21-13. File Description Specifications for IBM-Supplied Subroutines

Columns 1-2 (Page)

Entry	Explanation
Blank	No page number is used.
01-99	Page number.

Use columns 1 and 2 in the upper right corner of each sheet to number the specifications sheets, in ascending order, for your job. You can use more than one of each type of sheet, but keep all sheets of the same type together.

Columns 3-5 (Line)

Entry	Explanation
Blank	No line number is used.
Any numbers	Line numbers

Use columns 3 through 5 to number the lines on each page. Columns 3 and 4 are preprinted on each sheet so, in most cases, line numbering is already done.

Page and line numbers are optional entries and are not required to successfully run your program. Columns 1 through 5 are checked for ascending order, and RPG prints an S in the left margin of the RPG listing for any statement that is out of order. If you use SEU to enter the source program, you can request that SEU put the statements in order.

The control specification is always on line 01. Any other lines on the sheets can be skipped. The line numbers used need not be consecutive, but should be in ascending order.

Column 6 (Form Type)

An F must appear in column 6 to identify this line as a file description specification.

Column 7 (Comments)

Entry	Explanation
*	Comment line

Use an asterisk in column 7 to identify the line as a comment line. Use comments throughout your program to document the purpose of a certain section of coding. You can use any character in a comment line. Comments are not instructions to your program; they only document your program.

Columns 7-12 (/EJECT)

Entry	Explanation
/EJECT	The specifications following this entry are to begin on a new page of the compiler listing

The /EJECT specification is not printed on the compiler listing.

Columns 7-12 (/TITLE)

Entry	Explanation
/TITLE	The heading information (such as a title or security classification) that follows the /TITLE entry appears at the top of each page of the compiler listing. The heading information is entered in columns 14 through 74.

A program can contain more than one /TITLE statement. Each /TITLE statement provides heading information for the compiler listing until the next /TITLE statement is read. To print on the first page of the compiler listing, a /TITLE statement must be the first statement read. Information specified by the /TITLE statement is printed in addition to compiler heading information.

The /TITLE statement causes an eject to the next page before the title is printed. The /TITLE statement is not printed on the compiler listing.

Columns 7-14 (/SPACE)

Entry	Explanation
/SPACEƀn	Line spacing occurs at this point in the compiler listing. Valid entries for n are 1 to 3. If you do not specify n, 1 is assumed.

One blank (ƀ) must come before the value you specify for n. The value you specify for n indicates the number of blank lines to be spaced before the next specification line is printed. If n is greater than the number of lines remaining on the current page, the next specification line is printed on a new page. If you specify just /SPACE, one line is spaced.

/SPACE is not printed on the compiler listing but is replaced by the actual line spacing. The spacing indicated by /SPACE is in addition to the three blank lines that occur between specification types.

Columns 7-14 (Filename)

Entry	Explanation
A valid filename	Every file used in a program must have a separate name. The first character must be alphabetic. The remaining characters can be any combination of alphabetic and numeric characters; however, special characters are not allowed. Blanks cannot appear between characters in the filename. The filename can be from 1 to 8 characters long, and must begin in column 7.

Use columns 7 through 14 to assign a unique name to every file used in your program, with the following exceptions:

- Compile-time tables and arrays do not require a filename.
- If multiple tables or arrays are read in at preexecution time from the same device, multiple filenames are required.

For naming tables and arrays, see *Columns 27-32* in Chapter 22, *Extension Specifications*.

Column 15 (File Type)

Entry	Explanation
I	Input file
O	Output file
U	Update file
C	Combined (input and output) file

The file type indicates how a program will use a file (for input or for output). You code the file type in column 15 of the file description specifications.

Figure 21-14 shows the types of files that each device type can use.

Device Type	File Type
DISK	Input Output Update
WORKSTN	Combined
PRINTER	Output
SPECIAL	Input Output Update Combined
CONSOLE	Input
KEYBORD	Input
CRT	Output
BSCA	Input (receive) Output (transmit)

Figure 21-14. Types of Files That Each Device Type Can Use

Input Files

An input file contains records that the program reads.

Output Files

An output file contains records that the program writes.

Update Files

An update file is both an input file and an output file. The program reads a record from an update file, changes the data in some fields in the record, and writes the record back to the same place in the same file from which it was read. When an update file is processed, the output records contain both the changed and the unchanged fields from the input records.

Combined Files

A combined file is also both an input file and an output file. However, when a combined file is processed, the output records contain only the fields described on the output specifications. That is, the output records do not always contain the same fields as the input records.

Column 16 (File Designation)

Entry	Explanation
P	Primary file
S	Secondary file
F	Full-procedural file
C	Chained file
R	Record address file
T	Table file (preexecution-time arrays or tables)
D	Demand file
Blank	Output file (except chained output files)

Use column 16 to further identify the use of input, combined, and update files. Leave the column blank for all output files except chained output files.

Figure 21-15 shows which file designations each device type and file type can use.

Device Type	File Type	File Designation
DISK	Input	Primary Secondary Chained Record address Table or array Demand Full-procedural
	Output	Chained or blank
	Update	Primary Secondary Chained Demand Full-procedural
WORKSTN	Combined	Primary Demand
PRINTER	Output	
SPECIAL	Input	Primary Secondary Demand
	Output	
	Update	Primary Secondary Demand
	Combined	Primary Secondary Demand
CONSOLE	Input	Primary Secondary Demand Record address
KEYBORD	Input	Primary Demand
CRT	Output	
BSCA	Input	Primary Secondary Demand
	Output	

Figure 21-15. File Designations That Each Device Type and File Type Can Use.

Primary Files

The primary file is the main file from which the program reads input records. A program can have no more than one primary file, but it does not have to have any.

The primary file can be an input, update, or combined file. It can use any device except CRT or PRINTER.

Secondary Files

Secondary files are used only in programs that use multifile processing. For more information on record selection for multifile processing, see Chapter 11, *Using Primary and Secondary Files*.

A secondary file can be an input, update, or combined file. It can use the DISK, CONSOLE, SPECIAL, or BSCA device. Secondary files are processed in the order in which they are coded on the file description specifications. If no primary file is specified and one or more secondary files are specified, the first secondary file is assigned as the primary file. If WORKSTN is specified as the device for a primary file, no files in the program can be specified as secondary files.

Full-Procedural Files

A full-procedural file is a combination of a chained file and a demand file. A full-procedural file does not use the normal RPG cycle for input. A program can read records from a full-procedural file only when a CHAIN, READ, READE or READP operation occurs in calculations. The number of full-procedural, chained, and demand files used in a program cannot total more than 15.

A full-procedural file can be an input or an update file, and it must use the DISK device.

Chained Files

A chained file does not use the normal RPG program cycle for input. Instead, input occurs only when the program uses the CHAIN operation in calculations. The CHAIN operation reads input records randomly (that is, in no particular order) or loads a direct file that does not allow deletions. The number of chained, demand, and full-procedural files used by a program can total no more than 15. A chained file can be an input, output, or update file. It can use only the DISK device.

Record Address Files

A record address file contains either key-field limits or relative record numbers of records in a DISK file. (Key-field limits and relative record numbers are explained later in this chapter.) By providing these key-field limits or relative record numbers to the program, a record address file tells the program which records to read from the DISK file and in what order to read them. A program can use no more than one record address file. Record address files must be further defined on extension specifications.

A record address file must be an input file. If it contains key-field limits, it can use the DISK or CONSOLE device and can be used only with indexed files. If it contains relative record numbers, it can use only the DISK device and can be used with sequential, direct, or indexed files. Record address files that contain relative record numbers are called *addrout* (address output) files, and they are produced by a sort program.

Array or Table Files

An array or table file is an input file that contains preexecution-time array or table entries. Array or table files must be sequential files and must use the DISK device. When array or table files are read while the program is running, the program reads all the entries from the array or table before it begins to process records.

For more information about loading preexecution-time arrays or tables, see Chapter 13, *Using Arrays and Tables.*

Demand Files

A demand file can be an input, update, or combined file. It can use any device except CRT and PRINTER.

A demand file does not use the normal RPG program cycle for input. Instead, the program reads a demand file only when the READ operation occurs in calculations (or when the KEY operation occurs if the records come from a KEYBORD device). The number of demand, chained, and full-procedural files used by a program cannot total more than 15.

Column 17 (End Of File)

Entry	Explanation
Blank	The program can end whether or not all records from the file are processed. However, if column 17 is blank for all files, all records from every file must be processed before the program can end. This column must be blank for WORKSTN or KEYBORD files.
E	All records from the file must be processed before the program can end. This entry is not valid for files processed by record address files.

Use column 17 to indicate whether the program can end before all records from the file are processed. Column 17 applies only to primary and secondary files.

Column 17 can be used only for input, update, or combined files used as primary, secondary, or record address files. The devices associated with column 17 are DISK and CONSOLE. End of file for CONSOLE files occurs when the person using the display station presses command key 12.

A program that does processing with more than one input file could reach the end of one file before reaching the end of the others. Therefore, an entry in column 17 indicates whether the program is to continue reading records from the other files or is to end.

If the records from all files must be processed, column 17 must be blank or contain E's for all files.

Note: An entry cannot be made in column 17 for files assigned to the KEYBORD and WORKSTN devices. To end the program with a primary file assigned to the KEYBORD device, the last-record (LR) indicator must be set on by calculation specifications.

Column 18 (Sequence)

Entry	Explanation
Blank	No sequence checking is to be done. This column must be blank for a WORKSTN file.
A	Sequence checking is to be done. Records in the file are in ascending order.
D	Sequence checking is to be done. Records in the file are in descending order.

Use column 18 to indicate whether the program is to check the sequence of records. Column 18 applies to input, update, or combined files used as primary or secondary files. Sequence checking can be done for DISK files (except those processed randomly) and CONSOLE files. Use columns 61 and 62 of the input specifications to identify the record fields containing the sequence information.

Sequence checking is required when match fields are used in the records from the file. When a record from a matching input file is found to be out of sequence, error message RPG-9032, FILE CONTAINS A RECORD NOT IN SEQUENCE, is displayed. The operator has three options:

- Bypass the record out of sequence and read the next record from the same file.
- Bypass the record out of sequence, turn on the last-record (LR) indicator, and perform all end-of-job and final-total procedures.
- Cancel the entire program.

If column 18 contains an entry and matching records are specified, the entry in column 18 must be the same for all files. If column 18 is left blank and matching records are specified, then ascending order is assumed for a primary file, and the sequence of the primary file is assumed for all secondary files.

Column 19 (File Format)

Entry	Explanation
F or blank	Fixed-length records

An F in column 19 indicates that all records in the file are of the same length. If this column is blank, F is assumed.

Columns 20-23 (Block Length)

Entry	Explanation
Blank	The block length for this file equals the record length. These columns must be blank for a WORKSTN file and can be blank for any other file.
1-9999	Block length for a DISK file equals the record length or is a multiple of the record length.
1-198	Length of largest output record for a PRINTER file. (Entries from 133 through 198 should only be used for printers with 198 print positions.)
1-9999	Block length for a SPECIAL file equals the record length or is greater than the record length.
2-1518	Block length for a CONSOLE file, if entered, must equal the record length.
1-79	Length of largest field keyed for a KEYBORD file.
1-79	Length of largest output record for a CRT file.
1-4075	Block length for a BSCA file equals the record length or is a multiple of the record length.

Use columns 20 through 23 to specify the block length for the file. The entry made in columns 20 through 23 depends on the device named for the file. The block length entry must end in column 23, and leading zeros can be omitted (see Figure 21-16).

The function of the block length entry is to specify the amount of main storage to use for the input/output area. The maximum block length is 9999. The block length entered for DISK files must equal the record length or be a multiple of the record length. If the record length is entered but the block length is not specified, RPG assumes the block length equals the record length.

Columns 24-27 (Record Length)

Entry	Explanation
1-4096	Record length for DISK or SPECIAL files.
1-9999	Length of largest input or output record for a WORKSTN file.
1-198	Length of largest output record for PRINTER files. (Entries from 133 through 198 should only be used for printers with 198 print positions.)
2-1518	Record length for CONSOLE files.
1-79	Length of largest field keyed for KEYBORD files.
1-79	Length of largest output record for CRT files.
2-58	Twice the record address field length for a record address file assigned to the CONSOLE device.
1-4075	Record length for BSCA files.

Use columns 24 through 27 to indicate the length of the records in a file. An entry must be made for each file, and the entry depends on the device named for the file. Entries in these columns must end in column 27, and leading zeros can be omitted (see Figure 21-16).

All records in one file must be the same length. (For update files, the length of the record after the record is updated must be the same as it was before the record was updated.) The maximum length allowed depends upon the device assigned to the file (see Figure 21-16). The record length specified can be shorter than the maximum length allowed for the device but not longer.

The record length for KEYBORD files should be the length of the largest field to be typed in (that is, the record length equals the largest field length specified in columns 49 through 51 of the calculation specifications when the KEY operation code is used). If the KEY operation is used to display a message, you must also consider the length of the message when you specify the record length for the KEYBORD file. The maximum alphameric field length is 79 characters, and the maximum numeric field length is 15 characters. If the record length specified for a KEYBORD file is 40 or less, a display of six lines with 40 characters per line is centered both vertically and horizontally on the display screen. If the record length is greater than 40, the display consists of 24 lines with 79 characters per line.

Columns 40 through 46 (Device)	Columns 20 through 23 (Block Length)[1]	Columns 24 through 27 (Record Length)	Maximum Record Length
DISK	Record length or a multiple of record length	Record length	4096
WORKSTN	Must be blank	Length of longest input or output record	9999
PRINTER	Record length	Record length	198
SPECIAL	Record length or greater than the record length	Record length	4096
CONSOLE	Record length	Record length	1518
	Record address file record length	Record length	58
KEYBORD	Length of largest field to be keyed	Length of largest field to be keyed	79-alphameric 15-numeric
CRT	Length of longest output record	Length of longest output record	79
BSCA	Record length or a multiple of record length	Record length	4075
[1]Block length must be blank for a WORKSTN file and can be blank for any other file.			

Figure 21-16. Block Length and Record Length Entries

Column 28 (Mode Of Processing)

Entry	Explanation
Blank	Consecutive Sequential by key field Random by relative record number[1] Random by key field[1] Random by address output file[1]
L	Sequential within key-field limits
R	Random by key field Random by address output file Direct file load (random load)

Use column 28 to indicate the method by which records are to be read from the file, or to indicate that a direct file load (random load) is to take place.

For DISK files specified as primary, secondary, demand, chained, or full-procedural, the possible processing methods depend upon the organizations of the files (see Figure 21-17). For the other types of files, consecutive processing is the only possible method.

Column 31 further identifies the access method for the program. See *Column 31 (Record Address Type)* in this chapter.

File Organization	Possible Processing Methods
	Primary, Secondary, or Demand Files
Sequential	Consecutively Randomly by address output file
Direct	Consecutively Randomly by address output file (except demand files)
Indexed	Sequentially by key field Sequentially within key-field limits Randomly by address output file Consecutively (not using the index)
	Chained Files
Sequential	Randomly by relative record number
Direct	Randomly by relative record number
Indexed	Randomly by key field Randomly by relative record number (not using the index)
	Full-Procedural Files
Sequential	Consecutively Randomly by address output file Randomly by relative record number
Direct	Consecutively Randomly by address output file Randomly by relative record number
Indexed	Sequentially by key field Sequentially within key-field limits Randomly by address output file Randomly by key field Consecutively (not using the index) Randomly by relative record number (not using the index)

Figure 21-17. Possible Processing Methods for DISK Files

Columns 29-30 (Length Of Key Field Or Record Address Field)

Entry	Explanation
Blank	Records are read consecutively
1-99	Length of key field or relative record number

Use columns 29 and 30 to indicate:

- The length in bytes of the key fields in indexed files and record address files
- The total length in bytes of the noncontiguous key fields if a noncontiguous key is being used
- The length in characters of the relative record numbers in address output files, which is always 3

Columns 29 and 30 apply only to indexed files and record address files.

All of the key fields in the records in an indexed file must be the same length.

The maximum length of a key field is 99 positions, unless an indexed file is being processed sequentially within key-field limits using a CONSOLE device, in which case the maximum length is 29 positions. Key fields in packed decimal format can be up to 8 positions in length. All of the relative record numbers contained in an address output file are 3 characters long.

Column 31 (Record Address Type)

Entry	Explanation
Blank	Relative record numbers are used in processing sequential, direct and indexed files. A sequential or direct file is being loaded. Records are read consecutively. Key fields in the record address file are in the same format as key fields in the indexed files. Relative record numbers from the address output file are used to process this file (for full-procedural files only).
A	Key fields in zoned-decimal format are used in processing or loading indexed files and processing record address files.
I	Relative record numbers from the address output file are used to process the file, or the file is an address output file consisting of relative record numbers.
P	Key fields in packed-decimal format are used in processing or loading indexed files and processing record address files.

Use column 31 to indicate how records in a DISK file are identified. Column 31 applies to DISK files specified as input, update, or chained output files. Together, columns 28 and 31 indicate:

- The method by which records are read from the file
- A direct file load

Following are the specifications for retrieving records:

Processing Method	**Entry in Column 28**	**Entry in Column 31**[1]
	Primary, Secondary, or Demand Files	
Consecutive	Blank	Blank
By address output[2](except demand files)	R	I
Sequential by key field	Blank	A or P
Sequential within key-field limits	L	A or P
	Chained Files	
Random by relative record number	R	Blank
Random by key field	R	A or P
Direct file load (random load)	R	Blank[3]
	Full-Procedural Files	
Consecutive	Blank	Blank
By address output[2]	Blank	Blank
Sequential by key field	Blank	A or P
Sequential within key-field limits	L	A or P
Random by relative record number	Blank	Blank
Random by key field	Blank	A or P

[1]When creating a file with key fields in packed-decimal format (P in column 31), you must specify the key field as packed in your output specifications.

[2]For address output files, column 31 must contain an I, indicating that binary relative record numbers are used in processing. For full-procedural files processed by address output files, column 31 must be blank.

[3]For files that do not allow deletions, a direct file load requires an O in column 15 and a C in column 16. For files that allow deletions, a direct file load requires an O in column 15 and a blank in column 16.

Column 32 (File Organization Or Additional Input/Output Area)

Entry	Explanation
Blank	Sequential file, direct file, or indexed file processed consecutively or randomly, by relative record number. The program uses one input/output area for the file.
I	Indexed file processed sequentially or randomly, by key.
T	Address output file.
1-9	Sequential file. The program uses two input/output areas for the file.

Use column 32 to (1) identify the organization of all DISK files except address output files, (2) identify address output files, and (3) indicate whether one or two input/output areas are to be used for sequential files.

Columns 33-34 (Overflow Indicator)

Entry	Explanation
Blank	No overflow indicator is used.
OA-OG, OV	The overflow indicator you specify conditions records will be printed when overflow occurs.

Use columns 33 and 34 to specify an overflow indicator to condition the lines in each PRINTER file that will be printed when overflow occurs.

Only one overflow indicator can be assigned to a file. If more than one PRINTER file in a program is assigned an overflow indicator, the indicator must be different for each file.

For more information on overflow processing, see Chapter 7, *Using a PRINTER File.*

Columns 35-38 (Key Field Starting Location)

Entry	Explanation
Blank	Records are processed consecutively.
EXTK	The file uses noncontiguous fields as its key.
1-4096	Record position in which the key field begins if the key has only one field.

Use columns 35 through 38 to identify the beginning record position of the key field for an indexed file that only uses one field as its key, or to indicate that an index file uses noncontiguous fields as its key. Columns 35 through 38 apply only to indexed DISK files, and an entry must be made in these columns for an indexed DISK file. The key field of a record contains the information that identifies the record. This information is used in the index portion of the file. The key field must be in the same location in all of the records in the file. The entry in these columns must end in column 38. Leading zeros can be omitted.

Key fields can contain hex FF characters. However, if the first character or byte of a record is hex FF, the record is treated as a deleted record.

Column 39 (Extension Code)

Entry	Explanation
Blank	No extension or line counter specifications are used.
E	Extension specifications further describe the file.
L	Line counter specifications further describe the file.

Use column 39 to indicate whether the file is further described on the extension specifications or line counter specifications. Column 39 applies only to (1) preexecution-time array and table files, (2) record address files, and (3) output files assigned to the printer. Describe PRINTER files on the line counter specifications, and describe array, table, and record address files on the extension specifications.

Columns 40-46 (Device)

Entry	Explanation
DISK	Disk
WORKSTN	Display station
PRINTER	Printer
SPECIAL	Used for a device not supported directly by RPG
CONSOLE	Console data file or console record address file
KEYBORD	Keyboard
CRT	Display screen
BSCA	Binary synchronous communications adapter

Use columns 40 through 46 to identify the input/output device used for the file.

All entries must begin in column 40. The devices and the associated file types that can be used with each device are shown in Figure 21-18.

Device	Form of Data	File Type	Column 15	Column 16
DISK	Disk	Primary input	I	P
	Disk	Secondary input	I	S
	Disk	Record address file containing key-field limits	I	R
	Disk	Record address file containing relative record numbers (address output file)	I	R
	Disk	Full-procedural	I	F
	Disk	Chain input	I	C
	Disk	Demand	I	D
	Disk	Array or table (preexecution time only)	I	T
	Disk	Update (primary, secondary, full-procedural, chained, or demand)	U	P, S, F, C, or D
	Disk	Output	O	
	Disk	Direct file that does not allow deletions	O	C
WORKSTN	Typed in by operator	Demand	C	D
	Typed in by operator	Combined Primary	C	P
PRINTER	Printed lines	Output	O	

Figure 21-18 (Part 1 of 2). Devices and Associated File Types.

Device	Form of Data	File Type	Column 15	Column 16
SPECIAL	Special device	Primary input	I	P
	Special device	Secondary input	I	S
	Special device	Demand	I	D
	Special device	Update (primary, secondary, or demand)	U	P, S, or D
	Special device	Combined (primary, secondary, or demand)	C	P, S, or D
	Special device	Output	O	
CONSOLE	Typed in by operator	Primary input	I	P
	Typed in by operator	Secondary input	I	S
	Typed in by operator	Demand	I	D
	Typed in by operator	Record address files containing key-field limits	I	R
KEYBORD	Typed in by operator	Primary input	I	P
	Typed in by operator	Demand	I	D
CRT	Display lines	Output	O	
BSCA	Data communications line	Primary input	I	P
	Data communications line	Secondary input	I	S
	Data communications line	Demand	I	D
	Data communications line	Output	O	

Figure 21-18 (Part 2 of 2). Devices and Associated File Types.

Figure 21-19 shows the columns that can be used for the devices named.

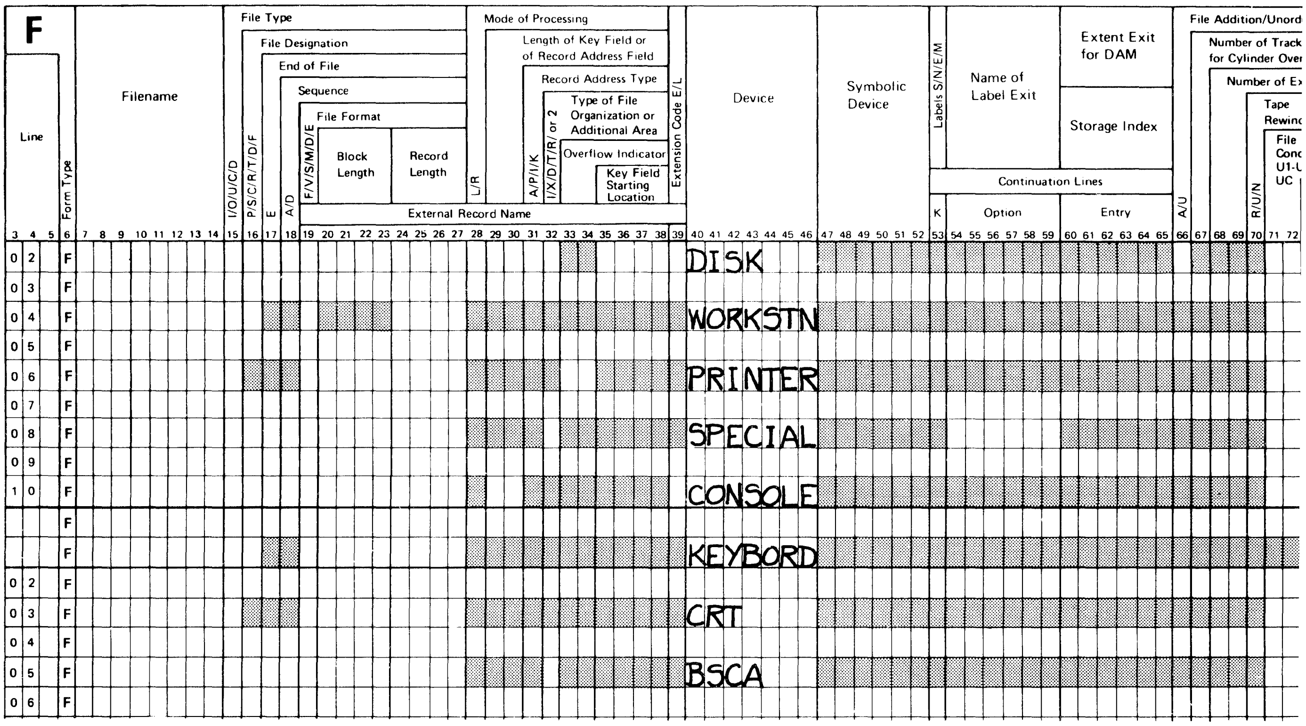


Figure 21-19. Columns That Apply to Device Named

Device Types

The device type tells what part of the computer provides input files to a program or receives output files from the program. The device type is coded in columns 40 through 46 of the file description specifications. The eight device types are: DISK, WORKSTN, PRINTER, SPECIAL, CONSOLE, KEYBORD, CRT, and BSCA.

DISK

The DISK device is an input/output device that allows RPG programs to process data stored on a magnetic disk drive (disk). Data stored on a disk is called a DISK file. A DISK file can be used for input, for output, or for both.

Chapter 5 explains how to code an RPG program that uses a DISK file.

WORKSTN

The WORKSTN device is an input/output device that allows RPG programs to communicate with one or more display stations. WORKSTN stands for work station, which is another name for display station. A display station consists of a display screen on which data is displayed and an attached keyboard from which data is entered. Use the System/36 display format specifications to define input and output for a WORKSTN file.

A program can use no more than one WORKSTN file, and a program that uses a WORKSTN file cannot use CONSOLE, KEYBORD, or CRT files.

Chapter 6 explains how to code a program that uses a WORKSTN file.

PRINTER

A PRINTER file can be used only for output. A program can use up to eight PRINTER files.

Chapter 7 explains how to code an RPG program that uses a PRINTER file.

SPECIAL

A SPECIAL file is handled by a device not directly supported by RPG. To use a SPECIAL file, you must use a subroutine to transfer data between the SPECIAL device and main storage.

Chapter 8 explains how to code an RPG program that uses a SPECIAL file.

CONSOLE

The CONSOLE device is an input device that allows RPG programs to read data records directly from a display station. CONSOLE is another word for display station. A CONSOLE file can be used as a record address file or as an input data file. If used as a record address file, the file must be further defined on extension specifications. A CONSOLE file can be used only as an input file. It cannot be used to display the records in a file.

Chapter 9 explains how to code a program that uses a CONSOLE file.

KEYBORD

The KEYBORD device is an input device that allows RPG programs to receive data via the KEY and SET operation codes. Input specifications are not used for KEYBORD files. Instead, the input data is defined in the KEY and SET operations.

Chapter 9 explains how to code an RPG program that uses a KEYBORD file.

CRT

The CRT device is an output device that allows RPG programs to write data to a display station. CRT stands for cathode ray tube, which means the display screen of a display station. A CRT file can be used only as an output file to display information on the screen. A person using the display station cannot change this displayed information.

Chapter 9 explains how to code an RPG program that uses a CRT file.

BSCA

A BSCA file allows an RPG program to send and receive binary synchronous data on a data communications network.

Chapter 10 explains how to code an RPG program that uses a BSCA file.

Columns 47-52

Columns 47 through 52 are not used. Leave them blank.

Column 53 (Continuation Lines-K)

Entry	Explanation
K	Continuation record

Use column 53 to indicate that a continuation record provides additional information about the DISK file, WORKSTN file, or SPECIAL file being defined. Only one continuation record can be specified for each DISK file or each SPECIAL file; however, several continuation records can be specified for a WORKSTN file. When you specify a continuation record for a SPECIAL device, columns 54 through 59 (continuation-line option) must be coded. When you specify a continuation record for a DISK or WORKSTN device, columns 54 through 65 must be coded. Figure 21-20 shows an example of the coding necessary for a continuation line on the file description specifications for a SPECIAL file.

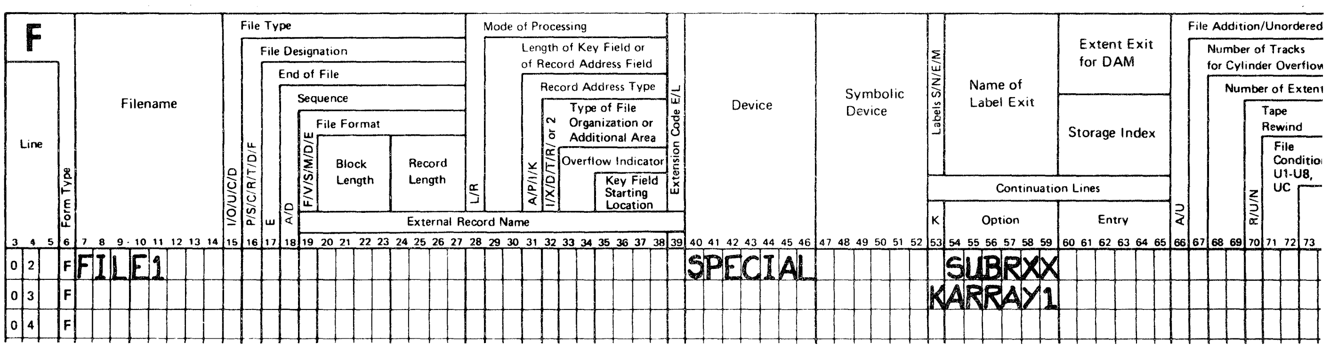


Figure 21-20. Specifications for a SPECIAL Device

Columns 54-59

Name of Label Exit

Entry	Explanation
Blank	No SPECIAL device is used.
SUBRxx	Name of the user-written or IBM-written subroutine that cannot be overlaid and that performs the input/output operation for a SPECIAL device. (For a user-written subroutine, x = any alphabetic character. Numeric characters are reserved for IBM-supplied subroutines.)
SRyzzz	Name of the IBM-written subroutine (5-character name in library is @yzzz) that performs the input/output operation for a SPECIAL device (y = any of the following 15 characters: B, C, D, F, G, H, I, L, M, O, P, R, S, T, or U; z = any of the following 16 characters: A, B, C, D, F, G, H, I, L, M, O, P, R, S, T, or U).

Note: Subroutines of the type SRyzzz can be overlaid. Modifications within the subroutine code may or may not be present the next time the subroutine is used.

Use columns 54 through 59 to specify the subroutine that performs the input/output operations for a file assigned to a SPECIAL device. Columns 54 through 59 must contain an entry for each file assigned to a SPECIAL device. The subroutine name entered in columns 54 through 59 can be from 4 to 6 characters long. For a user-written subroutine the first 4 characters must be SUBR; the remaining characters can be any alphabetic character.

Note: If the user-written subroutines are in a different library from the RPG source program, the name of the library containing the subroutines must be specified at compilation time.

Continuation-Line Option for DISK File

The RECNO option is used to randomly add records to, or to load, a DISK file that allows records to be deleted.

Option	Value
RECNO	Name of a numeric field that is 7 digits long with zero decimal positions. The name must be coded in the leftmost of columns 60 through 65. This field name must be specified if records are to be added randomly to a direct or sequential file that allows deletions. This field name is also required for a direct file load of a file that allows deletions. You must place in the RECNO field the relative record number of the record to be added to the file. It must be the relative

record number of a deleted record. A deleted record is one that has been initialized to hex FFs. RPG uses the relative record number in the RECNO field to determine where a record is to be loaded (direct file load) or added (ADD on output specifications).

Note: If the program successfully reads a record from a sequential or direct file by a CHAIN or READ operation, RPG places the relative record number of this record in the RECNO field.

Continuation-Line Options for WORKSTN File

The following options can be specified for a WORKSTN file if more than one device is attached to a program or if you want to specify the WORKSTN file information data structure (INFDS) or the WORKSTN exception/error-processing subroutine (INFSR). The NUM keyword is required if the program attaches more than one device to a file at the same time. Enter the keyword in columns 54 through 59 and the value in columns 60 through 65 (columns 60 through 67 can be used if the FMTS option is specified).

Note: For WORKSTN files, a device can be either a display station or an SSP-ICF session.

Option	**Value**
NUM	Maximum number of devices that can be attached to this file at one time. The number specified must be coded in the rightmost of columns 60 through 65. If a number is not specified, 1 is assumed. If a number is specified, NUM must be greater than or equal to the number of requesters specified by the MRTMAX parameter when the program is compiled plus the number of acquired devices (those specified on the control language WORKSTN statement or in the ACQ operation). The number specified on the MRTMAX parameter is reserved for requesters. The difference between the MRTMAX value and the NUM value is the maximum number of acquired devices that can be attached to the program at one time by using control language statements or the ACQ operation. For example, if the MRTMAX value is 5 and the NUM value is 6, only one acquired device can be attached to the program, even if only one requester is presently signed on.
SAVDS	Name of a data structure that is to be saved and restored for each device attached to this file. The name must be coded in the leftmost of columns 60 through 65. This data structure cannot be a display station local data area, and it cannot contain a compile-time array or a preexecution-time array. If SAVDS is not specified, no data area swapping is done.
IND	Number of indicators, beginning with 01, that are to be saved and restored by display station. Prior to reading from a WORKSTN file, RPG saves indicators 01

through the number specified, for the device from which the last input record was read. After the read, indicators are restored from the save area belonging to the current device. No store is done before the first read. Therefore, after the first read, the indicators are restored from a blank save area, thereby setting them all off.

If IND is not specified, no indicator swapping is done. The entry must be coded in the rightmost of columns 60 through 65.

Note: For SAVDS and IND, only one copy of the data structure and indicators is available at a time. The indicators and data structure that are available are those associated with the device from which the last input was read.

The data structure and indicators that are available change each time an input operation (either a primary file input or a demand file read) is executed. On an input operation, the present copy of the data structure and indicators in the program is written to a save area for the device from which the previous input was read. The data structure and indicators for the device now being read from are then written from the save area associated with the device to the program SAVDS and IND areas. After the first input operation for each device, all the restored indicators will be off and all the fields in the SAVDS data structure will be blank. (For more information, see Chapter 6, *Using a WORKSTN File.)*

SLN — Name of a 2-digit numeric field whose value determines the first line on the display screen where the display format is to begin if a variable starting line number (V in column 17 of the display format S specification) was specified in the format. The name must be coded in the leftmost of columns 60 through 65. If SLN is not specified, all formats having a variable starting line number begin on line 1.

FMTS — *NONE. Indicates that there are only SSP-ICF formats present in this program.

Name of the load member that contains the display formats. The compiler uses the name specified here as the name of the display format load member. The name entered can be from 1 to 8 characters in length and must be coded in the leftmost of columns 60 through 67. If a name is not entered, the compiler assumes that the name of the display format load member is the program name (from columns 75 through 80 of the control specification) with FM added to the end of the name.

ID — Name of a 2-character, self-defining alphameric field that contains the identification of the device that supplied the record being processed in this file. The name must be coded in the leftmost of columns 60 through 65. The ID field is updated whenever a record is read from the WORKSTN file. Therefore, it always contains the identification of the device from which the last record was read (unless your program moves a different identification into the ID field). This field is considered self-defining because it need not be specified as an input or result field. For a multiple device file, you can direct an operation to a device other than the one currently being processed by changing the value in the ID field to the symbolic ID of another device in the file before performing the output operation.

The device identifications are assigned at system configuration time. Display station identifications are in the form AX, where A is any alphabetic character (A-Z, #, @, or $) and X is any character. If a control language WORKSTN statement exists for the display station, the identification is the same as the value of the SYMID parameter.

SSP-ICF session identifications can be in two formats. They are either NN where N is numeric (0-9), or NA where N is numeric and A is alphabetic (A-Z, #, @, or $). If the format is NA, a control language SESSION statement must be specified with a SYMID parameter whose value is also in an NA format.

INFSR Name of the user-written calculation subroutine designated as the WORKSTN exception/error-processing subroutine. The name must be coded in the leftmost of columns 60 through 65. Control may be passed to this subroutine if an exception/error occurs during the following operations: ACQ, REL, NEXT, POST, input (READ or primary input), or output (EXCPT operation or normal cycle output). If INFSR is not specified, the program halts if an exception/error occurs. See *Handling Exceptions and Errors* in Chapter 6 for more information on INFSR.

INFDS Name of the data structure that contains the identification of the type of exception/error condition and an indication of the WORKSTN operation that was executing when the exception/error condition occurred. The name must be coded in the leftmost of columns 60 through 65. If INFDS is not specified, this information is not available to the RPG program.

CFILE Name of a communications file that associates a WORKSTN file with a communications format file defined through the interactive data definition utility (IDDU). For more information about IDDU, see *Interactive Data Definition Utility* in Chapter 6, and see the manual *Getting Started with the Interactive Data Definition Utility (IDDU)*, GC21-8003.

Continuation-Line Option for SPECIAL Device

Option	**Value**
Array/table	Name of array or table used by the user-written subroutine name.

Columns 60-65 (Storage Index)

Columns 60 through 65 are not used. Leave them blank if this is not a continuation line.

Column 66 (File Addition)

Entry	Explanation
A	New records are added to the file.
U	Records are to be loaded for an indexed file in unordered sequence (random sequence).

Use column 66 to indicate:

- The program is to add new records to the file. Records can be added at detail, total, or exception time during the program cycle.
- The program is to load records in an unordered sequence.

Column 66 applies to direct, sequential, and indexed DISK files.

Note: Adding records to a file also requires a corresponding ADD entry in columns 16 through 18 of the output specifications.

For more information about adding records to a DISK file, see Chapter 5.

Columns 67-70

Columns 67 through 70 are not used. Leave them blank.

Columns 71-72 (File Condition)

Entry	Explanation
Blank	The file is not conditioned by an external indicator.
U1-U8	The file is conditioned by the specified external indicator.

Use columns 71 and 72 to indicate whether the file is conditioned by an external indicator. Columns 71 and 72 apply to input (excluding table input files and KEYBORD files), update, and output files. A file conditioned by an external indicator is used only when the indicator is on. When the indicator is off, the file is treated as though the end of the file is reached; that is, no records can be read from or written to the file.

The external indicators are normally set prior to processing by the control language SWITCH statement or by a previous RPG program. Their setting can be changed during processing, allowing the program to alter the status of these indicators. However, if an external indicator conditions a file, that indicator must be set on when the program is loaded in order to use the file in the program. For information about how to save and restore the external indicators for each display station attached to a WORKSTN file, see Chapter 6.

If a file is conditioned by an external indicator, any calculations that are not done when the file is not used should also be conditioned by the same indicator.

Columns 73-74

Columns 73 and 74 are not used. Leave them blank.

Columns 75-80 (Program Identification)

Entry	**Explanation**
Blank	Program identification defaults to the program name specified on the control specification.
Any valid program name	Program identification. The first character must be alphabetic but cannot be #, $, or @. The remaining characters must be alphameric with imbedded blanks. No special character can be used.

Columns 75 through 80 can contain any characters. These columns can contain the program name used in the control specification, or they can contain any other characters to identify a certain portion of the program. These entries are ignored by the compiler but appear in the source program listing.

Note: To be compatible with other RPG systems, the specifications sheets show only 80 positions for each statement. However, each statement in an RPG source program can contain up to 96 characters. Columns 81 through 96 are available for comments.

Chapter 22. Extension Specifications

Chapter 22. Extension Specifications

Extension specifications describe all record address files; compile-time or preexecution-time tables; and compile-time, preexecution-time, or execution-time arrays used in a program. Write these specifications on the RPG Extension and Line Counter Specifications sheet (see Figure 22-1). Record address files require entries in columns 11 through 26. Preexecution-time arrays and tables require entries in columns 11 through 45. Compile-time arrays and tables require entries in columns 19 through 45. Execution-time arrays require entries in columns 27 through 32 and in columns 36 through 45. If you want to specify an alternating array or table with the array or table described in columns 11 through 45 or 19 through 45, the alternating array or table must be described in columns 46 through 57 of the same line. A maximum of 75 arrays, tables, or data structures can be used in a program; however, only 70 of these can be compile-time arrays or tables.

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

GX21-9091 UM/050*
Printed in U.S.A.

IBM International Business Machines Corporation

Program		Punching Instruction	Graphic	Card Electro Number
Programmer	Date		Punch	

Page (1 2) of ___ Program Identification (75 76 77 78 79 80)

Extension Specifications

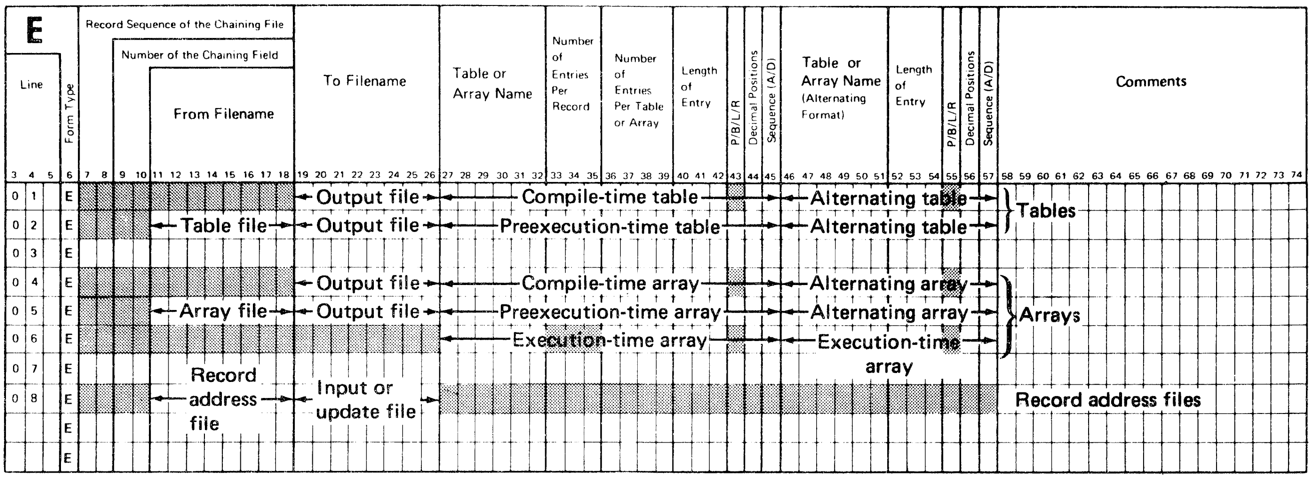
Line (3 4 5)	Form Type (6)	(7 8)	Record Sequence of the Chaining File / Number of the Chaining Field (9 10)	From Filename (11–18)	To Filename (19–26)	Table or Array Name (27–32)	Number of Entries Per Record (33–35)	Number of Entries Per Table or Array (36–39)	Length of Entry (40–42)	P/B/L/R (43)	Decimal Positions (44)	Sequence (A/D) (45)	Table or Array Name (Alternating Format) (46–51)	Length of Entry (52–54)	P/B/L/R (55)	Decimal Positions (56)	Sequence (A/D) (57)	Comments (58–74)
0 1	E																	
0 2	E																	
0 3	E																	
0 4	E																	
0 5	E																	
0 6	E																	
0 7	E																	
0 8	E																	
	E																	
	E																	

Line Counter Specifications

Line (3 4 5)	Form Type (6)	Filename (7–14)	1 Line Number (15–17)	1 FL or Channel Number (18–19)	2 Line Number (20–22)	2 OL or Channel Number (23–24)	3 Line Number (25–27)	3 Channel Number (28–29)	4 Line Number (30–32)	4 Channel Number (33–34)	5 Line Number (35–37)	5 Channel Number (38–39)	6 Line Number (40–42)	6 Channel Number (43–44)	7 Line Number (45–47)	7 Channel Number (48–49)	8 Line Number (50–52)	8 Channel Number (53–54)	9 Line Number (55–57)	9 Channel Number (58–59)	10 Line Number (60–62)	10 Channel Number (63–64)	11 Line Number (65–67)	11 Channel Number (68–69)	12 Line Number (70–72)	12 Channel Number (73–74)
1 1	L																									
1 2	L																									
	L																									

Figure 22-1. RPG Extension and Line Counter Specifications

Figure 22-2 shows possible extension specifications. See Chapter 13, *Using Arrays and Tables*, for a complete discussion of arrays and tables.



For tables and all arrays except execution-time arrays, columns 19 through 26 are optional. For all arrays and tables, columns 46 through 57 are optional.

Execution-time arrays are loaded by input and/or calculation specifications.

For record address files, columns 11 through 26 must have entries.

Figure 22-2. Possible Entries for Extension Specifications

Columns 1-2 (Page)

Entry	Explanation
Blank	No page number is used.
01-99	Page number.

Use columns 1 and 2 in the upper right corner of each sheet to number the specifications sheets, in ascending order, for your job. You can use more than one of each type of sheet, but keep all sheets of the same type together.

Columns 3-5 (Line)

Entry	Explanation
Blank	No line number is used.
Any numbers	Line numbers.

Use columns 3 through 5 to number the lines on each page. Columns 3 and 4 are preprinted on each sheet so, in most cases, line numbering is already done.

Page and line numbers are optional entries and are not required to successfully run an RPG program. Columns 1 through 5 are checked for ascending order, and RPG prints an S in the left margin of the RPG listing for any statement that is out of order. If you use SEU to enter the source program, you can request that SEU serialize the statements.

Column 6 (Form Type)

An E must appear in column 6 to identify this line as an extension specifications statement.

Column 7 (Comments)

Entry	Explanation
*	Comment line

Use an asterisk in column 7 to identify the line as a comment line. Use comments throughout your program to document the purpose of a certain section of coding. You can use any character in a comment line. Comments are not instructions to your program; they only document your program.

Columns 7-10

Columns 7 through 10 are not used to specify the record sequence of the chaining file or the number of the chaining field. Leave them blank.

Columns 7-12 (/EJECT)

Entry	Explanation
/EJECT	The specifications following this entry are to begin on a new page of the compiler listing.

The /EJECT specification is not printed on the compiler listing.

Columns 7-12 (/TITLE)

Entry	Explanation
/TITLE	The heading information (such as a title or security classification) that follows the /TITLE entry appears at the top of each page of the compiler listing. The heading information is entered in columns 14 through 74.

A program can contain more than one /TITLE statement. Each /TITLE statement provides heading information for the compiler listing until the next /TITLE statement is read. To print on the first page of the compiler listing, a /TITLE statement must be the first statement read. Information specified by the /TITLE statement is printed in addition to compiler heading information.

The /TITLE statement causes an eject to the next page before the title is printed. The /TITLE statement is not printed on the compiler listing.

Columns 7-14 (/SPACE)

Entry	Explanation
/SPACEƀn	Line spacing occurs at this point in the compiler listing Valid entries for n are 1 to 3. If you do not specify n, 1 is assumed.

One blank (ƀ) must come before the value you specify for n. The value you specify for n indicates the number of blank lines to be spaced before the next specification line is printed. If n is greater than the number of lines remaining on the current page, the next specification line is printed on a new page. If you specify just /SPACE, one line is spaced.

/SPACE is not printed on the compiler listing but is replaced by the actual line spacing. The spacing indicated by /SPACE is in addition to the three blank lines that occur between specification types.

Columns 11-18 (From filename)

Entry	Explanation
Blank	Array or table is loaded at compilation time if there is an entry in columns 33 through 35.
	Array is loaded at execution time (by input and/or calculation specifications) if there is no entry in columns 33 through 35.
Record address filename	Name of the record address file.
Array or table filename	Name of the array or table file loaded at preexecution time.

Use columns 11 through 18 to name an array file, table file, or record address file. Filenames must begin in column 11. The record address filename must always be entered in these columns and also on the file description specifications. The filename of every preexecution-time array or table used in the program must be entered in these columns and on the file description specifications. Leave columns 11 through 18 blank for compile-time arrays or tables and for arrays loaded by input and/or calculation specifications (execution-time arrays).

When an array or table is loaded at compilation time, it is compiled along with the source program and included in the object program. Such an array or table does not need to be loaded separately every time the program is run. Only those arrays and tables that contain constant data should be compiled with the program.

When arrays or tables are compiled with the program, the array or table records must always follow the RPG source program. A record with **ƀ (ƀ = blank) in positions 1 through 3 must separate the RPG source program from the array or table records. Arrays or tables must be separated from each other by records with **ƀ in positions 1 through 3. Because **ƀ in positions 1 through 3 indicates the start of an array or table, **ƀ must not be specified in positions 1 through 3 of the array or table input records.

Short tables (tables that contain blank entries) can be compiled with the program, but a warning is issued. See *Columns 36-39* in this chapter for a discussion of short tables.

Columns 19-26 (To Filename)

Entry	Explanation
Name of an input or update file	File processed by the record address file named in columns 11 through 18.
Name of an output file	Output file to which an array or table is to be written at end of job.

Use columns 19 through 26 to define the relationship between a file named in these columns and a file named in columns 11 through 18 or to name the file to which an array or table is to be written at end of job. Filenames must begin in column 19.

If a record address file is named in columns 11 through 18, the name of the input or update file that contains the data records to be processed must be entered in columns 19 through 26. Do not enter the record address filename in these columns.

If an array or table is to be written at end of job (that is, after last record processing), enter the filename of the output file in columns 19 through 26. This output file must be named previously in the file description specifications. An array or table can be written to only one output device. Leave columns 19 through 26 blank if the array or table is not to be written.

If an array or table is assigned to an output file, it is automatically written after all other records are written. The array or table is written in the same format in which it was entered.

Because there is no program control over the output format when an entry is made in columns 19 through 26, those cases where formatting is required should be provided for in the program through the output specifications or by the EXCPT operation that writes one item at a time. For more information, see Chapter 28, *Operation Codes*. Arrays or tables should be written only after all records are processed (last-record indicator LR is on).

Columns 27-32 (Array or Table Name)

Entry	Explanation
Array or table name	Name of array or table used in the program

Use columns 27 through 32 to name the array or table. No two arrays or tables can have the same name. The rules for forming array and table names are discussed in the following text.

Array Name

Each array used in a program must be given a separate name that does not begin with the letters TAB. The name can be from 1 to 6 characters long and must begin with an alphabetic character. This array name is used throughout the program. To refer to the entire array, use the array name alone. To refer to a single element of the array, use the array name plus an index. See *Array Name and Index* in Chapter 13 for more information on array names and on referencing array entries.

Table Name

Every table used in your program must have a name that begins with the letters TAB. The entire table name can be from 3 to 6 characters long.

After the letters TAB, 1 to 3 alphabetic or numeric characters can be used (no special characters are allowed). Blanks cannot appear between characters in the table name. Any name in columns 27 through 32 that does not begin with TAB is considered an array name.

The table name entered in columns 27 through 32 is used throughout the program. However, different results can be obtained depending upon how the table name is used. When the table name is used in factor 2 or the result field of the calculation specifications with a LOKUP operation, the name refers to the entire table. When the table name is used with any other operation code, the name refers to the table entry last selected from the table by a LOKUP operation (see *Operation Codes, LOKUP* in Chapter 28 and *Using Arrays and Tables* in Chapter 13).

Table files are processed in the order they are specified on the extension specifications.

Therefore, if you have more than one table file, the files should be loaded in the same order as they appear on the extension specifications.

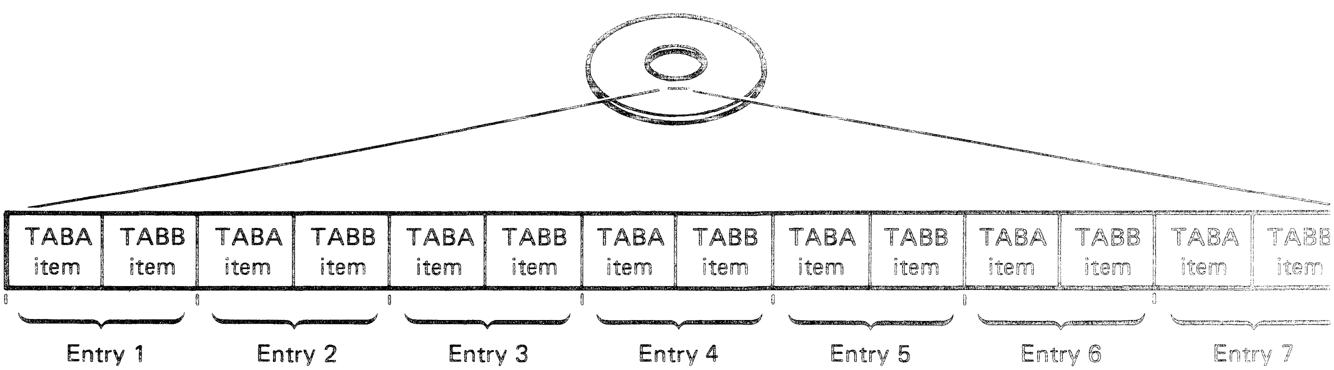
If two tables are in alternating format in one file, the table whose entry appears first must be named in columns 27 through 32. The second table is named in columns 46 through 51 (see Figure 22-3).

Two tables (TABA and TABB) are described in alternating format. An item for TABA appears first. Thus, TABA is named in columns 27 through 32 of the extension specifications sheet (see Part 2 of this figure); TABB is named in columns 46 through 51.

Table A (Account Number)	Table B (Amount Due)
00126	56.75
03240	39.00
03648	156.72
15632	17.98
28887	2.97
29821	290.98
30001	579.95
5 Positions	7 Positions

The account number and the amount due for that account number are corresponding table items.

Note: The decimal points shown in Table B are only for illustration purposes. Decimal points are not a part of array or table input data.



The corresponding items from the tables are entered in the system in alternating format. Corresponding items from the two tables are considered as one entry.

Figure 22-3 (Part 1 of 2). Related Tables

Table entries for the two tables, A and B, are entered in alternating format. A1 and B1, the corresponding items in tables A and B, are considered one entry. Even though 14 table items are listed, there are only seven table entries.

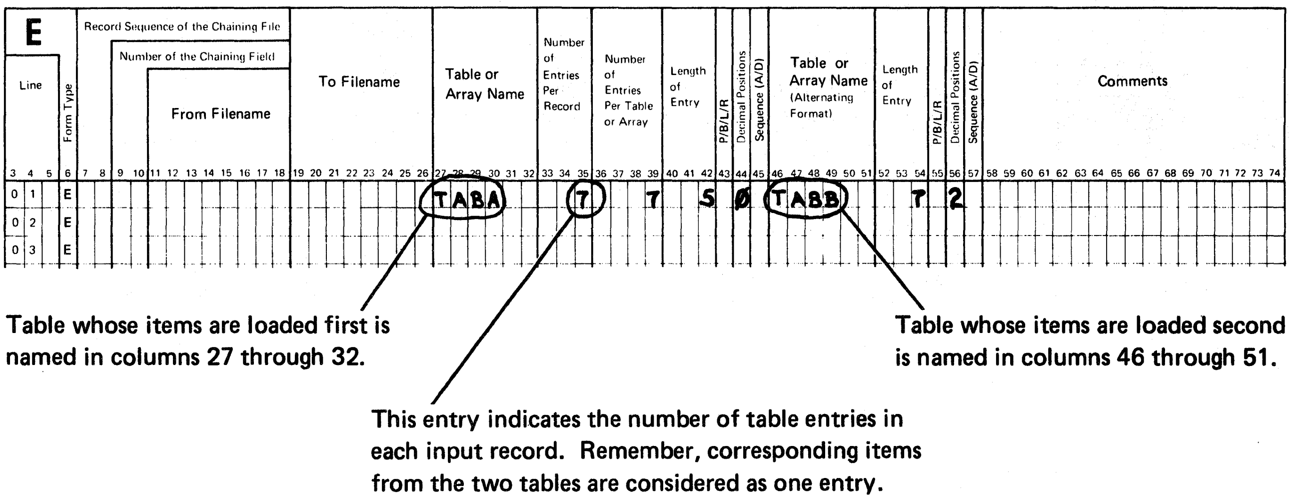


Figure 22-3 (Part 2 of 2). Related Tables

Columns 33-35 (Number of Entries per Record)

Entry	Explanation
1-999	Number of array or table entries in each table or array input record

Use columns 33 through 35 to indicate the exact number of array or table entries in each array or table input record. The number must end in column 35. Every array or table input record except the last must contain the same number of entries as indicated in columns 33 through 35. The last record can contain fewer entries than indicated, but not more. Comments can be entered on table input records in the positions following the table entries.

If two arrays or tables are in alternating format in one file, each array or table input record must contain the corresponding entries from each array or table. The corresponding entries from the two arrays or tables are considered one entry and must be on the same record.

When columns 27 through 32 contain an array name, the following rules apply to the use of columns 11 through 18 and 33 through 35:

- For a preexecution-time array, columns 11 through 18 must contain a filename and columns 33 through 35 must have an entry.
- For a compile-time array, columns 11 through 18 must be blank and columns 33 through 35 must have an entry.
- For an execution-time array, columns 11 through 18 and columns 33 through 35 must be blank.

Columns 36-39 (Number of Entries per Table or Array)

Entry	Explanation
1-9999	Maximum number of array or table entries

Use columns 36 through 39 to indicate the maximum number of entries that can be contained in the array or table named in columns 27 through 32. This number applies to one array or table or to two arrays or tables in alternating format. The number entered must end in column 39.

Because the number of entries for two arrays or tables written in alternating format must be the same, the number in these columns also gives the number of entries in the second array or table specified in columns 46 through 51.

If the array or table is full, these columns give the exact number of entries in it. However, if the array or table is not full, these columns give the number of entries that can be put into it (see Figure 22-4). An array or table that is not full is one that contains unused entries and is known as a *short array* or *table.*

A compile-time array or table should be full. However, if it is not full (a short array or table), the array or table is compiled with the program, and a warning is issued. In storage, the unused entries in a short array or table are filled with blanks or zeros (for alphameric or numeric arrays or tables, respectively).

A preexecution-time array or table need not be full.

TABPRT (Part Number)	TABAMT (Price)
001	127.62
002	198.32
003	.27
004	.01
005	1.98
009	3.79
010	5.67
014	2.33
026	14.67
045	29.33
096	29.34
097	.05
098	.09
099	1.19
100	2.22
101	126.73
110	596.74
115	393.75
126	697.75
137	1.92

If this data is entered into the system, TABPRT and TABAMT will be full (20 entries fill the table).

TABPRT (Part Number)	TABAMT (Price)
001	127.62
002	198.32
003	.27
004	.01
005	1.98

If this data is entered into the system, TABPRT and TABAMT will not be full.

Note:
The decimal points shown in these tables are only for illustration purposes. Decimal points are not part of table input data.

E Line (3-5)	Form Type (6)	Record Sequence of the Chaining File (7-8)	Number of the Chaining Field (9-10)	From Filename (11-18)	To Filename (19-26)	Table or Array Name (27-32)	Number of Entries Per Record (33-35)	Number of Entries Per Table or Array (36-39)	Length of Entry (40-42)	P/B/L/R (43)	Decimal Positions (44)	Sequence (A/D) (45)	Table or Array Name (Alternating Format) (46-51)	Length of Entry (52-54)	P/B/L/R (55)	Decimal Positions (56)	Sequence (A/D) (57)	Comments (58-74)
01	E					TABPRT	12	20	3		Ø		TABAMT	5		2		
02	E																	

This entry indicates that TABPRT and TABAMT can both have a maximum of 20 entries.

Figure 22-4. Table Entries (Number per Table)

Columns 40-42 (Length of Entry)

Entry	Explanation
1-15	Length of a numeric entry
1-256	Length of an alphameric entry

Use columns 40 through 42 to specify the length of each entry in the array or table named in columns 27 through 32. The number entered must end in column 42. For numeric arrays or tables in packed decimal format, enter the zoned decimal length in columns 40 through 42. For numeric arrays or tables in binary format, enter the number of digits required in storage for the binary field. For a 2-position binary field, the entry in columns 40 through 42 is 4; for a 4-position binary field, the entry is 9.

All array or table entries must have the same number of characters. It is almost impossible, however, for every item to be the same length. Therefore, add leading zeros for numeric entries and add blanks after alphameric entries to make them the same length (see Figure 22-5).

List of Months	TABMO
JANUARY	JANUARYƀƀ
FEBRUARY	FEBRUARYƀ
MARCH	MARCHƀƀƀƀ
APRIL	APRILƀƀƀƀ
MAY	MAYƀƀƀƀƀƀ
JUNE	JUNEƀƀƀƀƀ
JULY	JULYƀƀƀƀƀ
AUGUST	AUGUSTƀƀƀ
SEPTEMBER	SEPTEMBER
OCTOBER	OCTOBERƀƀ
NOVEMBER	NOVEMBERƀ
DECEMBER	DECEMBERƀ

All entries must have the same length. Those items that are not as long as the longest item must be padded with blanks (ƀ).

Figure 22-5. Length of Table Entries

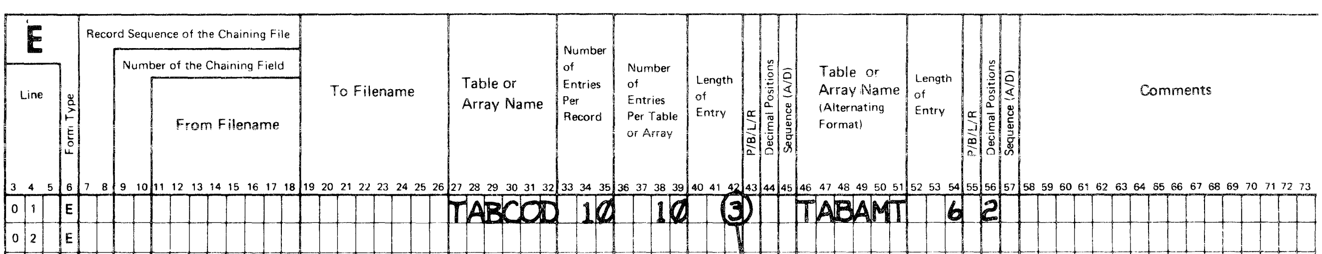
If two arrays or tables are entered in alternating format, the specification in columns 40 through 42 applies to the array or table whose entry appears first in the record (see Figure 22-6).

See Chapter 13, *Using Arrays and Tables*, for more information.

TABCOD (Code)	TABAMT (Amount)
021	217.43
022	93.06
023	8.14
040	2166.58
041	39.23
060	1741.78
117	83.33
118	5.12
143	72.03
352	253.96
3 Positions	6 Positions

Two tables are entered in alternating format, TABCOD and TABAMT. Each item in TABCOD is 3 characters long; each item in TABAMT is 6 characters long. Since TABCOD is entered in the system first, its length, 3, is specified in columns 40 through 42. The length of items in TABAMT is in columns 52 through 54.

Note: The decimal points shown in these tables are only for illustration purposes. Decimal points are not a part of table input data.



The length of the table item that is first entered in the system must appear in columns 40 through 42.

Figure 22-6. Length of Corresponding Table Items

Column 43 (Packed or Binary Field)

Entry	Explanation
Blank	Data for array or table is in zoned decimal format or is alphameric.
P	Data for array or table is in packed decimal format on disk.
B	Data for array or table is in binary format on disk.

Use column 43 to indicate that a numeric field in a preexecution-time array or table file is in packed or binary format. Leave column 43 blank if the field is in zoned decimal format. See *Column 43* under *Field Description Entries* in Chapter 25 for more information on packed or binary format.

Column 44 (Decimal Positions)

Entry	Explanation
Blank	Alphameric array or table
0-9	Number of positions to the right of the decimal in numeric array or table items

Use column 44 to indicate the number of decimal positions in a numeric array or table entry. Column 44 must always have an entry for a numeric array or table. If the entries in an array or table have no decimal positions, enter a 0.

If two arrays or tables are entered in alternating format, the specification in this column applies to the array or table containing the entry that appears first on the record.

Column 45 (Sequence)

Entry	Explanation
Blank	No particular order
A	Ascending order
D	Descending order

Use column 45 to describe the sequence (either ascending or descending) of the data in an array or table.

When an entry is made in column 45, the array or table is checked for the specified sequence. If a compile-time array or table is out of sequence, a terminal error occurs, and the compiler halts. If a preexecution-time array or table is out of sequence, an error occurs, and the program halts immediately. The program can be restarted from the point where it halted if you do not want to correct the out-of-sequence condition; however, if you do correct the out-of-sequence condition, program execution must be restarted from the beginning.

Ascending order means that the array or table entries start with the lowest data entry (according to the collating sequence) and proceed to the highest. *Descending order* means that the array or table entries start with the highest data entry and proceed to the lowest.

If two arrays or tables are entered in alternating format, the entry in column 45 applies to the array or table containing the entry that appears first on the record. When the LOKUP operation is used to search an array or table for an entry to determine whether the entry is high or low compared with the search word, the array or table must be in either ascending or descending order. See *LOKUP* in Chapter 28 for more information.

An execution-time array (built by input and/or calculation specifications) is not sequence-checked. However, an A or D entry must be specified if a high or low LOKUP operation is performed.

Columns 46-57

Entry	Explanation
Array or table name and description	Name and description of the alternating array or table

Use columns 46 through 57 only to describe a second array or table that is entered in an alternating format with the array or table specified in columns 27 through 45. All fields in this section have the same significance and require the same entries as the fields with corresponding titles in columns 27 through 45. See the previous discussion on those columns for information about correct specifications. Leave these columns blank for a single array or table.

Columns 58-74 (Comments)

Columns 58 through 74 can be used for comments to document the purpose of each specification line.

Columns 75-80 (Program Identification)

Entry	Explanation
Blank	Program identification defaults to the program name specified on the control specifications.
Any valid program name	Program identification. The first character must be alphabetic but cannot be #, $, or @. The remaining characters must be alphameric with no imbedded blanks. No special character can be used.

Columns 75 through 80 can contain any characters. These columns can contain the program name used in the control specifications, or they can contain any other characters to identify a certain portion of the program. These entries are ignored by the compiler but appear in the source program listing.

Note: To be compatible with other RPG systems, the specifications sheets show only 80 positions for each statement. However, each statement in an RPG source program can contain up to 96 characters. Columns 81 through 96 are available for comments.

Chapter 23. Line Counter Specifications

Chapter 23. Line Counter Specifications

Line counter specifications indicate at what line overflow occurs and the length of the form used in the printer. Both of these entries may be specified on the RPG Extension and Line Counter Specifications sheet (see Figure 23-1).

Line counter specifications may be used for each PRINTER file in your program. If no line counter specifications exist, the form length used is the form length specified on the PRINTER OCL statement. (See the LINES parameter of the PRINTER OCL statement in the *System Reference* manual for a description of the defaults for the form length.) In this instance, the overflow line is assumed to be six lines less than the specified form length.

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

GX21-9091 UM/050*
Printed in U.S.A.

IBM International Business Machines Corporation

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

Extension Specifications

E Line (3 4 5)	Form Type (6)	Record Sequence of the Chaining File (7 8)	Number of the Chaining Field (9 10)	From Filename (11–18)	To Filename (19–26)	Table or Array Name (27–32)	Number of Entries Per Record (33–35)	Number of Entries Per Table or Array (36–39)	Length of Entry (40–42)	P/B/L/R (43)	Decimal Positions (44)	Sequence (A/D) (45)	Table or Array Name (Alternating Format) (46–51)	Length of Entry (52–54)	P/B/L/R (55)	Decimal Positions (56)	Sequence (A/D) (57)	Comments (58–74)
0 1	E																	
0 2	E																	
0 3	E																	
0 4	E																	
0 5	E																	
0 6	E																	
0 7	E																	
0 8	E																	
	E																	
	E																	

Line Counter Specifications

L Line (3 4 5)	Form Type (6)	Filename (7–14)	1 Line Number (15–17)	1 FL or Channel Number (18 19)	2 Line Number (20–22)	2 OL or Channel Number (23 24)	3 Line Number (25–27)	3 Channel Number (28 29)	4 Line Number (30–32)	4 Channel Number (33 34)	5 Line Number (35–37)	5 Channel Number (38 39)	6 Line Number (40–42)	6 Channel Number (43 44)	7 Line Number (45–47)	7 Channel Number (48 49)	8 Line Number (50–52)	8 Channel Number (53 54)	9 Line Number (55–57)	9 Channel Number (58 59)	10 Line Number (60–62)	10 Channel Number (63 64)	11 Line Number (65–67)	11 Channel Number (68 69)	12 Line Number (70–72)	12 Channel Number (73 74)
1 1	L																									
1 2	L																									
	L																									

Figure 23-1. RPG Extension and Line Counter Specifications

Columns 1-2 (Page)

Entry	Explanation
Blank	No page number is used.
01-99	Page number.

Use columns 1 and 2 in the upper right corner of each sheet to number the specifications sheets, in ascending order, for your job. You can use more than one of each type of sheet, but keep all sheets of the same type together.

Columns 3-5 (Line)

Entry	Explanation
Blank	No line number is used.
Any numbers	Line numbers.

Use columns 3 through 5 to number the lines on each page. Columns 3 and 4 are preprinted on each sheet so, in most cases, line numbering is already done.

Page and line numbers are optional entries and are not required to successfully run an RPG program. Columns 1 through 5 are checked for ascending order, and RPG prints an S in the left margin of the RPG listing for any statement that is out of order. If you use SEU to enter the source program, you can request that SEU serialize the statements.

Column 6 (Form Type)

An L must appear in column 6 to identify this line as a line counter specifications statement.

Column 7 (Comments)

Entry	Explanation
*	Comment line

Use an asterisk in column 7 to identify the line as a comment line. Use comments throughout your program to help document the purpose of a certain section of coding. You can use any character in a comment line. Comments are not instructions to your program; they only document your program.

Column 7-12 (/EJECT)

Entry	Explanation
/EJECT	The specifications following this entry are to begin on a new page of the compiler listing.

The /EJECT specification is not printed on the compiler listing.

Columns 7-12 (/TITLE)

Entry	Explanation
/TITLE	The heading information (such as a title or security classification) that follows the /TITLE entry appears at the top of each page of the compiler listing. The heading information is entered in columns 14 through 74.

A program can contain more than one /TITLE statement. Each /TITLE statement provides heading information for the compiler listing until the next /TITLE statement is read. To print on the first page of the compiler listing, a /TITLE statement must be the first statement read. Information specified by the /TITLE statement is printed in addition to compiler heading information.

The /TITLE statement causes an eject to the next page before the title is printed. The /TITLE statement is not printed on the compiler listing.

Columns 7-14 (/SPACE)

Entry	Explanation
/SPACEƀn	Line spacing occurs at this point in the compiler listing. Valid entries for n are 1 to 3. If you do not specify n, 1 is assumed.

One blank (ƀ) must come before the value you specify for n. The value you specify for n indicates the number of blank lines to be spaced before the next specification line is printed. If n is greater than the number of lines remaining on the current page, the next specification line is printed on a new page. If you specify just /SPACE, one line is spaced.

/SPACE is not printed on the compiler listing but is replaced by the actual line spacing. The spacing indicated by /SPACE is in addition to the three blank lines that occur between specification types.

Columns 7-14 (Filename)

Entry	Explanation
A valid filename	Filename of the printer output file as previously defined on the file description specifications sheet. The filename must begin in column 7.

Use columns 7 through 14 to identify the output file to be printed on the printer.

Columns 15-17 (Line Number--Number of Lines per Page)

Entry	Explanation
1-112	Number of printing lines available is 1 to 112.

Use columns 15 through 17 to specify the exact number of lines available on the page you want to use. The entry must end in column 17. Leading zeros can be omitted.

Columns 18-19 (Form Length)

Entry	Explanation
FL	Form length

Use columns 18 and 19 to indicate that the entry in columns 15 through 17 is the form length. Columns 18 and 19 must contain the entry FL.

Columns 20-22 (Line Number--Overflow Line)

Entry	Explanation
1-112	The line number specified is the overflow line.

Use columns 20 through 22 to specify the line number that is the overflow line. The entry must end in column 22. Leading zeros can be omitted. The entry must be less than or equal to the form length specified in columns 15 through 17. When the line that is specified as the overflow line is printed, the overflow indicator turns on. When the overflow indicator is on and fetch overflow is not specified, the following occurs before forms advance to the next page:

1. Detail lines are printed (if this part of the program cycle has not already been completed).

2. Total lines are printed (if conditions are met).

3. Total lines conditioned by the overflow indicator are printed.

Because all these lines are printed on the page after the overflow line, specify the overflow line high enough on the page to allow all these lines to print. See *Handling Overflow* in Chapter 7 and *Overflow Indicators* in Chapter 12 for more information.

Note: If the number of lines per page entry equals the overflow line entry, no overflow occurs.

Columns 23-24 (Overflow Line)

Entry	Explanation
OL	Overflow line

Use columns 23 and 24 to indicate that the entry in columns 20 through 22 is the overflow line. Columns 23 and 24 must contain OL.

Columns 25-74

Columns 25 through 74 are not used. Leave them blank.

Columns 75-80 (Program Identification)

Entry	Explanation
Blank	Program identification defaults to the program name specified on the control specifications.
Any valid program name	Program identification. The first character must be alphabetic but cannot be #, $, or @. The remaining characters must be alphameric with no imbedded blanks. No special character can be used.

Columns 75 through 80 can contain any characters. These columns can contain the program name used in the control specifications, or they can contain any other characters to identify a certain portion of the program. These entries are ignored by the compiler but appear in the source program listing.

Note: To be compatible with other RPG systems, the specifications sheets show only 80 positions for each statement. However, each statement in an RPG source program can contain up to 96 characters. Columns 81 through 96 are available for comments.

Chapter 24. Telecommunications Specifications

Chapter 24. Telecommunications Specifications

Telecommunications specifications describe the information necessary to establish and maintain the batch BSC (binary synchronous communications) link. Each BSCA file defined on the file description specifications must have a corresponding specification on the RPG Telecommunications Specifications sheet (see Figure 24-1).

RPG data communications programming enables you to send and receive binary synchronous data via a data communications network. RPG data communications support performs all the functions necessary to establish the line connections, exchange identification sequences, send and receive data, and execute the correct termination or disconnect procedures.

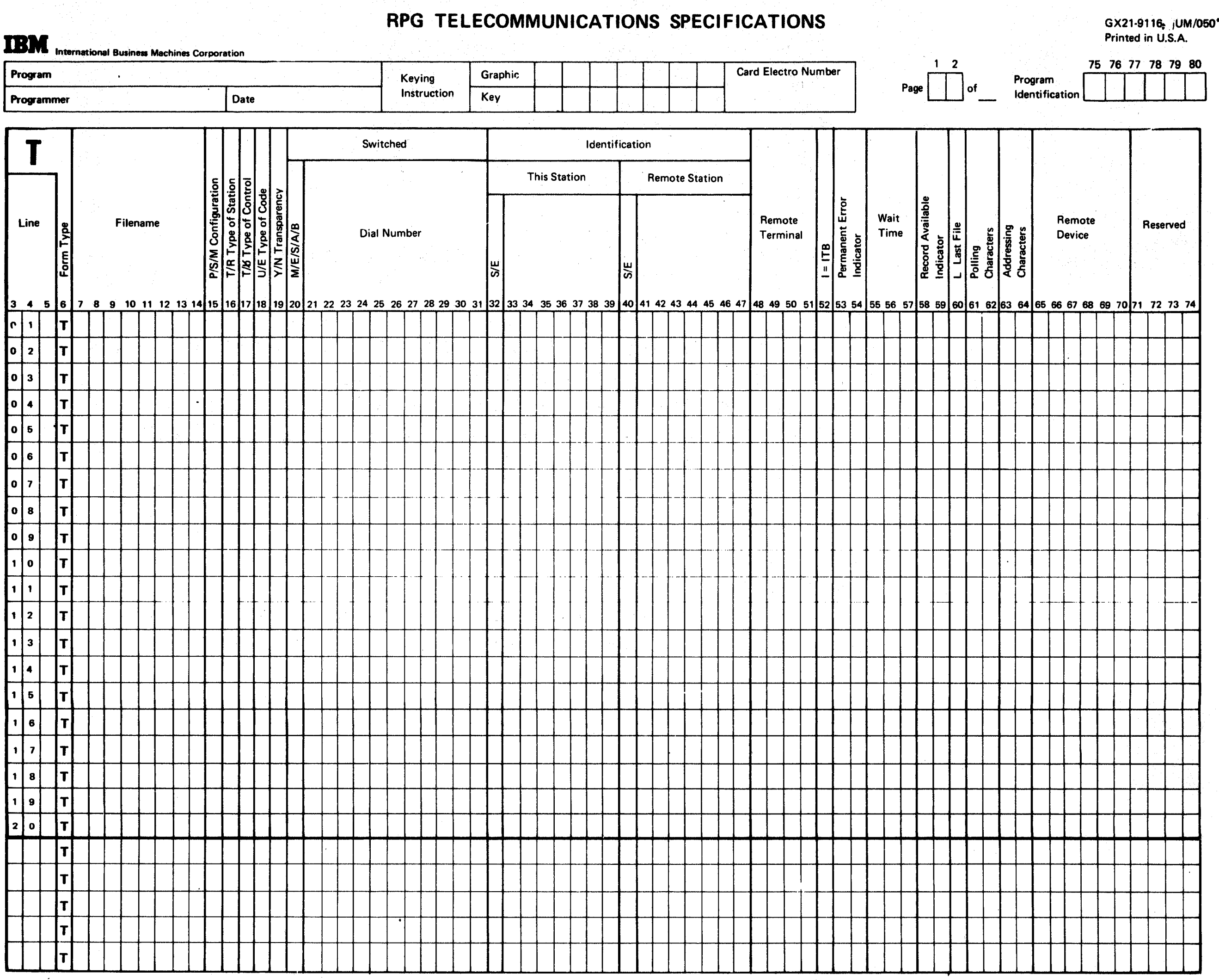


Figure 24-1. RPG Telecommunications Specifications

RPG batch BSC permits System/36 to function as any of the following station types:

- Receive only (receive input data from a remote station).
- Send only (send data to a remote station).
- Send and receive, but no conversational reply. Three modes of operation are possible:
  - Sending a file, then receiving another file
  - Receiving a file, then sending another file
  - Sending records of one file interspersed with receiving records of another file

BSC is a flexible form of line control that provides a set of rules for communications between devices. For a description of the basic characteristics and operational concepts of BSC, a description of the RPG interface to BSC, and a complete description of RPG data communications programming, see Chapter 10, *Using a BSCA File.*

Note: Telecommunications specifications are used only for RPG data communications programming (batch BSC). Telecommunications specifications are not used for the Interactive Communications Feature (SSP-ICF).

Columns 1-2 (Page)

Entry	Explanation
Blank	No page number is used.
01-99	Page number.

Use columns 1 and 2 in the upper right corner of each sheet to number the specifications sheets, in ascending order, for your job. You can use more than one of each type of sheet, but keep all sheets of the same type together.

Columns 3-5 (Line)

Entry	Explanation
Blank	No line number is used.
Any numbers	Line numbers.

Use columns 3 through 5 to number the lines on each page. Columns 3 and 4 are preprinted on each sheet so, in most cases, line numbering is already done.

Page and line numbers are optional entries and are not required to successfully run an RPG program. Columns 1 through 5 are checked for ascending order, and RPG prints an S in the left margin of the RPG listing for any statement that is out of order. If you use SEU to enter the source program, you can request that SEU number the specifications in ascending order.

Column 6 (Form Type)

A T must appear in column 6 to identify this line as a telecommunications specification.

Column 7 (Comments)

Entry	Explanation
*	Comment line

Use an asterisk in column 7 to identify the line as a comment line. Use comments throughout your program to help document the purpose of a certain section of coding. You can use any character in a comment line. Comments are not instructions to the RPG II program; they only document your program.

Columns 7-12 (/EJECT)

Entry	Explanation
/EJECT	The specifications following this entry are to begin on a new page of the compiler listing.

The /EJECT specification is not printed on the compiler listing.

Columns 7-12 (/TITLE)

Entry	Explanation
/TITLE	The heading information (such as a title or security classification) that follows the /TITLE entry appears at the top of each page of the compiler listing. The heading information is entered in columns 14 through 74.

A program can contain more than one /TITLE statement. Each /TITLE statement provides heading information for the compiler listing until the next /TITLE statement is read. To print on the first page of the compiler listing, a /TITLE statement must be the first statement read. Information specified by the /TITLE statement is printed in addition to compiler heading information.

The /TITLE statement causes an eject to the next page before the title is printed. The /TITLE statement is not printed on the compiler listing.

Columns 7-14 (/SPACE)

Entry	Explanation
/SPACEƀn	Line spacing occurs at this point in the compiler listing. Valid entries for n are 1 to 3. If you do not specify n, 1 is assumed.

One blank (ƀ) must come before the value you specify for n. The value you specify for n indicates the number of blank lines to be spaced before the next specification line is printed. If n is greater than the number of lines remaining on the current page, the next specification line is printed on a new page. If you specify just /SPACE, one line is spaced.

/SPACE is not printed on the compiler listing but is replaced by the actual line spacing. The spacing indicated by /SPACE is in addition to the three blank lines that occur between specification types.

Columns 7-14 (Filename)

Entry	Explanation
A valid filename	Filename previously defined on the file description specifications for the BSC device.

Column 15 (Configuration)

Entry	Explanation
P or blank	This is a point-to-point nonswitched line.
M	This is a multipoint line where the control station selects the tributary station through polling or addressing. System/36 cannot be the control station.
S	This is a point-to-point switched line.

If this column contains an M, column 17 must contain a T.

Column 16 (Type of Station)

Entry	Explanation
T	This station sends (transmits) information from the file named in columns 7 through 14. The file must be designated as an output file on the file description specifications and must be defined on the output specifications.
R	This station receives information in the file named in columns 7 through 14. The file must be designated as an input file on the file description specifications and must be defined on the input specifications.

Note: This entry is independent of the entry in column 20.

Column 17 (Type of Control)

Entry	Explanation
Blank	Polling is not used.
T	This is a tributary station on a multipoint network. Column 17 must contain a T if column 15 contains an M.

System/36 cannot be the control station.

Column 18 (Type of Code)

Entry	Explanation
A or U	ASCII (formerly referred to as USASCII) transmission control characters are used. An A or U entry causes the necessary file translation to be done for System/36.
E or blank	EBCDIC transmission control characters are used.

ASCII and EBCDIC characters are listed in the *System Reference* manual.

If your BSC program halts because of an invalid ASCII character in your data, check your data and the ASCII translation table.

Column 19 (Transparency)

Entry	Explanation
Y	EBCDIC transparency is used. The data being transferred can contain transmission control characters and/or packed numeric or alphameric characters. Column 18 must be E or blank.
N or blank	EBCDIC transparency is not used. Zoned decimal numeric or alphameric data is sent and received. The data being transferred cannot contain transmission characters.

Column 20 (Switched)

Entry	Explanation
Blank	This is not a switched line.
M	The operator using this program makes the connection by dialing the number (manual dial).
A	This program uses autoanswer.
B	This program uses manual answer.

Note:

1. This entry is independent of the entry in column 16.
2. If you are using an autocall line, the switch type specified has no effect. However, if no phone list is specified in the control language COMM statement, the switch type specified here is established unless it has been overridden with the ALTERCOM procedure.

Columns 21-31

Columns 21 through 31 are not used. Leave them blank.

Column 32 (Location of Identification--This Station)

Entry	Explanation
Blank	No identification is used for this station.
S	This station's identification is at the position specified by the symbolic name in columns 33 through 39. This entry only applies to switched lines.
E	The entry in columns 33 through 39 is this station's identification. This entry only applies to switched lines.

Columns 33-39 (Identification--This Station)

Entry	Explanation
Alphameric characters	When column 32 contains an E, this entry is the actual identification sequence of this station (minimum of 2 characters). When column 32 contains an S, this entry is the symbolic name of the location of this station's identification.

If columns 33 through 39 contain a symbolic name, it must not be an array name. If the BSCA file is a primary or secondary file, the symbolic name must refer to the first entry of a table (the table might have only one entry) to make sure that the station identification is in storage before the communications line is open.

The station identification referred to by the symbolic name can be from 2 to 15 characters long, but it must not contain a transmission control character. The station identification is translated if the BSCA files are translated.

Column 40 (Location of Identification--Remote Station)

Entry	Explanation
Blank	No identification is used for the remote station.
S	The remote station's identification is at the position specified by the symbolic name in columns 41 through 47. This entry only applies to switched lines.
E	The entry in columns 41 through 47 is the remote station's identification. This entry only applies to switched lines.

Columns 41-47 (Identification--Remote Station)

Entry	Explanation
Alphameric characters	When column 40 contains an E, this entry is the actual identification sequence of the remote station (minimum of 2 characters). When column 40 contains an S, this entry is the symbolic name of the location of the remote station's identification.

If columns 41 through 47 contain a symbolic name, it must not be an array name. If the BSCA file is a primary or secondary file, this symbolic name must refer to the first entry in a table (the table might have only one entry) to make sure that the station identification is in storage before the communications line is open.

The station identification referred to by the symbolic name can be from 2 to 15 characters long, but must not contain a transmission control sequence character. The station identification is translated if the BSCA files are translated. The identification received from the remote station is compared with this entry. The session continues only if the identification matches.

Columns 48-51

Columns 48 through 51 are not used. Leave them blank.

Column 52 (ITB)

Entry	Explanation
Blank	Intermediate block checking is not used.
I	Intermediate block checking is used.

Intermediate block checking (ITB) can be used only if the records are blocked. ITB and EBCDIC transparency cannot both be specified for the same BSC output file.

Columns 53-54 (Permanent-Error Indicator)

Entry	Explanation
Blank	No permanent-error indicator is specified. If a permanent error occurs, a system halt occurs and the program cannot be restarted.
01-99, L1-L9,LR, H1-H9	A permanent-error indicator can be specified for every BSCA file. The indicator does not have to be unique for each file. You should specify a permanent-error indicator when the system is running and no one is attending to the system.

Use columns 53 and 54 to specify a permanent-error indicator for every BSCA file. When a permanent error occurs, the specified error indicator and the identification indicator of the record causing the error turn on (however, no hardware diagnostics are performed). The permanent-error indicator can then be used to condition the appropriate programming response, such as printing a message or performing a controlled cancel.

Do not attempt to send information while the permanent-error indicator is on. This includes attempts to send more than one record during detail, total, or exception output. Further attempts to send information can be prevented if each record to be sent is conditioned with the not-permanent-error indicator in columns 9 through 11 of the calculation specifications or columns 23 through 31 of the output specifications.

To retry an operation after a permanent error occurs, turn off the permanent-error indicator. The RPG program can then access the BSCA file on which the error occurred. If an error occurs on the retried operation, the permanent-error indicator is turned on again; otherwise, processing continues.

Consider the following points when retrying an operation:

- The permanent-error indicator is the only indication to the RPG program that an error occurred. A BSC information message describing the type of error is displayed. If a halt (H1 through H9) is not issued as part of the permanent-error routine, the BSC information message may not be preserved on the display screen. You can find the message by running the HISTORY procedure. For more information on the HISTORY procedure, see the *System Reference* manual.

- Any data in the BSC buffers at the time of an error is lost. The record in your buffers is not the same as the record in the BSC buffers. Therefore, retrying the last operation will still result in lost data.

- Switched lines are not disabled when an error occurs unless a disconnect sequence is received or the hardware detects disconnect.

- Any data sent while the permanent-error indicator is on is invalid. Unless your program is designed to recognize all data, the error condition can cause an unidentified record halt.

- A limit should be imposed by the RPG program on the number of times an error can occur before the program is stopped.

Note: Avoid using H1 through H9 as permanent-error indicators if you are going to condition operations on the permanent-error indicator being off. Because H1 through H9 are reset at the end of the detail logic cycle, they can be set off before the program cycle in which the error occurred is completed. If H1 through H9 are used as permanent-error indicators, the H1 through H9 display can preempt the system halt display. If the H1 through H9 display appears before the system display, the person using the display station should take the 0 option to prompt the system halt display.

Columns 55-57 (Wait Time)

Entry	Explanation
Blank	The system convention for time-out, 180 seconds, is used.
Numeric	The length in time in seconds, 1 to 999, that BSC waits with no messages being sent or received before a permanent error occurs.

A permanent error is recognized by the system whenever the wait time on an idle line elapses. Therefore, when determining the wait time, consider the time the person using the display station might require to respond to halts and other processing interruptions, and also time the program might require for special operations such as table searches and computing square roots.

The wait time limit specified applies only to delays caused by the System/36 program and does not apply to the remote device. In addition, the time limit applies only during the transmission or reception of a file, not between file transmissions.

The occurrence of a permanent error indicates the end of processing of a file, but not the end of file.

Columns 58-59 (Record-Available Indicator)

Entry	Explanation
01-99, L1-L9, LR, H1-H9	A record-available indicator should be specified if a reverse interrupt (RVI) is to be received. This indicator turns on whenever a reverse interrupt (RVI) is received.

Column 60 (Last File)

Entry	Explanation
Blank	This BSCA file may not be the last input file processed.
L	This BSCA file is processed only after all other input files are processed. All secondary files should have L in column 60.

The entry in column 60 does not affect demand files.

Columns 61-62 (Polling Characters)

Entry	Explanation
Blank	This station is not part of a multipoint network.
Alphameric characters	The polling identification of this station is required if this station is part of a multipoint network and the BSCA file is a sending (output) file. Polling and addressing characters must be used in pairs.

Columns 63-64 (Addressing Characters)

Entry	Explanation
Blank	This station is not part of multipoint network.
Alphameric character	The addressing identification of this station is required if this station is part of a multipoint network and the BSCA file is a receiving (input) file. Polling and addressing characters must be used in pairs.

Enter polling and addressing characters in EBCDIC; the compiler converts the characters to the form required by the code specified in column 18. (If ASCII was specified, enter uppercase addressing characters; they are converted to lowercase ASCII characters.)

Columns 65-74

Columns 65 through 74 are not used. Leave them blank.

Columns 75-80 (Program Identification)

Entry	Explanation
Blank	Program identification defaults to the program name specified on the control specification.
Any valid program name	Program identification. The first character must be alphabetic but cannot be #, $, or @. The remaining characters must be alphameric with no imbedded blanks. No special character can be used.

Columns 75 through 80 can contain any characters. These columns can contain the program name used in the control specification, or they can contain any other characters to identify a certain portion of the program. These entries are ignored by the compiler but appear in the source listing.

Note: To be compatible with other RPG systems, the specifications sheets show only 80 positions for each statement. However, each statement in an RPG source program can contain up to 96 characters. Columns 81 through 96 are available for comments.

Chapter 25. Input Specifications

Chapter 25. Input Specifications

Input specifications describe the data files, records, and fields of the records used in the program. All input files are described on the input specifications except files assigned to the device KEYBORD, record address files, address output files, and table files. KEYBORD files are described on the calculation specifications when the KEY operation code is used. Record address files, address output files, and table files are described on the extension specifications.

Input specifications are also used to describe data structures.

The input specifications are divided into two categories:

- File and record-type identification entries (columns 7 through 42) describe the input record and its relationship to other records in the file.

- Field description entries (columns 43 through 74) describe the fields in the records. These specifications must start on the line below the file and record-type identification specifications.

Write these specifications on the RPG Input Specifications sheet (see Figure 25-1).

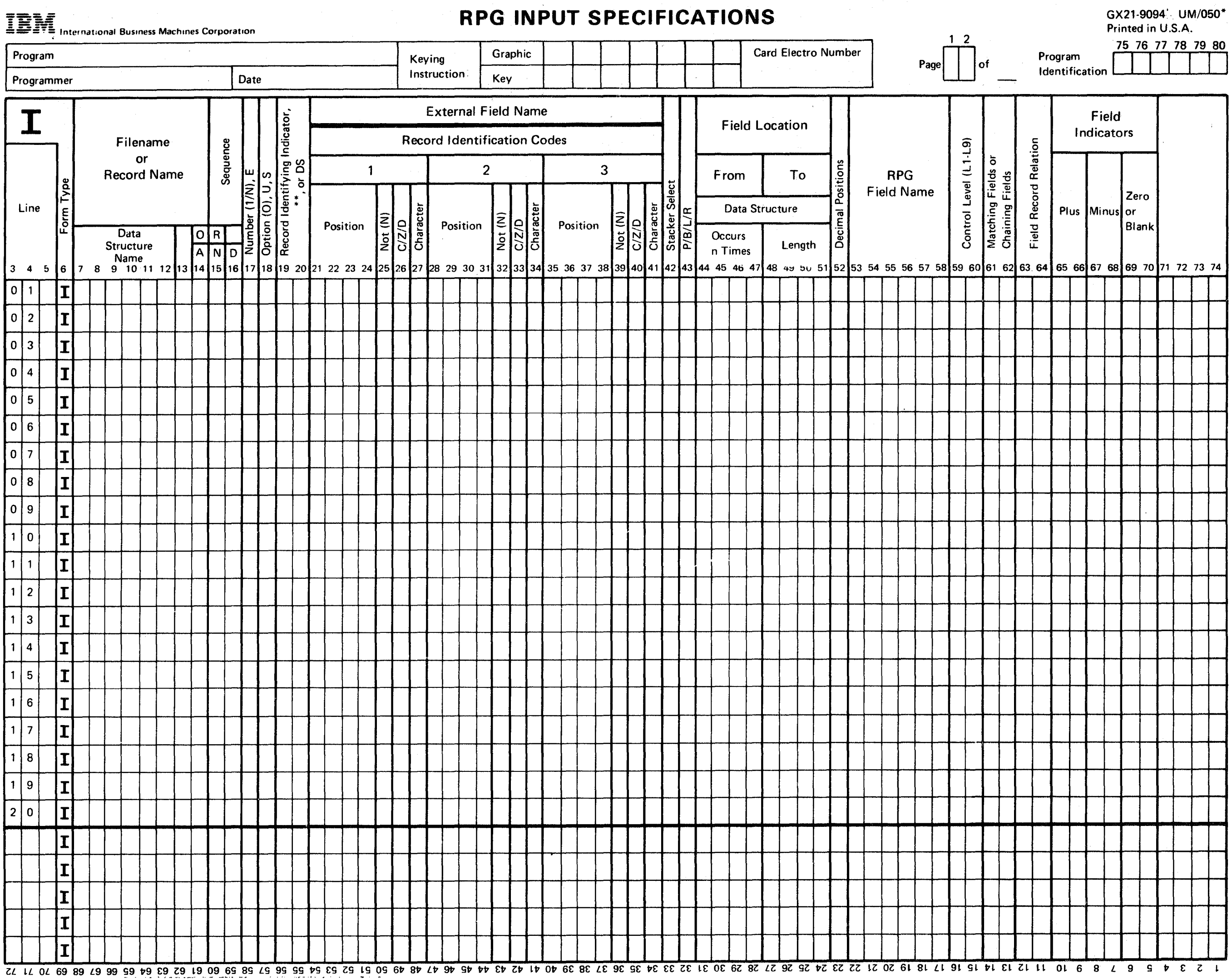


Figure 25-1. RPG Input Specifications

File and Record-Type Identification Entries

Columns 1-2 (Page)

Entry	Explanation
Blank	No page number is used.
01-99	Page number.

Use columns 1 and 2 in the upper right corner of each sheet to number the specifications sheets, in ascending order, for your job. You can use more than one of each type of sheet, but keep all sheets of the same type together.

Columns 3-5 (Line)

Entry	Explanation
Blank	No line number is used.
Any numbers	Line numbers.

Use columns 3 through 5 to number the lines on each page. Columns 3 and 4 are preprinted on each sheet so, in most cases, line numbering is already done.

Page and line numbers are optional entries and are not required to successfully run an RPG program. Columns 1 through 5 are checked for ascending order, and RPG prints an S in the left margin of the RPG listing for any statement that is out of order. If you use SEU to enter the source program, you can request that SEU sequence the specifications in ascending order.

Column 6 (Form Type)

An I must appear in column 6 to identify this line as an input specification.

Column 7 (Comments)

Entry	Explanation
*	Comment line

Use an asterisk in column 7 to identify the line as a comment line. Use comments throughout your program to help document the purpose of a certain section of coding. You can use any character in a comment line. Comments are not instructions to the RPG II program; they only document your program.

Columns 7-12 (/EJECT)

Entry	Explanation
/EJECT	The specifications following this entry are to begin on a new page of the compiler listing.

The /EJECT specification is not printed on the compiler listing.

Columns 7-12 (/TITLE)

Entry	Explanation
/TITLE	The heading information (such as a title or security classification) that follows the /TITLE entry appears at the top of each page of the compiler listing. The heading information is entered in columns 14 through 74.

A program can contain more than one /TITLE statement. Each /TITLE statement provides heading information for the compiler listing until the next /TITLE statement is read. To print on the first page of the compiler listing, a /TITLE statement must be the first statement read. Information specified by the /TITLE statement is printed in addition to compiler heading information.

The /TITLE statement causes an eject to the next page before the title is printed. The /TITLE statement is not printed on the compiler listing.

Columns 7-14 (/SPACE)

Entry	Explanation
/SPACEƀn	Line spacing occurs at this point in the compiler listing. Valid entries for n are 1 to 3. If you do not specify n, 1 is assumed.

One blank (ƀ) must come before the value you specify for n. The value you specify for n indicates the number of blank lines to be spaced before the next specification line is printed. If n is greater than the number of lines remaining on the current page, the next specification line is printed on a new page. If you specify just /SPACE, one line is spaced.

/SPACE is not printed on the compiler listing but is replaced by the actual line spacing. The spacing indicated by /SPACE is in addition to the three blank lines that occur between specification types.

Columns 7-14 (Filename)

Entry	Explanation
A valid filename or data structure name	Same filename that appears on the file description specifications for the input file or the name of a data structure.

If a data structure is specified (DS in columns 19 and 20), columns 7 through 14 can contain:

- Blanks
- A name up to 6 characters long
- A name previously referenced in columns 53 through 58 of the input specifications

Data structure entries must be the last statements on the input specifications.

Columns 14-16

Entry	Explanation
AND or OR	AND/OR indicates a relationship between record-identifying indicators or record types. The entry must begin in column 14.

See *Columns 21-41 (Record Identification Codes)* and *Columns 53-58 (Field Name)* in this chapter for more information on AND/OR relationship.

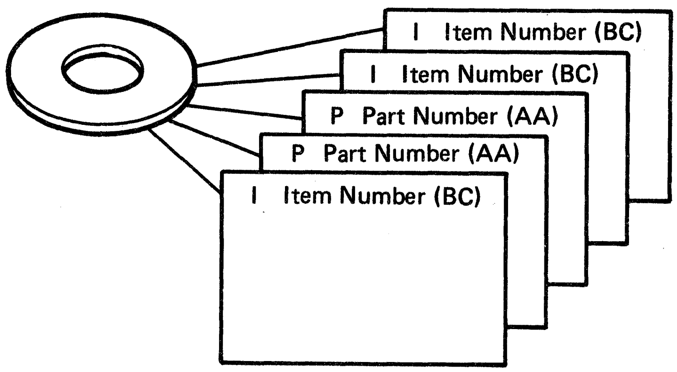
Columns 15-16 (Sequence)

Entry	Explanation
Any two alphabetic characters	Program does not check for special sequence. Alphabetic characters must be used for full-procedural files, chained files, demand files (except CONSOLE demand files), WORKSTN files, and look-ahead records.

Entry	Explanation
01-99	Program checks for special sequence.

Use an entry (01 through 99) in columns 15 and 16 to assign a special sequence number to different record types in a file. The first sequence number must be 01. Gaps in sequence numbers are allowed, but the numbers must be in ascending order.

If the types of records do not need to be in any special order, use two alphabetic characters (see Figure 25-2). Within one file, all record types having alphabetic entries in columns 15 and 16 must be described before those types with numeric entries.



File RECORDA has two types of records (part number and item number) that can appear in any order. Because they are not to be checked for sequencing, they are assigned 2 alphabetic characters in columns 15 and 16 (AA and BC, respectively) instead of numbers.

Line (3-5)	Form Type (6)	Filename or Record Name (7-14)	Sequence (15-16)	Number (1/N), E (17)	Option (O), U, S (18)	Record Identifying Indicator, **, or DS (19-20)	Record Identification Codes 1: Position (21-24)	Not (N) (25)	C/Z/D (26)	Character (27)
01	I	RECORDA	AA			Ø1	1		C	P
02	I									
03	I									
04	I									
05	I									
06	I									
07	I		BC			Ø2	1		C	I
08	I									

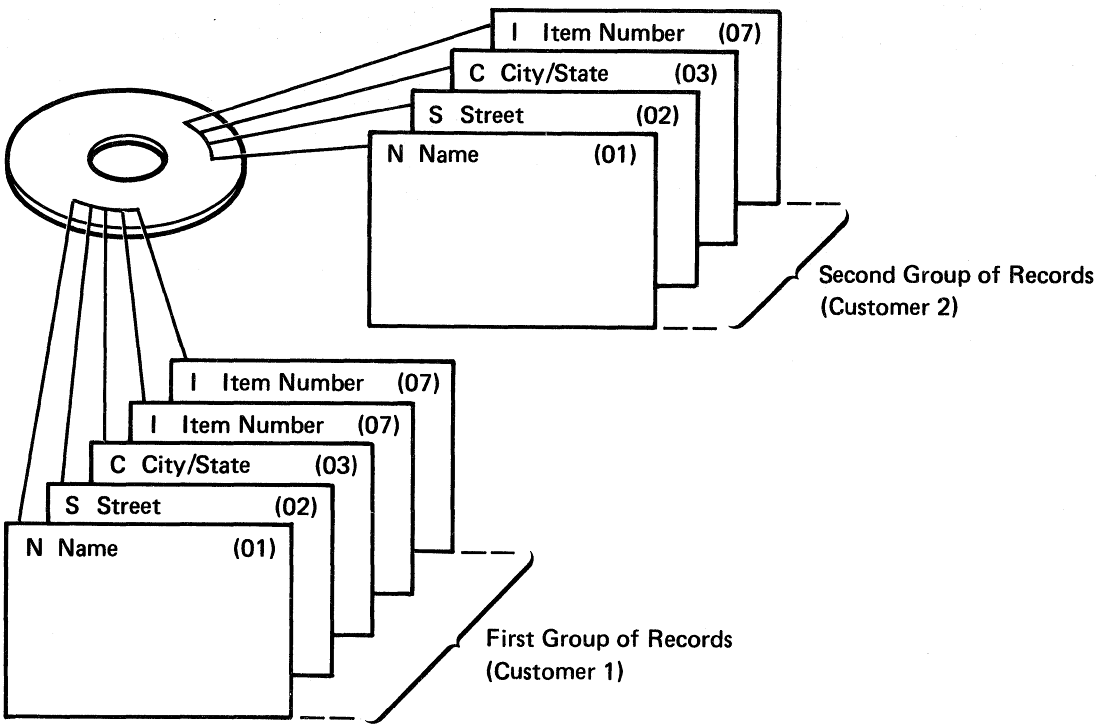
Figure 25-2. Unsequenced Record Types in a File

Assigning Sequence Numbers

Enter a numeric character in columns 15 and 16 if one record type (identified by a record identification code) must be read before another record type in a sequenced group. To specify sequence checking, each record type must have a record identification code, and the record types must be numbered in the order they should appear. The program checks this order as the records are read (see Figure 25-3). If a record type is out of sequence, the program stops and error message RPG-9031, FILE CONTAINS A RECORD NOT IN SEQUENCE, is displayed. You can continue the program by selecting option 0 and pressing an entry function key. The program bypasses the record that caused the halt and reads the next record from the same file.

Sequence numbers make sure that all records of the lowest record type come before the records of the next highest record type. The sequence numbers do not make sure that records within a record type are in any certain order. Sequence numbers are unrelated to control levels and do not provide for checking data in fields of a record for a special sequence (see Figure 25-4). Use columns 61 and 62 to indicate that data in fields of a record be checked for a special sequence.

Records in an OR or AND line cannot have a sequence entry in these columns. The entry in columns 15 and 16 on the previous line also applies to the OR or AND line.



This file contains four different kinds of records. The records are arranged in groups according to a customer name control field. The name record is first in each group and is assigned sequence number 01. Street record is next and is assigned 02. City/state record is 03. (Remember, gaps are allowed). Item number record is 07. More than one item number record can be present (N in column 17).

Line (3-5)	Form Type (6)	Filename or Record Name (7-14)	Sequence (15-16)	Number (1/N), E (17)	Option (O), U, S (18)	Record Identifying Indicator, **, or DS (19-20)	Record Identification Codes 1: Position (21-24)	Not (N) (25)	C/Z/D (26)	Character (27)	Field Location From (44-47)	Field Location To (48-51)	Decimal Positions (52)	RPG Field Name (53-58)	Control Level (L1-L9) (59-60)	Matching Fields or Chaining Fields (61-62)	Field Record Relation (63-64)	Field Indicators Plus (65-66)	Minus (67-68)	Zero or Blank (69-70)
01	I	CUST	Ø1	1			1		C	N										
02	I										5	25		NAME						
03	I		Ø2	1	O		1		C	S										
04	I										2	26		STREET						
05	I		Ø3	1			1		C	C										
06	I										5	21		CTYST						
07	I		Ø7	N	O		1		C	I										
08	I										1Ø	16		ITEM						
09	I										18	23		QTY						
10	I																			

Figure 25-3. Sequence Checking of Record Type

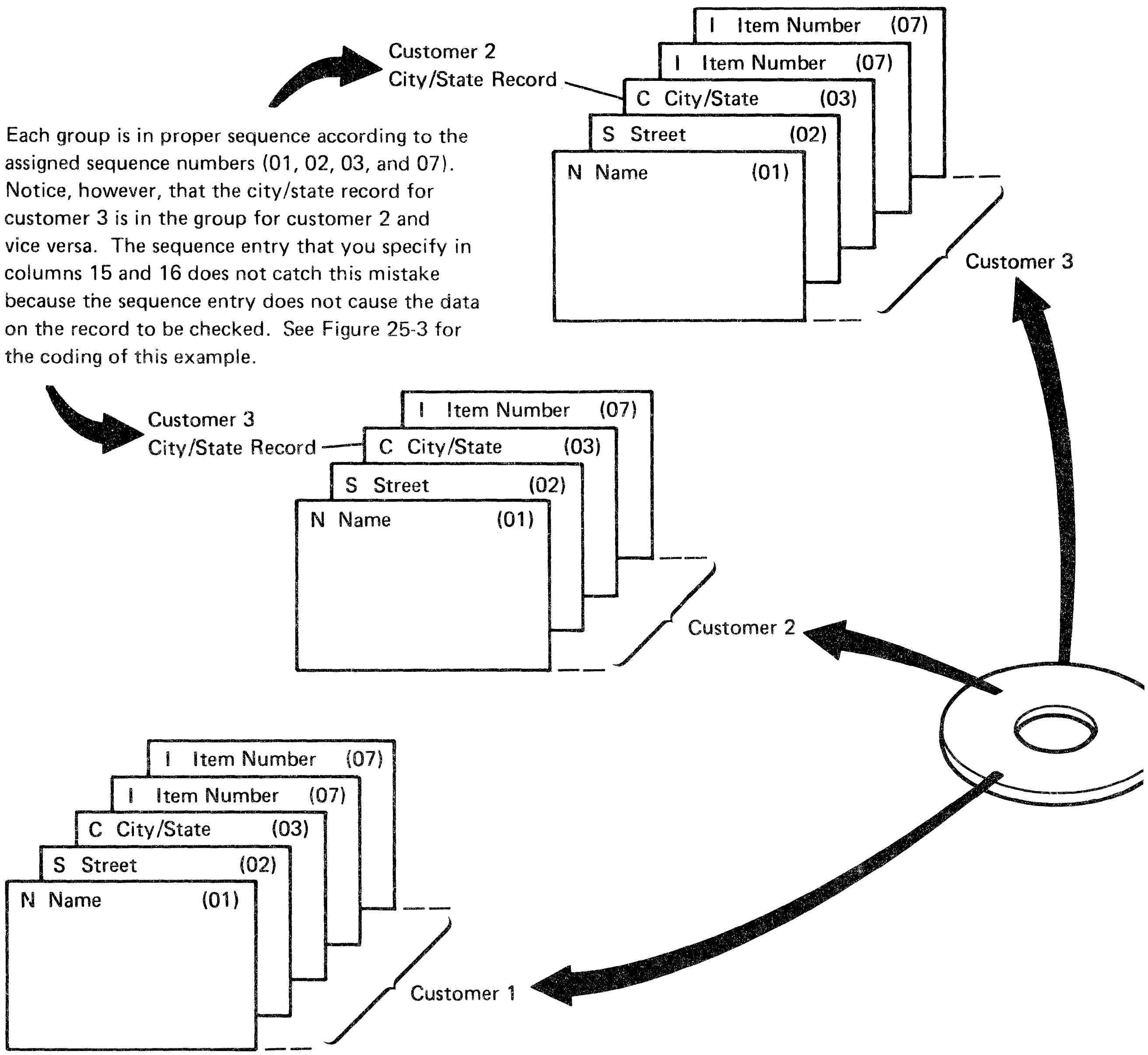


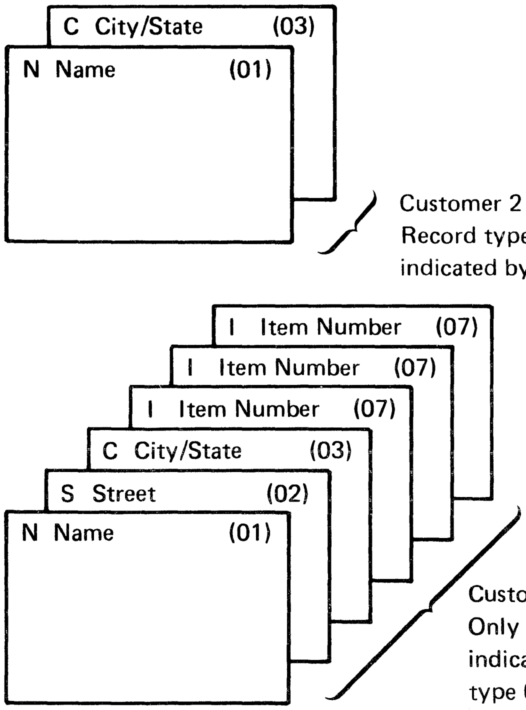
Figure 25-4. Correct Record Sequence (Incorrect Data within Groups)

Column 17 (Number)

Entry	Explanation
Blank	Program does not check record types for a special sequence (columns 15 and 16 have alphabetic entries).
1	Only one record of this type can be present in the sequenced group.
N	One or more records of this type can be present in the sequenced group.

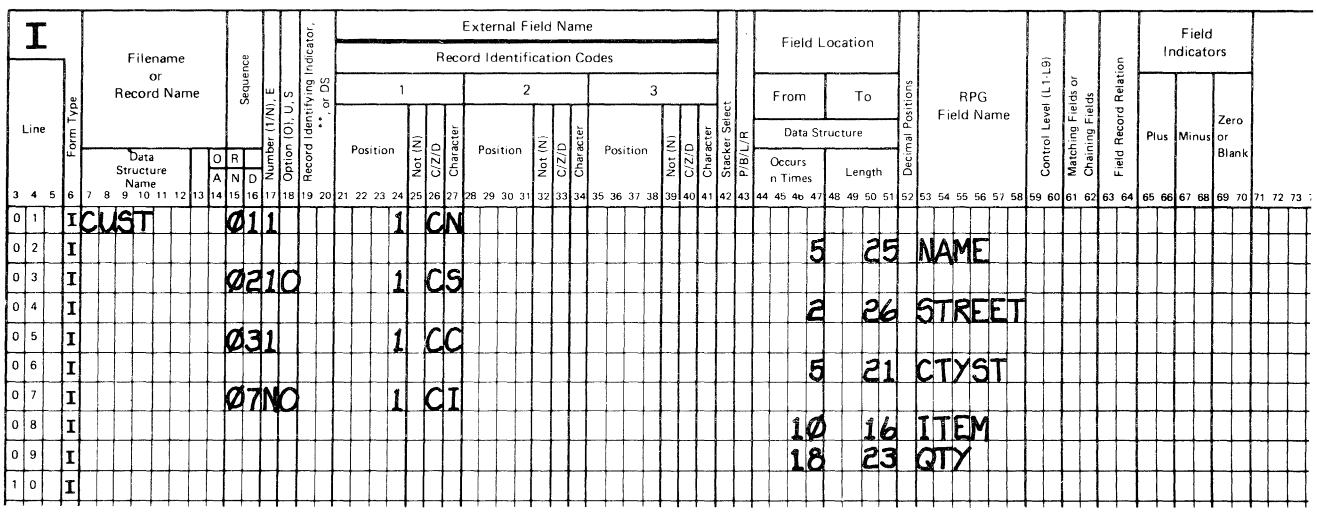
Use column 17 only if columns 15 and 16 contain a numeric entry specifying sequence checking (see Figure 25-5).

OR lines (columns 14 and 15 contain OR) and AND lines (columns 14 through 16 contain AND) should not have an entry in this column. The entry in column 17 on the previous line also applies to the OR or AND line. See *Columns 53-58 (Field Name)* in this chapter for more information on OR lines.



Customer 2
Record types 02 and 07 are optional as indicated by O in column 18.

Customer 1
Only one record of types 01, 02, and 03 can be present as indicated by 1 in column 17; however, any number of record type 07 can be present as indicated by N in column 17.



Line	Form Type	Filename or Record Name	Sequence	Number (1/N),E	Option (O),U,S	Position	Not (N)	C/Z/D	Character	From	To	RPG Field Name
01	I	CUST	Ø1	1		1		C	N			
02	I									5	25	NAME
03	I		Ø2	1	O	1		C	S			
04	I									2	26	STREET
05	I		Ø3	1		1		C	C			
06	I									5	21	CTYST
07	I		Ø7	N	O	1		C	I			
08	I									1Ø	16	ITEM
09	I									18	23	QTY
10	I											

Figure 25-5. Sequenced Record File

Column 18 (Option)

Entry	Explanation
Blank	Record type must be present if sequence checking is specified.
O	Record type is optional (that is, it may or may not be present) if sequence checking is specified.
U	The program uses the data structure defined on this specification line as a display station local data area.

Use column 18 only if columns 15 and 16 contain a numeric entry specifying sequence checking, or if the data structure defined on the following specification line is used as a display station local data area.

If sequence checking is specified and all record types are optional, no sequence error is found.

OR and AND lines should not have an entry in this column. The entry in column 18 on the previous line also applies to the OR or AND line. See *Columns 53-58 (Field Name)* in this chapter for more information on OR lines.

Columns 19-20 (Record-Identifying Indicator, **, DS)

Entry	Explanation
01-10	Record-identifying indicator for CONSOLE files. Record-identifying indicators 01 through 10 for CONSOLE files correspond to command keys 1 through 10.
01-99	Record-identifying indicator.
L1-L9	Control-level indicator used for a record-identifying indicator when a record type rather than a control field signals the start of a new control group.
LR	Last-record indicator.
H1-H9	Halt indicator used for a record-identifying indicator when checking for a record type that causes an error condition.
**	Look-ahead field. Look-ahead can be used only with input or update files; however, these files cannot be full-procedural, chained, or demand files. Look-ahead fields are not valid with CONSOLE files or WORKSTN files.

DS Data structure. A data structure is considered to be alphameric data and can be from 1 to 9999 characters in length. Data structure entries must be the last entries on the input specifications. For more information about data structures, see Chapter 14.

Look-Ahead

A look-ahead field allows you to:

- Determine when the last record of a control group is being processed
- Extend the RPG matching-record capability

Because an RPG program processes one record at a time, normally only the information from the record being processed is available for use. However, look-ahead allows information to be made available from records that follow the one currently being processed. This information can then be used to determine what operation should be done next.

Any or all of the fields in a file can be described as look-ahead fields. The description applies to all records in the file regardless of their type. Look-ahead fields can be described before or after the field descriptions for any of the records in the file. The line that signals that look-ahead fields are to be described must contain an alphabetic entry in columns 15 and 16 and must contain ** in column 19 and 20. All the other columns must be blank. Remember that specifications with an alphabetic sequence in columns 15 and 16 must precede specifications with a numeric sequence in columns 15 and 16.

Look-ahead fields are described on the lines immediately following the line that contains ** in column 19 and 20 (see Figure 25-6). Make the following entries for each look-ahead field description line:

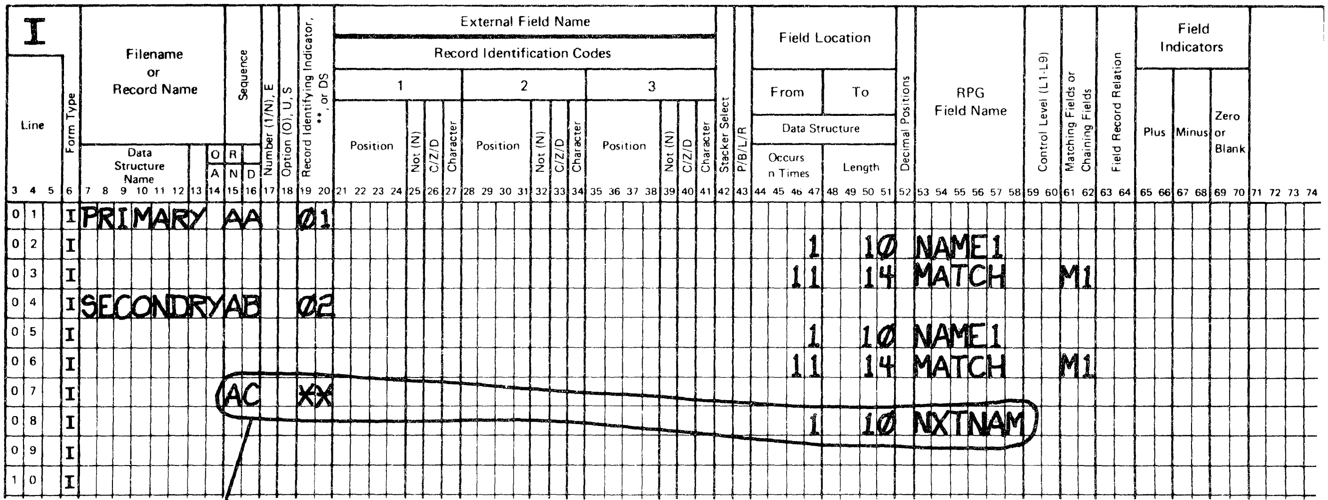
- *Columns 44-51*: Identify the record positions in which the field is located.
- *Column 52*: If the field is numeric, enter the number of digits to the right of the decimal point in column 52. If there are no decimal positions, enter a 0. If the field is alphameric, leave this column blank.
- *Columns 53-58*: Enter the name of the look-ahead field. If the field is also one of the normal fields in the record, use a different name for the look-ahead field.

Line	Form Type	Filename	File Type (I/O/U/C/D)	File Designation (P/S/C/R/T/D/F)	End of File (E)	Sequence (A/D)	File Format (F/V/S/M/D/E)	Block Length	Record Length	Device
02	F	PRIMARY	U	P			F	96	96	DISK
03	F	SECONDRY	I	S	E		F	96	96	DISK
04	F									

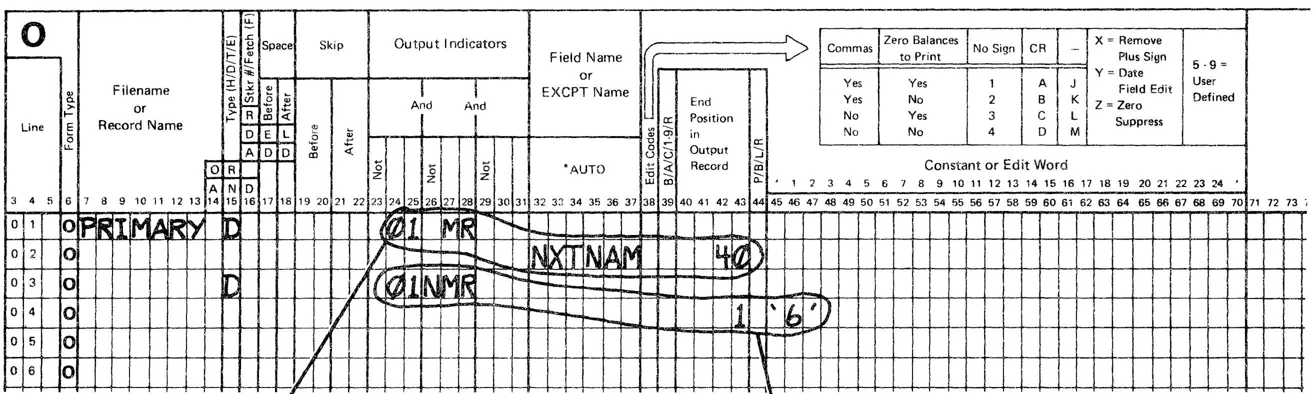
The program reads records from two disk files. The primary file is named PRIMARY; the secondary file, SECONDRY. If a record from the primary file matches one from the secondary file, the information in positions 1 through 10 of the secondary file record is placed in positions 31 through 40 of the primary file record. When there is no match, a 6 is placed in position 1 of the primary file record. The 6 indicates an unmatched record in the primary file.

Because the primary file record is processed first when it matches a secondary file record, the information from the secondary file can be made available only by a look-ahead field.

Figure 25-6 (Part 1 of 2). Look-Ahead Fields



Look-Ahead Field (field from secondary file records needed in primary file records)



Place the look-ahead field from secondary records into positions 31 through 40 of the primary record if the two records match.

Place a 6 in position 1 of the primary record if the re[cord] matches no secondary record.

Figure 25-6 (Part 2 of 2). Look-Ahead Fields

For input files, look-ahead fields always apply to the next record in the file, provided the file is not an update file. Thus, if the information is used both before and after the record is selected for processing, describe the field twice, once as a look-ahead field and once as a normal field. See Figure 25-7 for an example of how records are selected for processing from two input files when look-ahead fields are used.

For update files, the look-ahead fields apply to the next record in the file only if the record currently selected for processing was read from another file.

Therefore, when the program is reading from only one file and that file is an update file, look-ahead fields always apply to the current record and contain the same information as a normal field. See Figure 25-8 for an example of how records are selected for processing from an update file and an input file when look-ahead fields are used.

As the last record from a file is processed, every look-ahead field for the file is automatically filled with 9s. For example, a look-ahead field that is 3 characters long will contain 999. The 9s remain in the field until the job ends. The blank-after option (B in column 39 of the output specifications) cannot be used with look-ahead fields.

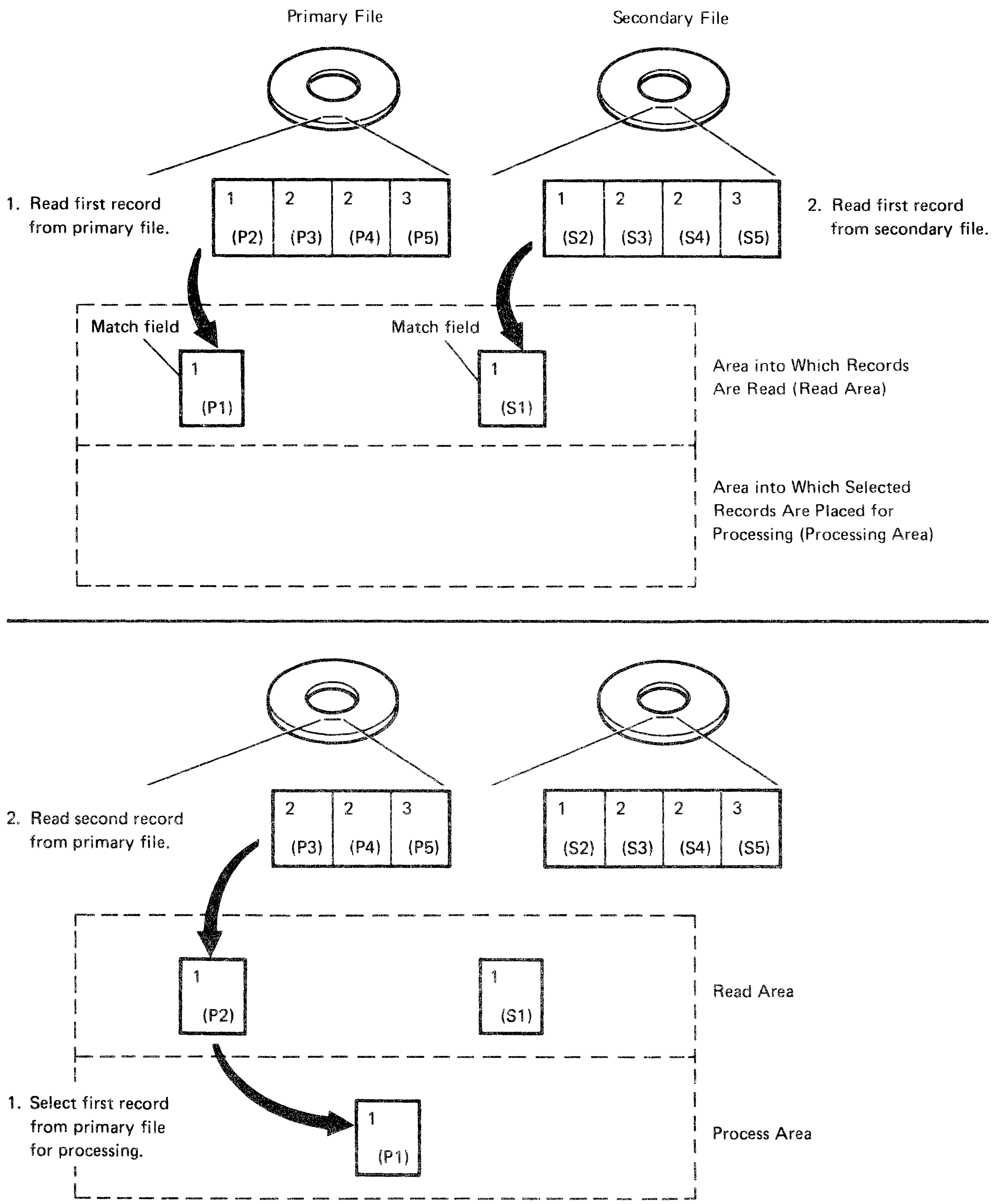


Figure 25-7 (Part 1 of 3). Available Records: Two Input Files

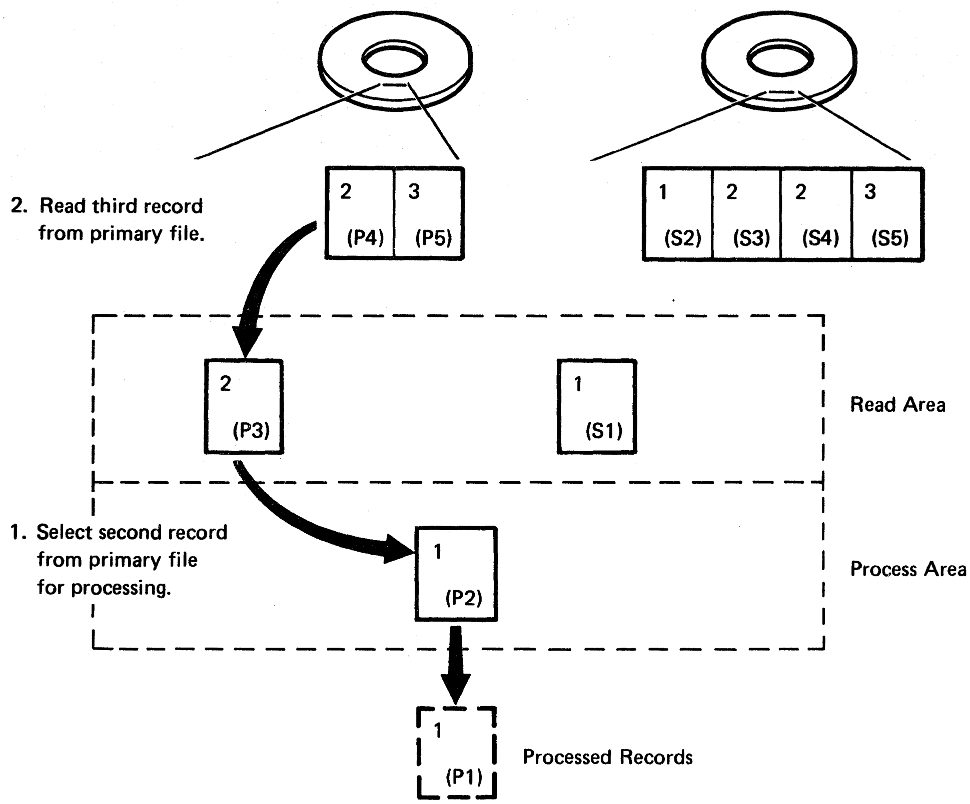
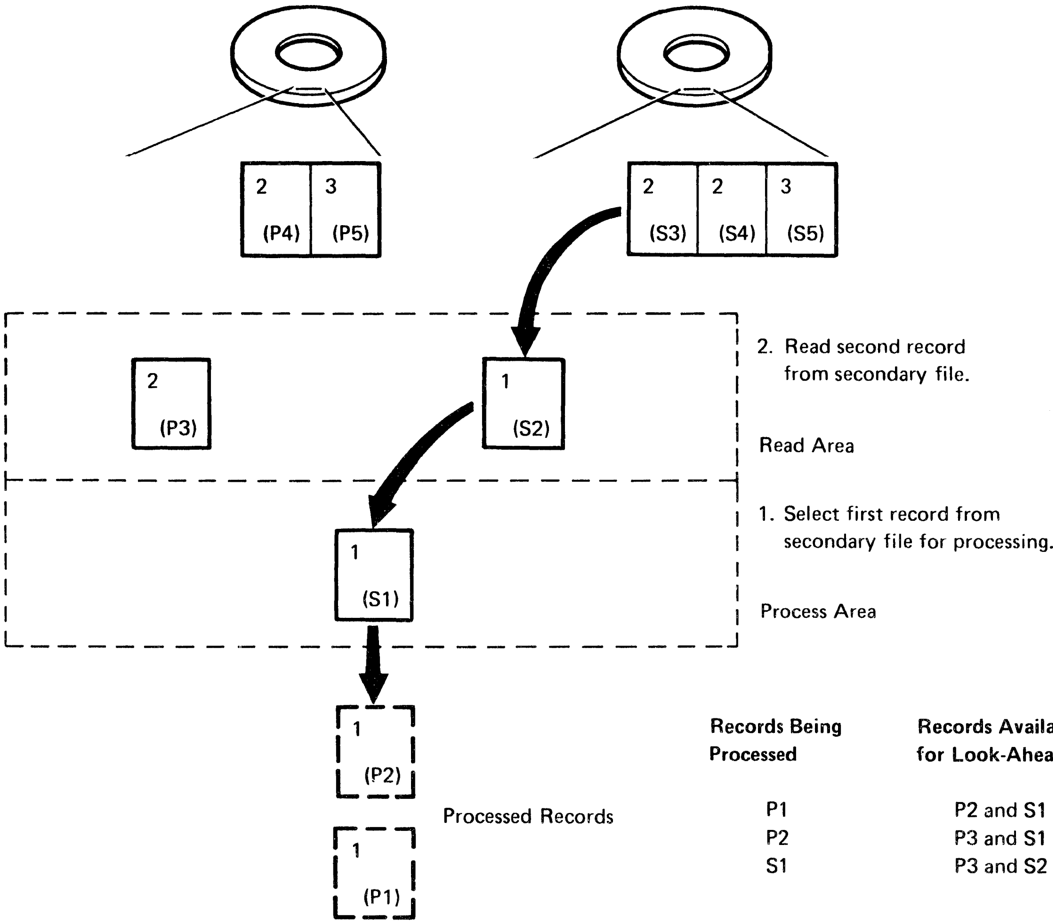


Figure 25-7 (Part 2 of 3). Available Records: Two Input Files



Records Being Processed	Records Available for Look-Ahead
P1	P2 and S1
P2	P3 and S1
S1	P3 and S2

Figure 25-7 (Part 3 of 3). Available Records: Two Input Files

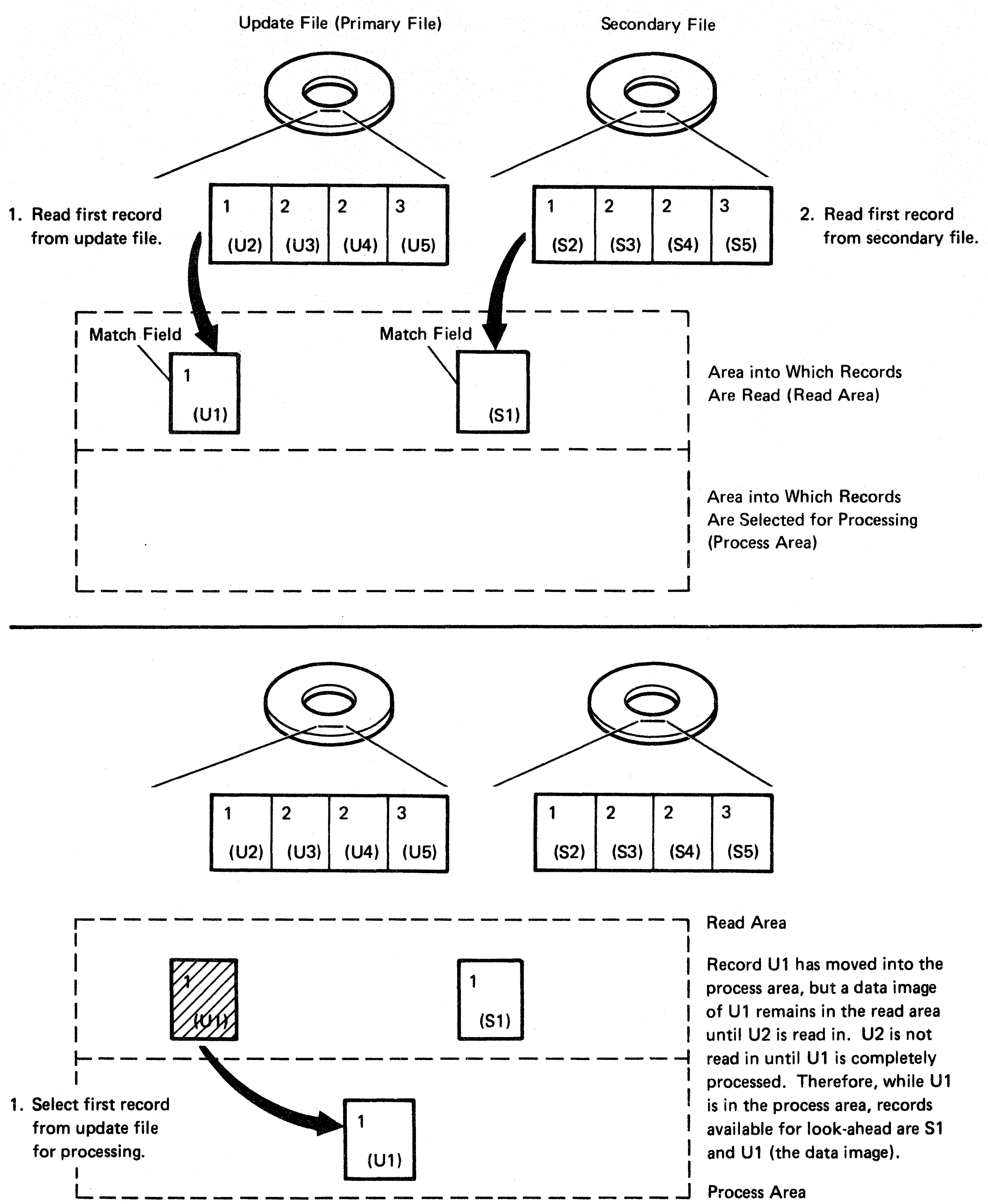


Figure 25-8 (Part 1 of 3). Available Records: One Input File, One Update File

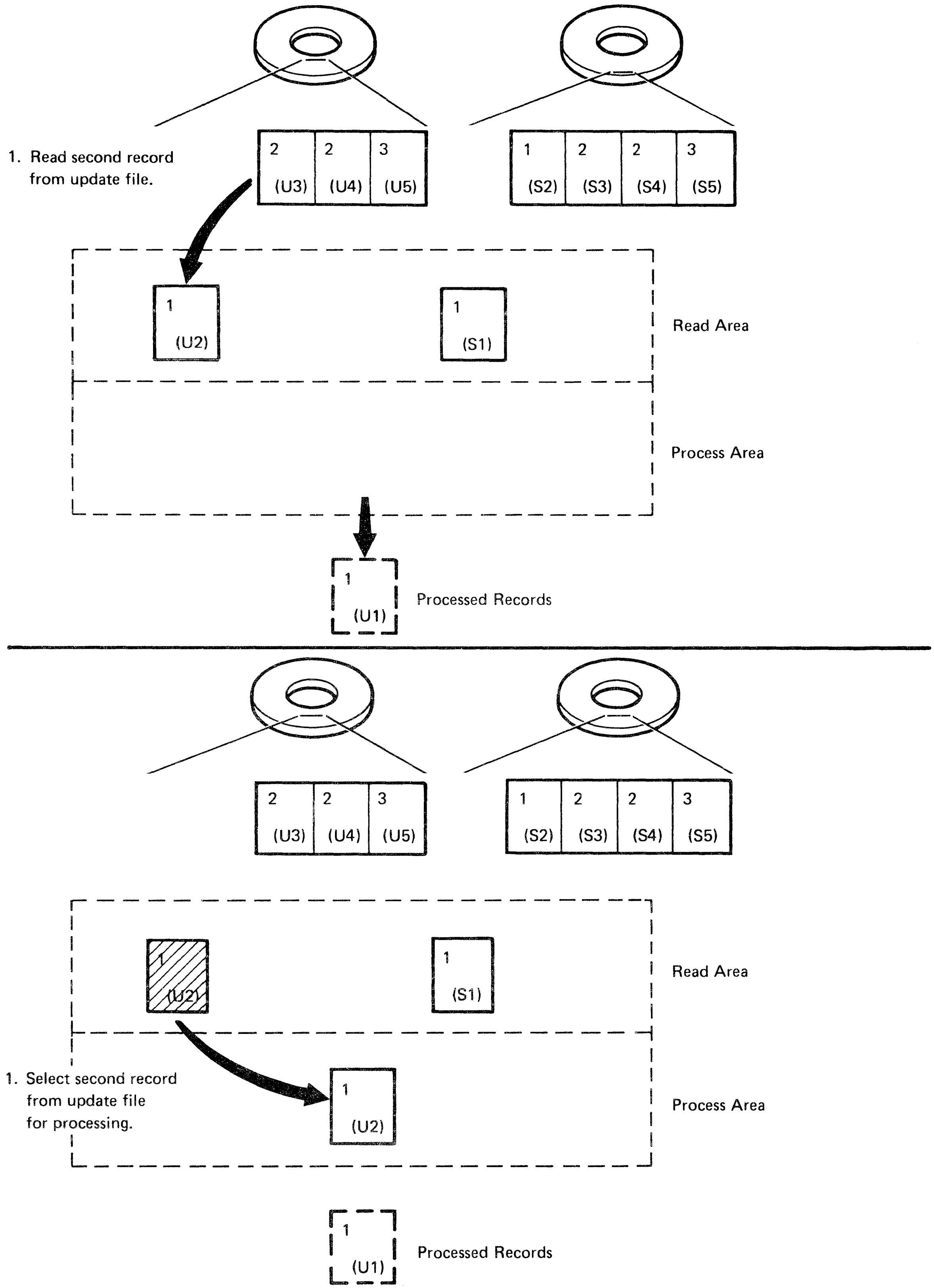
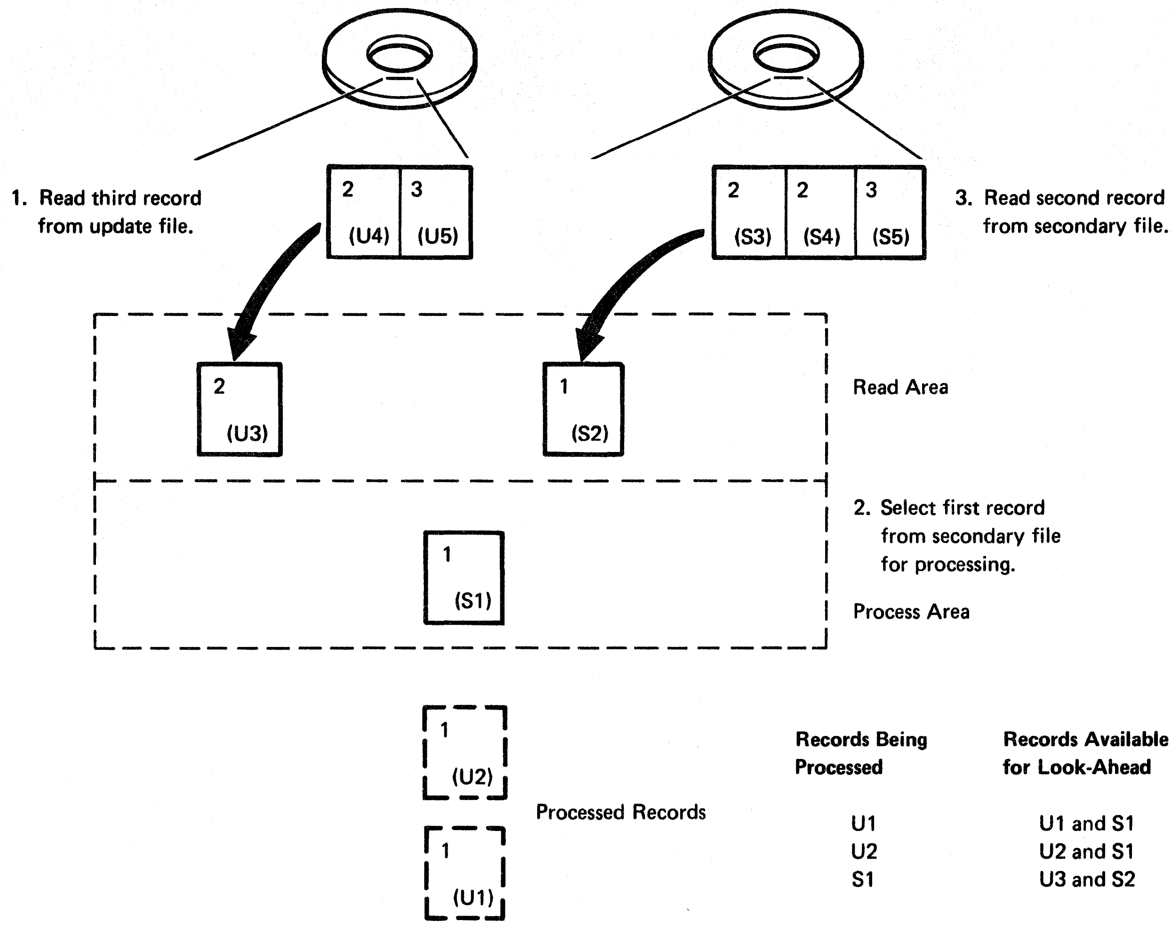


Figure 25-8 (Part 2 of 3). Available Records: One Input File, One Update File



Records Being Processed	Records Available for Look-Ahead
U1	U1 and S1
U2	U2 and S1
S1	U3 and S2

Figure 25-8 (Part 3 of 3). Available Records: One Input File, One Update File

Columns 21-41 (Record Identification Codes)

Use columns 21 through 41 to describe the information that identifies a record type. If all records are to be processed alike regardless of their type, or if there is only one record type, leave columns 21 through 41 blank.

Note: Only columns 21 through 34 are valid for CONSOLE files (see Chapter 9, *Using a CONSOLE, KEYBORD, or CRT File*, for more information).

When one file contains more than one record type, each record type is identified by a code consisting of a character or a combination of characters in certain positions in the record. If different operations are to be performed for each record type, this code must be described in columns 21 through 41 so that the program can determine the type of record selected for processing. Only one type of record is selected for processing during a program cycle, and the record-identifying indicator for that record turns on at the time of selection.

Seven columns are used for the description of one character in the record identification code. Each specification line contains three sets of seven columns: columns 21 through 27, 28 through 34, and 35 through 41. Each set consists of four fields: Position, Not, C/Z/D, and Character. Coding is the same for all three sets.

Note: Any record that is read by the system and is not described by a record identification code in columns 21 through 41 causes the program to halt. The person using the display station can continue, however, by selecting the appropriate option. The record that causes the halt is not processed, and the next record in that file is read.

Position (Columns 21-24, 28-31, and 35-38)

Entry	Explanation
Blank	No record identification code is needed.
1-4096	Record position of one character in the record identification code.

Use these columns to give the location in the record of every character in the identification code. These entries must end in columns 24, 31, and 38 respectively.

Not (N) (Columns 25, 32, and 39)

Entry	Explanation
Blank	Character is present in the specified record position.
N	Character should not be present in the specified record position (not valid for CONSOLE files; see Chapter 9).

Use these columns to indicate that a certain character should not be present in a specified position.

C/Z/D (Columns 26, 33, and 40)

Entry	Explanation
C	Entire character. C must be used for CONSOLE files (see Chapter 9).
Z	Zone portion of character.
D	Digit portion of character.

Use these columns to indicate what portion of a character is used as part of the record identification code. Only the zone portion, only the digit portion, or both portions (the whole character) can be used (see Figure 25-9). When establishing record identification codes, remember that many characters have either the same zone or the same digit portion. For a list of characters that have identical zone or digit portions, see Figure 25-10.

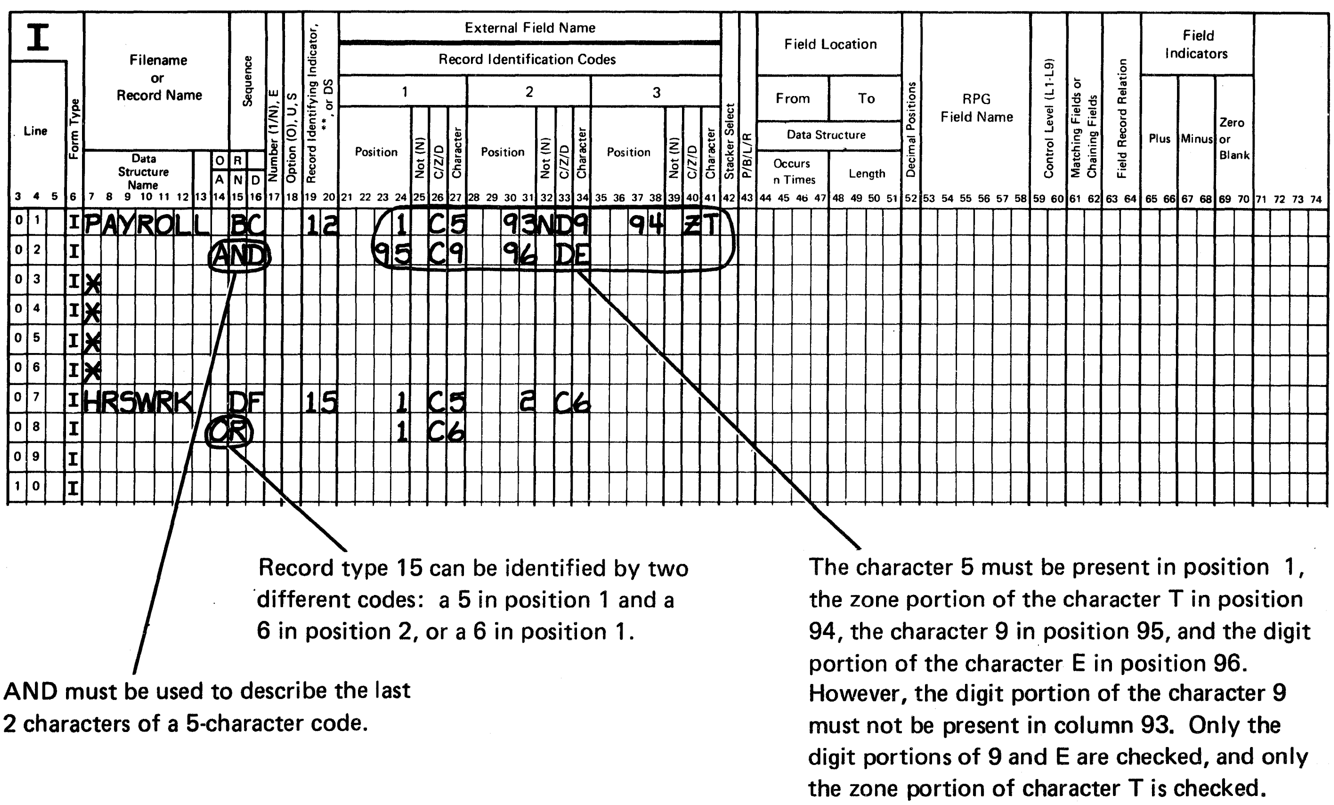


Figure 25-9. Record Identification Codes

Character Grouping by Zone (Z)		
Zone 4	**Zone 8**	**Zone D**
blank	a	- (minus)
¢	b	}
.	c	J
<	d	K
(	e	L
+	f	M
|	g	N
	h	O
	i	P
		Q
		R
Zone 5	**Zone 9**	**Zone E**
&	j	\
!	k	S
$	l	T
*	m	U
)	n	V
;	o	W
¬	p	X
	q	Y
	r	Z
Zone 6	**Zone A**	**Zone F**
- (minus)	~	blank
/	s	0
¦	t	1
, (comma)	u	2
%	v	3
_ (underscore)	w	4
>	x	5
?	y	6
	z	7
		8
		9
Zone 7	**Zone C**	
`	&	
:	{	
#	A	
@	B	
' (apostrophe)	C	
=	D	
"	E	
	F	
	G	
	H	
	I	

Character Grouping by Digit (D)		
Digit 0	**Digit 6**	**Digit C**
blank	f	<
&	o	*
- (minus)	w	%
}	F	@
{	O	
\	W	
0	6	
Digit 1	**Digit 7**	**Digit D**
/	g	(
a	p	)
j	x	_ (underscore)
~	G	' (apostrophe)
A	P	
J	X	
1	7	
Digit 2	**Digit 8**	**Digit E**
b	h	+
k	q	;
s	y	>
B	H	=
K	Q	
S	Y	
2	8	
Digit 3	**Digit 9**	**Digit F**
c	i	|
l	r	¬
t	z	?
C	I	"
L	R	
T	Z	
3	9	
Digit 4	**Digit A**	
d	¢	
m	!	
u	¦	
D	:	
M		
U		
4		
Digit 5	**Digit B**	
e	.	
n	$	
v	,	
E	#	
N		
V		
5		

Figure 25-10. Characters Interpreted as Having the Same Zone or Digit

Character (Columns 27, 34, and 41)

In these columns, enter the alphabetic character, special character, or numeric character that is used in the record as the identification code or part of the code.

Character Grouping by Zone or Digit

When characters are used for record identification purposes on a digit or zone only basis, all characters having the same zone or digit are selected by the system as meeting record identification requirements. When a character is read into the system, it is converted into an 8-bit code. The program tests this 8-bit code to see whether the character meets the requirements of the record identifying character in the input specifications.

Figure 25-10 lists the characters that have identical zones or digits. For example, if column 26 contains D, which specifies digit only, and column 27 contains A, all records having a slash (/), A, J, or 1 in the specified column are selected as having the correct record identification code. If column 26 contains Z and column 27 contains A, all records containing & or A through I are selected as having the correct code.

The following three special cases are exceptions:

- The hexadecimal representation of an & (ampersand) is 50. However, when the ampersand is coded in the character entry, it is treated as though its hexadecimal representation were C0, that is, as if it had the same zone as the characters A through I. An ampersand in the input data satisfies two zone checks, for either a hexadecimal 5 zone or a hexadecimal C zone.

- The hexadecimal representation of a - (minus sign) is 60. However, when the minus sign is coded in the character entry, it is treated as though its hexadecimal representation were D0, that is, as if it had the same zone as the characters J through R. A minus sign in the input data satisfies two zone checks, for either a hexadecimal 6 zone or a hexadecimal D zone.

- The hexadecimal representation of a blank is 40. However, when the blank is coded in the character entry, it is treated as though its hexadecimal representation were F0, that is, as if it had the same zone as the numeric characters 0 through 9. A blank in the input data satisfies two zone checks, for either a hexadecimal 4 zone or a hexadecimal F zone.

AND Relationship

A maximum of three identifying characters can be described in one specification line. If the identification code consists of more than 3 characters, an AND line must be used to describe the additional characters. Write the word AND in columns 14 through 16 to indicate an AND line (see Figure 25-9).

Any number of AND lines can be used to describe the record identification code for a record sequence. The record must contain all the characters indicated as its record identification code before the record-identifying indicator turns on. AND lines are not allowed on CONSOLE files used for interactive data entry.

OR Relationship

If a particular record type can be identified by two different codes, OR lines must be used to indicate that either of the codes can be present to identify the record. A maximum of 20 OR lines can appear for each record sequence. Write the word OR in columns 14 and 15 to indicate an OR line (see Figure 25-9).

Note: If AND lines and OR lines are combined the total number of OR lines for one record sequence cannot exceed 20 and any number of AND lines can be used.

Column 42

Column 42 is not used. Leave it blank.

Field Description Entries

The field description entries (columns 43 through 74) must begin one line below the file and record identification entries (columns 7 through 42) for each file.

Column 43 (Packed-Decimal or Binary Field)

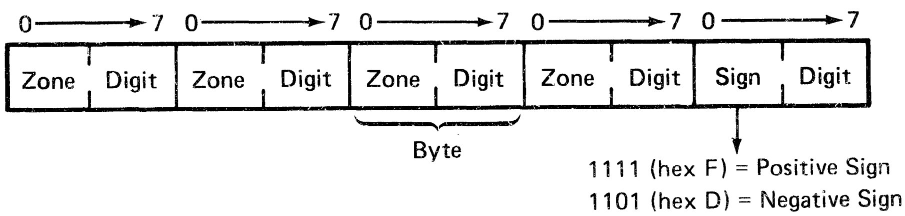
Entry	Explanation
Blank	Field is in zoned-decimal format or is alphameric. (This column must be blank for CONSOLE files.)
P	Field named in columns 53 through 58 is in packed-decimal format.
B	Field named in columns 53 through 58 is in binary format.

Use column 43 to indicate that a numeric field is in packed-decimal or binary format. DISK files support packed-decimal or binary fields for read or write operations. Numeric data fields in packed-decimal or binary format are converted by the system to the zoned-decimal format before they are processed. This conversion ignores decimal points.

Packed or binary input to arrays should have a P or B in this column. The from and to columns should then define the positions the array occupies in the record in the packed or binary format. The zoned-decimal length of each array element is defined on the extension specifications.

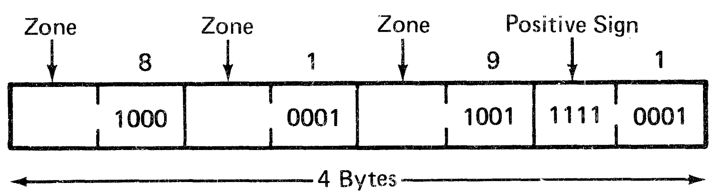
Zoned-Decimal Format (Blank)

Zoned-decimal format means that each byte can contain 1 character. That character can be a decimal number or an alphabetic or special character. In the zoned-decimal format, each byte is divided into a 4-bit zone portion and a 4-bit digit portion. The zoned-decimal format looks like this:



Note: RPG does not perform data verification on numeric data. The value of the digit portion of a character is assumed to be the numeric value of that character.

The zone portion of the low-order byte indicates whether the decimal number is positive or negative. A positive value is indicated by a hexadecimal F, and a negative value is indicated by a hexadecimal D. In zoned-decimal format, each digit in a decimal number includes a zone portion; however, only the low-order zone portion serves as the sign. The decimal number 8191 looks like this in zoned-decimal format:

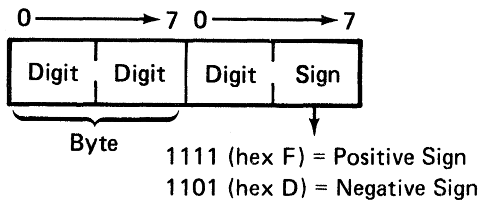


For more efficient use of disk storage, you may want to enter your numeric data (decimal numbers) in packed-decimal or binary format. However, you will have to convert your data to zoned-decimal format before it can be processed.

Packed-Decimal Format (P)

Packed-decimal format means that each byte (except for the low-order byte) can contain two digits. Because many of the fields in a DISK file contain decimal numbers, you can conserve storage by using the packed-decimal format.

In the packed decimal format, each byte, except the low-order byte, is divided into two 4-bit digit portions. The rightmost portion of the low-order byte contains the sign (plus or minus) for that field. The packed-decimal format looks like this:



The sign portion of the low-order byte indicates whether the numeric value represented in the digit portions is positive or negative. A positive value is indicated by a hexadecimal F, and a negative value is indicated by a hexadecimal D. In the packed-decimal format, the sign is included for each decimal number; however, the zone portion is not given for each digit in the number. Compare how the decimal number 8191 is represented in packed-decimal format with its zoned-decimal representation shown before (see Figure 25-11).

Because processing requires the zoned-decimal format once it is inside the computer, you must indicate when input fields are in another format. Entering a P in column 43 indicates that the input field is in the packed-decimal format and that the system must convert this field to the required zoned-decimal format.

When a packed-decimal field is converted to a zoned-decimal field, the zoned-decimal field always contains an odd number of bytes. If a zoned-decimal field with an even number of bytes is converted to a packed-decimal field and then converted back to a zoned-decimal field, the resulting zoned-decimal field also contains an odd number of bytes.

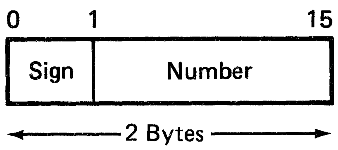
Packed-decimal fields can be up to 8 bytes long. The following chart shows the packed-decimal equivalents for zoned-decimal fields up to 15 bytes long, which is the maximum length.

Zoned-Decimal Length in Bytes	**Packed-Decimal Length in Bytes**
15 14	8
13 12	7
11 10	6
9 8	5
7 6	4
5 4	3
3 2	2
1	1

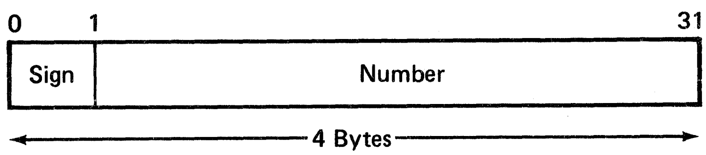
Binary Format (B)

Binary format means that 2 bytes can contain a 4-digit number, and that 4 bytes can contain a 9-digit number. The binary format allows you to save even more disk storage space than you can save using the packed-decimal format. In the binary format, each field on disk must be either 2 or 4 bytes long.

Each 2-byte binary field consists of a 1-bit sign followed by a 15-bit numeric value. In binary format, a decimal number as high as 9999 requires only 2 bytes of disk storage. For each 2-byte binary field, the RPG compiler automatically sets aside 4 bytes of storage to accommodate the field when it is unpacked. A 2-byte field in binary format looks like this:



Each 4-byte binary field consists of a 1-bit sign followed by a 31-bit numeric value. In binary format, a decimal number as high as 999 999 999 requires only 4 bytes of disk storage. For each 4-byte binary field stored on disk, the RPG compiler automatically sets aside 9 bytes of storage to accommodate the field when it is converted. A 4-byte field in binary format looks like this:



In each case, the sign portion of the high-order byte indicates whether the numeric value is positive (sign bit off) or negative (sign bit on). Positive numbers are represented in true binary notation with a 0 bit in the sign position. Negative numbers are represented in twos-complement notation with a 1 bit in the sign position. The bits between the sign position and the leftmost significant bit of the integer are always the same as the sign bit. When the number is positive, all bits to the left of the most significant bit, including the sign bit, are 0′s. When the number is negative, all bits to the left of the most significant bit, including the sign bit, are 1′s. Notice that, in the binary format, the zone position of the decimal number is not given. Compare how the decimal number 8191 is represented in binary format with packed-decimal and zoned-decimal representation (see Figure 25-11).

Because processing requires the zoned-decimal format once it is inside the computer, you must indicate when input fields are in another format. Entering a B in column 43 indicates that the input field is in the binary format and that the system must convert this field to the required zoned-decimal format.

Note: Although packed-decimal and binary fields require less disk storage space, the conversion routines needed to handle such data increase the program size.

Binary Format

Positive Sign	4096 +	2048 +	1024 +	512 +	256 +	128 +	64 +	32 +	16 +	8 +	4 +	2 +	1	= 8191[1]
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

2 Bytes

Packed-Decimal Format

0	8	1	9	1	Positive Sign
0000	1000	0001	1001	0001	1111

3 Bytes

Zoned-Decimal Format[2]

Zone		Zone	8	Zone	1	Zone	9	Positive Sign	1
	0000		1000		0001		1001	1111	0001

5 Bytes

[1] To obtain the numeric value of a positive binary number, add the values of the bits that are on (1); the sign bit is not included. To obtain the numeric value of a negative binary number, add the values of the bits that are off (0) plus one; the sign bit is not included (twos-complement notation).

[2] If 8191 is read into storage as a zoned-decimal field, it occupies 4 bytes. However, if it is converted to packed-decimal format, it occupies 3 bytes; then when it is converted back to zoned-decimal format, it occupies 5 bytes.

Figure 25-11. Binary, Packed-Decimal, and Zoned-Decimal Representation of 8191

Columns 44-51 (Field Location)

Entry	Explanation
1-9999	Beginning of a field (from) or end of a field (to). See Chapter 9 for CONSOLE file considerations. For a WORKSTN file, the from and to positions refer to the location of the fields in the input record and not to their location in the display format.

Use columns 44 through 51 to describe the location on the record of the field named in columns 53 through 58. Enter the number of the record position in which the field begins in columns 44 through 47. Enter the number of the record position in which the field ends in columns 48 through 51. The entries must end in columns 47 and 51. Leading zeros can be omitted.

Define a single-position field by entering the same number in both the from (columns 44 through 47) and to (columns 48 through 51) positions. If a field of more than one position is defined, the number entered in columns 44 through 47 must be smaller than the number entered in columns 48 through 51.

The maximum field length for a zoned-decimal numeric field is 15 positions (8 if the field is packed-decimal and 4 if it is binary). The maximum field length for an alphameric field is 256 characters, and the maximum length for a data structure is 9999 characters.

Column 52 (Decimal Positions)

Entry	Explanation
Blank	Alphameric field
0-9	Number of decimal positions in numeric field

Use column 52 to indicate the number of decimal positions in any numeric field named in columns 53 through 58. Column 52 must contain an entry when the field named in columns 53 through 58 is numeric. To define a field as numeric with no decimal position, enter a 0. If a field is to be used in arithmetic operations or is to be edited, it must be numeric. If the number of decimal positions specified for a field exceeds the length of that field, the number of decimal positions is assumed equal to the length of the field.

Columns 53-58 (Field Name)

Entry	Explanation
1-6 alphameric characters	Field name, array name, or array element
PAGE, PAGE1-PAGE7	Special words

Use columns 53 through 58 to name a field, array, or array element found on your input records. When referencing an array, additional entries may be needed in these columns (see *Using an Array Name and Index* in Chapter 13, *Using Arrays and Tables).*

Use this name throughout the program whenever you refer to this field. Indicate the names of the fields for all types of records using a separate line for each field. However, name only the fields that you use. For example, if you use only the first 10 positions of a record that is 96 positions long, define positions 1 through 10 on the input specifications.

For CONSOLE files, whole array names must be entered in one of the following ways:

- Define the whole array as a subfield within a field.
- Define each element of the array with an index and place this entry in columns 53 through 58 of the input specifications. The index must be an integer value.

Field Names

A field name can be from 1 to 6 characters long and must begin in column 53. The first character must be an alphabetic character. The remaining characters can be any combination of alphabetic and numeric characters (special characters are not allowed). Blanks cannot appear between characters in the name.

All fields in one type of record should have different names. If two or more fields on the same record type have the same name, only the field described last is used. However, fields from different record types can have the same name if the fields are the same length and contain the same type of data. This applies even if the fields are in different locations in each record type.

Numeric fields can have a maximum length of 15 digits. Alphameric fields can have a maximum length of 256 characters (66 for CONSOLE files). A data structure can have a maximum length of 9999 characters. Subfields can have a maximum length of 256 characters for alphameric subfields and 15 digits for numeric subfields.

If a data structure subfield is specified in columns 53 through 58, only field-record-relation indicators (columns 63 and 64) can be specified. Entries for control-level indicators (columns 59 and 60), match field values (columns 61 and 62), and field indicators (columns 65 through 70) are not allowed. A data structure name cannot be specified as a subfield in a data structure.

Fields that are used in arithmetic operations (see Chapter 28, *Operation Codes)* or fields that are edited or zero suppressed (see *Column 38* and *Columns 45-70* in Chapter 27, *Output Specifications)* must be defined as numeric. Therefore, column 52 must have a decimal position entry (0 through 9).

Field Names in OR Relationship

If two or more record types contain identical fields, you must describe each field. To eliminate duplicate coding of identical fields from different record types, use the OR relationship (see Figure 25-10). A maximum of 20 OR lines can be used for each record sequence group if no AND lines are specified.

An OR relationship means that the fields named can be found in either of the record types. You can use OR lines when:

- Two or more record types have the same fields in the same positions (see Figure 25-12).
- Two or more record types have some fields that are identical and some fields that differ in location, length, or type of data. See *Columns 63-64* in this chapter for sample coding of such record types.

Write OR in columns 14 and 15 to indicate an OR line. If there are several AND or OR lines, field description lines start after the last record identification line.

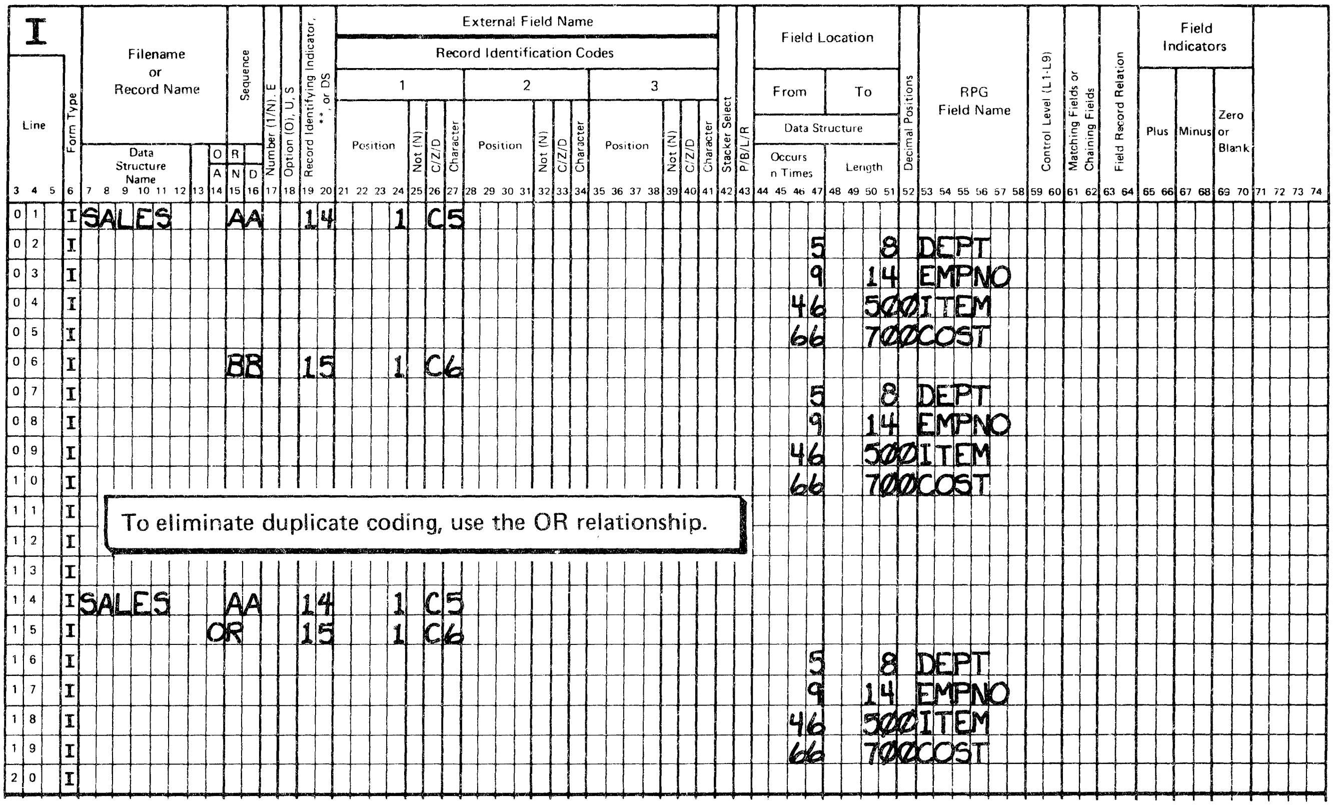
Special Words (PAGE, PAGE1-PAGE7)

If a printed report has several pages that are to be numbered, use the special word PAGE to indicate that page numbering is to be done. When you use a PAGE entry on the output specifications, page numbering automatically starts with 1 (see *Columns 32-37* in Chapter 27, *Output Specifications).*

To start at a page number other than 1, enter that page number in a field of an input record and name that field PAGE in columns 53 through 58. The number entered in the PAGE field should be one less than the starting page number. If numbering starts with 24, enter a 23 in the PAGE field. The PAGE field can be 1 to 15 digits long, but must have zero decimal positions (see Figure 25-13). If a PAGE field is used but it is not defined, the PAGE field is assumed to be 4 digits long with zero decimal positions. Any entry in the PAGE field should be coded in the rightmost columns, such as 0023.

Page numbering can be restarted during a program run when a number is specified in a PAGE field of any input record. The PAGE field can be defined as a numeric field, 1 to 15 digits in length, with zero decimal positions, and used in calculations like any other field.

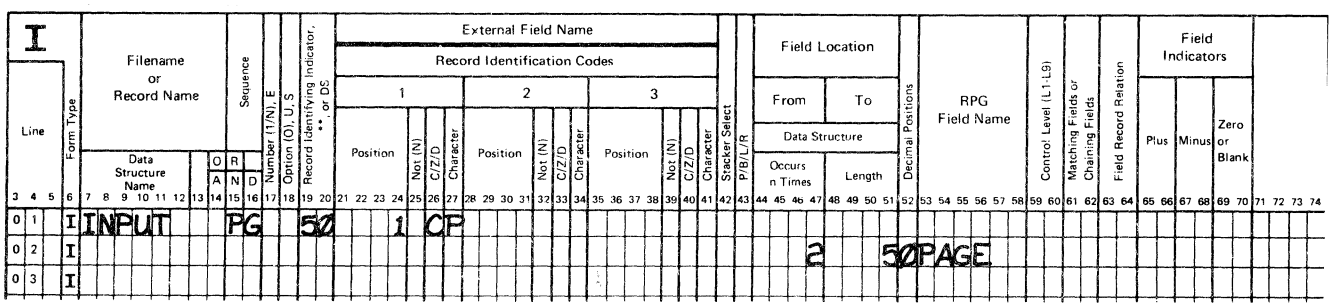
The eight possible PAGE entries (PAGE, PAGE1, PAGE2, PAGE3, PAGE4, PAGE5, PAGE6, and PAGE7) are provided for numbering different page types in the output file or for numbering the pages for different PRINTER files.



Line	Form Type	Filename or Record Name	Sequence / OR	Record Identifying Indicator	Position	C/Z/D	Character	From	To	Decimal Positions	RPG Field Name
01	I	SALES	AA	14	1	C	5				
02	I							5	8		DEPT
03	I							9	14		EMPNO
04	I							46	50	0	ITEM
05	I							66	70	0	COST
06	I		BB	15	1	C	6				
07	I							5	8		DEPT
08	I							9	14		EMPNO
09	I							46	50	0	ITEM
10	I							66	70	0	COST
11	I										
12	I										
13	I										
14	I	SALES	AA	14	1	C	5				
15	I		OR	15	1	C	6				
16	I							5	8		DEPT
17	I							9	14		EMPNO
18	I							46	50	0	ITEM
19	I							66	70	0	COST
20	I										

To eliminate duplicate coding, use the OR relationship.

Figure 25-12. Record Types with Identical Fields



Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator	Position	C/Z/D	Character	From	To	Decimal Positions	RPG Field Name
01	I	INPUT	PG	50	1	C	P				
02	I							2	5	0	PAGE
03	I										

Figure 25-13. Page Record Description

Columns 59-60 (Control Level)

Entry	Explanation
L1-L9	Any control-level indicator. Control-level indicators cannot be used with full-procedural, chained, demand, or WORKSTN files or with a data structure.

Use columns 59 and 60 to assign control-level indicators to input fields. Use control-level indicators to specify when calculation or output operations are to be performed. For more information, see Chapter 12, *Using Indicators.*

Columns 61-62 (Matching Fields)

Entry	Explanation
M1-M9	Any matching level

Use columns 61 and 62 to specify match fields and sequence checking. Match fields and sequence checking cannot be specified for chained files, full-procedural files, demand files, WORKSTN files, or a data structure.

An entry in columns 61 and 62 indicates:

- Match fields and sequence checking when you have two or more input or update files with match fields
- Sequence checking only when you have just one input or update file

The match levels are ranked in order of importance, with M1 being the least significant.

Match Fields

In processing more than one input file, specify match fields to compare records from two or more input or update files to determine which record is to be selected for processing. You can use one field, many fields, or an entire record to match records. Whenever the contents of the match field from the primary file record are the same as the contents of the match field from a secondary file record, the matching-record (MR) indicator turns on. The matching-record indicator can then be used to condition those operations that are to be done only when records match (see *Columns 9-17* in Chapter 26, *Calculation Specifications; Columns 23-31* in Chapter 27, *Output Specifications*; and *Matching-Record Indicator* in Chapter 12, *Using Indicators).*

As many as nine match fields can be indicated when you use the values M1 through M9.

M1 through M9 only identify the fields by which the records are matched; they are not indicators, but they cause the matching-record indicator to turn on.

For a complete description of how to assign match fields and how records are selected for processing, see Chapter 11, *Using Primary and Secondary Files*.

Sequence Checking

To check the data in the fields of a record in one input or update file for a special sequence, assign a value of M1 through M9 to the field to be checked. As many as nine fields can be checked. The sequence (ascending or descending) of the record file must be specified in column 18 of the file description specifications (see Chapter 21). See Figure 25-14 for an example of sequence checking.

To check the sequence of record types in a file, see *Columns 15-16 (Sequence)* in this chapter.

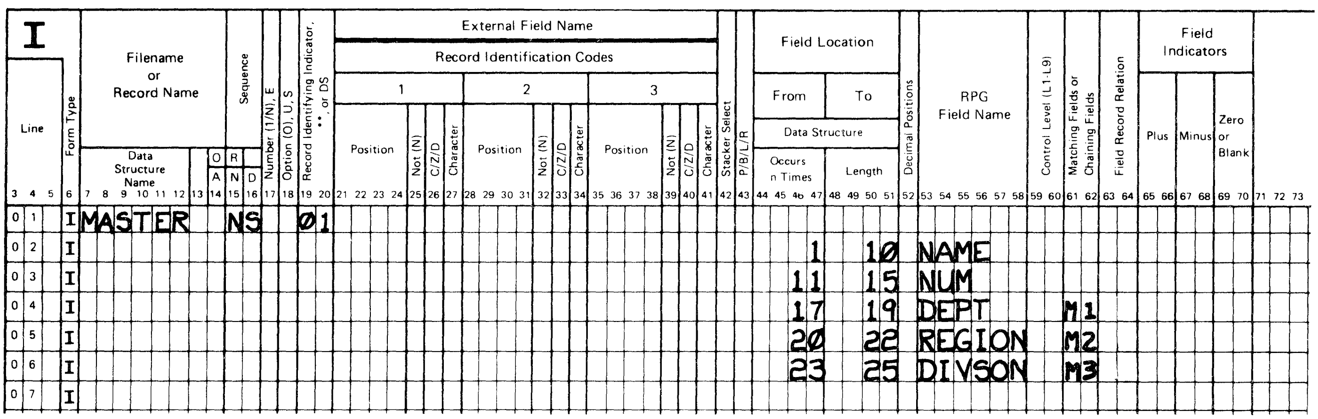


Figure 25-14. Match Fields (Sequence Checking within a File)

An input file called MASTER is to be sequence-checked through three fields. Data from two records is shown below:

Data from First Record		**Data from Second Record**	
DEPT	008	DEPT	003
REGION	051	REGION	025
DIVSON	003	DIVSON	005

In sequence checking, all fields are treated as one continuous field. Thus, the match fields look like:

	M3	M2	M1
Record 1	003	051	008
Record 2	005	025	003

The match field from record 1 is compared with the match field from record 2. If the file is specified to be in ascending sequence, the records are in order because 005025003 is higher than 003051008. However, if the file is specified as having a descending sequence, record 2 is out of order.

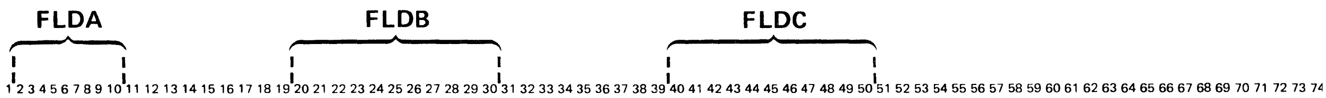
Columns 63-64 (Field Record Relation)

Entry	Explanation
Blank	Columns must be blank for CONSOLE files.
01-99	Record-identifying indicator assigned to a record type, or an indicator set on elsewhere in the program.
L1-L9	Control-level indicator previously used.
MR	Matching-record indicator.
U1-U8	External indicator previously set.
H1-H9	Halt indicator previously used.

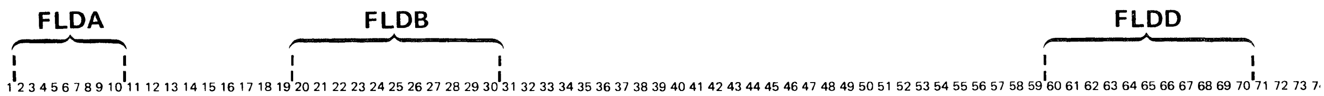
Use a record-identifying indicator in columns 63 and 64 to relate a field to a particular record type.

When several record types are specified in an OR relationship, all fields that do not have a field-record-relation indicator in columns 63 and 64 are associated with all record types in the OR relationship. To relate a field to just one record type, enter the record-identifying indicator assigned to that record type in columns 63 and 64 (see Figure 25-15).

Columns 63 and 64 can also be used to specify that the program accept and use data from a particular field only when a certain condition occurs (such as matching records, a control break, or an external indicator is on). Data from the field named in columns 53 through 58 is accepted only when the field-record-relation indicator is on.



Record Identification Code = 5



Record Identification Code = 6

The file contains two different types of records, one identified by a 5 in position 1 and the other by a 6 in position 1. FLDC is related by record-identifying indicator 14 to the record type identified by a 5 in position 1. FLDD is related to the record type having a 6 in position 1 by record-identifying indicator 16. This means that FLDC is found on only one type of record (that identified by 5 in position 1) and FLDD is found only on the other type. FLDA is conditioned by indicator 07, which was previously defined elsewhere in the program. FLDB is found on both types because they are not related to any one type by a record-identifying indicator.

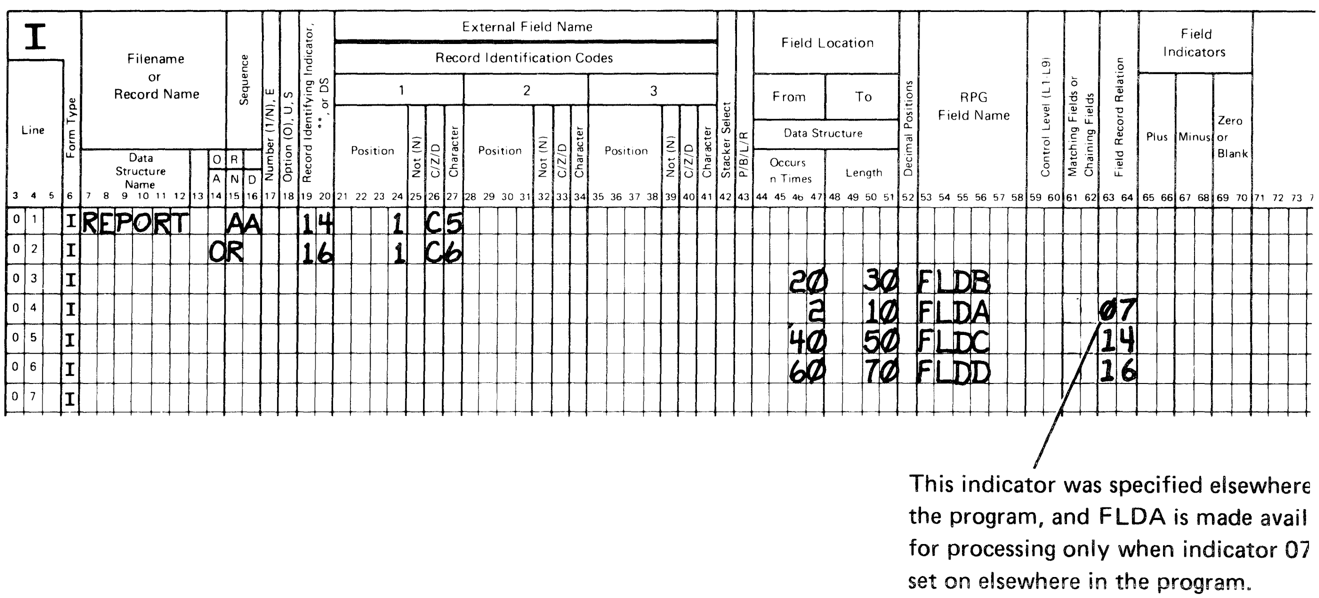


Figure 25-15. Field-Record-Relation Indicator

Columns 65-70 (Field Indicators)

Entry	Explanation
01-99	Numeric indicator
H1-H9	Halt indicator (when checking for an error condition in the data)

Use columns 65 through 70 to check the condition of the numeric fields. Use columns 69 and 70 to check the condition of an alphameric field. These columns cannot be used for a data structure. The three conditions are:

- Plus (columns 65 and 66). An indicator entered in columns 65 and 66 turns on if the numeric field named in columns 53 through 58 is greater than zero.
- Minus (columns 67 and 68). An indicator entered in columns 67 and 68 turns on if the numeric field in columns 53 through 58 is less than zero.
- Zero or blank (columns 69 and 70). An indicator entered in columns 69 and 70 turns on if a numeric field named in columns 53 through 58 is all zeros or if an alphameric field is all blanks. A numeric field that is all blanks turns on an indicator specified for zeros. However, if an alphameric field is all zeros, the field does not turn on the indicator specified for all blanks.

Columns 65 through 70 must be blank when table or array names are specified in input specifications. However, an entry can be made for an array element.

Field indicators assigned in these columns can also be set on or set off by SETON or SETOF operations in the calculation specifications.

Columns 71-74

Columns 71 through 74 are not used. Leave them blank.

Columns 75-80 (Program Identification)

Entry	Explanation
Blank	Program identification defaults to the program name specified on the control specification.
Any valid program name	Program identification. The first character must be alphabetic but cannot be #, $, or @. The remaining characters must be alphameric with no imbedded blanks. No special characters can be used.

Columns 75 through 80 can contain any characters. These columns can contain the program name used in the control specification, or they can contain any other characters to identify a certain portion of the program. These entries are ignored by the compiler but appear in the source program listing.

Note: To be compatible with other RPG systems, the specifications sheets show only 80 positions for each statement. However, each statement in an RPG source program can contain up to 96 characters. Columns 81 through 96 are available for comments.

Chapter 26. Calculation Specifications

Chapter 26. Calculation Specifications

Calculation specifications describe the calculations you want performed on your data and the order in which you want them performed. Each calculation specifications statement can be divided into three parts:

- When the operation is to be performed (columns 7 through 17). The indicators entered in these columns determine under what conditions the specified operation is to be done.
- What kind of operation is to be performed (columns 18 through 53). Entries in these fields describe the kind of operation to be done and specify the data upon which the operation is to be performed.
- What tests are to be made on the results of the operation (columns 54 through 59). The indicators entered in these columns signal the result of the operation and can be used to condition other operations.

Calculation specifications must be specified in the following order: detail, total, subroutine.

Write these specifications on the RPG Calculation Specifications sheet (see Figure 26-1).

IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

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

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

C Line (3 4 5)	Form Type (6)	Control Level (L0-L9, LR, SR, AN/OR) (7 8)	Indicators: Not (9), 10 11	And: Not (12), 13 14	And: Not (15), 16 17	Factor 1 (18–27)	Operation (28–32)	Factor 2 (33–42)	Result Field: Name (43–48)	Result Field: Length (49–51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators — Arithmetic: Plus / Compare: 1 > 2 / Lookup(Factor 2) is: High (54 55)	Minus / 1 < 2 / Low (56 57)	Zero / 1 = 2 / Equal (58 59)	Comments (60–74)
0 1	C															
0 2	C															
0 3	C															
0 4	C															
0 5	C															
0 6	C															
0 7	C															
0 8	C															
0 9	C															
1 0	C															
1 1	C															
1 2	C															
1 3	C															
1 4	C															
1 5	C															
1 6	C															
1 7	C															
1 8	C															
1 9	C															
2 0	C															
	C															
	C															
	C															
	C															
	C															

*Number of sheets per pad may vary slightly.

Figure 26-1. RPG Calculation Specifications

Columns 1-2 (Page)

Entry	Explanation
Blank	No page number is used.
01-99	Page number.

Use columns 1 and 2 in the upper right corner of each sheet to number the specifications sheets, in ascending order, for your job. You can use more than one of each type of sheet, but keep all sheets of the same type together.

Columns 3-5 (Lines)

Entry	Explanation
Blank	No line number is used.
Any numbers	Line numbers.

Use columns 3 through 5 to number the lines on each page. Columns 3 and 4 are preprinted on each sheet so, in most cases, line numbering is already done.

Page and line numbers are optional entries and are not required to successfully run an RPG program. Columns 1 through 5 are checked for ascending order, and RPG prints an S in the left margin of the RPG listing for any statement that is out of order. If you use SEU to enter the source program, you can request that SEU put the statements in order.

Column 6 (Form Type)

A C must appear in column 6 to identify this line as a calculation specification.

Column 7 (Comments)

Entry	Explanation
*	Comment line

Use an asterisk in column 7 to identify the line as a comment line. Use comments throughout your program to help document the purpose of a certain section of coding. You can use any character in a comment line. Comments are not instructions to your program; they only document your program.

Columns 7-8 (Control Level)

Entry	Explanation
Blank	Calculation operation is done at detail calculation time for each program cycle if the indicators in columns 9 through 17 allow it; or if calculation is part of a subroutine.
L0	Calculation operation is done at total calculation time for each program cycle after total calculation processing has started. *Note:* If no control-level indicators are specified on input specifications, total calculation time processing starts during the second program cycle. If control-level indicators are specified on the input specifications, total calculation time processing starts during the program cycle after the first record containing control fields is processed or at LR time. Totals are always processed at LR time.
L1-L9	Calculation operation is done when the appropriate control break occurs at total calculation time.
LR	Calculation operation is done after the last record has been processed.
SR	Calculation operation is part of a subroutine. A blank entry is also valid for calculations that are part of a subroutine.
AN, OR	Establishes AND and OR relationships between lines of indicators.

Use columns 7 and 8 to:

- Perform total calculation operations when the appropriate control break occurs.
- Perform calculation operations that are done only after the last record has been read.
- Indicate that an operation is part of a subroutine. However, columns 7 and 8 can also be blank for calculations that are part of a subroutine.
- Specify that certain lines of indicators are in an AN/OR relationship.

For more information on the 2-character entries L0 and L1 through L9, see Chapter 12, *Using Indicators.*

Subroutine Lines (SR)

An SR entry in columns 7 and 8 indicates that this specification line is part of a subroutine (see *Subroutine Operations* in Chapter 28, *Operation Codes).* You do not have to use SR on a calculation specification line that is part of a subroutine, you can leave columns 7 and 8 blank. Subroutine lines must be specified last.

AN/OR Lines

Use columns 7 and 8 to specify that lines of indicators are in an AN/OR relationship. When you use the AN/OR relationship, many lines of indicators can be grouped together to condition an operation. A maximum of seven AN lines or seven OR lines or any combination thereof can condition an operation. For more information, see Chapter 12, *Using Indicators*.

Columns 7-12 (/EJECT)

Entry	Explanation
/EJECT	The specifications following this entry are to begin on a new page of the compiler listing.

The /EJECT specification is not printed on the compiler listing.

Columns 7-12 (/TITLE)

Entry	Explanation
/TITLE	The heading information (such as a title or security classification) that follows the /TITLE entry appears at the top of each page of the compiler listing. The heading information is entered in columns 14 through 74.

A program can contain more than one /TITLE statement. Each /TITLE statement provides heading information for the compiler listing until the next /TITLE statement is read. To print on the first page of the compiler listing, a /TITLE statement must be the first statement read. Information specified by the /TITLE statement is printed in addition to compiler heading information.

The /TITLE statement causes an eject to the next page before the title is printed. The /TITLE statement is not printed on the compiler listing.

Columns 7-14 (/SPACE)

Entry	Explanation
/SPACEb̸n	Line spacing occurs at this point in the compiler listing. Valid entries for n are 1 to 3. If you do not specify n, 1 is assumed.

One blank (b̸) must come before the value you specify for n. The value you specify for n indicates the number of blank lines to be spaced before the next specification line is printed. If n is greater than the number of lines remaining on the current page, the next specification line is printed on a new page. If you specify just /SPACE, one line is spaced.

/SPACE is not printed on the compiler listing but is replaced by the actual line spacing. The spacing indicated by /SPACE is in addition to the three blank lines that occur between specification types.

Columns 9-17 (Indicators)

Entry	Explanation
Blank	Operation is performed on every program cycle.
01-99	Field indicators, record-identifying indicators, or resulting indicators assigned elsewhere in the program.
KA-KN, KP-KY	Command-key indicators assigned elsewhere.
L1-L9	Control-level indicators assigned elsewhere. These indicators are on as detail indicators when the first record of a new control group is processed.
LR	Last-record indicator.
MR	Matching-record indicator.
H1-H9	Halt indicators assigned elsewhere.
U1-U8	External indicators previously set.
OA-OG, OV	Overflow indicators previously assigned.

Use columns 9 through 17 to assign indicators that control the conditions under which an operation is done. You can use from one to three separate fields (columns 10 and 11, 13 and 14, and 16 and 17) on each line, one for each indicator. If the indicator must be off to condition the operation, place an N before the appropriate indicator (columns 9, 12, 15).

The indicators specified in columns 9 through 17 on one specification line are in an AND relationship with each other. The indicators on one line or indicators in grouped lines plus the control-level indicator (if used in columns 7 and 8) must all be exactly as specified before the operation is done.

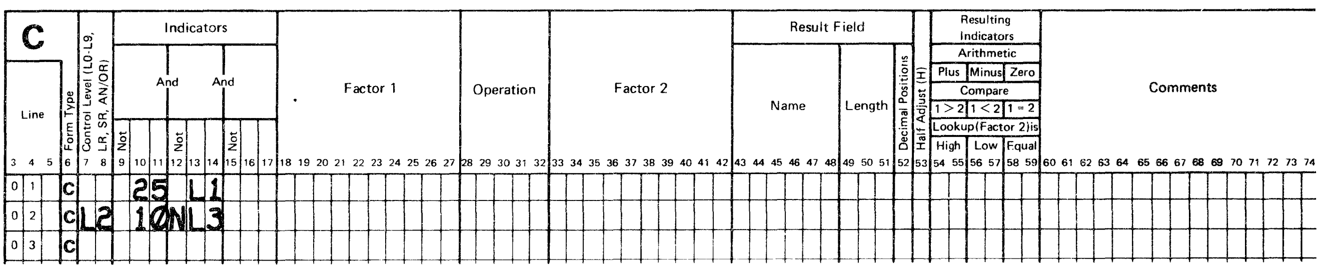
An indicator that is specified in columns 9 through 17 of a calculation specification can also be entered as a resulting indicator on the same line. If the indicator in columns 9 through 17 is on, the calculation is performed.

Relationship between Columns 7-8 and Columns 9-17

In one program cycle, all operations conditioned by control-level indicators in columns 7 and 8 (total time) are done before operations conditioned by control-level indicators in columns 9 through 17 (see Figure 26-2).

When a control-level indicator is used in columns 9 through 17 and columns 7 and 8 are not used (detail time), the operation conditioned by the indicator is done only on the record that causes a control break or any higher-level control break.

When a control-level indicator is specified in columns 7 and 8 (total time) and the matching-record indicator (MR) is specified in columns 9 through 17, MR indicates the matching condition of the previous record and not the record just read that caused the control break. After all operations conditioned by the control-level indicators (specified in columns 7 and 8 of the calculation specifications) are done, MR then indicates the matching condition of the record just read.



Assume that indicator 25 represents a record type and that a control level 2 break occurred when record type 25 was read. L1 and L2 are both on. All operations conditioned by the control-level indicators in columns 7 and 8 are performed before operations conditioned by control-level indicators in columns 9 through 17. Thus, the operation in line 02 occurs before the operation in line 01. The operation in line 01 is done on the first record of the new control group indicated by 25, whereas the operation in line 02 is a total operation done for all records of the previous control group.

The operation in line 02 can be done when the L2 indicator is on provided the other conditions are met. Indicator 10 must be on. The L3 indicator must not be on.

The operation conditioned by both L2 and NL3 is done only when a control level 2 break occurs. These two indicators are used together because this operation is not to be done when a control level 3 break occurs, even though L2 is also on.

Figure 26-2. Conditioning Operations Using Control Level Indicators

Columns 18-27 (Factor 1)

Use columns 18 through 27 to name the field or to give the actual data (literal) to be used in the operation to be performed. See Figure 26-3 for a summary of the operation codes.

The entries you can use for factor 1 are:

- The name of any field that has been defined
- Any alphameric or numeric literal
- Any subroutine, table, array name, or array element
- Any date field name (UDATE, UMONTH, UDAY, UYEAR)
- The special names PAGE, PAGE1, PAGE2, PAGE3, PAGE4, PAGE5, PAGE6, or PAGE7
- The special qualifier *LIKE for the DEFN operation.
- Any figurative constant (*BLANK, *BLANKS, *ZERO, *ZEROS)
- A label for a TAG, BEGSR, or ENDSR operation

The following restrictions apply to entries in factor 1:

- A data structure name *cannot* be specified in factor 1 or factor 2.
- A data structure subfield name can be used in factor 1 or factor 2; however, overlapping subfields in a data structure cannot be used in the same calculation. A subfield is considered to be an overlapping subfield if its from or to position occurs within the from and to positions of another subfield within the same data structure. If factor 1, factor 2, or the result field references a subfield in a data structure that is an array or array element with a variable index, the entire array is used to determine whether overlap exists. The same array name can be referenced in the appropriate factors of a calculation specification without violating the overlap rule. See Figure 26-4 for examples of the overlap rule.
- Figurative constants cannot be used with move zone operations, bit operations, or the SET, KEY, SQRT, or DEBUG operation codes.

An entry in factor 1 must begin in column 18.

Entries for factor 1 depend upon the operation code used in columns 28 through 32. Some operations require entries in both factors, some require entries in only one, and some require no entries at all. See *Columns 28-32 (Operation)* for more information on operation codes. For information on how to name a subroutine, see *Subroutine Operations* in Chapter 28, *Operation Codes.*

Operation Code	Control-Level Indicators	Conditioning Indicators	Factor 1	Factor 2	Result Field	Resulting Indicators		
	Columns					Columns		
	7–8	9–17				54–55	56–57	58–59
ACQ	Optional	Optional	Required	Required			Optional	
ADD	Optional	Optional	Optional	Required	Required	Optional	Optional	Optional
BEGSR	SR or blank		Required					
BITOF	Optional	Optional		Required	Required			
BITON	Optional	Optional		Required	Required			
CASxx (CASE)	Optional	Optional	Optional	Optional	Required	Optional	Optional	Optional
CHAIN	Optional	Optional	Required	Required		Optional [1]		
COMP	Optional	Optional	Required	Required		Optional [2]	Optional [2]	Optional [2]
DEBUG	Optional	Optional	Optional	Required	Optional			
DEFN	Optional		*LIKE	Required	Required			
DIV	Optional	Optional	Optional	Required		Optional	Optional	Optional
DO	Optional	Optional	Optional	Optional	Optional			
DOUxx (DO UNTIL)	Optional	Optional	Required	Required				
DOWxx (DO WHILE)	Optional	Optional	Required	Required				
ELSE (ELSE DO)	Optional							
END/DO	Optional	Optional		Optional				
END/DOU	Optional	Optional						
END/DOW	Optional	Optional						
END/IF	Optional							
END/CAS	Optional							
ENDSR	SR or blank		Optional	Optional				
EXCPT	Optional	Optional		Optional				
EXIT	Optional	Optional		Required				
EXSR	Optional	Optional		Required				
FORCE		Optional		Required				
GOTO	Optional	Optional		Required				
IFxx (IF/THEN)	Optional	Optional	Required	Required	Required			
KEYnn [3]	Optional	Optional	Optional			Optional	Optional [4]	Optional
LOKUP(Array)	Optional	Optional	Required	Required	Optional	Optional [4]	Optional [4]	Optional [4]
LOKUP(Table)	Optional	Optional	Required	Required	Required	Optional [4]	Optional	Optional [4]
MHHZO	Optional	Optional		Required	Required			
MHLZO	Optional	Optional		Required	Required			
MLHZO	Optional	Optional		Required	Required			
MLLZO	Optional	Optional		Required	Required			
MOVE	Optional	Optional		Required	Required			
MOVEA	Optional	Optional		Required	Required			
MOVEL	Optional	Optional		Required	Required			
MULT	Optional	Optional	Optional	Required	Required	Optional	Optional	Optional
MVR	Optional	Optional				Optional	Optional	Optional
NEXT	Optional	Optional	Required	Required	Required		Optional	
POST	Optional	Optional	Required				Optional	
READ	Optional	Optional		Required			Optional [5]	Optional [1]
READE	Optional	Optional	Required	Required				Required
READP	Optional	Optional		Required				Required
REL	Optional	Optional	Required	Required			Optional	
RLABL								
SETnn	Optional	Optional	Optional	Optional		Optional	Optional	Optional

Figure 26-3 (Part 1 of 2). Operation Codes

Operation Code	Control-Level Indicators	Conditioning Indicators	Factor 1	Factor 2	Result Field	Resulting Indicators		
	Columns					Columns		
	7-8	9-17				54-55	56-57	58-59
SETOF	Optional	Optional				Optional [2]	Optional [2]	Optional [2]
SETON	Optional	Optional				Optional [2]	Optional [2]	Optional [2]
SETLL	Optional	Optional	Required	Required				
SHTDN	Optional	Optional				Required		
SORTA	Optional	Optional		Required				
SQRT	Optional	Optional		Required	Required			
SUB	Optional	Optional	Optional	Required	Required	Optional	Optional	Optional
TAG	Optional		Required					
TESTB	Optional	Optional		Required	Required	Optional [2]	Optional [2]	Optional [2]
TESTZ	Optional	Optional			Required	Optional [2]	Optional [2]	Optional [2]
TIME	Optional	Optional			Required			
XFOOT	Optional	Optional		Required	Required	Optional	Optional	Optional
Z-ADD	Optional	Optional		Required	Required	Optional	Optional	Optional
Z-SUB	Optional	Optional		Required	Required	Optional	Optional	Optional

Fields without entries must be blank.

SR = The only allowable nonblank characters in columns 7 and 8
for the BEGSR and ENDSR operation codes

[1] This indicator is required if the file specified in factor 2 is a full-procedural file.
[2] At least one resulting indicator must be specified in columns 54 through 59.
[3] The nn entries in columns 31 and 32 are for message indicator numbers. If the result field of a SET operation contains the keyword ERASE, factor 2 must contain the name of the CONSOLE file. Otherwise, factor 2 and the result field must be blank.
[4] At least one resulting indicator must be specified in columns 54 through 59, but no more than two can be used.
[5] Columns 56 and 57 can contain an indicator when the READ operation is used with a WORKSTN device.

Figure 26-3 (Part 2 of 2). Operation Codes

Line	Form Type	Filename or Record Name / Data Structure Name	Record Identifying Indicator, **, or DS	Field Location From	Field Location To	Decimal Positions	RPG Field Name
01	I	DATADS	DS				
02	I			1	5		ALPHA1
03	I			6	1Ø		ALPHA2
04	I			4	8		ALPHA3
05	I			7	9		ALPHA4
06	I			6	1Ø		ALPHA5
07	I			11	14	Ø	NUM1
08	I			11	15	Ø	NUM2
09	I			11	15	Ø	NUM3
10	I			15	2Ø	1	NUM4
11	I			17	18	Ø	NUM5
12	I			21	5Ø		ARR1
13	I			41	7Ø		ARR2
14	I						
15	I						
16	I						
17	I						
18	I						
19	I						
20	I						

The data structure DATADS contains subfields and arrays that are defined as overlapping (that is, occupying part of the same area). ARR1 (on line 12) has six elements, and each element is five positions long for a total length of 30. ARR2 (on line 13) has five elements, and each element is six positions long for a total length of 30.

Figure 26-4 (Part 1 of 3). Examples of Valid and Invalid Calculations with Overlapping Subfields in a Data Structure

Line	Form Type	Factor 1	Operation	Factor 2	Result Field Name
01	C				
02	C				
03	C				
04	C				

The following individual calculations are valid because the subfields do not overlap.

Line	Form Type	Factor 1	Operation	Factor 2	Result Field Name
05	C		MOVE	ALPHA1	ALPHA2
06	C		MOVE	ALPHA2	ALPHA1
07	C	NUM1	ADD	NUM4	NUM4
08	C		Z-ADD	1	NUM3
09	C		Z-ADD	5	NUM2
10	C		Z-ADD	NUM5	NUM1
11	C		MOVE	' '	ALPHA3
12	C		MOVE	ALPHA1	ARR1,X
13	C				
14	C				
15	C				
16	C				
17	C				
18	C				

The following individual calculations are valid because the subfields are determined to be the same area, and execution will not cause invalid results.

Line	Form Type	Factor 1	Operation	Factor 2	Result Field Name
19	C				
20	C		MOVE	ALPHA2	ALPHA5
	C		MOVEL	ALPHA5	ALPHA2
	C	NUM2	ADD	NUM3	NUM3
	C		Z-ADD	NUM3	NUM2
	C				
	C				
01	C				
02	C				
03	C				
04	C				

The following individual calculations are invalid because the subfields occupy part of the same area, and execution could cause invalid results.

Line	Form Type	Factor 1	Operation	Factor 2	Result Field Name
05	C				
06	C		MOVE	ALPHA1	ALPHA3
07	C		MOVE	ALPHA3	ALPHA4
08	C	NUM1	ADD	NUM2	NUM2
09	C		Z-ADD	NUM5	NUM4
10	C		Z-ADD	NUM3	NUM1
11	C				

Figure 26-4 (Part 2 of 3). Examples of Valid and Invalid Calculations with Overlapping Subfields in a Data Structure

The following individual calculations involve the same array and are valid calculations.

```
C                           MOVE ARR1,1    ARR1,5
C                           MOVE ARR1,X    ARR1,2
C                           MOVE ARR2,X    ARR2,Y
```

The following individual calculations are valid because the array elements associated with the constant indexes do not overlap.

```
C                           MOVE ARR1,1    ARR2,1
C                           MOVELARR1,2    ARR2,5
```

The following individual calculations are invalid because the array elements associated with the constant indexes overlap, or variable indexes are specified and the entire array is required to determine overlap.

```
C                           MOVE ARR2,1    ARR1,5
C                           MOVE ARR1,X    ARR2,X
C                           MOVE ARR1,X    ARR2,1
C                           MOVE ARR1,1    ARR2,X
```

Figure 26-4 (Part 3 of 3). Examples of Valid and Invalid Calculations with Overlapping Subfields in a Data Structure

Literals

A literal is the actual data used in an operation rather than the field name representing that data. A literal can be either alphameric or numeric.

Alphameric Literals

Consider the following rules when using an alphameric literal (see Figure 26-5):

- Any combination of characters can be used in an alphameric literal. Blanks are also valid.
- The maximum length of an alphameric literal is 8 characters.
- Alphameric literals must be enclosed in apostrophes (').
- An apostrophe required as part of a literal is represented by two apostrophes. For example, the literal O'CLOCK is coded as 'O''CLOCK'.
- Alphameric literals cannot be used for arithmetic operations.

Numeric Literals

Consider the following rules when using a numeric literal (see Figure 26-5):

- A numeric literal consists of any combination of the digits 0 through 9. A decimal point or sign can also be included.
- The sign (+ or -), if present, must be the leftmost character. An unsigned literal is treated as a positive number.
- The maximum total length of a numeric literal is 10 characters including the sign and decimal point.
- Blanks cannot appear in a numeric literal.
- Numeric literals must not be enclosed in apostrophes (').
- Numeric literals are used in the same way as a numeric field.

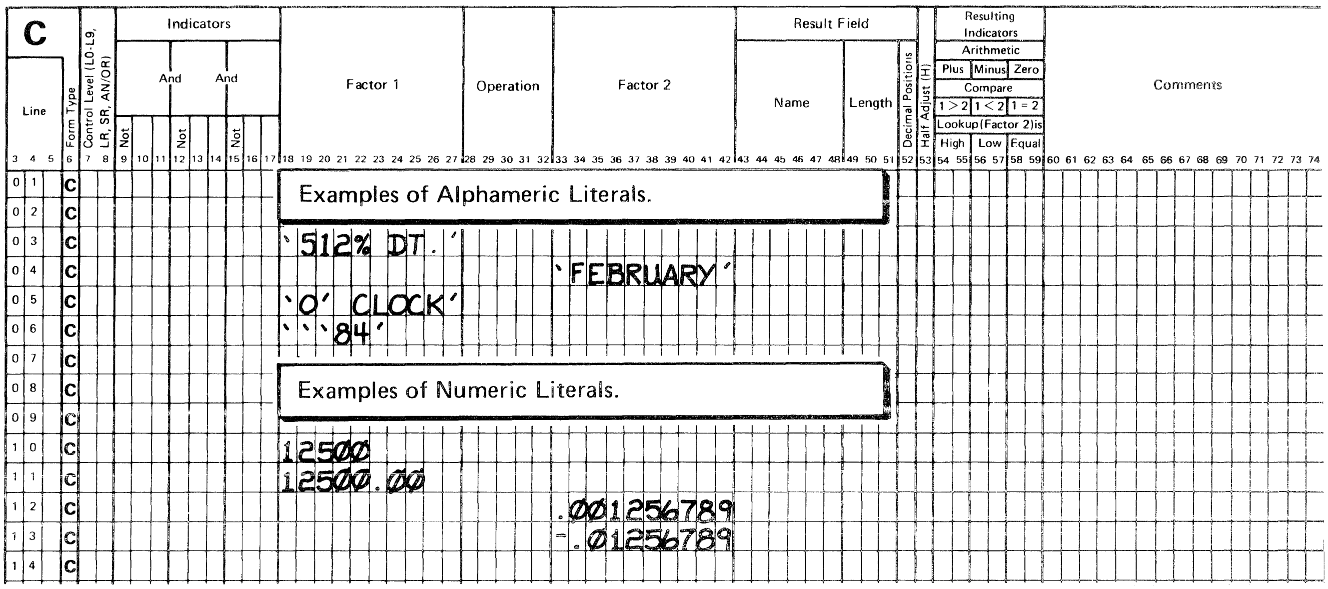


Figure 26-5. Alphameric and Numeric Literals

Figurative Constants

The figurative constants *BLANK, *BLANKS, *ZERO, and *ZEROS can be specified as literals. The following rules apply for figurative constants:

- The figurative constants *BLANK and *BLANKS can only be used with alphameric fields.
- The figurative constants *ZERO and *ZEROS can be used with either alphameric or numeric fields.
- The length of the figurative constant is assumed to be equal to the length of the other factor field, if present. Otherwise, the length of the figurative constant is assumed to be equal to the length of the result field.
- Figurative constants are considered to be elementary items, and, if used in conjunction with an array, act like a field. For example:

Line (3-5)	Form Type (6)	Control Level (L0-L9, LR, SR, AN/OR) (7-8)	Indicators (9-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field Name (43-48)	Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators (54-59)	Comments (60-74)
01	C				MOVE	*ZEROS	ARR					
02	C											

If ARR has 4-character elements, each element of ARR contains '0000' after the move is executed.

- The logical placement of a figurative constant in the collating sequence can be altered by specifying an alternative collating sequence.

Columns 28-32 (Operation)

Use columns 28 through 32 to specify the kind of operation to be performed using factor 1, factor 2, and/or the result field. The operation code must begin in column 28. A special set of operation codes must be used to indicate the type of operation to be performed.

Every operation code used requires certain entries on the same specification line. See Figure 26-3 for a summary of all the operation codes and the entries required for each code. For further information on the operation codes, see Chapter 28, *Operation Codes*.

The program performs the operations in the order specified on the calculation specifications sheet.

Columns 31-32

Entry	Explanation
Blank or 01-99	Message identification code (MIC) of user message member to be displayed for SET or KEY operations unless overridden by a factor 1 entry

Use columns 31 and 32 for all KEY and SET operations in which command key indicators are specified in columns 54 through 59, unless an entry is made in factor 1. Entries in columns 31 and 32 are ignored when factor 1 is specified on the same line as the SET or KEY operation.

The same combination of message identification codes should not be assigned to more than one KEY or SET operation except when the SET operation immediately precedes a KEY operation conditioned by the same indicators (columns 9 through 17) and the special SET-KEY combination is used. See *SET* and *KEY* in Chapter 9 for more information.

Columns 33-42 (Factor 2)

Use columns 33 through 42 to name the field or to give the actual data (literal) to be used in the operation to be performed. See Figure 26-3 for a summary of the operation codes.

The entries you can use for factor 2 are:

- The name of any field that has been defined
- Any alphameric or numeric literal
- Any subroutine, table, array name, or array element
- Any date field name (UDATE, UMONTH, UDAY, UYEAR)
- The special names PAGE, PAGE1, PAGE2, PAGE3, PAGE4, PAGE5, PAGE6, or PAGE7
- Any figurative constant (*BLANK, *BLANKS, *ZERO, *ZEROS)
- A label for a GOTO or EXSR operation
- A filename for a SETLL, CHAIN, DEBUG, READ, READE, READP, FORCE, ACQ, REL, or NEXT operation
- An EXCPT name for an EXCPT operation
- A subroutine name for an EXIT operation
- An array name for a SORTA operation.

The following restrictions apply to entries in factor 2:

- A data structure name *cannot* be specified in factor 1 or factor 2.
- A data structure subfield name can be used in factor 1 or factor 2; however, overlapping subfields in a data structure cannot be used in the same calculation. A subfield is considered to be an overlapping subfield if its from or to position occurs within the from and to positions of another subfield within the same data structure. If factor 1, factor 2, or the result field references a subfield in a data structure that is an array or array element with a variable index, the entire array is used to determine whether overlap exists. The same array name can be referenced in the appropriate factors of a calculation specification without violating the overlap rule. See Figure 26-4 for examples of the overlap rule.
- Figurative constants cannot be used with move zone operations, bit operations, or the SET, KEY, SQRT, or DEBUG operation codes.

An entry in factor 2 must begin in column 33.

Entries for factor 2 depend upon the operation code used in columns 28 through 32. Some operations require entries in both factors, some require entries in only one, and some require no entries at all. See *Columns 28-32 (Operation)* for more information on operation codes. For information on how to name a subroutine, see *Subroutine Operations* in Chapter 28, *Operation Codes.*

Columns 43-48 (Result Field)

Entry	Explanation
ERASE	Erase the CONSOLE file buffer by using the SET operation code.
Field name, table name, array name, array element, data structure subfield name, or data structure name	The field specified contains the result of, or is the object of, the operation specified in columns 28 through 32. A data structure name can be specified as a result field only if the operation code in columns 28 through 32 is RLABL or POST.
INxx (xx = any RPG indicator)	The indicator to be transferred to an external subroutine in an RLABL operation.
Subroutine name	Name of a subroutine to branch to if the condition specified in xx portion of a CASxx statement is met. A subroutine name can be specified as a result field only if the operation code in columns 28 through 32 is CASxx.

Erase

Enter ERASE in columns 43 through 48 to blank or erase the entire buffer for the CONSOLE file. The filename of the CONSOLE file must be entered in columns 33 through 42. ERASE indicates to the system that the buffer should be set to blanks just before the system gets a record at the beginning of the next RPG cycle.

Because the buffer is not erased until the beginning of the next RPG cycle, processing of the current record continues after the ERASE operation is read. If the ERASE operation is executed because of invalid input data, you should insert code in your program to avoid further calculations and to return to the start of the RPG cycle. A correct form of the record containing the invalid input data and any records that were entered after that record can then be reentered.

Field Name, Table Name, Array Name, Array Element, or Data Structure

Use columns 43 through 48 to name the field, data structure subfield, table, array, array element, or data structure that holds the result of the operation specified in columns 28 through 32, or that is the field upon which an operation is performed. Use the name of a field, table, array, array element, data structure, or data structure subfield that has already been defined either by the input, extension, or calculation specifications; or define a new field by entering a field name that is not already used. Any field defined in the result field is created when the program is compiled. The result field can be either numeric or alphameric.

A field used in arithmetic operations (see *Columns 28-32 (Operation)* or numeric compare operations or a field edited or zero suppressed by output specifications must be numeric.

A data structure name can be used as the result field only if the operation specified in columns 28 through 32 is RLABL or POST. Overlapping subfields in a data structure cannot be used in the same calculation. If factor 1, factor 2, or the result field references a subfield in a data structure that is an array or array element with a variable index, the entire array is the entire array is used to determine whether overlap exists. The same array name can be referenced in the appropriate factors of a calculation specification without violating the overlap rule. See Figure 26-4 for examples of the overlap rule.

The result field name must begin with an alphabetic character in column 43 and contain no blanks or special characters.

If columns 43 through 48 contain the name of a field that is not defined elsewhere, columns 49 through 52 should also contain entries. If the field is defined elsewhere, entries in columns 49 through 52 are not necessary but, if specified, must agree with the previous definition of that field.

Columns 49-51 (Field Length)

Entry	Explanation
Blank	Result field is described elsewhere
1-256	Result field length

Use columns 49 through 51 to specify the length of the result field. If the result field is defined elsewhere, no entry is required for the length. However, if the length is specified, it must be the same as the previously defined length, with the same number of decimal positions. If the result field is a new field, consider the form your data is in because the result field must be large enough to hold the largest possible result. If the result field is too small, significant digits can be lost.

For example, to add field A (8 characters long, four decimal positions) to field B (10 characters long, six decimal positions), the result field, field C, must be large enough to contain 11 characters:

```
 9999.0000      Field A
 0001.111111    Field B
10000.111111    Field C (result field)
```

In this example, field C must be defined as 11 characters long with six decimal positions. Some of the numbers to the right of the decimal could be lost without changing the meaning of the result greatly. However, if field C was defined as 10 characters long with six decimal positions, a significant digit to the left of the decimal would be lost. Field C in this case would be 0000.111111; the meaning of the result has greatly changed.

Figure 26-6 shows how the contents of a result field can change after a multiplication operation, depending on the decimal position (column 52) and field length (columns 49 through 51) specifications. The result field for a multiply operation should be as long as the sum of the lengths of the two factor fields.

Numeric fields have a maximum length of 15 characters. Alphameric fields can be up to 256 characters long.

If the result field contains the name of a table or array, an entry in these columns is optional. If used, the entry must agree with the length described by the extension specifications.

Multiplication: 98.76 x 1.234 = 121.86984

Decimal Positions for Result Field (Column 52)	Result Field Length (Columns 49 through 51)									
	10	9	8	7	6	5	4	3	2	1
9	1.869840000	.869840000								
8	21.86984000	1.86984000	.86984000							
7	121.8698400	21.8698400	1.8698400	.8698400						
6	0121.869840	121.869840	21.869840	1.869840	.869840					
5	00121.86984	0121.86984	121.86984	21.86984	1.86984	.86984				
4	000121.8698	00121.8698	0121.8698	121.8698	21.8698	1.8698	.8698			
3	0000121.869	000121.869	00121.869	0121.869	121.869	21.869	1.869	.869		
2	00000121.86	0000121.86	000121.86	00121.86	0121.86	121.86	21.86	1.86	.86	
1	000000121.8	00000121.8	0000121.8	000121.8	00121.8	0121.8	121.8	21.8	1.8	.8
0	0000000121	000000121	00000121	0000121	000121	00121	0121	121	21	1

Not permitted

Permitted but inaccurate

Recommended

Figure 26-6. Result Field Contents Based on Various Field-Length and Decimal-Position Specifications

Column 52 (Decimal Positions)

Entry	Explanation
Blank	Alphameric or numeric result field is described elsewhere, or the newly defined result field is alphameric
0-9	Number of decimal positions in a newly defined numeric result field

Use column 52 to indicate the number of positions to the right of the decimal in a numeric result field. If the numeric result field contains no decimal positions, enter a 0 (zero). This column must be blank if the result field is alphameric. This column can be left blank if the result field is numeric but was described by input or calculations specifications. In this case, field length (columns 49 through 51) must also be left blank.

The number of decimal positions must never be greater than the length of the field. The number can, however, be larger or smaller than the number of decimal positions that actually result from an operation. If the number of decimal positions specified is greater than the number of decimal places that actually result from an operation, zeros are filled in to the right. If the number specified is smaller than the number that results from the operation, the rightmost digits are dropped.

Column 53 (Half-Adjust)

Entry	Explanation
Blank	Do not half-adjust
H	Half adjust

Use column 53 to indicate that the contents of the result field are to be half-adjusted (rounded). Half-adjusting is when the single digit to the right of the last decimal position specified is added to the same position in the result field. All decimal positions to the right of the position specified of the position specified for that field are then dropped (see Figure 26-7).

The half-adjust entry is allowed only with arithmetic operations. See *Columns 28-32 (Operation)*. However, half-adjusting cannot be specified for an MVR operation or for a DIV operation followed by an MVR operation.

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators	Comments
01	C											
02	C											
03	C			30	ADD	5.7968	RESULT	4	2	H		
04	C											

This calculation line shows a result field being half-adjusted to two decimal positions (2 in column 52 and H in column 53).

Second Position
↓

35.7968	Result of an add operation.
6	Add the digit to the right of the last decimal position specified to the same position in the result field.
35.80xx	Drop all decimal positions to the right at the position specified.
35.80	Result after half-adjusting.

Figure 26-7. Half-Adjust

Columns 54-59 (Resulting Indicators)

Entry	Explanation
01-99	Any 2-digit number
KA-KN, KP-KY	Any command-key indicator (allowed only with SET or SETOF operation)
H1-H9	Any halt indicator
L1-L9	Any control-level indicator
LR	Last-record indicator
OA-OG, OV	Any overflow indicator
U1-U8	Any external indicator

Columns 54 through 59 have three purposes:

- To test the value of the result field after an arithmetic operation or to test the result of a CHAIN, KEY, LOKUP, COMP, READ, READE, READP, CAS, TESTB, TESTZ, ACQ, REL, NEXT, POST, or SHTDN operation. For more information on each specific operation, see Chapter 28, *Operation Codes*.
- To specify which command keys to press for a SET operation.
- To specify which indicators are to be turned on or off by the SETON and SETOF operations.

Test Results

You can use an indicator in columns 54 through 59 to test the value of the result field, or to indicate an end-of-file condition, a no-record-found condition, or an exception/error condition. Normally, only the 2-character entries 01 through 99 and H1 through H9 are used as resulting indicators for testing. The indicator specified turns on only if the result field satisfies the condition being tested for. If the condition tested for is not met, the indicator is turned off.

You can use three fields (columns 54 and 55, 56 and 57, and 58 and 59) for testing the results. Each field is used to test for different conditions. You can specify testing for any or all conditions at the same time.

For more information on using resulting indicators for testing, see Chapter 12, *Using Indicators*.

Allowing Command Keys To Be Pressed (SET)

Columns 54 through 59 can contain command-key indicators (KA through KN, KP through KY) for a SET operation. When a SET operation occurs, only the command keys in columns 54 through 59 for that SET operation can be pressed at that time. From one to three command keys can be entered for each SET operation. If one or two command keys are specified, they can appear in any of the three sets of columns. For more information on the SET operation, see Chapter 28, *Operation Codes*.

Columns 60-74 (Comments)

Use columns 60 through 74 to enter any meaningful comments that will help you understand the purpose of each statement. Comments are not instructions to the RPG program; they serve only as a means of documenting your program.

Columns 75-80 (Program Identification)

Entry	Explanation
Blank	Program identification defaults to the program name specified on the control specification.
Any valid program name	Program identification. The first character must be alphabetic but cannot be #, $, or @. The remaining characters must be alphameric with no imbedded blanks. No special character can be used.

Columns 75 through 80 can contain any characters. These columns can contain the program name used in the control specification, or they can contain any other characters to identify a certain portion of the program. These entries are ignored by the compiler but appear in the source program listing.

Note: To be compatible with other RPG systems, the specifications sheets show only 80 positions for each statement. However, each statement in an RPG source program can contain up to 96 characters. Columns 81 through 96 are available for comments.

Chapter 27. Output Specifications

Chapter 27. Output Specifications

You use output specifications to describe the records and fields in the output file and the conditions under which output operations are to be performed. These specifications can be divided into two general categories:

- File and record identification entries (columns 7 through 37) that describe the output file, the records, and the indicators that condition the output.

- Field description entries (columns 23 through 74) that describe the position and format of data on the output record. These entries must begin one line below the file and record identification entries.

Write these specifications on the RPG Output Specifications sheet (see Figure 27-1).

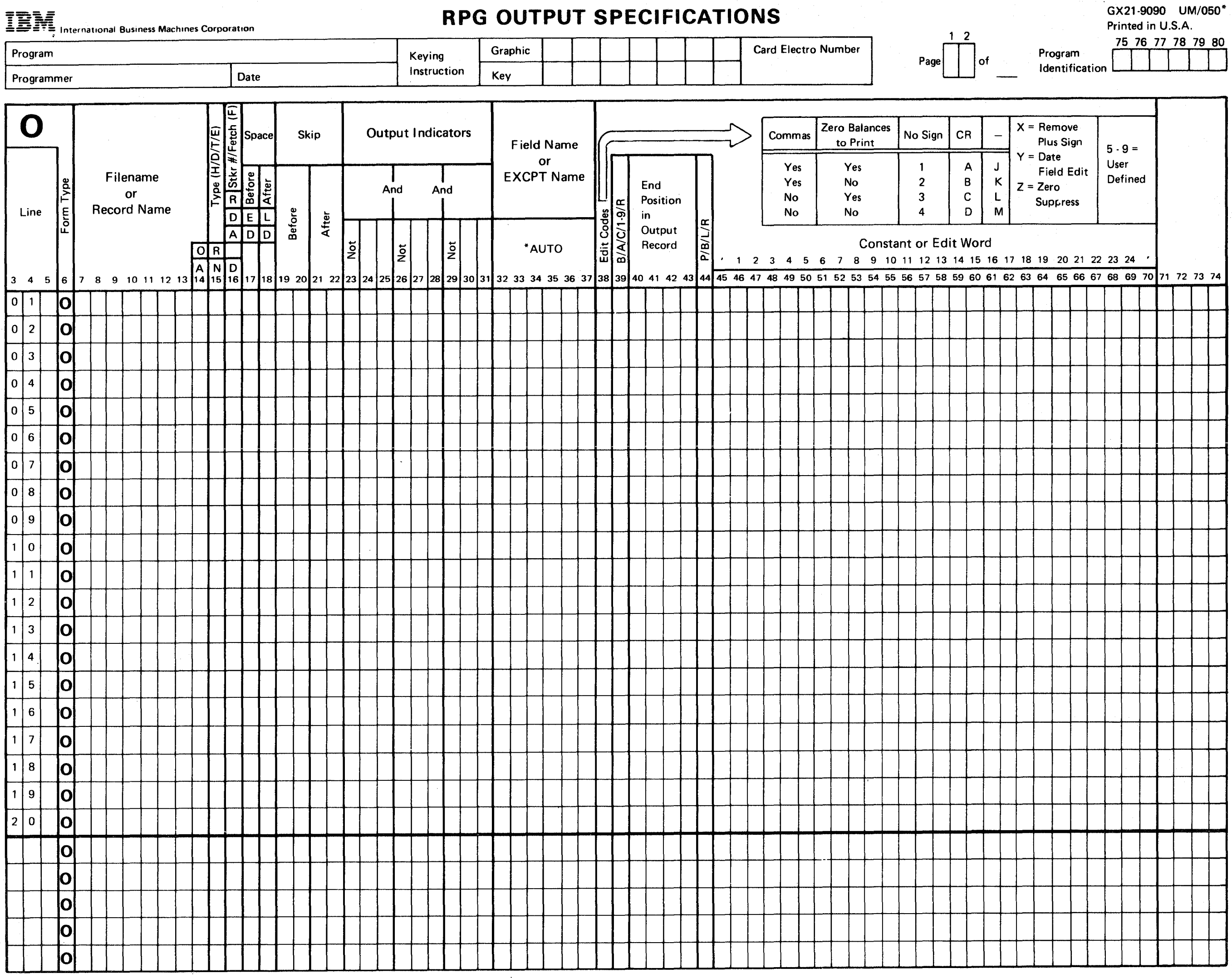


Figure 27-1. RPG Output Specifications

Columns 1-2 (Page)

Entry	Explanation
Blank	No page number is used.
01-99	Page number.

Use columns 1 and 2 in the upper right corner of each sheet to number the specifications sheets, in ascending order, for your job. You can use more than one of each type of sheet, but keep all sheets of the same type together.

Columns 3-5 (Line)

Entry	Explanation
Blank	No line number is used.
Any numbers	Line numbers.

Use columns 3 through 5 to number the lines on each page. Columns 3 and 4 are preprinted on each sheet so, in most cases, line numbering is already done.

Page and line numbers are optional entries and are not required to successfully run an RPG program. Columns 1 through 5 are checked for ascending order, and RPG prints an S in the left margin of the RPG listing for any statement that is out of order. If you use SEU to enter the source program, you can request that SEU put the statements in order.

Column 6 (Form Type)

An O must appear in column 6 to identify this line as an output specification.

Column 7 (Comments)

Entry	Explanation
*	Comment line

Use an asterisk in column 7 to identify the line as a comment line. Use comments throughout your program to help document the purpose of a certain section of coding. You can use any character in a comment line. Comments are not instructions to your program; they only document your program.

Columns 7-12 (/EJECT)

Entry	Explanation
/EJECT	The specifications following this entry are to begin on a new page of the compiler listing.

The /EJECT specification is not printed on the compiler listing.

Columns 7-12 (/TITLE)

Entry	Explanation
/TITLE	The heading information (such as a title or security classification) that follows the /TITLE entry appears at the top of each page of the compiler listing. The heading information is entered in columns 14 through 74.

A program can contain more than one /TITLE statement. Each /TITLE statement provides heading information for the compiler listing until the next /TITLE statement is read. To print on the first page of the compiler listing, a /TITLE statement must be the first statement read. Information specified by the /TITLE statement is printed in addition to compiler heading information.

The /TITLE statement causes an eject to the next page before the title is printed. The /TITLE statement is not printed on the compiler listing.

Columns 7-14 (/SPACE)

Entry	Explanation
/SPACEƀn	Line spacing occurs at this point in the compiler listing. Valid entries for n are 1 to 3. If you do not specify n, 1 is assumed.

One blank (ƀ) must come before the value you specify for n. The value you specify for n indicates the number of blank lines to be spaced before the next specification line is printed. If n is greater than the number of lines remaining on the current page, the next specification line is printed on a new page. If you specify just /SPACE, one line is spaced.

/SPACE is not printed on the compiler listing but is replaced by the actual line spacing. The spacing indicated by /SPACE is in addition to the three blank lines that occur between specification types.

Columns 7-14 (Filename)

Entry	Explanation
A valid filename	Same filename that appears on the file description specifications for the output, combined, update, or add file

Use columns 7 through 14 to identify the output file you want to describe. The filename must begin in column 7.

The filename should be specified only on the first line. However, if another output file is specified and more specifications are then required for the first file, the first filename must be repeated in columns 7 through 14 (see Figure 27-2).

Line	Form Type	Filename or Record Name	Type (H/D/T/E)
01	O	FILEA	H
02	O		
03	O		D
04	O		
05	O		
06	O		T
07	O		
08	O		E
09	O		
10	O	FILEB	D
11	O		
12	O		
13	O		T
14	O		

Line	Form Type	Filename or Record Name	Type (H/D/T/E)
01	O	FILEA	H
02	O		
03	O		H
04	O		
05	O	FILEB	H
06	O		
07	O		D
08	O		
09	O	FILEA	D
10	O		
11	O		D
12	O		
13	O		T
14	O		
15	O		T
16	O		
17	O	FILEB	T
18	O		
19	O	FILEA	E
20	O		

Note: The filename need not be repeated in columns 7 through 14 unless another output file is specified and then further specifications are required for the first file.

Figure 27-2. Specifying Filename

Columns 14-16 (AND/OR)

Entry	Explanation
AND or OR	AND/OR indicates a relationship between lines of output indicators.

Use columns 14 through 16 to specify AND/OR lines for output operations. For an AND relationship, the condition for all indicators must be satisfied before the output operation is done. You can use any number of AND lines for an output operation. For an OR relationship, only one condition is met between several indicators or groups of indicators before the output operation is done. You can use a maximum of 20 OR lines for an output operation. If you use a combination of AND and OR lines for an output operation, you can use any number of AND lines but you cannot use more than 20 OR lines.

You can use AND and OR lines to condition entire output lines, but you must not use them to condition fields. However, you can condition an output field with more than three indicators by using the SETON operation in calculations.

Column 15 (Type)

Entry	Explanation
H	Heading records
D	Detail records
T	Total records
E	Exception records (lines to be written during calculation time)

Use column 15 to indicate the type of record to be written. Column 15 must have an entry for every output record (see Figure 27-2).

Heading Records (H)

Heading records usually contain constants identifying information such as column headings, page number, and date.

Detail Records (D)

Detail records usually contain data that comes directly from the input record or is the result of calculations performed on data from the input record.

Total Records (T)

Total records usually contain data that is the end result of specific calculations on several detail records. Total output cannot be specified for primary or secondary update files. Records can be added to indexed primary and secondary files at total time if add is specified (A in column 66) on the file description specifications.

Exception Records (E)

Exception records are written during calculation time. Exception records can be specified only when the operation code EXCPT is used. See Chapter 28, *Operation Codes*, for more information on the EXCPT operation code.

Columns 16-18 (ADD/DEL)

Entry	Explanation
ADD	Add a record to an indexed, direct, or sequential file defined as an input, output, or update file.
DEL	Delete the last record read on the identified update file.

ADD

When ADD is specified in columns 16 through 18 to add a record to an indexed, direct, or sequential file, column 66 of the file description specifications must contain an A for the file to which records are being added. The output device for this file must be DISK.

The ADD entry must not be used in an OR line. An ADD entry in columns 16 through 18 of the previous line also applies to the record in the OR relationship. For a detailed description of adding records to a file, see *Column 66 (File Addition)* in Chapter 21 or *Adding Records* in Chapter 5.

DEL

If a record is to be deleted from a file, the file must be defined as delete-capable when it is built. For more information on defining a delete-capable file, see *FILE Statement* in the *System Reference* manual. If you attempt to delete a record from a file that is not delete-capable, an execution-time error message is displayed.

DEL must be specified in columns 16 through 18 of the main output record line. DEL applies to all the OR extensions to the main line. When records are deleted from a file, the file must be defined as an update file (column 15 of the file description specifications contains U).

Note: Record deletion is not dependent on the file organization and mode of processing entries.

Records are not physically removed from a file when they are deleted. Deleted records are filled with hex FFs.

When a file containing deleted records is processed sequentially or consecutively (primary, secondary, demand, or full-procedural files), a deleted record is not returned to the program when it is accessed. It is bypassed, and the next record is read. This process is repeated until a nondeleted record is found or the end of the file is reached. When a file containing deleted records is processed randomly using CHAIN, the no-record-found indicator is turned on when a deleted record is accessed. If this indicator is not specified in columns 54 and 55 of the calculation specification specifying the CHAIN operation, error message RPG-9035, NO RECORD FOUND ON GET OPERATION FOR FILE, is displayed.

Column 16 (Fetch Overflow or Release)

Entry	Explanation
F	Fetch overflow routine
R	Release the device (display station or SSP-ICF session) after output

Fetch Overflow

Use fetch overflow when printing a particular line causes overflow, and not enough space is left on the page to print the remaining detail, exception, or total output lines. To determine when to fetch the overflow routine, study all possible overflow situations. By counting lines and spaces, you can calculate what happens if overflow occurs on each detail and total line.

Use column 16 to specify fetch overflow for a PRINTER file only. Column 16 of each OR line must contain an F if the overflow routine is to be used for each record in the OR relationship. Fetch overflow cannot be used when an overflow indicator is specified in columns 23 through 31 on the same specification line. If this occurs, the overflow routine is not fetched. Specifying fetch overflow allows you to alter the RPG overflow logic (see *Columns 33-34* in Chapter 21). You can advance forms when total, detail, or exception records are printed instead of waiting for the usual time in the program cycle. The fetched overflow routine does not automatically cause forms to advance; that is, the entry in columns 21 and 22 of the output specifications must contain a 2-digit entry that is less than the number of the lines the printer is currently on. Fetching the overflow routine can prevent printing over the page perforation and can use as much of the page as possible. For more information on fetch overflow, see Chapter 7, *Using a PRINTER File*.

Release

You can release a device from your program after output to that device has been written. To release the device, enter an R in column 16. You can specify OR lines; however, column 16 must contain an R for each OR line. The device is released when that output specification is read during the output operations. If you specify a format name on a field description line for the record that contains an R in column 16, the format is written, and then the device is released.

If the WORKSTN file is a primary file and the program does not have a NEP attribute, RPG sets on the last-record (LR) indicator when all devices have been released. If the program has a NEP attribute, RPG sets on the last-record (LR) indicator when all devices have been released and the system operator enters the STOP SYSTEM command.

Note: For WORKSTN files, a device can be either a display station or an SSP-ICF session.

Columns 17-22 (Spacing and Skipping)

Column 17 (Space Before)

Entry	Explanation
0-3	Number of lines to be spaced before a line is displayed for a CRT file or printed for a PRINTER file.

Column 18 (Space After)

Entry	Explanation
0-3	Number of lines to be spaced after a line is displayed for a CRT file or printed for a PRINTER file.

Columns 19-20 (Skip Before)

Entry	Explanation
01	Display screen is blanked immediately for a CRT file.
01-99	Skip to lines 01 to 99 before printing for PRINTER files.
A0-A9	Skip to lines 100 to 109 before printing for PRINTER files.
B0-B2	Skip to lines 110 to 112 before printing for PRINTER files.

Columns 21-22 (Skip After)

Entry	Explanation
01-99	Skip to lines 01 to 99 after printing for PRINTER files.
A0-A9	Skip to lines 100 to 109 after printing for PRINTER files.
B0-B2	Skip to lines 110 to 112 after printing for PRINTER files.

Use columns 17 through 22 to specify line spacing and skipping for PRINTER and CRT files. Spacing refers to advancing one line at a time, and skipping refers to jumping from one print line to another.

If you make an incorrect entry in these columns, the compiler drops the entry and assumes a blank specification. If columns 17 through 22 are blank, single spacing occurs after each line is printed. You can specify different spacing and skipping for OR lines. If you do not specify spacing or skipping entries for the OR line, spacing and skipping are done according to the specifications for the line that comes before the OR line. You cannot specify spacing or skipping on AND lines.

Columns 23-31 (Output Indicators)

Entry	Explanation
01-99	Any resulting indicator, field indicator, or record-identifying indicator previously specified.
KA-KN, KP-KY	Any command-key indicator previously specified in a SET operation or used with a WORKSTN file.
L0-L9	Any control-level indicators previously specified.
H1-H9	Any halt indicators previously specified.
U1-U8	Any external indicator set prior to program execution.
OA-OG, OV	Any overflow indicator previously assigned to this file.
MR	Matching-record indicator.
LR	Last-record indicator.
1P	First-page indicator. The first-page indicator cannot be specified for a WORKSTN file.

You can specify one indicator in each of the three separate output indicator fields (columns 23 through 25, 26 through 28, and 29 through 31). If these indicators are on, the output operation is done. An N in the column that comes before each indicator (column 23, 26, or 29) means that the output operation is done only if the indicator is not on. This is a negative indicator. No output line should be conditioned by all negative indicators. At least one of the indicators should be positive. You should not specify all negative indicators to condition a heading or detail operation because the operation is performed at the beginning of the program cycle when the first page lines are written.

If no output indicators are specified, the line is produced at output every time that record is checked for output. If no output indicators are specified on a heading or detail line, that record is also produced as output at the beginning of the program cycle.

If you need more than three indicators to condition an output operation, use an AND line or an OR line. For more information, see *Columns 14-16* earlier in this chapter.

Columns 32-37 (Field Name)

In columns 32 through 37, use one of the following types of names to specify each field that is to be written out:

- Any field name or data structure name that you used earlier in this program
- The special words PAGE, PAGE1 through PAGE7, *PLACE, UDATE, UDAY, UMONTH, or UYEAR
- A table name, array name, or array element
- An EXCPT name

Field Names

The field names you use must be the same as the field names on the input specifications (columns 53 through 58) or the calculation specifications (columns 43 through 48). Do not enter a field name if a constant is used in columns 45 through 70. If a field name is entered in columns 32 through 37, columns 7 through 22 must be blank.

Fields can be listed on the specifications sheet in any order because the order in which they appear on the output record is determined by the entry in columns 40 through 43. However, the fields are usually listed in order. If fields overlap, the last field specified is the only field completely written.

The sign (+ or -) of a numeric field is in the units position (rightmost digit). The units position prints as a letter unless the field is edited. See *Column 38 (Edit Codes)* or *Columns 45-70 (Constant or Edit Word).*

Rules for Field Names

A field name can be from 1 to 6 characters long. The first character must be alphabetic. The remaining characters can be any combination of alphameric characters.

Special Words

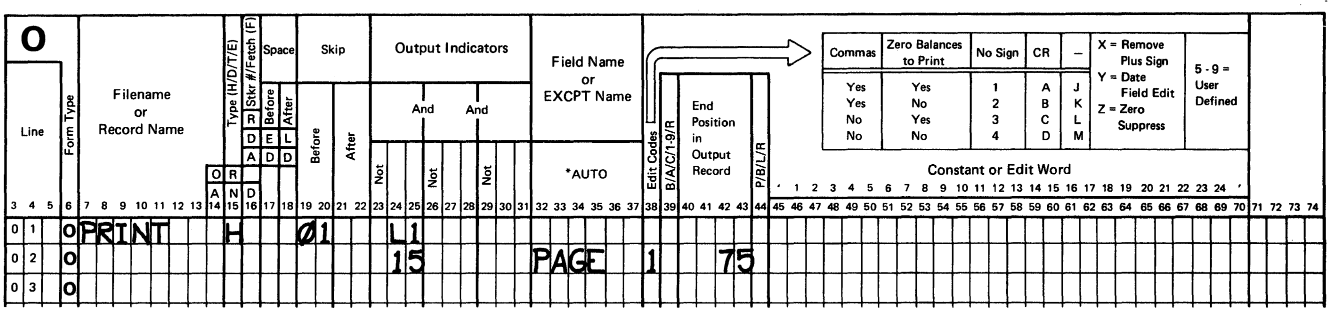
Page Numbering (PAGE, PAGE1-PAGE7)

PAGE is a special word that causes automatic numbering of the pages. Enter the word PAGE or PAGE1 through PAGE7 in these columns if the pages are to be numbered. When a PAGE field is named in these columns without being defined elsewhere, it is assumed to be a 4-digit, numeric field with zero decimal positions. Leading zeros are replaced with blanks automatically. A PAGE field can also be defined on input or calculations specifications as a numeric field from 1 to 15 digits long, with zero decimal positions.

The page number starts with 0001 unless otherwise specified, and 1 is automatically added for each new page. See *Columns 53-58 (Field Name)* in Chapter 25, *Input Specifications*, for information concerning page numbering that starts at a number other than 1.

Page numbering can be restarted at any point in a job. To do this, set the PAGE field to zero before it is printed by specifying either blank after in column 39 or an output indicator. If the status of the indicator is as specified, the PAGE field is reset to zero, and 1 is added to the PAGE field before it is printed (see Figure 27-3).

The eight possible PAGE entries (PAGE, PAGE1 through PAGE7) may be needed for numbering different types of output pages or for numbering pages for different PRINTER files.



When indicator 15 is on, the PAGE field is reset to zero and a 1 is added before the field is printed. When 15 is off, a 1 is added to the contents of the PAGE field before it is printed.

Figure 27-3. Resetting the PAGE Fields to Zero

Repeating Output Fields (*PLACE)

*PLACE is a special RPG word that allows you to write the same fields in several locations on one record without naming the fields and giving their end position each time the fields are to be written. The fields repeated by means of *PLACE are written ending in the position specified in columns 40 through 43 of the same specifications line. For example, if FIELDS A, B, and C appear twice on one record, the fields can be specified in two ways:

- Define each field and its corresponding end position each time the field is to be written (see Figure 27-4).
- Use the special word *PLACE (see Figures 27-4 and 27-5).

Both coding methods shown in Figure 27-4 produce a record that looks like this:

Ending Record Positions	10	20	30	40	50	60	75
Fields	FIELDA	FIELDB	FIELDC	FIELDA	FIELDB	FIELDC	FIELDD

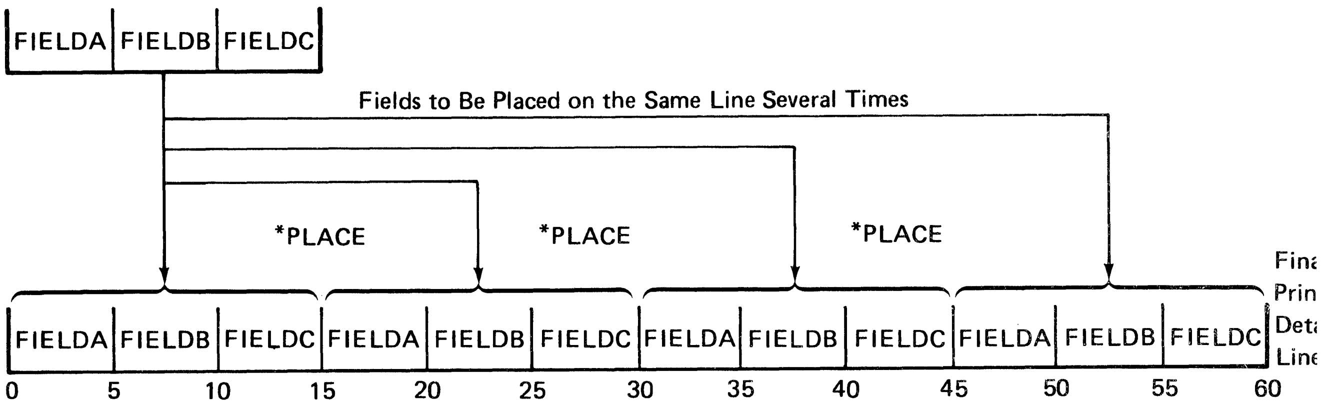
Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Field Name or EXCPT Name	End Position in Output Record
01	O	OUTPUT	D		
02	O			FIELDA	10
03	O			FIELDB	20
04	O			FIELDC	30
05	O			FIELDA	40
06	O			FIELDB	50
07	O			FIELDC	60
08	O			FIELDD	75
09	O				

To repeat an output field, each field can be defined each time it is to be printed or written to disk.

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Field Name or EXCPT Name	End Position in Output Record
01	O	OUTPUT	D		
02	O			FIELDA	10
03	O			FIELDB	20
04	O			FIELDC	30
05	O			*PLACE	60
06	O			FIELDD	75
07	O				

Or the special word *PLACE can be used to repeat a group of fields.

Figure 27-4. Writing Fields Twice on the Same Record



*PLACE can also be used to print the same group of fields several times on the same line. FIELDS A, B, and C are to be printed four times on one line as shown above. They are printed once when they are named and once for every *PLACE entry.

*PLACE is specified after the fields that are to be printed several times on the same line (below). All fields to which *PLACE applies appear on the same record. FIELD D, which appears on the total record, is not affected by *PLACE.

Notice that an end position is given for every *PLACE. FIELDS A, B, and C have a total length of 15 characters. Thus, the end positions given for the *PLACE entries allow room for the printing of 15 characters. This eliminates any overlapping.

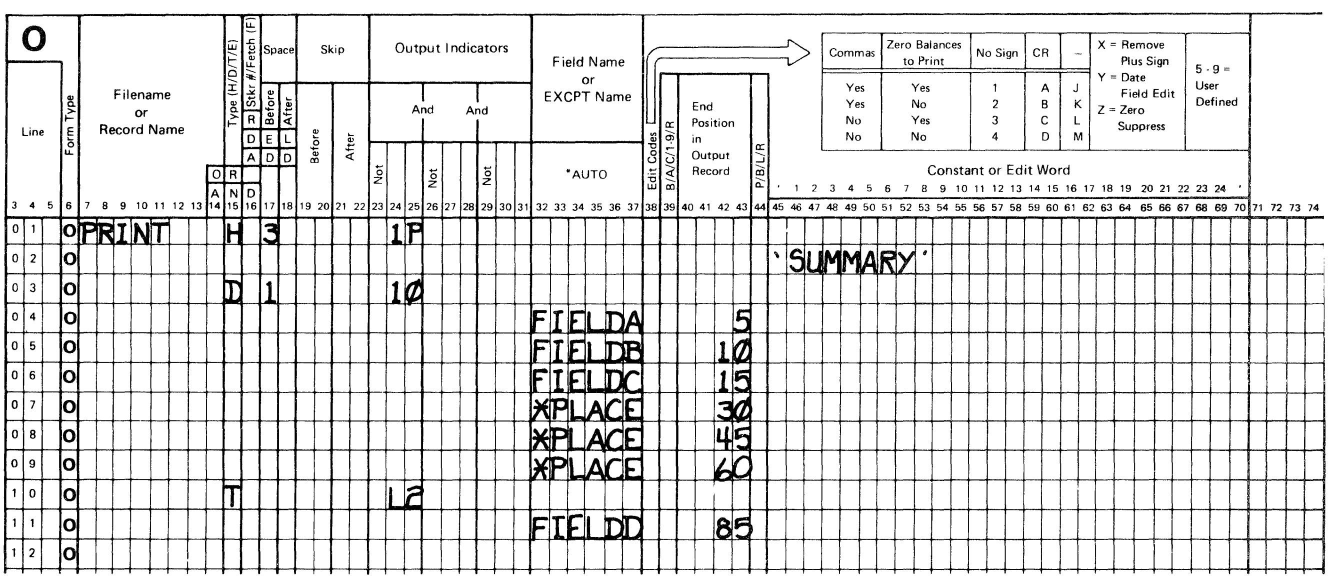


Figure 27-5. *PLACE

When you specify *PLACE, all fields named for each record type (H/D/T/E) are written as usual in the location specified. The entry *PLACE then causes all of these fields to be written ending at the position specified in columns 40 through 43 of the *PLACE specification. When you specify *PLACE, consider the following:

- *PLACE must be specified after the field names that are to be written in different positions in one record (see Figure 27-5).
- *PLACE causes all fields within a record type to be written, not just the field name on the line immediately above the *PLACE entry.
- *PLACE must appear on a separate specification line each time a field or a group of fields is to be written.
- An end position no greater than 256 must be specified for every *PLACE line. Allow enough space for all fields to be written (see Figure 27-5); otherwise, overlapping occurs.
- Multiple or successive *PLACE entries can be specified if the fields preceding the first *PLACE specification are to be repeated more than once.
- The leftmost position of the fields to be written by the *PLACE specification is always assumed to be position 1.
- Additional fields or constants can be specified after the *PLACE specification and are not affected by any preceding *PLACE specification.

Note: Attempts to use the *PLACE function for other than its defined purpose may produce unpredictable results.

Date Fields (UDATE, UMONTH, UDAY, UYEAR)

To have the date printed on a report or program listing, use special words UDATE, UMONTH, UDAY, or UYEAR. The date fields are established at job setup time. UDATE contains the program date that may not be the same as the date in the result field of the TIME operation. The result field of the TIME operation contains the system date. See the *System Reference* manual for a complete discussion of the system date, program date, and the DATE OCL statement. The following rules apply to date fields:

- UDATE prints a 6-character numeric date field in one of three formats:

  ```
  Month/day/year
  ```

  ```
  Year/month/day
  ```

  ```
  Day/month/year
  ```

  Use columns 19 and 20 of the control specifications to specify the date format and the editing to be done. If columns 19 and 20 are blank, the date format is determined by the contents of column 21 of the control specifications.

- Use UDAY for the day only, UMONTH for the month only, and UYEAR for the year only.

- These fields cannot be changed by any operations specified in the program. Thus, these fields are generally used only in compare and test operations.

EXCPT Names

When the record type is an exception record (indicated by an E in column 15), a name can be placed in columns 32 through 37 of the record line. The EXCPT operation can specify the name assigned to a group of records to be written. This name is called an EXCPT name. An EXCPT name must follow the rules for field names. Also, an EXCPT name cannot be the same as a filename, field name, data structure name, array name, table name, label, or subroutine name. A group of any number of output records can use the same EXCPT name, and the records do not have to be consecutive records. The maximum number of different EXCPT names is 64. An EXCPT with a blank name field is counted as one of the 64.

When the EXCPT operation is specified without an EXCPT name, only those exception records without an EXCPT name are checked and written if the conditioning indicators are satisfied.

When the EXCPT operation specifies an EXCPT name, only those exception records with that EXCPT name are checked and written if the conditioning indicators are satisfied.

The EXCPT name is specified on the main record line and applies to all AND/OR lines.

Column 38 (Edit Codes)

Use column 38 to:

- Suppress leading zeros in a numeric field
- Omit a sign from the low-order position of a numeric field
- Punctuate a numeric field without establishing an edit word

For more information on edit codes, see Chapter 16.

Column 39 (Blank After)

Entry	**Explanation**
Blank	Field is not reset.
B	Field specified in columns 32 through 37 is reset to blank or zero after the output operation is complete.

Use column 39 to reset a numeric field to zeros or an alphameric field to blanks. If the field is conditioned by indicators in columns 23 through 31, the blank after is also conditioned. This column must be blank for look-ahead and UDATE fields.

Resetting fields to zeros is useful when totals are accumulated and written for each control group in a program. After the total is accumulated and written for one control group, the total field can be reset to zeros before accumulation begins on the total for the next control group.

If blank after (column 39) is specified for a field to be written more than once, the B should be entered on the last line specifying output for that field. When blank after is specified with a table name, the field that is blanked contains the last element found by a successful LOKUP. If no LOKUP or no successful LOKUP occurred, the first element of the table is blanked.

If the file description specifications for the file to which the field on this output specification is to be written contains an external indicator in columns 70 and 71, you may want to use the same external indicator in columns 23 through 31 of this specification to prevent the field from being blanked when the file is not being used by the program.

Columns 40-43 (End Position in Output Record)

Entry	Explanation
1-4096	End position for DISK or SPECIAL file
1-4075	End position for BSCA file
1-1919	End position for WORKSTN file
1-198	End position for 198-position printer
1-79	End position for CRT file
K1-K8	Length of format name for a WORKSTN file

Use columns 40 through 43 to define the end position of a field or constant on the output record. All entries in these columns must end in column 43. Enter only the position of the rightmost character in the field or constant.

Note: If columns 40 through 43 are left blank, the field or constant is placed in the output record immediately following the field specified in the previous output specification for that record. If no previous field specification exists for the record, the high-order position of the field is placed in position 1. A blank end position with *PLACE causes the *PLACE to be ignored.

Column 44 (Packed-Decimal or Binary Field)

Entry	Explanation
Blank	Field is zoned-decimal numeric data or alphameric data. Leave this column blank for nondisk files.
P	Field is to be written on disk in packed-decimal format.
B	Field is to be written on disk in binary format.

Use column 44 to specify whether a numeric field (decimal number) is to be written to disk or to SSP-ICF output in packed-decimal or binary format. Packed-decimal and binary fields cannot be displayed or printed; these fields can be written only to disk or to SSP-ICF output. Column 44 must be blank for *PLACE.

After decimal numbers are processed, they can be left in the zoned-decimal format. However, for more efficient use of disk space, convert decimal numbers into packed-decimal or binary format. When binary output is specified, a numeric field 1 to 4 digits long (zoned-decimal in storage) is converted into a 2-byte binary field when it is written on disk; a numeric field 5 to 9 digits long is converted into a 4-byte binary field. When packed-decimal output is specified, a byte of disk storage (except for the low-order byte) can contain two decimal numbers. See *Column 43 (Packed or Binary Field)* in Chapter 25, *Input Specifications*, for a description of how data fields are represented in zoned-decimal, packed-decimal, and binary formats.

Note: Although packed-decimal and binary fields require less disk storage space, the conversion routines needed to handle such data increase the program size (and execution time).

Columns 45-70 (Constant or Edit Word)

Use columns 45 through 70 to specify a constant, the format name for a WORKSTN file, or an edit word. If you are using edit codes, you can also use columns 45 through 47 to specify a floating currency symbol or asterisk fill.

Constants

A constant is any unchanging information that is to appear on a report. Constants are usually words used for report headings or column headings.

The following rules apply to constants (see Figure 27-6 for examples):

- Field name (columns 32 through 37) must be blank.
- A constant must be enclosed in apostrophes. Enter the leading apostrophe in column 45.
- An apostrophe in a constant must be represented by two apostrophes. For example, if the word *you're* appears in a constant it must be coded as 'YOU''RE'.
- Numeric data can be used as a constant.
- Up to 24 characters of constant information can be placed in one line. Additional lines can be used, but each line must be treated as a separate line of constants. The end position is specified in columns 40 through 43. If no end position is specified, the constant is placed in the output record immediately following the field or constant specified in the previous output specification line for that record (see *Columns 40-43, End Position in Output Record*, in this chapter).

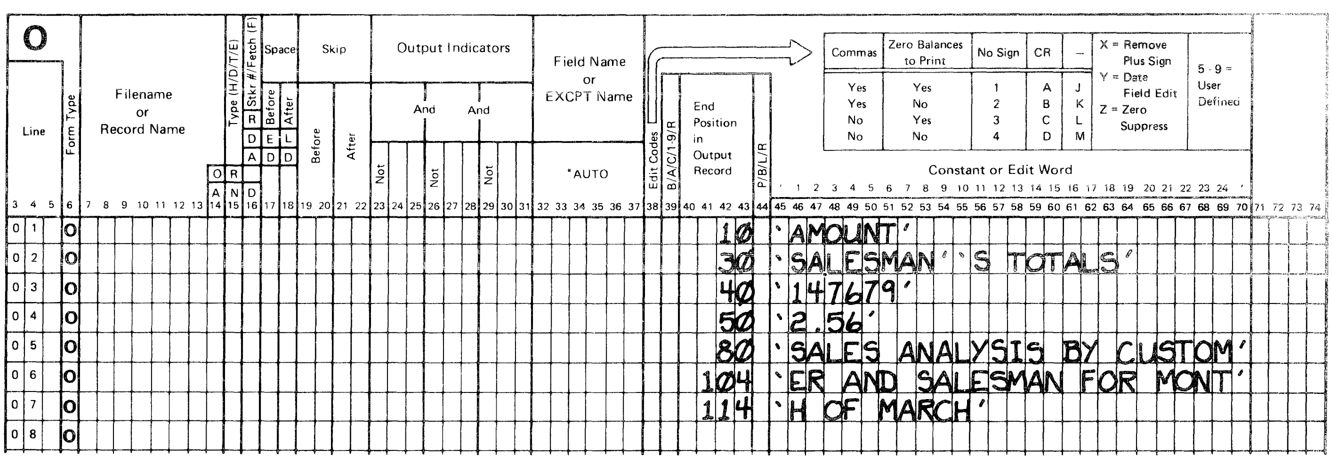


Figure 27-6. Examples of Output Constants

Format Name

The name of the display format that is used by the WORKSTN file must be specified in columns 45 through 54. One format name is required for each output record for the WORKSTN file; the specification of more than one format name per record is not allowed. The format name must be enclosed in apostrophes. This is the same name that is specified in columns 7 through 14 of the S specification line on the display screen format specifications. You must also enter Kn in the rightmost of columns 40 through 43, where n is the length of the format name. For example, if the format name is FORM1, enter K5 in columns 42 and 43.

For more information on the display screen format, see Chapter 6, *Coding an RPG Program That Uses a WORKSTN File.*

Note: The output specifications line containing the format name cannot be conditioned by any indicators.

Edit Words

See Chapter 16 for a complete discussion on edit words.

Columns 71-74

Columns 71 through 74 are not used. Leave them blank.

Columns 75-80 (Program Identification)

Entry	Explanation
Blank	Program identification defaults to the program name specified on the control specification.
Any valid program name	Program identification. The first character must be alphabetic but cannot be #, $, or @. The remaining characters must be alphameric with no imbedded blanks. No special character can be used.

Columns 75 through 80 can contain any characters. These columns can contain the program name used in the control specification, or they can contain any other characters to identify a certain portion of the program. These entries are ignored by the compiler but appear in the source program listing.

Note: To be compatible with other RPG systems, the specifications sheets show only 80 positions for each statement. However, each statement in an RPG source program can contain up to 96 characters. Columns 81 through 96 are available for comments.

Chapter 28. Operation Codes

Chapter 28. Operation Codes

The RPG language allows you to perform many different types of operations on your data. Special codes, which are entered on the calculation specifications, indicate the operations to be performed. Usually these codes are just abbreviations of the names of the operations.

Many operation codes can be placed into categories. The first part of this chapter includes general information about these categories. The rest of the chapter describes each operation code in alphabetical order and shows one or more examples for most of the operations. Figure 28-1 is a summary of the specifications for each operation code.

Operation Code	Control-Level Indicators	Conditioning Indicators	Factor 1	Factor 2	Result Field	Resulting Indicators		
	Columns					Columns		
	7–8	9–17				54–55	56–57	58–59
ACQ	Optional	Optional	Required	Required			Optional	
ADD	Optional	Optional	Optional	Required	Required	Optional	Optional	Optional
BEGSR	SR or blank		Required					
BITOF	Optional	Optional		Required	Required			
BITON	Optional	Optional		Required	Required			
CASxx (CASE)	Optional	Optional	Optional	Optional	Required	Optional	Optional	Optional
CHAIN	Optional	Optional	Required	Required		Optional [1]		
COMP	Optional	Optional	Required	Required		Optional [2]	Optional [2]	Optional [2]
DEBUG	Optional	Optional	Optional	Required	Optional			
DEFN	Optional		*LIKE	Required	Required			
DIV	Optional	Optional	Optional	Required		Optional	Optional	Optional
DO	Optional	Optional	Optional	Optional	Optional			
DOUxx (DO UNTIL)	Optional	Optional	Required	Required				
DOWxx (DO WHILE)	Optional	Optional	Required	Required				
ELSE (ELSE DO)	Optional							
END/DO	Optional	Optional		Optional				
END/DOU	Optional	Optional						
END/DOW	Optional	Optional						
END/IF	Optional							
END/CAS	Optional							
ENDSR	SR or blank		Optional	Optional				
EXCPT	Optional	Optional		Optional				
EXIT	Optional	Optional		Required				
EXSR	Optional	Optional		Required				
FORCE		Optional		Required				
GOTO	Optional	Optional		Required				
IFxx (IF/THEN)	Optional	Optional	Required	Required	Required			
KEYnn [3]	Optional	Optional	Optional			Optional	Optional [4]	Optional
LOKUP(Array)	Optional	Optional	Required	Required	Optional	Optional [4]	Optional [4]	Optional [4]
LOKUP(Table)	Optional	Optional	Required	Required	Required	Optional [4]	Optional	Optional [4]
MHHZO	Optional	Optional		Required	Required			
MHLZO	Optional	Optional		Required	Required			
MLHZO	Optional	Optional		Required	Required			
MLLZO	Optional	Optional		Required	Required			
MOVE	Optional	Optional		Required	Required			
MOVEA	Optional	Optional		Required	Required			
MOVEL	Optional	Optional		Required	Required			
MULT	Optional	Optional	Optional	Required	Required	Optional	Optional	Optional
MVR	Optional	Optional				Optional	Optional	Optional
NEXT	Optional	Optional	Required	Required	Required		Optional	
POST	Optional	Optional	Required				Optional	
READ	Optional	Optional		Required			Optional [5]	Optional [1]
READE	Optional	Optional	Required	Required				Required
READP	Optional	Optional		Required				Required
REL	Optional	Optional	Required	Required			Optional	
RLABL								
SETnn	Optional	Optional	Optional	Optional		Optional	Optional	Optional

Figure 28-1 (Part 1 of 2). Summary of Operation Code Specifications

Operation Code	Control-Level Indicators	Conditioning Indicators	Factor 1	Factor 2	Result Field	Resulting Indicators		
	Columns					Columns		
	7-8	9-17				54-55	56-57	58-59
SETOF	Optional	Optional				Optional [2]	Optional [2]	Optional [2]
SETON	Optional	Optional				Optional [2]	Optional [2]	Optional [2]
SETLL	Optional	Optional	Required	Required				
SHTDN	Optional	Optional				Required		
SORTA	Optional	Optional		Required				
SQRT	Optional	Optional		Required	Required			
SUB	Optional	Optional	Optional	Required	Required	Optional	Optional	Optional
TAG	Optional		Required					
TESTB	Optional	Optional		Required	Required	Optional [2]	Optional [2]	Optional [2]
TESTZ	Optional	Optional			Required	Optional [2]	Optional [2]	Optional [2]
TIME	Optional	Optional			Required			
XFOOT	Optional	Optional		Required	Required	Optional	Optional	Optional
Z-ADD	Optional	Optional		Required	Required	Optional	Optional	Optional
Z-SUB	Optional	Optional		Required	Required	Optional	Optional	Optional

Fields without entries must be blank.

SR = The only allowable nonblank characters in columns 7 and 8 for the BEGSR and ENDSR operation codes

[1] This indicator is required if the file specified in factor 2 is a full-procedural file.
[2] At least one resulting indicator must be specified in columns 54 through 59.
[3] The nn entries in columns 31 and 32 are for message indicator numbers. If the result field of a SET operation contains the keyword ERASE, factor 2 must contain the name of the CONSOLE file. Otherwise, factor 2 and the result field must be blank.
[4] At least one resulting indicator must be specified in columns 54 through 59, but no more than two can be used.
[5] Columns 56 and 57 can contain an indicator when the READ operation is used with a WORKSTN device.

Figure 28-1 (Part 2 of 2). Summary of Operation Code Specifications

Arithmetic Operations

Arithmetic operations (ADD, SUB, MULT, DIV, Z-ADD, and Z-SUB) can be performed only on numeric fields or numeric literals. The result field must also be numeric. Decimal alignment is performed for all arithmetic operations. Even though truncation can occur, the position of the decimal point in the result field is not affected. For arithmetic operations in which all three fields are used:

- Factor 1, factor 2, and the result field can be three different fields.
- Factor 1, factor 2, and the result field can all be the same field.
- Factor 1 and factor 2 can be the same field but different from the result field.
- Either factor 1 or factor 2 can be the same as the result field.

The length of any field specified in an arithmetic operation cannot exceed 15 characters. If the result exceeds 15 characters, characters are dropped from either or both ends depending on the location of the decimal point. The results of all operations are signed (+ or -). Any data placed in the result field replaces the data that was there before.

Move Operations

Move operations (MOVE, MOVEA, and MOVEL) move all or part of factor 2 to the result field. Factor 2 remains unchanged. Factor 1 must be blank, and no resulting indicators can be specified in columns 54 through 59.

The MOVE and MOVEL operations can be used to change numeric fields to alphameric fields and alphameric fields to numeric fields. To change a numeric field to an alphameric field, enter the name of the numeric field in factor 2 and specify an alphameric result field. To change an alphameric field to a numeric field, enter the name of the alphameric field in factor 2 and specify a numeric result field.

When an alphameric field is moved into a numeric result field, the digit portion of each character is converted to its corresponding numeric character and then moved to the result field. Blanks are transferred as zeros. For the MOVE operation, the zone portion of the rightmost alphameric character is converted to its corresponding sign and is moved to the rightmost position of the numeric field where it becomes the sign of the field. For the MOVEL operation, the zone portion of the rightmost character of factor 2 is converted and used as the sign of the result field whether or not the rightmost character is included in the move operation.

When move operations are specified to move data into numeric fields, the decimal positions specified for the factor 2 field are ignored. For example, if the data 1.00 is moved into a numeric field with one decimal position, the result is 10.0.

The MOVEA operation can be used to move several contiguous array elements to a single field, a single field to several contiguous array elements, or contiguous elements of one array to contiguous elements of another array. All arrays and fields used in a MOVEA operation can be alphameric or numeric.

Move Zone Operations

The move zone operations (MHHZO, MHLZO, MLHZO, and MLLZO) move only the zone portion of a character. A minus (-) sign in a move zone operation does not yield a negative character in the result field, because a minus sign is represented by a hex 60 internally and a D zone is required for a negative character. Characters J through R have D zones and can be used to obtain a negative value (J = hex D1, . . . , R = hex D9).

Note: *Whenever the word high is used in a move zone operation, the field involved must be alphameric; whenever low is used, the field involved can be either alphameric or numeric.*

Compare and Testing Operations

The compare and testing operations test fields for certain conditions. These operations are COMP and TESTZ. Another group of compare and testing operations consists of IFxx, DO, DOUxx, DOWxx, and CASxx (structured programming operations). The following rules of comparing fields apply to all compare and testing operations:

- If numeric fields are compared, fields of unequal length are aligned at the implied decimal point. The shorter field is filled with zeros to the left or right of the decimal point to make the fields of equal length. The maximum field length for numeric fields to be compared is 15 digits.

- If alphameric character fields are compared, fields of unequal length are aligned at their leftmost character. The shorter field is filled with blanks to equal the length of the longer field for comparison. The maximum field length for alphameric fields to be compared is 256 characters.

- All numeric comparisons are algebraic. A positive value (+) is always greater than a negative (-) value.

- Blanks within numeric fields are assumed to be zeros.

- Numeric fields are converted to packed-decimal format, if necessary, before they are compared.

- If an alternate collating sequence (position 26 of the control specification) has been specified for the comparison of character fields, the fields are translated into the alternate sequence before comparison.

- An alphameric field cannot be compared to a numeric field.

- An array name and a data structure name cannot be specified in a compare operation, but an array element, a table element, and a data structure subfield can.

With the COMP and TESTZ operations, the resulting indicators assigned in positions 54 through 59 are set according to the results of the operation.

With the CASxx operation, the branch to a subroutine specified in the result field occurs if the condition specified in the xx portion of the operation is met.

No fields are changed by compare and testing operations.

Structured Programming Operations

The structured programming operations are:

- DO (Do)
- DOWxx (Do While)
- DOUxx (Do Until)
- IFxx (If/Then)
- CASxx (Case)
- END (End)
- ELSE (Else Do)

The DO operation allows an operation or a series of operations to be performed a fixed number of times. You indicate how many times the operation(s) will be performed by specifying the starting value in factor 1, incrementing each time by the value in factor 2 of the associated END operation or by 1 if factor 2 on the END operation is not specified, until the index value (result field) exceeds the limit value (factor 2).

The DOWxx and DOUxx operations allow an operation or a series of operations to be performed one or more times based on the results of comparing factor 1 with factor 2.

The IFxx/ELSE operationsallow an operation or a series of operations to be performed based on the results of comparing factor 1 with factor 2.

The CASxx operation allows conditional branching to a subroutine based on the results of comparing factor 1 with factor 2.

The xx portion of the IFxx, DOUxx, DOWxx, and CASxx operations can be:

GT Factor 1 is greater than factor 2.

LT Factor 1 is less than factor 2.

EQ Factor 1 is equal to factor 2.

NE Factor 1 is not equal to factor 2.

GE Factor 1 is greater than or equal to factor 2.

LE Factor 1 is less than or equal to factor 2.

Blanks Factor 1 is not compared to factor 2 (unconditional execution). This is valid for the CASxx operation only, if the xx portion of it contains blanks.

Conditioning indicators can be specified.

Factor 1 and factor 2 can contain a character literal, a numeric literal, an array element, a table name, a data structure subfield, a field name, or blanks (blanks are valid only for CASƀƀ). If factor 1 and factor 2 are not blanks, both must be character data or both must be numeric. Only numeric literals, field names, array elements, table names, or data structure subfields with zero decimal positions can be specified in factor 1 and factor 2 of the DO operation.

The rules for comparing factor 1 and factor 2 on the IFxx, DOUxx, DOWxx, and CASxx operation codes are the same as those given under *Compare and Testing Operations* in this chapter. The same rules apply to comparing a result field (index value) and factor 2 (limit value) on the DO operation.

The group of operations that begins with a DO, DOUxx, DOWxx, or IFxx operation and ends with an END operation, is called a do group. Each do group must end with an END operation, which either ends the do group or allows the do group (except the IF group) to continue executing.

If a do group contains another complete do group, together they form a nested do group. The following is an example of nested do groups, three levels deep:

```
┌──────── DOU/DOW
│    ┌─── DOU/DOW
│    └─── END
│  ┌───── IF
│  │ ┌─── DOU/DOW
│  │ └─── END
│  │ ┌─── IF
│  │ └─── END
│  └───── END
└──────── END
```

Remember the following when specifying do groups:

- Do groups can be nested to a maximum depth of 100 levels.
- Each do group must contain both a DO, DOUxx, DOWxx, or IFxx operation and an associated END operation.
- A do group must be contained in either detail, total, LR, or subroutine calculations; it cannot be split between the different calculation times.
- If you branch into a do group from outside the do group, the loop controls will not have been initialized, which may yield undesirable results.

A CAS group can contain only CASxx operations. An END operation must follow the last CASxx operation to denote the end of the CAS group.

After the subroutine is executed, the program continues at the next operation following the END operation for the CAS group, unless the subroutine is the INFSR and factor 2 on the ENDSR operation is not blank, in which case the return point for the subroutine is specified in factor 2 of the ENDSR operation.

Bit Operations

The bit operations (BITON, BITOF, and TESTB) set and test individual bits. Use the individual bits as switches in a program in order to save storage for binary-type switches.

When you use the BITON, BITOF, and TESTB operations, any field named in factor 2 or result field must be a one-position alphameric field. A field is considered alphameric if there are no entries in the decimal positions column of the input or calculation specifications. The field specified as factor 2 or as the result field can be an array element if each element in the array is a one-position alphameric element.

If a field is defined in a BITON or BITOF operation, the initial value of the field is hex 40.

SETON and SETOF Operations

The operation codes SETON and SETOF turn indicators on or off. Any indicator to be turned on or off is specified in columns 54 through 59. The headings for these columns (plus or high, minus or low, zero or equal) have no meaning in these operations. When setting indicators, consider:

- The following indicators cannot be turned on by the SETON operation: first-page (1P), matching-record (MR), level-zero (L0), command-key (KA through KN, KP through KY).

- The following indicators cannot be turned off by the SETOF operation: first-page (1P), matching-record (MR), level-zero (L0), and last-record (LR).

- If the last-record (LR) indicator is turned on by a SETON operation that is conditioned by a control-level indicator (columns 7 and 8 of the calculation specifications), processing stops after all total output operations are finished. If it is turned on by a SETON operation at detail time (not conditioned by a control-level indicator in columns 7 and 8), processing stops after the next total output operation is completed.

- If the halt indicators (H1 through H9) are turned on and not turned off before the detail output operations are complete, the system stops. The operator can continue processing by responding to the halt for every halt indicator that is on.

- Turning control-level indicators (L1 through L9) on or off does not automatically turn any lower control-level indicators on or off.

- Control-level indicators (L1 through L9) and the record-identifying indicators always turn off after the next detail output operations are completed regardless of the previous SETON or SETOF operation.

- Whenever a new record is read, record-identifying indicators (01 through 99) and field indicators turn on or off to reflect conditions on the new record. The setting from any previous SETON or SETOF operation does not apply then.

- If an indicator with the 2-character entries 01 through 99 is turned on and is not changed in other calculations, it remains on until it is turned off by another calculation specification.

Branching within RPG

Operations are normally performed in the order in which they appear on the calculation specifications. There may be times, however, when the operations should be performed in a different order, such as when:

- Several operations should be skipped when certain conditions occur.

- Certain operations should be performed for several, but not all, record types.

- Several operations should be repeated.

See *Conditional Branching* and *Repeating an Operation* in Chapter 18 for the details.

Subroutine Operations

The operation codes BEGSR, ENDSR, EXSR, and CASxx are used only for subroutines. In an RPG program, a subroutine is a group of calculation specifications that can be performed several times in one program cycle. A subroutine must be coded after all other calculation operations for a program. Subroutine specifications must be identified by SR or blanks in columns 7 and 8 on the calculation specifications. Therefore, individual operations within a subroutine cannot be conditioned by control-level indicators in columns 7 and 8. Within a subroutine, SR or blanks in columns 7 and 8 can be intermixed.

Linking to External Subroutines

To link from an RPG program to an assembler-language subroutine, use the EXIT and RLABL operations. You can use the EXIT and RLABL operation codes to link to the IBM-supplied subroutines SUBR20, SUBR21, SUBR23, and SUBR95.

During compilation, the name of the user library containing the assembler-language subroutines can be specified. The input library name is assumed when the subroutine library name is not specified.

WORKSTN Operations

The operation codes ACQ and REL are used only with the WORKSTN file. For these operations, factor 1 specifies either the name of a 2-character field that contains the device identification or a 2-character alphameric literal that is the device identification. Factor 2 specifies the name of the WORKSTN file for which the operation is requested. Columns 56 and 57 on the calculation specifications can contain a resulting indicator that turns on if an exception or error occurs.

Note: For WORKSTN files, a device can be either a display station or an SSP-ICF session.

Programmed Control of Input and Output

The normal program cycle can be changed to allow input and output operations during calculations. (See Chapter 19 for a description of the program cycle.) The following operations provide this capability:

- EXCPT (Exception Output)
- READ (Read)
- READE (Read Equal Key)
- READP (Read Previous Record)
- FORCE (Force)
- NEXT (Next)
- CHAIN (Chain)
- KEY (Key)
- SET (Set)
- SETLL (Set Lower Limits)

OPERATION CODES

The rest of this chapter discusses individual operation codes in alphabetical order.

ACQ (Acquire)

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional	Optional	Required	ACQ	Required	Blank	Blank	Optional	Blank

The ACQ operation acquires the device specified in factor 1 for the program. Factor 2 must contain the name of the WORKSTN file.

If the device is available, ACQ attaches it to the program. If it is not available or is already attached to the program, an error occurs. If an indicator is specified in columns 56 and 57, the indicator turns on.

If no indicator is specified but the program contains the INFSR (WORKSTN exception/error-processing) subroutine, the INFSR subroutine automatically receives control when an exception or error occurs.

If no indicator is specified and the program does not contain the INFSR subroutine, the program halts when an exception or error occurs. No input or output operation occurs when the ACQ operation is performed. For more information about the ACQ operation, see Chapter 6, *Using WORKSTN Files.*

ADD (Add)

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional	Optional	Optional	ADD	Required	Required	Optional	Optional	Optional

Factor 2 is added to factor 1. The sum is placed in the result field. Factor 1 and factor 2 are not changed by the operation. If factor 1 is not present, factor 2 is added to the result field, and the sum is placed in the result field.

BEGSR (Begin Subroutine)

Indicators						Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Result Field	54-55	56-57	58-59
Optional: SR	Blank	Required	BEGSR	Blank	Blank	Blank	Blank	Blank

The BEGSR operation serves as the beginning point of a subroutine. Factor 1 must contain the name of the subroutine. The control-level entry (columns 7 and 8) can be SR or blank. Columns 9 through 17 must not contain any conditioning indicators.

The subroutine name can be from 1 to 6 characters long. It must begin with an alphabetic character in column 18. The remaining characters can be any combination of alphabetic or numeric characters. However, special characters are not allowed, and blanks cannot appear between characters in the name. Every subroutine must have a different name. This name cannot be used as the label of a TAG or ENDSR operation.

BITOF (Set Bit Off)

Indicators						Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Result Field	54-55	56-57	58-59
Optional	Optional	Blank	BITOF	Required	Required	Blank	Blank	Blank

The BITOF operation causes bits identified in factor 2 to be set off (be set to 0) in the field named as the result field. Factor 2 is always a source of bits for the result field. The result field is the field in which the bits are set off.

Factor 2 can contain:

- *Bit numbers 0-7*: From 1 to 8 bits can be set off per operation. The bits to be set off are identified by the numbers 0 through 7 (0 is the leftmost bit). The bit numbers must be enclosed in apostrophes, and the entry must begin in column 33. For example, to set off bits 0, 2, and 5, enter '025' in factor 2.

- *Field name* The name of a one-position alphameric field, array element, or table element can be specified in factor 2. In this case, the bits that are on in the field, array element, or table element are set off in the result field; bits that are off are not affected.

See Figure 28-2 for a summary of BITOF operations.

The operation code BITOF must appear in columns 28 through 32. Conditioning indicators can be used in columns 7 through 17. However, factor 1, decimal positions, half-adjust, and the resulting-indicator columns must be blank.

The following BITOF operation sets bit 5 off in the field named BITSW. The field is defined in the same line with a field length of 1.

```
BITOF'5'       BITSW    1
```

The following operation sets bits 1, 2, 4, and 6 off in the field named BITSW. The one-position field has been previously defined.

```
BITOF'1246'    BITSW
```

The following operation uses a one-position alphameric field as a source of bits. Any bits that are on in the field named ALPHA cause corresponding bits to be set off in the field named BITSW. If bits 5 and 7 are on in the field named ALPHA, the BITOF operation sets bits 5 and 7 off in the field named BITSW.

```
BITOFALPHA     BITSW
```

The following operations use a one-position alphameric array element either as a source of bits or as a result field, or both. In the first operation, any bits that are on in the field named ALPHA cause corresponding bits to be set off in the array element ARR,NX.

```
BITOFALPHA     ARR,NX
BITOF'137'     ARR,NX
BITOFARR,NX    ARE,12
```

BITS is a one-position field containing hex F0 (numeric zero). To change hex F0 to hex 40 (blank), set bits 0, 2, and 3 off:

```
BITOF'023'     BITS
```

To create a hex 1C (dup character) in the one-position field ASTRSK, set all bits off, then set on bits 3, 4, and 5.

```
BITOF'01234567'ASTRSK
BITON'345'     ASTRSK
```

*Number of sheets per pad may vary slightly.

Figure 28-2. Summary of BITOF Operations

BITON (Set Bit On)

Indicators						Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Result Field	54-55	56-57	58-59
Optional	Optional	Blank	BITON	Required	Required	Blank	Blank	Blank

The BITON operation causes bits identified in factor 2 to be set on (be set to 1) in the field named as the result field. Factor 2 is always a source of bits for the result field. The result field is the field in which the bits are set on.

Factor 2 can contain:

- *Bit numbers 0-7*: From 1 to 8 bits can be set on per operation. The bits to be set on are identified by the numbers 0 through 7 (0 is the leftmost bit). The bit numbers must be enclosed in apostrophes, and the entry must begin in column 33. For example, to set on bits 0, 2, and 5, enter '025' in factor 2.

- *Field name*: The name of a one-position alphameric field, array element, or table element can be specified in factor 2. In this case, the bits that are on in the field, array element, or table element are set on in the result field; bits that are off are not affected.

See Figure 28-3 for a summary of BITON operations.

The operation code BITON must appear in columns 28 through 32. Conditioning indicators can be used in columns 7 through 17. However, factor 1, decimal positions, half-adjust, and the resulting-indicator columns must be blank.

Line	Form Type	Factor 1	Operation	Factor 2	Result Field Name	Length
01–04	C	The following BITON operation sets bit 4 on in the field named BITS. The field is defined in the same line with a field length of 1.				
05	C		BITON	'4'	BITS	1
07–10	C	The following operation sets bits 0, 3, 5, and 7 on in the field named BITS. This one-position field has been previously defined.				
11	C		BITON	'0357'	BITS	
13–17	C	The following operation uses a one-position alphameric field as a source of bits. Any bits that are on in the field named ALPHA cause corresponding bits to be set on in the field named BITS. If bits 5 and 7 are on in the field named ALPHA, the BITON operation sets bits 5 and 7 on in the field named BITS.				
19	C		BITON	ALPHA	BITS	
	C	The following operations use a one-position alphameric array element either as a source of bits or as a result field, or both. In the first operation, any bits that are on in the array element ARR,NX cause corresponding bits to be set on in the array element ARE,12.				
02	C		BITON	ARR,NX	ARE,12	
04	C		BITON	'0246'	ARR,NX	
06	C		BITON	ALPHA	ARR,NX	

Figure 28-3. Summary of BITON Operations

CASxx (Case)

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional	Optional	Optional	CASxx	Optional	Required	Optional	Optional	Optional

The CASxx operation allows a subroutine to be conditionally selected for execution. The selection is based on the relationship between factor 1 and factor 2, as specified by the xx portion of the CASxx operation. See *Structured Programming Operations* in this chapter for options available under the xx portion of the CASxx operation.

Conditioning indicators can be specified. Conditioning indicators on the CASxx operation control whether this particular CASxx operation is performed.

Factor 1 and factor 2 can contain a character literal, a numeric literal, an array element, a table element, a data structure subfield, a field name, or blanks (blanks are valid only for CASƀƀ). If factor 1 and factor 2 are not blanks, both must be character data or both must be numeric. The rules for comparing factor 1 and factor 2 on the CASxx operation are the same as those given under *Compare and Testing Operations* in this chapter.

The result field must contain the name of a valid RPG subroutine. If the relationship denoted by xx exists between factor 1 and factor 2, control passes to the subroutine specified in the result field. If the relationship denoted by xx does not exist, the program continues with the next CASxx operation in the CAS group.

A CAS group can contain only CASxx operations. An END operation must follow the last CASxx operation to denote the end of the CAS group. After the subroutine is executed, the program continues at the next operation following the END operation for the CAS group.

Note: For the INFSR subroutine, an optional factor 2 entry on the ENDSR operation specifies the return point for the subroutine. See *Coding the INFSR Subroutine* in Chapter 6 for the description of entries allowed in factor 2 of the ENDSR operation used with the INFSR subroutine. For all other subroutines, factor 2 of the ENDSR operation must be blank.

You must not use conditioning indicators on the END operation for a CAS group.

In a CASƀƀ operation, factor 1 and factor 2 are required only if resulting indicators are specified in positions 54 through 59.

The CASƀƀ operation with no resulting indicators specified in position 54 through 59 is functionally identical to an EXSR operation because it causes the unconditional execution of the subroutine named in the result field of the CASƀƀ operation. Any CASxx operations that follow an unconditional CASƀƀ operation in the same CAS group are never tested. Therefore, the normal placement of an unconditional CASƀƀ operation is after all other CASxx operations in the CAS group.

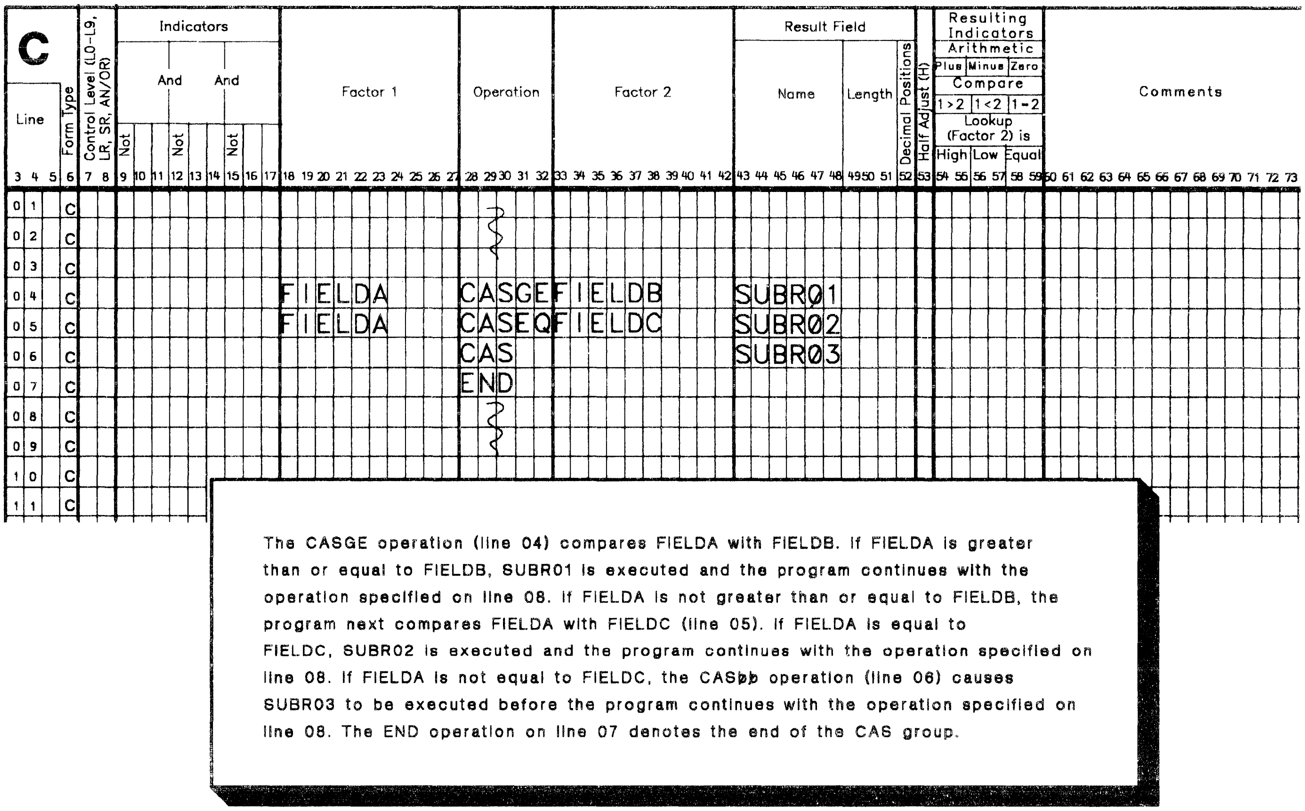


Figure 28-4. CASxx Operation

CHAIN (Chain)

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional	Optional	Required	CHAIN	Required	Blank	Optional	Blank	Blank

The CHAIN operation causes one record to be read from a DISK file during calculations. The CHAIN operation can be used either to read records randomly from a sequential, direct, or indexed file, or to load a direct file that does not allow deletions. For more information on loading a direct file that does not allow deletions, see *Direct Files* in Chapter 5.

Enter the operation code CHAIN in columns 28 through 32. Factor 1 defines the relative record number or the key field of the record to be selected for processing. If you wish to use noncontiguous keys, the key field name may be the name of a data structure subfield. Alternative Indexes and data structures may be used to create noncontiguous key fields. See *Creating an Alternative Index File for an Indexed File* in Chapter 5. Factor 2 names the chained file or full-procedural file from which the record is read. This file must be defined with a C or F entry in column 16 of the file description specifications.

Indicators can be used in columns 7 through 17, but columns 43 through 53 and 56 through 59 must be blank. If the chained file is conditioned by an external indicator on the file description specifications, the CHAIN statement should be conditioned by the same external indicator. A maximum of 15 full-procedural, chained, and/or demand files are allowed per program.

Columns 54 and 55 should specify an indicator. This indicator must be specified for full-procedural files. If the record is not found (or, for a direct file load, if the record location does not exist in the file), the indicator turns on. No update is permitted to a chained update file when the specified record is not found; however, adding records to a file is allowed. Records with duplicate key fields are possible in the file after an unsuccessful chain to an update-add file if the key field is changed before an add to the file. If the original record is found, the indicator turns off.

If an indicator is not specified in columns 54 and 55 and the record is not found, the program halts, and the person using the display station must respond to the error message. When chaining to a file with key fields in packed-decimal format, the field specified in factor 1 of the CHAIN operation must have a packed-decimal length that is the same as the length of the key field in the chained file. Packed-decimal key fields can be up to 8 bytes long. The packed-decimal field equivalents for zoned-decimal fields up to 15 bytes long are shown in a chart under *Packed-Decimal Format* in Chapter 25, *Input Specifications*.

Note: If you chain to one or more files during the same RPG cycle, record-identifying indicators assigned to the chained file or files remain on throughout the cycle if the previous chain operations were performed successfully. If you chain to the same file more than once during an RPG cycle, only the last record processed is updated during output time unless an exception output is associated with each CHAIN operation.

Random Processing

To read a record from a sequential or direct file with the CHAIN operation, the record must be identified by relative record number. To read a record from an indexed file with the CHAIN operation, the record must be identified by a key field. A field can be specified to contain the relative record number or key field.

If the record has been deleted from the file, the no-record-found indicator is turned on. If the no-record-found indicator is not specified, a message is displayed.

Factor 1 must contain a relative record number, a key field, or the name of a field that contains a relative record number or key field. Factor 2 must contain the name of the file from which the record is read.

Figure 28-5 shows an example of chaining to and updating an indexed file.

Line	Form Type	Filename	File Type	File Designation	End of File	Sequence	File Format	Block Length	Record Length	Length of Key Field	Record Address Type	Type of File Organization	Key Field Starting Location	Device
02	F	RECIN	I	P	E		F	96	96					DISK
03	F	MASTINV	U	C			F	120	120	R 9	A	I	1	DISK
04	F	MESSAGE	I	D			F	9	9					KEYBORD
05	F													

RECIN file consists of records sorted by item number, with each record containing a quantity ordered.

Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator	Position	Not (N)	C/Z/D	Character	From	To	Decimal Positions	RPG Field Name	Control Level (L1-L9)
01	I	RECIN	AA	01	96		C	X					
02	I								21	29	0	ITEMNO	L1
03	I								30	36	0	QTY	
04	I	MASTINV	BB	02	120		C	1					
05	I								1	9	0	ITEMNO	
06	I								10	18	0	QOH	
07	I												

ITEMNO is used as a control field. When all the quantities for one item number are added, a control break occurs.

Figure 28-5 (Part 1 of 2). CHAIN Operation

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Resulting Indicators (High / Plus)	Comments
01	C		Ø1	QTY	ADD	TOTQTY	TOTQTY	7	Ø		DETAIL CALCS
02	C	L1		ITEMNO	CHAIN	MASTINV				2Ø	FIND MASTER REC
03	C	L1	2Ø	ITEMNO	SET						FOUND?NO-DISPLY
04	C	L1	N2Ø	QOH	SUB	TOTQTY	QOH				YES-TOTAL CALCS
05	C				Other total calculations						
06	C										
07	C										

The CHAIN operation then uses ITEMNO to find the master record and update it. If it is not found, indicator 20 turns on and a SET operation displays the item number on the screen. If the master record is found, the total quantity for the item number is subtracted from the quantity on hand.

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Output Indicators	Field Name or EXCPT Name	End Position in Output Record
01	O	MASTINV	T	L1 Ø2		
02	O				QOH	18
03	O					

After the total calculations, the QOH field in the master record is updated.

Figure 28-5 (Part 2 of 2). CHAIN Operation

COMP (Compare)

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional	Optional	Required	COMP	Required	Blank	One required		

The COMP operation compares factor 1 with factor 2. As a result of the compare, indicators turn on as follows:

High Factor 1 is greater than factor 2.

Low Factor 1 is less than factor 2.

Equal Factor 1 equals factor 2.

Indicators for conditions not met remain off, or turn off if they had been turned on previously.

Factor 1 and factor 2 must be both alphameric or both numeric.

At least one resulting indicator must be specified in positions 54 through 59.

The fields are automatically aligned before they are compared. If the fields are alphameric, they are aligned on their leftmost character. If one is shorter, the unused positions are filled with blanks (see Figure 28-6). The maximum field length for alphameric fields to be compared is 256 characters.

If the fields are numeric, they are aligned on the decimal point. Any missing digits are filled with zeros (see Figure 28-7). The maximum field length for numeric fields to be compared is 15 digits.

If an alternative collating sequence is specified, alphameric fields are compared according to the alternative sequence.

Figure 28-8 shows some examples of specifications for compare operations.

C C C C C C

C C C C C C

Equal-Length Alphameric Fields

C C C C C C

C C C ƀ ƀ ƀ

Unequal-Length Alphameric Fields

Figure 28-6. Comparison of Alphameric Fields

1 5 6 7 9 5

2 1 0 5 7 6

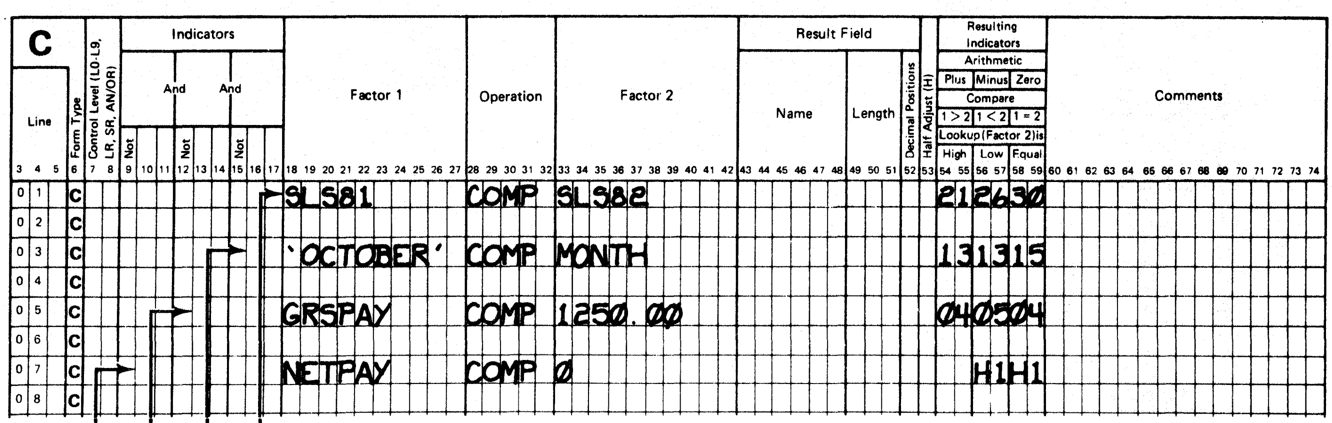
Equal-Length Numeric Fields

0 0 1 7 5 6

2 5 4 3 2 0

Unequal-Length Numeric Fields

Figure 28-7. Comparison of Numeric Fields



The contents of the field SLS81 (1981 sales) are compared with the contents of SLS82. If 1981 sales exceed 1982 sales, resulting indicator 21 turns on; if they are less, indicator 26 turns on; if the two years had equal sales, indicator 30 turns on.

The alphameric constant OCTOBER is compared with the contents of the field named MONTH, which must also be defined as alphameric. If the MONTH field does not contain the word OCTOBER, indicator 13 turns on; if it does, indicator 15 turns on.

The contents of the field named GRSPAY, which must be defined as numeric, are decimal-aligned with numeric constant 1250.00. If the value in field GRSPAY is greater than or equal to 1250.00, indicator 04 turns on; if its value is less than 1250.00, indicator 05 turns on.

The contents of the field NETPAY, which must be defined as numeric, are decimal-aligned with numeric constant 0 and then compared to it. If NETPAY is greater than zero, indicator H1 remains off; however if NETPAY is zero or negative, indicator H1 turns on.

Figure 28-8. Compare Operations

DEBUG (Debug)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Optional	Optional	Optional	DEBUG	Required	Optional	Blank	Blank	Blank

The DEBUG operation is an RPG function that helps you find errors in a program that is not working properly. Either one or two records containing information helpful for finding programming errors are written to an output file as a result of this operation. All DEBUG output in a program is written to the same file.

The DEBUG operation code can be specified at any point or at several points in the calculation specifications. Whenever the program encounters the DEBUG operation, either one or two records are written, depending upon the specifications entered. The first record contains a list of all indicators that are on at the time the DEBUG operation was performed. The second record, if specified, shows the contents of the field specified in the result field.

Factor 1 can contain a literal or the name of a field to help identify the particular DEBUG operation. The length of the specified field can be from 1 to 8 characters. The contents of the field or the literal are written in the first record. If factor 1 is not used, the RPG-created statement number of the DEBUG operation code is written in the first record. Factor 2 must contain the name of an output file on which the DEBUG lines are written and can be any valid output file. A WORKSTN file is not a valid output file for the DEBUG operation. The same output file name must appear in factor 2 for all DEBUG statements in a program. The result field can contain the name of a field or array whose contents are to be written in the second record. Any valid indicator can be used in columns 7 through 17. Columns 49 through 59 must be blank.

To use the DEBUG operation, you must enter one of the following:

- 1 in column 15 of the control specification
- DEBUG in response to the prompt *Override debug option in source* on the RPGONL, RPGC, or AUTOC procedure. See Chapter 3 for information about these procedures.

If one of these entries is not made, the DEBUG operation code and its conditioning indicators are treated as a comment. See *Column 15 (DEBUG)* in Chapter 20, *Control Specification*, for more information.

Records Written for DEBUG

For a DEBUG operation, the first record is always written and appears in the following format:

Output Positions	Information
1-8	DEBUG--
9-16	Literal, contents of field entered in factor 1 (optional), or the statement number of the DEBUG operation code in the program.
17	Blank
18-32	INDICATORS ON--
33-any position (depending on length of field)	The names of all indicators that are on, each separated by a blank. More than one record may be needed.

The second record is written only when an entry is made in the result field. The record is written in the following format:

Output Positions	Information
1-14	FIELD VALUE--
15-any position (depending on length of field)	The contents of the result field (up to 256 characters). If the result field is an array, more than one output record may be needed to contain the array.

*LIKE DEFN (Field Definition)

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional	Blank	*LIKE	DEFN	Required	Required	Blank	Blank	Blank

The DEFN operation with *LIKE in factor 1 defines a field based on the attributes of another field. These attributes include length, decimal positions, and type (character or numeric).

Factor 1 must always contain the entry *LIKE.

Factor 2 must contain the name of the field that provides the attributes for the field being defined. Factor 2 cannot be a literal or a data structure name. If factor 2 is an array, an array element, or a table name, the attributes of an element of the array or table are used to define the field.

The result field must contain the name of the field being defined. The result field cannot be an array, an array element, a table name, or a data structure name.

The DEFN operation can be specified anywhere within calculations. The control-level entry can be blank or can contain an L1 through L9 indicator, the LR indicator, or an L0 entry to group the statement within the appropriate section of the program. Conditioning indicator entries (columns 9 through 17) are not permitted.

Columns 49 through 51 (field length) can be used to make the result field entry longer or shorter than the factor 2 entry. A plus sign (+) in column 49 indicates a length increase; a minus sign (-) in column 49 indicates a length decrease. Columns 50 and 51 can contain the increase or decrease in length (right-adjusted) or can be blank. If columns 49 through 51 are blank, the result field entry is defined with the same length as the factor 2 entry.

The number of decimal positions in the new field cannot be determined by an entry in column 52 (decimal positions). The new field will have the same number of decimal positions as the factor 2 entry.

Resulting indicators are not permitted.

See figure 28-9 for examples of *LIKE DEFN.

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators	Comments
01	C			*LIKE 1	DEFN	FLDA	FLDP					
02	C	*										
03	C			*LIKE 2	DEFN	FLDA	FLDQ	+ 2				
04	C	*										
05	C			*LIKE 3	DEFN	FLDA	FLDR	- 1				
06	C	*										
07	C			*LIKE 4	DEFN	FLDB	FLDS					
08	C	*										
09	C			*LIKE 5	DEFN	FLDB	FLDT	+ 1				
10	C	*										
11	C			*LIKE 6	DEFN	FLDB	FLDU	- 2				
12	C	*										
13	C			*LIKE 7	DEFN	FLDU	FLDX					
14	C											

FLDA is a 7-position character field.

FLDB is a 5-digit field with 2 decimal positions.

1 FLDP is a 7-position character field.

2 FLDQ is a 9-position character field.

3 FLDR is a 6-position character field.

4 FLDS is a 5-position numeric field with 2 decimal positions.

5 FLDT is a 6-position numeric field with 2 decimal positions.

6 FLDU is a 3-position numeric field with 2 decimal positions.

7 FLDX is a 3-position numeric field with 2 decimal positions.

Figure 28-9. DEFN Operations.

DIV (Divide)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Optional	Optional	Optional	DIV	Required	Required	Optional	Optional	Optional

Factor 1 (dividend) is divided by factor 2 (divisor). The quotient (result) is placed in the result field. Factor 1 and factor 2 are not changed. If factor 1 is 0, the result of the divide operation is 0. Factor 2 cannot be 0. If it is, the job stops immediately. The person using the display station can continue processing, however, by responding to the error message. When processing continues, the result and remainder are set to 0. If factor 1 is not present, the result field is divided by factor 2, and the quotient is placed in the result field. Any remainder resulting from the divide operation is lost unless the move remainder (MVR) operation is specified as the next operation. When you use the move remainder operation to save the remainder, you cannot half-adjust the result of the divide operation.

DO (Do)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Optional	Optional	Optional (start)	DO	Optional (limit)	Optional (index)	Blank	Blank	Blank

The DO operation begins a group of operations you want to perform a fixed number of times. You indicate how many times this group of operations has to be performed by specifying a starting value, a limit value, and an index value on the DO statement line.

An associated END operation marks the end of the do group.

In factor 1 (starting value) specify a numeric literal, a numeric field name, an array element, a table name, or a data structure subfield with zero decimal positions. If factor 1 is not specified, the starting value is assumed to be 1.

In factor 2 (limit value) specify a numeric literal, a numeric field name, an array element, a table name, or a data structure subfield with zero decimal positions. If factor 2 is not specified, the limit value is assumed to be 1.

In the result field, which is used to contain the current index value, specify a numeric field name, an array element, a table name, or a data structure subfield with zero decimal positions. If you do not specify an index field, the compiler will generate one for internal use.

Note that any value in the index field is replaced by factor 1 when the DO operation begins.

An increment value of the DO operation is specified in factor 2 on the associated END statement line. It can be a numeric positive literal, field, array element, table name, or data structure subfield with zero decimal positions. If factor 2 on the associated END operation is not specified, the increment value is 1.

Conditioning indicators can be specified. In addition to the DO operation itself, the conditioning indicators on the DO and END statements control the do group, as explained below:

1. If the conditioning indicators on the DO statement are satisfied, the DO operation is performed (step 2). If the indicators are not satisfied, control passes to the next executable operation following the associated END statement (step 7).

2. RPG begins the DO operation by moving the starting value (factor 1) to the index field (result field).

3. If the index value is greater than the limit value, control passes to the next executable operation following the associated END statement.

Otherwise, control passes to the first operation after the DO statement (step 4).

4. Each of the operations between the DO statement and the associated END statement is performed.

5. If the conditioning indicators on the END statement are not satisfied, control passes to the next executable operation following the associated END statement (step 7). Otherwise, the END operation is performed (step 6).

6. RPG performs the END operation by adding the increment to the index field. Control passes to step 3. Note that, unlike step 1, the conditioning indicators on the DO statement are not tested again when control passes to step 3.

7. The operation after the associated END statement is performed when the conditioning indicators on the DO or END statements are not satisfied (steps 1 or 5), or when the index value is greater than the limit value (step 3).

The rules applied when comparing the result field (index value) and factor 2 (limit value) are the same as those given for numeric comparison under *Compare and Testing Operations* earlier in this chapter.

Remember the following when specifying the DO operation and the associated END operation:

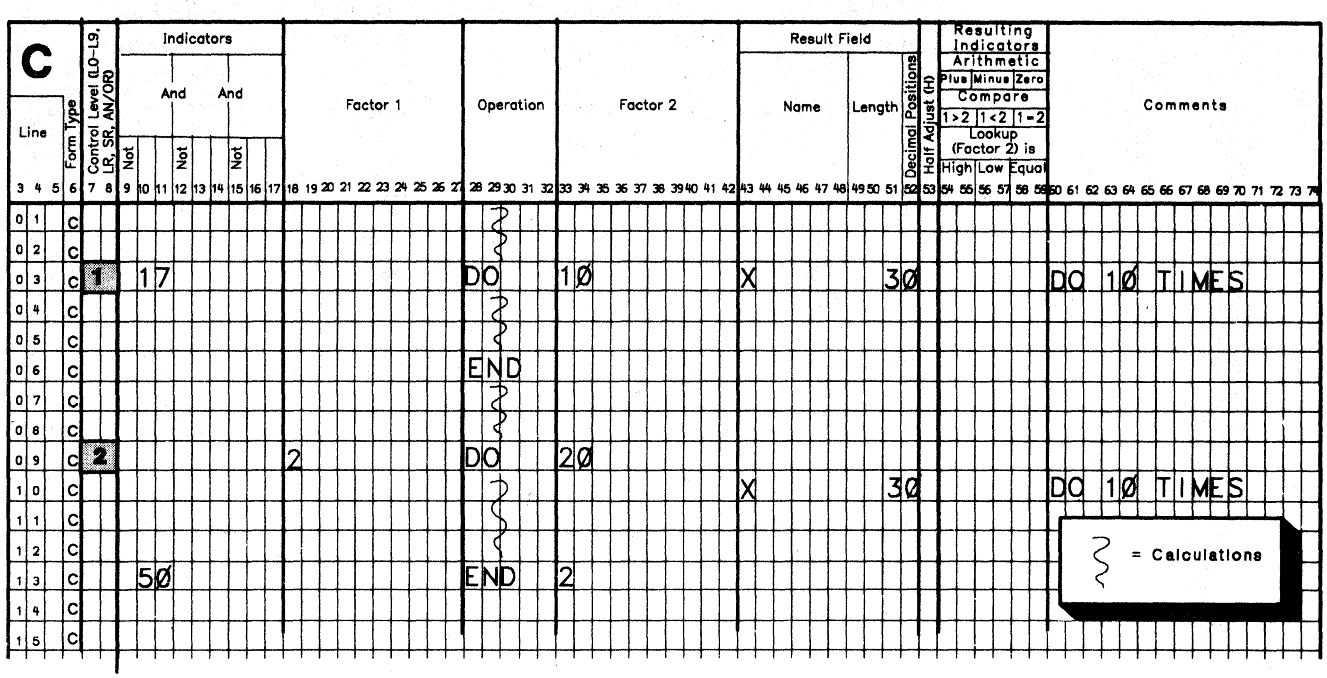
- The limit value you specify in factor 2 must be equal to or greater than the starting value specified in factor 1, otherwise the do group will never be executed.

- Specifying a negative or zero increment value in factor 2 of the associated END operation can cause the program to loop indefinitely.

- The index value, increment, limit value, and indicators can be modified within the do group to affect the termination of the loop.

- The RPG-generated index field always has the length of 15.

Figure 28-10 illustrates how the DO operation works.



Indicator 17 is tested only once. If indicator 17 is off, the execution will continue following the END statement (line 07). If indicator 17 is on, the do group (lines 2 through 6) is executed 10 times. The execution stops when the index value in field X (the result field) is greater than the limit value (10) in factor 2. Control then passes to the operation immediately following the END operation (line 7). Since factor 1 on the DO statement line is not specified, the starting value is 1. Since factor 2 on the END statement line is not specified, the increment value is 1.

The do group (lines 9 through 13) can be executed 10 times. The execution stops when the value in the index field (generated by the compiler since the result field is not specified) is greater than the limit value (20) in factor 2, or if indicator 50 is not on when the END operation is encountered. When indicator 50 is not on, the END operation is not executed; therefore, control passes to the operation following the END operation. The starting value of 2 is specified in factor 1 of the DO operation, and the increment value of 2 is specified in factor 2 of the END operation.

Figure 28-10. DO Operation

DOUxx (Do Until)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7–8	9–17					54–55	56–57	58–59
Optional	Optional	Required	DOUxx	Required	Blank	Blank	Blank	Blank

The DOUxx operation begins a group of operations which are performed until a condition specified in the xx portion of the DOUxx operation is met. See *Structured Programming Operations* in this chapter for options available under the xx portion of the DOUxx operation code. An associated END operation marks the end of the do group.

Factor 1 and factor 2 can contain a character literal, a numeric literal, an array element, a table element, a data structure subfield, or a field name. Both factor 1 and factor 2 must be character data or both must be numeric. The rules for comparing factor 1 and factor 2 on the DOUxx operation are the same as those given under *Compare and Testing Operations* in this chapter.

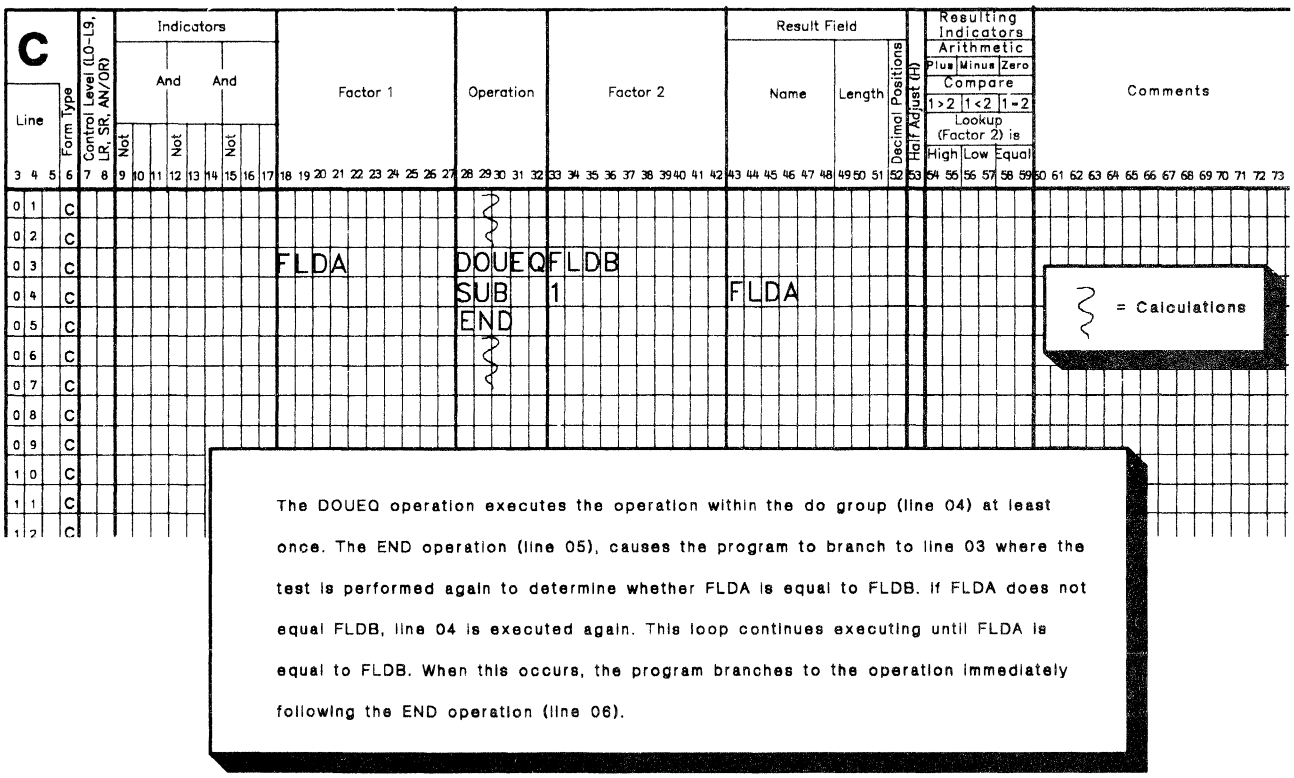


Figure 28-11. DOUxx Operation

Conditioning indicators can be specified. In addition to the DOUxx operation itself, the conditioning indicators on the DOUxx and END operations control the do group. The conditioning indicators on the DOUxx operation control whether the DOUxx operation is begun, while the conditioning indicators on the associated END operation control whether the do group is repeated.

Figure 28-12 shows how the DOUxx operation with conditioning indicators works:

1. If the conditioning indicators on the DOUxx operation are satisfied, the DOUxx operation is performed (line 2). If the indicators are not satisfied, control passes to the operation following the associated END operation (line 6).

2. RPG performs the DOUxx operation by passing control to the next operation (line 3). The DOUxx operation does not compare factor 1 and factor 2 at this point.

3. Each operation of the do group is performed.

4. If the conditioning indicators on the END operation are not satisfied, control passes to the operation following the END operation (line 6). Otherwise, the END operation is performed (line 5).

5. RPG performs the END operation by comparing factor 1 and factor 2 of the DOUxx operation. If the relationship xx exists between factor 1 and factor 2, the do group is finished and control passes to the next operation after the END operation (line 6). If the relationship xx does not exist between factor 1 and factor 2, the operations in the group are repeated (line 3). Note that at this point the conditioning indicators on the DOUxx operation are not tested again.

6. Control passes to the operation following the END operation when the conditioning indicators on the DOUxx or END statements are not satisfied (lines 2 or 5), or when the relationship xx exists between factor 1 and factor 2 at line 5.

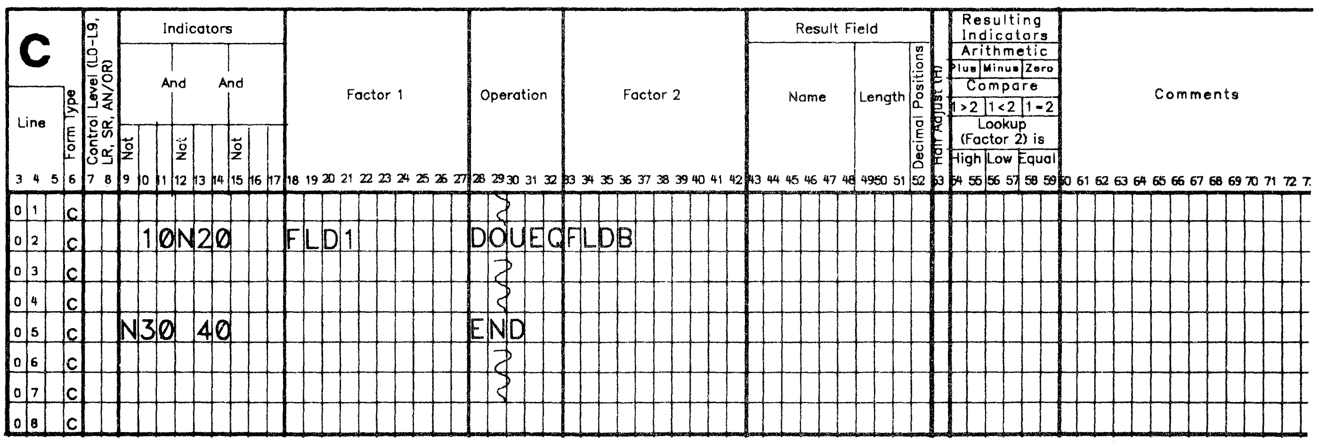


Figure 28-12. DOUxx Operation Using Conditioning Indicators

DOWxx (Do While)

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional	Optional	Required	DOWxx	Required	Blank	Blank	Blank	Blank

The DOWxx operation begins a group of operations performed while the relationship xx exists between factor 1 and factor 2. See *Structured Programming Operations* in this chapter for options available under the xx portion of the DOWxx operation code. An associated END operation marks the end of the do group.

Factor 1 and factor 2 can contain a character literal, a numeric literal, an array element, a table element, a data structure subfield, or a field name. Both factor 1 and factor 2 must be alphameric data, or both must be numeric. The rules for comparing factor 1 and factor 2 on the DOWxx operation are the same as those given under *Compare and Testing Operations* in this chapter.

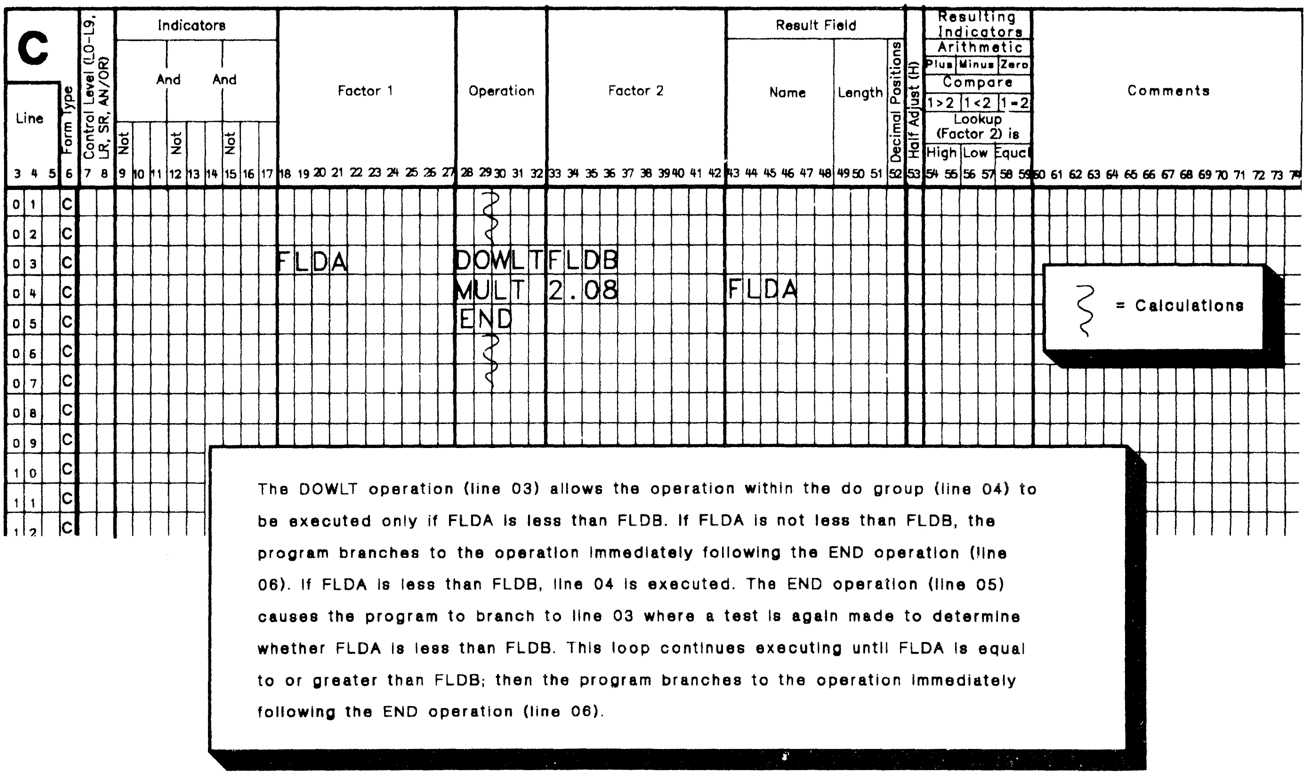


Figure 28-13. DOWxx Operation

Conditioning indicators can be specified. In addition to the DOWxx operation itself, the conditioning indicators on the DOWxx operation control whether the DOWxx operation is begun. The conditioning indicators on the associated END operation control whether the do group is repeated again.

Figure 28-14 shows how the DOWxx operation with conditioning indicators works:

1. If the conditioning indicators on the DOWxx operation are satisfied, the DOWxx operation is performed (line 2). If the indicators are not satisfied, control passes to the operation following the associated END operation (line 6).

2. RPG performs the DOWxx operation by comparing factor 1 and factor 2. If the relationship xx does not exist between factor 1 and factor 2, the do group is finished and control passes to the operation following the END operation (line 6). If the relationship xx does exist between factor 1 and factor 2, the operations in the do group are performed (line 3).

3. Each of the operations in the do group is performed.

4. If the conditioning indicators on the END operation are not satisfied, control passes to the next executable operation following the END operation (line 6). Otherwise, the END operation is performed (line 5).

5. RPG performs the END operation by passing control to the DOWxx operation (line 2). Note that the conditioning indicators on the DOWxx operation at line 2 are not tested again.

6. Control passes to the operation following the END operation when the conditioning indicators on the DOWxx or END operation are not satisfied (lines 2 or 5), or when the relationship xx does not exist between factor 1 and factor 2 at line 2.

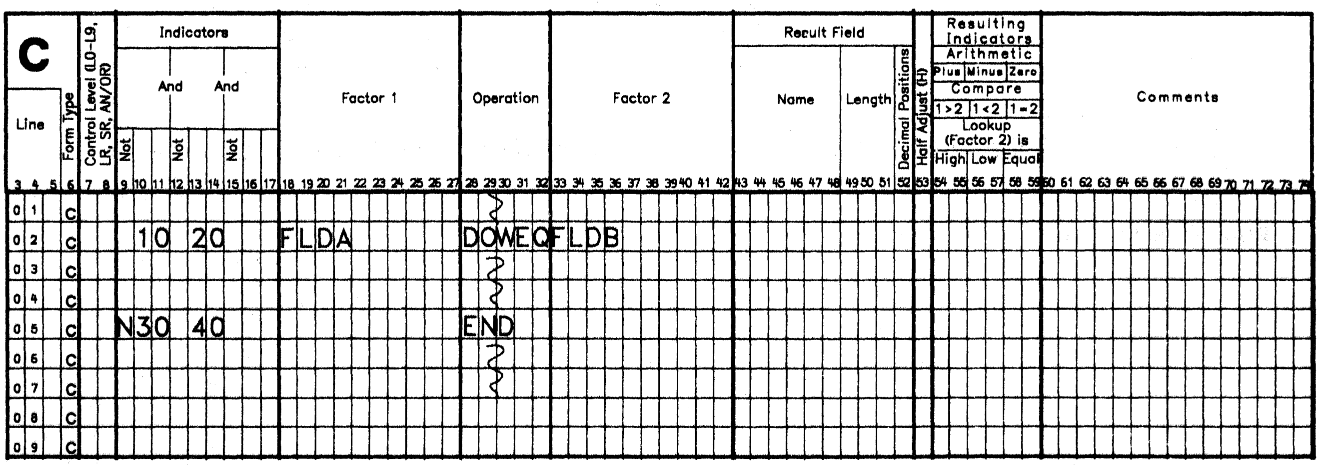


Figure 28-14. DOWxx Operation Using Conditioning Indicators

ELSE (Else Do)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Optional	Blank	Blank	ELSE	Blank	Blank	Blank	Blank	Blank

The ELSE operation is optional with the IFxx operation. ELSE is specified immediately following the calculations that are performed if the IFxx comparison is met, and is immediately followed by the calculations to be performed if the IFxx comparison is not met.

The control level entry (positions 7 and 8) can be blank or can contain an L1 through L9 indicator, an LR indicator, or an L0 entry to group the statements within the appropriate section of the program. The control level entry is for documentation purposes only. Conditioning indicator entries (positions 9 through 17) are not permitted.

An END operation must be used to close the IFxx/ELSE group.

END (End)

The END operation specifies the end of a CASxx, DO, DOUxx, DOWxx, or IFxx group.

This is how the END operation for the DO operation should be specified:

Indicators						Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Result Field	54-55	56-57	58-59
Optional	Optional	Blank	END	Optional (increment)	Blank	Blank	Blank	Blank

The table below shows how the END operation for the DOUxx and DOWxx operations should be specified:

Indicators						Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Result Field	54-55	56-57	58-59
Optional	Optional	Blank	END	Blank	Blank	Blank	Blank	Blank

And the following table shows how the END operation for the IFxx and CASxx operations should be specified:

Indicators						Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Result Field	54-55	56-57	58-59
Optional	Blank	Blank	END	Blank	Blank	Blank	Blank	Blank

Note that a factor 2 entry is allowed only on an END operation associated with a DO operation. In this case, factor 2 of the END operation contains the increment value of the DO operation. If specified, factor 2 must contain a numeric positive value, which can be a literal, a field name, an array element, a table name, or a data structure subfield with zero decimal positions. If factor 2 is not specified on the associated END operation, the increment value of the DO operation is 1.

For an explanation of how conditioning indicators affect the END operation, see descriptions of the DO, DOUxx and DOWxx operation codes in this chapter. Do not use conditioning indicators on the END operation for the CASxx or IFxx operations.

ENDSR (End Subroutine)

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional: SR	Blank	Optional	ENDSR	Optional	Blank	Blank	Blank	Blank

The ENDSR operation defines the end of a subroutine; therefore, it must be the last statement in the subroutine. Factor 1 can contain a name that can be used as a point to which a GOTO operation within the subroutine can branch. The control-level entry (columns 7 and 8) can be SR or blank. Columns 9 through 17 must not contain any conditioning indicators.

The ENDSR operation ends the subroutine and automatically causes a branch back to the statement that follows the EXSR operation unless the subroutine is the INFSR (exception/error-processing) subroutine. For the INFSR subroutine, an optional factor 2 entry on the ENDSR operation specifies the return point for the subroutine. The valid entries for factor 2 for the INFSR subroutine are described in Chapter 6 under *Coding the INFSR Subroutine.* For all other subroutines, factor 2 must not contain an entry.

EXCPT (Exception Output)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Optional	Optional	Blank	EXCPT	Optional	Blank	Blank	Blank	Blank

The EXCPT operation allows your program to write records during detail or total calculation time instead of the normal time during the RPG program cycle. Consider the following when specifying the EXCPT operation:

- On the calculation specifications:
  - Columns 28 through 32 must contain EXCPT to indicate when records are to be written during calculation time.
  - Columns 7 through 17 can contain indicators.
  - Factor 2 can contain an EXCPT name. This EXCPT name can specify a group of exception lines to be written, reducing the need for indicators to condition which exception lines are to be written. This name must follow the rules for field names. See *Rules for Field Names* in Chapter 27.
  - All other columns must be blank.

- On the output specifications:
  - Column 15 must contain an E to indicate the lines that are to be written during calculation time.
  - Columns 23 through 31 can contain indicators with or without group names specified.
  - Columns 32 through 37 can contain an EXCPT name for a group of records to be written during calculation time. This name must be the same name specified in factor 2 on a calculation specification containing the EXCPT operation code. The same name can be on multiple EXCPT output record lines.
  - Columns 38 through 74 must be blank.

- Only exception records, not heading, detail, or total records, can contain an EXCPT name.

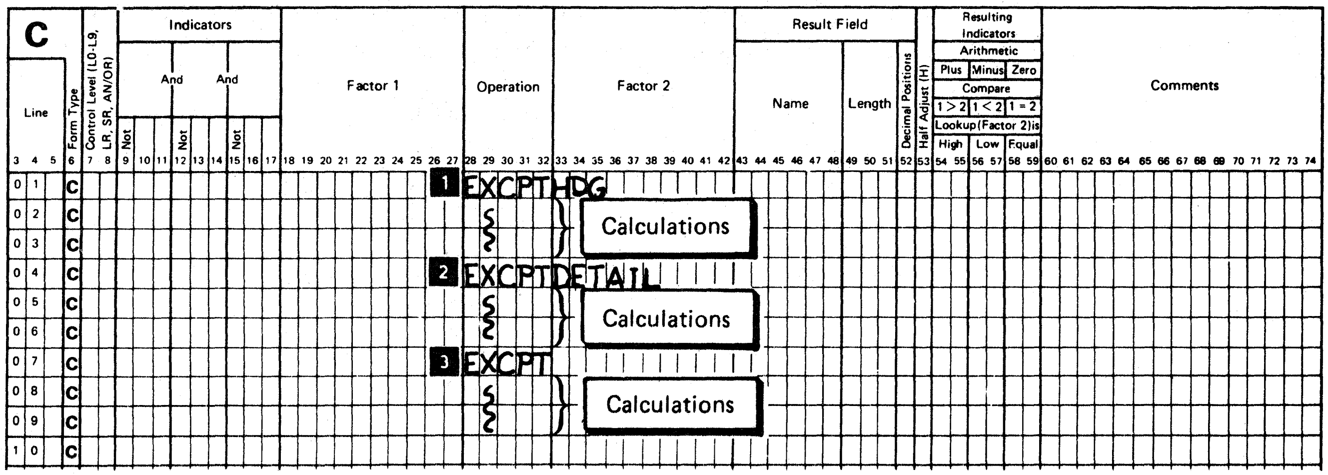
- When the EXCPT operation with a name in factor 2 on the calculation specifications occurs, only those exception records (E in column 15) with the same name in columns 32 through 37 on the output specifications are written if the conditioning indicators are satisfied.

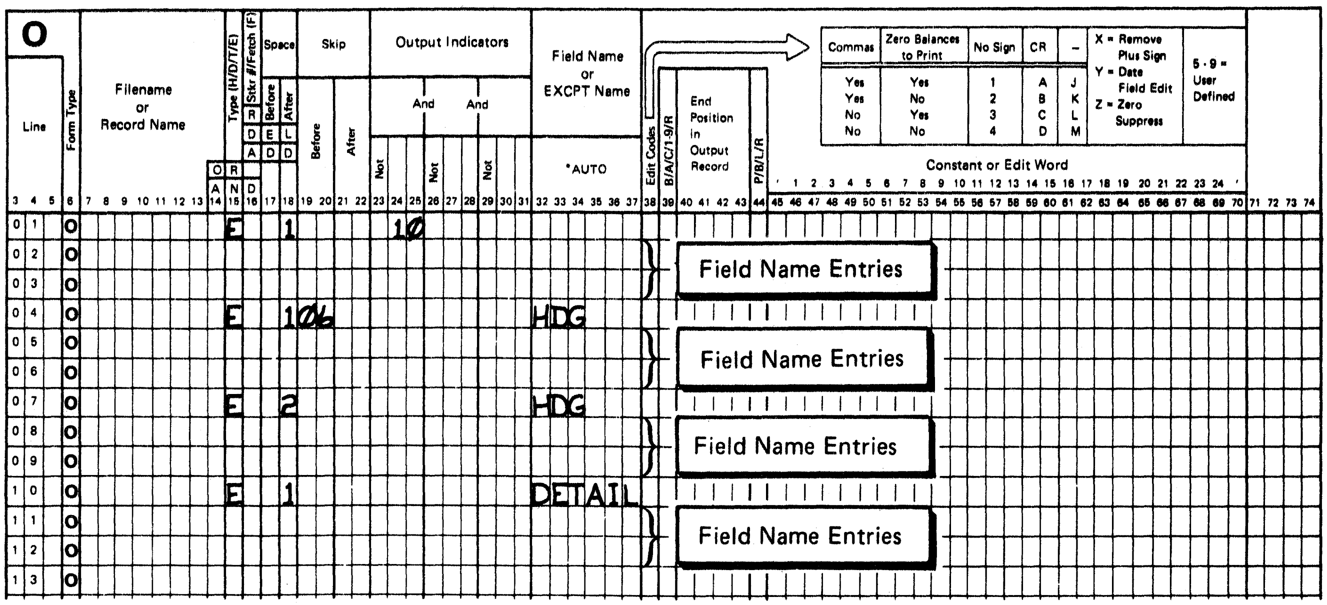
- When factor 2 on the calculation specifications with the EXCPT operation code is blank, only those exception records with no name in columns 32 through 37 on the output specifications are written if the conditioning indicators are satisfied.

- Overflow indicators cannot be used in columns 23 through 31 of the output specifications when there is an E in column 15.

- The maximum number of different EXCPT names is 64. Only one EXCPT name can be blank. These names cannot be the same as a filename, field name, data structure name, array name, table name, label, or subroutine name used in your program.

See Figures 28-15 through 28-17 for examples of the EXCPT operation.



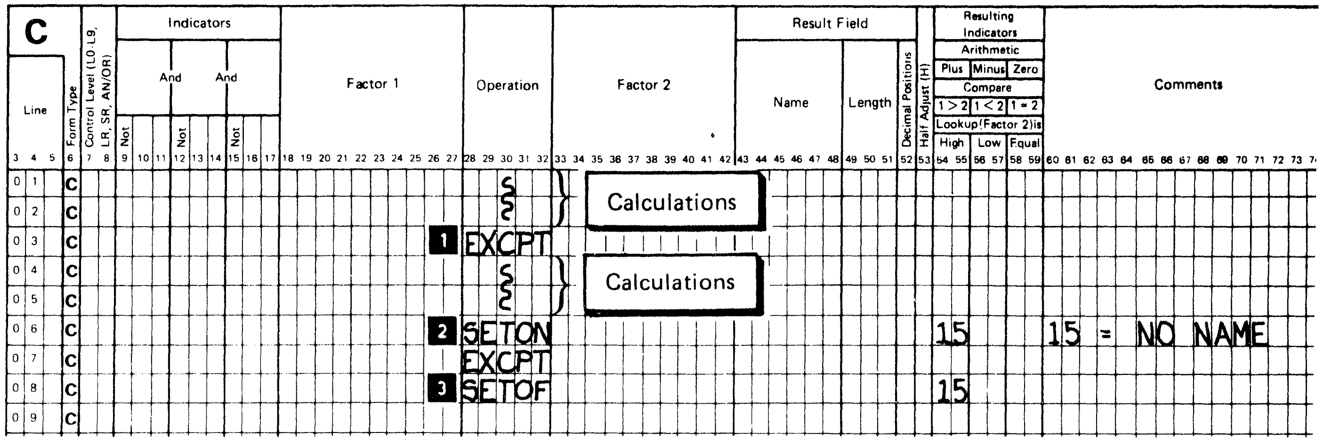


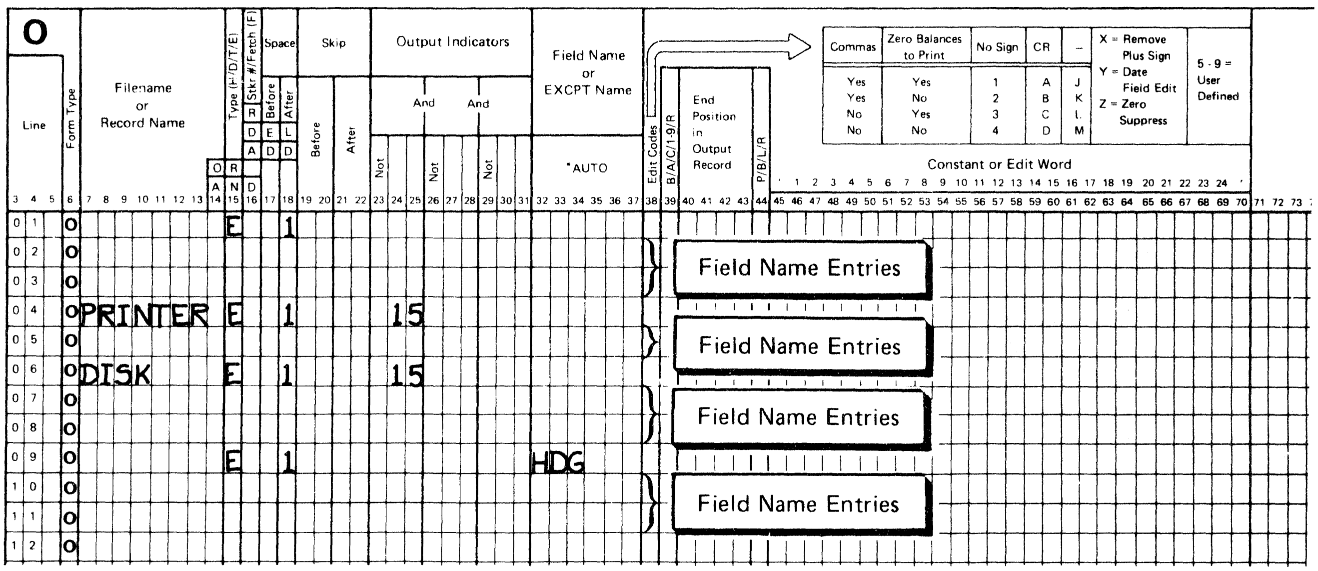
1 When the EXCPT operation with HDG specified in factor 2 is performed (line 01 in calculation specifications), all exception records with the EXCPT name HDG (lines 04 and 07 of the output specifications) are written.

2 When the EXCPT operation with DETAIL specified in factor 2 is performed (line 04 of the calculation specifications), all exception records with the EXCPT name DETAIL (line 10 of the output specifications) are written.

3 When the EXCPT operation with no entry in factor 2 is performed (line 07 of the calculation specifications), all exception records that do not have an EXCPT name specified in positions 32 through 37 (such as line 01 of the output specifications) are written if the conditioning indicators are satisfied. Any exception records without conditioning indicators and without an EXCPT name are always written by an EXCPT operation with no entry in factor 2.

Figure 28-15. EXCPT Operation with/without Factor 2 Specified





1 When the EXCPT operation is encountered in the calculation specifications (line 03), all exception records in the output specifications that do not have an EXCPT name are written if the conditioning indicators allow output. Line 01 of the output specifications is always output by the EXCPT operation because it is unconditioned.

2 When the SETON operation sets on indicator 15 in the calculation specifications (line 06) and the EXCPT operation on line 07 is performed, all lines in the output specifications that have no EXCPT name in positions 32 through 37 and that are unconditioned or conditioned by indicator 15 (lines 01, 04, and 06) are written.

3 The SETOF operation sets off indicator 15 (line 08).

Figure 28-16. EXCPT Operation without Factor 2 Specified

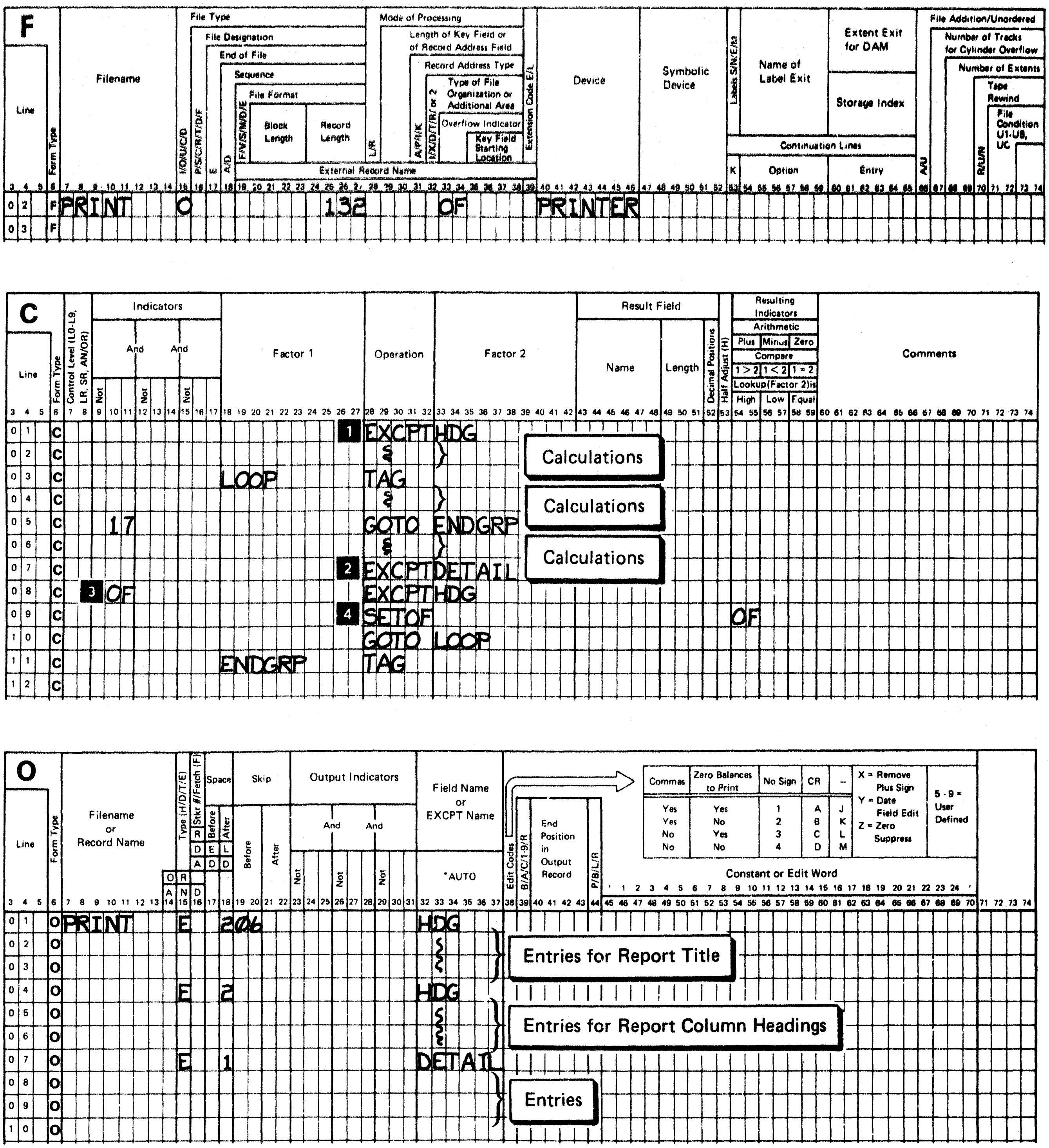


Figure 28-17 (Part 1 of 2). EXCPT Output with an Overflow Indicator

This example shows the coding for EXCPT output with an overflow indicator when you are printing the title and column headings on each page of a report.

1 The EXCPT operation with HDG in factor 2 (line 01 of the calculation specifications) causes all lines with the group EXCPT name HDG (lines 01 and 04 of the output specifications) to be printed.

2 The EXCPT operation with DETAIL specified in factor 2 (line 07 of the calculation specifications) causes all lines with the group EXCPT name DETAIL (line 07 of the output specifications) to be printed.

3 When the overflow indicator is set on, the EXCPT operation (line 08 of the calculation specifications) prints all HDG lines (lines 01 and 04 of the output specifications) on the overflow page.

4 The SETOF operation sets off the OF indicator (line 09 of the calculation specifications), and the program branches to the label specified in the GOTO operation.

Figure 28-17 (Part 2 of 2). EXCPT Output with an Overflow Indicator

EXIT (Exit to an External Subroutine)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Optional	Optional	Blank	EXIT	Required	Blank	Blank	Blank	Blank

The EXIT operation designates the point in the calculation specifications at which control is to be transferred from an RPG program to an assembler-language subroutine.

The rules for use of the EXIT operation on the calculation specifications are as follows:

Columns	Entry
Operation (28-32)	EXIT
Factor 1 (18-27)	Blank
Factor 2 (33-42)	The name of the subroutine to which control is to be passed. The name must consist of 5 or 6 characters, the first 4 of which are SUBR. The remaining characters must be alphabetic for user-written subroutines. (Numeric characters are reserved for IBM-- supplied subroutines.) The module name and entry point name must be the same.
Result field (43-48)	Blank
Resulting indicators (54-59)	Blank

The EXIT operation can be controlled by a control-level indicator (columns 7 and 8) and conditioning indicators (columns 9 through 17). If no control-level indicator is used, the EXIT operation occurs at detail calculation time.

The position of the EXIT operation in the calculation specifications of the RPG program determines when the actual subroutine execution occurs (see the table below).

To specify linkage to a non-I/O subroutine for a SPECIAL file, use the EXIT operation. You must keep track of the EXIT that is taken because index register 2 does not point to the DTF on an EXIT operation.

Note: The maximum number of user-written assembler subroutines that can be used in a program is 256.

The table below shows the relationship between the position of the EXIT operation and execution of the subroutine.

Position	**Execution of Subroutine**
First detail line in calculation specifications	Immediately following data routine file, that is, after data is extracted from input record
Last detail line in calculation specifications	Immediately before heading records output time
First total line in calculation specifications	Immediately following input routine (after determination of record type and testing for control-level break)
Last total line in calculation specifications	Immediately before total records output time
Any other detail or total line in calculation specifications	Immediately following the previous calculation operation

EXSR (Execute Subroutine)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Optional	Optional	Blank	EXSR	Required	Blank	Blank	Blank	Blank

The EXSR operation causes control to be given to the subroutine named in factor 2. The EXSR operation can appear anywhere in the program. Whenever it appears, the subroutine is given control. After operations in the subroutine are performed, the operation in the line following the EXSR operation is performed.

The EXSR operation can be conditioned by any indicators; thus, the subroutine is given control only when all conditions are satisfied. Any valid indicator can be used in columns 7 through 17. If no indicators are used, the subroutine is always given control.

Factor 2 must contain the name of the subroutine that is to be performed. This name must appear on a BEGSR operation.

Coding Subroutines

All RPG operations can be performed within a subroutine, and these operations can be conditioned by any valid indicators in columns 9 through 17. Because SR or blanks must appear in columns 7 and 8, control-level indicators cannot be used in these columns. However, AND/OR lines within the subroutine can be indicated in columns 7 and 8.

Fields used in a subroutine can be defined either in the subroutine or in the main program. In either instance, the fields can be used by both the main program and the subroutine.

Any number of subroutines can be included in a program; however, a subroutine cannot contain another subroutine. One subroutine can call another subroutine; that is, a subroutine can contain an EXSR or CASxx operation code. However, a subroutine cannot call itself directly or via another subroutine.

Subroutines do not have to be specified in the order they are used. Each subroutine must have a unique name and must contain a BEGSR and ENDSR operation.

See Figure 28-18 for an example of coding a subroutine.

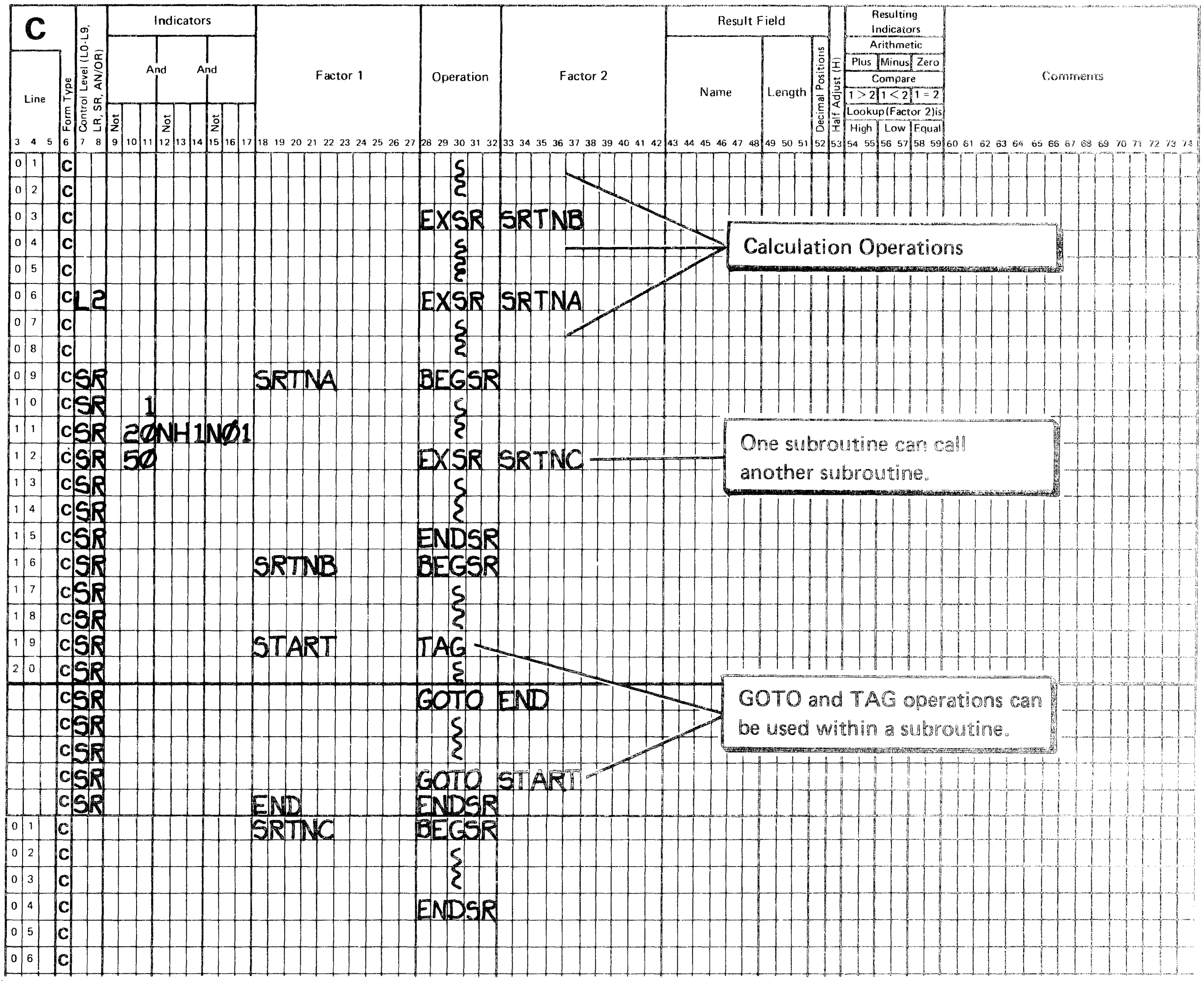


Figure 28-18. Example of Coding Subroutines

FORCE (Force)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Blank	Optional	Blank	FORCE	Required	Blank	Blank	Blank	Blank

The FORCE operation allows selection of the file from which the next record is to be read. The FORCE operation can be used for primary or secondary input and update files; however, it cannot be used to read from files assigned to a KEYBORD or WORKSTN device.

Factor 2 in a FORCE operation identifies the file from which the next record is to be selected. If the operation is performed, the record is read at the start of the next program cycle. If more than one FORCE operation is performed during the same program cycle, all but the last are ignored. FORCE should not be specified at total time.

FORCE operations override the multifile processing method by which the program normally selects records. However, the first record to be processed is always selected by the normal method. The remaining records can be selected by FORCE operations.

Figure 28-19 shows how the FORCE operation can be used to control input from primary and secondary files.

```
02 FINPUT1  IPE F  64  64            DISK
03 FINPUT2  IS  F  64  64            DISK
04 FOUTPUT  O   F  64  64            DISK
05 F
```

Figure 28-19 (Part 1 of 2). Example of FORCE Operation Controlling Input

Line	Form Type	Filename or Record Name	Sequence	Record Identifying Indicator	Position	C/Z/D	Character	From	To	Decimal Positions	RPG Field Name
01	I	INPUT1	AA	Ø1	1	C	1				
02	I							2	2	Ø	NBR
03	I							3	8		FIELDA
04	I							9	24		FIELDB
05	I							25	30		FIELDC
06	I	INPUT2	BB	Ø2	1	C	1				
07	I							13	18		FIELDA
08	I							2Ø	35		FIELDB
09	I							38	43		FIELDC
10	I										

The NBR field of each primary record contains the number of secondary records to be read and written after each primary record is read. If NBR is less than or equal to zero, a halt occurs. No primary or secondary records are read. Processing begins with the next primary record according to normal selection.

Line	Form Type	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Resulting Indicators High	Resulting Indicators Low	Comments
01	C	Ø1	NBR	COMP	Ø		H1	H1	
02	C	NH1	NBR	SUB	1	NBR	Ø3		
03	C	NØ3NH1		FORCE	INPUT2				NEXT CYCLE SEC
04	C	Ø3NH1		FORCE	INPUT1				NEXT CYCLE SEC
05	C								

If NBR is greater than zero, the field is reduced by 1 and tested (line 02 of the calculation specifications). If the result is not negative, the FORCE operation calls for input on the next program cycle from the secondary file. The primary record is written, and secondary records are read and written until NBR is negative (indicator 03 is on). The FORCE operation in line 04 then calls for input on the next primary file.

Line	Form Type	Filename or Record Name	AND/OR	Type (H/D/T/E)	Output Indicators	Field Name or EXCPT Name	End Position in Output Record
01	O	OUTPUT		D	Ø1		
02	O		OR		Ø2		
03	O					FIELDA	1Ø
04	O					FIELDB	27
05	O					FIELDC	33
06	O						

Figure 28-19 (Part 2 of 2). Example of FORCE Operation Controlling Input

GOTO (Branch To)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Optional	Optional	Blank	GOTO	Required	Blank	Blank	Blank	Blank

The GOTO operation allows operations to be skipped by instructing the program to go to (or branch to) another operation. A GOTO operation can be used to specify a branch:

- To a previous or a succeeding specification line
- From a detail calculation line to another detail calculation line
- From a total calculation line to another total calculation line

However, a branch cannot be made:

- From a detail calculation line to a total calculation line or vice versa.
- From calculations conditioned by L0 through L9 to calculations conditioned by LR or vice versa. (A total calculation line is defined as one that is conditioned by a control-level indicator in columns 7 and 8 of the calculation specifications.)
- From a subroutine to other calculations or vice versa.

Factor 2 must contain the name of the label to which the program is to branch. This label is entered in factor 1 of a TAG operation. If the GOTO is within the subroutine, the label can be specified in factor 1 of the ENDSR operation. The label can be from 1 to 6 characters long and must begin with an alphabetic character in column 33. The remaining characters can be any combination of alphabetic or numeric characters. Blanks must not appear between characters in the label.

Factor 1 and the result field are not used in this operation. The GOTO operation can be conditioned by any indicators. If no indicators are specified, the operation is always done.

See Figures 28-20 and 28-21 for examples of the GOTO operation.

Line	Form Type	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Plus / High (54-55)	Minus / Low (56-57)	Zero / Equal (58-59)	Comments
01	C		FIELDA	SUB	FIELDB	FIELDB	5	2		10		
02	C	10		GOTO	RTN1							
03	C		FIELDA	MULT	4	SAVE	8	2				
04	C			GOTO	RTN1							
05	C		RTN2	TAG								
06	C											
07	C			Calculations								
08	C											
09	C		RTN1	TAG								
10	C											
11	C			Calculations								
12	C										15	
13	C	N15		GOTO	RTN2							
14	C			TESTZ		FIELDC			20		20	
15	C	20		GOTO	END							
16	C			MHLZO	FIELDC	FIELDD						
17	C		END	TAG								
18	C											

1. If the result of the subtraction in line 01 is minus (indicator 10 is on), a branch is taken to RNT1 (routine 1) named by the TAG operation code in line 09. Notice that neither the GOTO (line 02) nor the TAG (line 09) is conditioned by control-level indicators.

2. If the branch is not taken in line 02, the multiplication in line 03 is performed. Then the branch to RTN1 (line 09) must be taken because this branch is not conditioned by indicators.

3. Operations in lines 10 through 12 are then done. If the operation in line 12 does not turn indicator 15 on, a branch is taken backwards to RTN2 (line 05).

4. Operations are then performed again in the order specified from lines 06 through12. Nothing is done in line 09 because TAG gives only a name. These same operations are performed again and again until indicator 15 does turn on.

5. When indicator 15 is on, the branch to RTN2 is not taken. The TESTZ operation is then performed. If this operation causes indicator 20 to turn on, a branch is taken to line 17 (GOTO END). If indicator 20 is not on, the operation in line 16 is performed.

Figure 28-20. Using GOTO and TAG (Skipping Operations)

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators Plus / High (1>2)	Minus / Low (1<2)	Zero / Equal (1=2)	Comments
01	C				EXCPT									
02	C				EXCPT									
03	C				EXCPT									
04	C				EXCPT									
05	C				EXCPT									
06	C				EXCPT									
07	C				EXCPT									
08	C				EXCPT									
09	C													

Assume you want to make eight mailing labels for every customer you have. The customer's name and address are found on an input record. Because you want to write eight labels for each record, you can use exception lines and the EXCPT operation instead of coding eight identical output line specifications. (See *EXCPT (Exception Output)* in this chapter for further information.)

However, by using branching, you can code it all in six lines as shown below. An EXCPT line is printed out. One is added to COUNT to keep track of how many times the line is printed. Then COUNT is compared to 8. If COUNT does not equal 8, a branch is taken back to the beginning (GOTO DOAGIN). If COUNT equals 8, the branch is not taken. Instead, the COUNT field is set to zero for the next cycle.

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators Plus / High (1>2)	Minus / Low (1<2)	Zero / Equal (1=2)	Comments
01	C			DOAGIN	TAG									
02	C				EXCPT									
03	C			1	ADD	COUNT	COUNT							
04	C			COUNT	COMP	8							20	
05	C		N20		GOTO	DOAGIN								
06	C				Z-ADD	0	COUNT							
07	C													

Figure 28-21. Using GOTO and TAG to Eliminate Duplicate Coding

IFxx (If/Then)

Indicators					Result	Resulting Indicators		
7–8	9–17	Factor 1	Operation	Factor 2	Field	54–55	56–57	58–59
Optional	Optional	Required	IFxx	Required	Blank	Blank	Blank	Blank

The IFxx operation allows a group of calculations to be performed if a relationship specified in xx portion of the IFxx operation exists between factor 1 and factor 2. See *Structured Programming Operations* in this chapter for options available under the xx portion of the IFxx operation code.

Conditioning indicators can be specified. Conditioning indicators on the IFxx operation control whether the IFxx operation (and the entire IFxx/ELSE group) will be performed, or control will be passed to the operation immediately following the associated END operation.

Factor 1 and factor 2 can contain a character literal, a numeric literal, an array element, a table element, a data structure subfield, or a field name. Both factor 1 and factor 2 must be character data, or both must be numeric. The rules for comparing factor 1 and factor 2 on the IFxx operation are the same as those given under *Compare and Testing Operations* in this chapter.

If the relationship between factor 1 and factor 2 does not exist and an ELSE operation is not specified, control passes to the first executable operation following the associated END operation. If the relationship between factor 1 and factor 2 does not exist and an ELSE operation is specified, control passes to the first operation following the ELSE operation.

Conditioning indicator entries on the END operation associated with IFxx must be blank.

An END operation must be used to close an IFxx group. If an IFxx operation is followed by an ELSE operation, an END operation is needed after the ELSE operation but not after the IFxx operation.

Figure 28-22 is an example of code using IFxx/END and IFxx/ELSE/END structures.

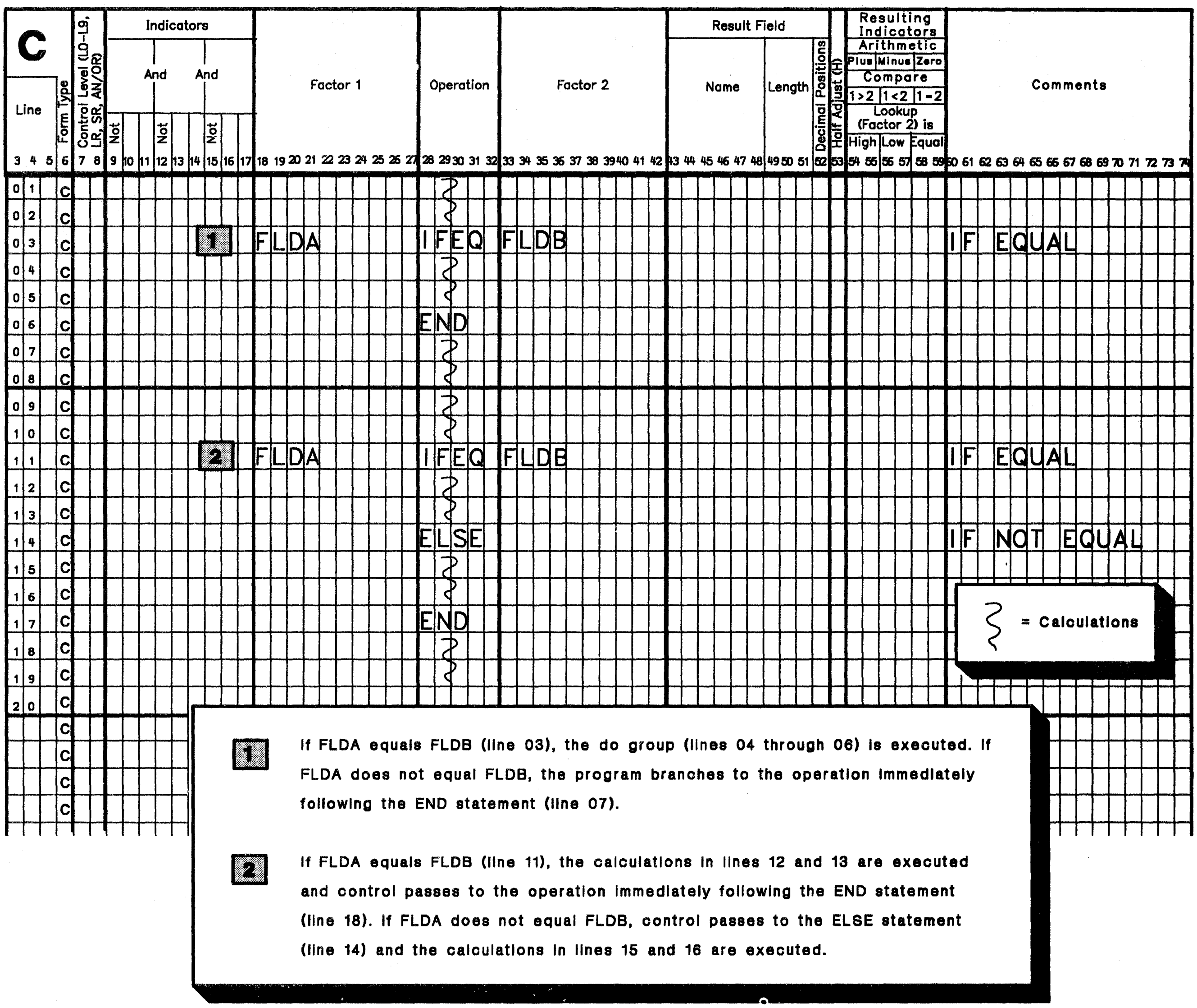


Figure 28-22. IFxx/END and IFxx/ELSE/END designs.

KEY (Key)

Indicators						Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Result Field	54-55	56-57	58-59
Optional	Optional	Optional	KEY	Blank	Required	Blank	Blank	Blank

The KEY operation causes a pause in calculations during which the person using the display station can enter data from the keyboard. All KEY operations are directed to the display station that loaded the program.

Factor 1 can contain the constant, literal, field name, or table or array element to be displayed. Factor 2 must be blank. The result field can contain the name of the field to be entered. See *Calculation Specifications for a KEY Operation* in Chapter 9 for more information on coding the KEY operation.

To use the KEY operation code, you must specify the device name KEYBORD in columns 40 through 46 of the file description specifications. KEY can be used only with a KEYBORD input file. As the person enters data, it is displayed on the screen in one of two formats:

- If the record length is 40 or less, the display consists of six lines, with 40 characters per line, centered both vertically and horizontally on the screen.
- If the record length is greater than 40, the display consists of 12 lines with 79 characters per line (1 character is reserved for field attributes).

When the KEY operation is used, the contents of the result field are determined by the person's response. The possible responses are:

- The person types in the data and presses an entry function key. If not all positions of a field are entered, numeric fields are right-justified and padded to the left with zeros; alphameric fields are padded to the right with blanks.
- The person presses only an entry function key, which causes any data in the result field to be changed to zero or blank.
- The person presses the Dup key and then an entry function key, which does not modify the data in the result field.

Note: The person can use any one of the following four keys as an entry function key: Field Exit, Field-, Field+, or Enter. However, if data has been entered into a numeric field, the Enter key cannot be used as an entry function key.

LOKUP (Lookup)

Array LOKUP

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional	Optional	Required	LOKUP	Required	Blank	One required		

Table LOKUP

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional	Optional	Required	LOKUP	Required	Optional	One required		

The LOKUP operation causes a search to be made for a particular element in an array or table. The array or table is named in factor 2. Factor 1 is the search word (data for which you want to find a match in the array or table named).

Factor 1, the search word, can be:

- An alphameric or numeric constant
- A field name
- An array element
- A table name

When a table is named in factor 1, it refers to the element of the table last selected in a LOKUP operation, not to the whole table.

Resulting indicators are always used with a LOKUP operation. The indicators first specify the type of search to be made and then reflect the result of the search. The specified indicator turns on only if the search is successful.

Resulting indicators specify the type of search and reflect the result of the search in the following manner:

- A resulting indicator assigned to equal (columns 58 and 59) instructs the program to search for an entry in the array or table equal to the search word. The first equal entry found turns on the indicator assigned to equal.

- An indicator assigned to low (columns 56 and 57) instructs the program to find the entry in the array or table that is nearest to, yet lower in sequence than, the search word. The first such entry found turns on the indicator assigned to low.

- The indicator assigned to high (columns 54 and 55) instructs the program to find the entry in the array or table that is nearest to, yet higher in sequence than, the search word. The first such entry found turns on the indicator assigned to high.

At least one resulting indicator must be assigned, but no more than two can be used. Resulting indicators can be assigned to equal and high or to equal and low. The program searches for an entry that satisfies either condition with equal given precedence; that is, if no equal entry is found, the nearest lower or nearest higher entry is selected. If resulting indicators are assigned both to high and low, the indicator assigned to low is ignored.

When you use the LOKUP operation, consider the following:

- The search word and each array or table element must have the same length and the same format (alphameric or numeric).

- A search can be made for high, low, high and equal, or low and equal only if the array or table is in sequence. The sequence must be indicated in column 45 of the Extension specifications.

- No resulting indicator turns on if the search is not successful.

- If an index is used with an array, the search starts at that element.

- If a variable index is used with an array, the field used as the index is set to the number of the array element found if the search is successful. If the search is unsuccessful, the index is set to 1.

For more information, see *Searching Arrays and Tables* in Chapter 13.

MHHZO (Move High to High Zone)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Optional	Optional	Blank	MHHZO	Required	Required	Blank	Blank	Blank

The MHHZO operation moves the zone from the leftmost position of factor 2 to the leftmost position of the result field. Factor 2 and the result field must be alphameric.

MHLZO (Move High to Low Zone)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Optional	Optional	Blank	MHLZO	Required	Required	Blank	Blank	Blank

The MHLZO operation moves the zone from the leftmost position of factor 2 to the rightmost position of the result field. Factor 2 must be alphameric. The result field can be alphameric or numeric.

MLHZO (Move Low to High Zone)

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional	Optional	Blank	MLHZO	Required	Required	Blank	Blank	Blank

The MLHZO operation moves the zone from the rightmost position of factor 2 to the leftmost position of the result field. Factor 2 can be numeric or alphameric, but the result field must be alphameric.

MLLZO (Move Low to Low Zone)

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional	Optional	Blank	MLLZO	Required	Required	Blank	Blank	Blank

The MLLZO operation moves the zone from the rightmost position of factor 2 to the rightmost position of the result field. Factor 2 and the result field can be either alphameric or numeric.

Functions of the four move zone operations are shown in Figure 28-28.

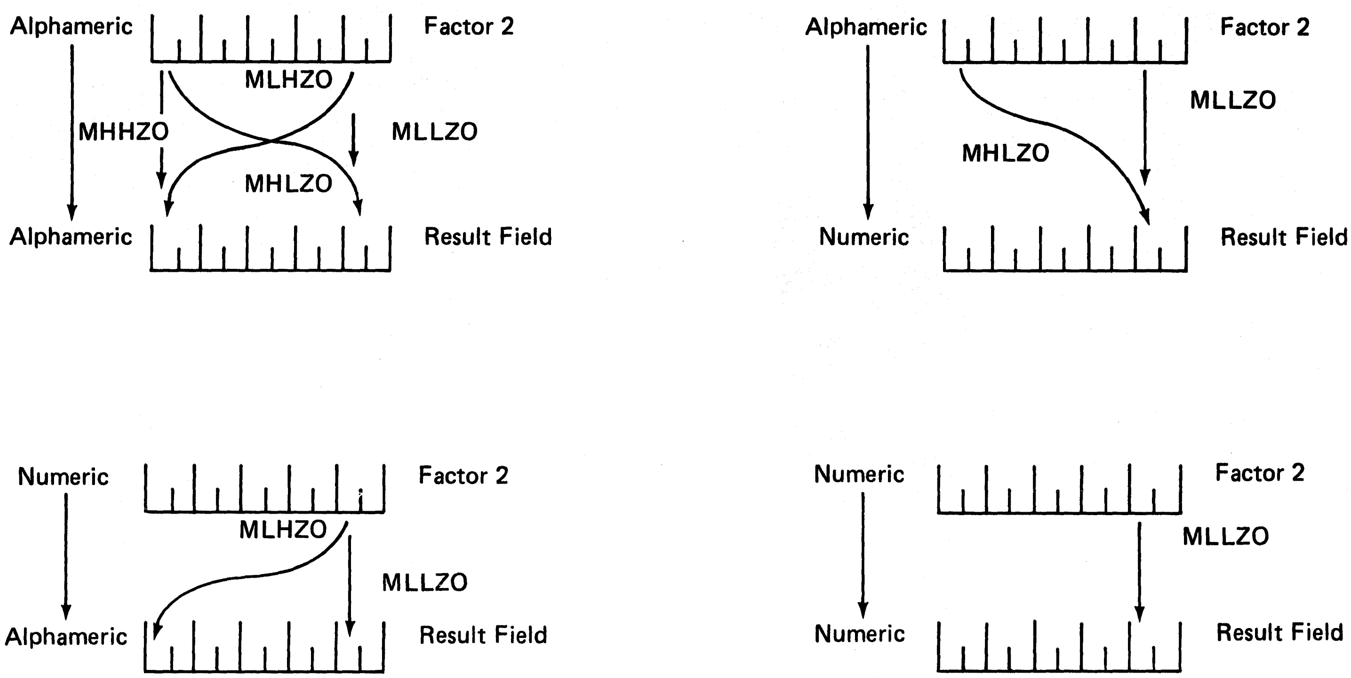


Figure 28-23. Functions of Move Zone Operations

MOVE (Move)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Optional	Optional	Blank	MOVE	Required	Required	Blank	Blank	Blank

The MOVE operation transfers characters from factor 2 to the rightmost positions in the result field. Moving starts with the rightmost character of factor 2. If factor 2 is longer than the result field, the excess leftmost characters of factor 2 are not moved. If the result field is longer than factor 2, the excess leftmost characters in the result field are unchanged.

When an alphameric field is moved to a numeric field, only the digit portion of each alphameric character is moved to the digit portion of the corresponding numeric character. The zone of the rightmost numeric character is set to hexadecimal D (negative) if the zone of the rightmost alphameric character is hexadecimal D (characters J through R). Otherwise, it is set to hexadecimal F.

The MOVE operation is summarized in Figure 28-24.

Factor 2 Shorter Than Result Field

	Factor 2		Result Field	
a. Alphameric	P H 4 S N	Before MOVE	1 2 3 4 5 6 7 8 4	Alphameric
	P H 4 S N	After MOVE	1 2 3 4 P H 4 S N	
b. Alphameric	P H 4 S N	Before MOVE	1 2 3 4 5 6 7 8 4	Numeric
	P H 4 S N	After MOVE	1 2 3 4 7 8 4 2 $\bar{5}$	
c. Numeric	1 2 7 8 4 2 5	Before MOVE	1 2 3 4 5 6 7 8 9	Numeric
	1 2 7 8 4 2 5	After MOVE	1 2 1 2 7 8 4 2 5	
d. Numeric	1 2 7 8 4 2 5	Before MOVE	A C F G P H 4 S N	Alphameric
	1 2 7 8 4 2 5	After MOVE	A C 1 2 7 8 4 2 5	

Factor 2 Longer Than Result Field

	Factor 2		Result Field	
a. Alphameric	A C E G P H 4 S N	Before MOVE	5 6 7 8 4	Alphameric
	A C E G P H 4 S N	After MOVE	P H 4 S N	
b. Alphameric	A C E G P H 4 S N	Before MOVE	5 6 7 8 4	Numeric
	A C E G P H 4 S N	After MOVE	7 8 4 2 $\bar{5}$	
c. Numeric	1 2 7 8 4 2 5	Before MOVE	5 6 7 4 8	Numeric
	1 2 7 8 4 2 5	After MOVE	7 8 4 2 5	
d. Numeric	1 2 7 8 4 2 5	Before MOVE	P H 4 S N	Alphameric
	1 2 7 8 4 2 5	After MOVE	7 8 4 2 5	

Factor 2 and Result Field Same Length

	Factor 2		Result Field	
a. Alphameric	P H 4 S N	Before MOVE	5 6 7 8 4	Alphameric
	P H 4 S N	After MOVE	P H 4 S N	
b. Alphameric	P H 4 S N	Before MOVE	5 6 7 8 4	Numeric
	P H 4 S N	After MOVE	7 8 4 2 $\bar{5}$	
c. Numeric	7 8 4 2 $\bar{5}$	Before MOVE	5 6 7 8 4	Numeric
	7 8 4 2 $\bar{5}$	After MOVE	7 8 4 2 $\bar{5}$	
d. Numeric	7 8 4 2 $\bar{5}$	Before MOVE	A L T 5 F	Alphameric
	7 8 4 2 $\bar{5}$	After MOVE	7 8 4 2 N	

Figure 28-24. MOVE Operations

MOVEA (Move Array)

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional	Optional	Blank	MOVEA	Required	Required	Blank	Blank	Blank

The MOVEA operation transfers characters from the leftmost positions of factor 2 to the leftmost positions of the result field. Factor 2 and the result field cannot reference the same array even if the array is indexed. Arrays and fields specified by a MOVEA operation can be alphameric or numeric. The length of the move is determined by the shorter of the lengths of factor 2 and the result field. If factor 2 is longer than the result field, the excess rightmost characters of factor 2 are not moved; if the result field is longer than factor 2, the rightmost characters in the result field are unchanged.

The length of factor 2 or the result field is the length of the entire array if the array is not indexed, or the length from the element specified to the end of the array if the array is indexed.

The MOVEA operation makes it possible to:

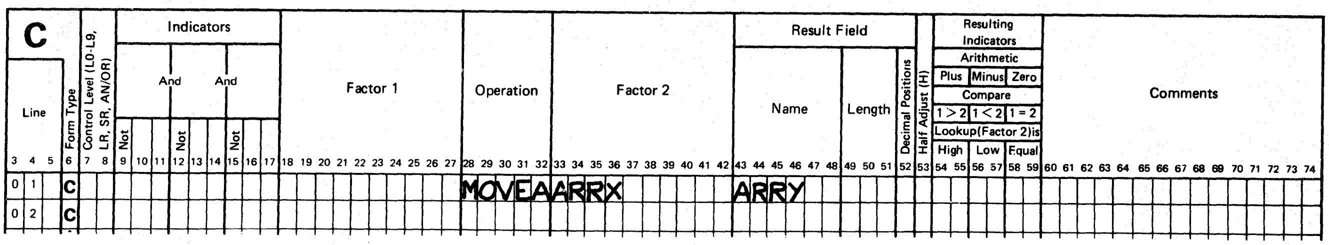
- Move several contiguous array elements to a single field.
- Move a single field to several contiguous array elements.
- Move contiguous elements of one array to contiguous elements of another array.

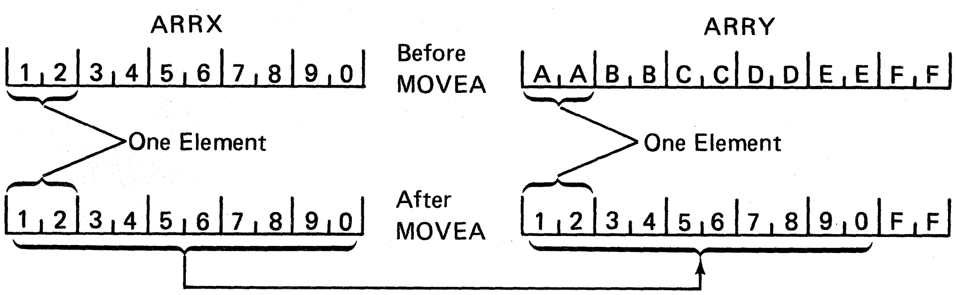
Movement of data starts with the first element of an array if the array is not indexed or with the element specified if the array is indexed. The movement of data ends when the last array element is moved or filled or when the number of characters moved equals the length of the shorter field specified by factor 2 and the result field. Therefore, the move could end in the middle of an array element. Because array boundaries are not considered when the MOVEA operation is performed, all numeric data is treated as alphanumeric data. As a result, numeric data is moved without regard for the sign. When you are moving data to a numeric array or numeric field, you should ensure that the result field will contain valid numeric data.

If you use the MOVEA operation with a figurative constant (*BLANK, *BLANKS, *ZERO, or *ZEROS) in factor 2 and an array in the result field, the figurative constant is moved into the array. The figurative constant begins at the array element specified in the result field; it ends at the end of the array.

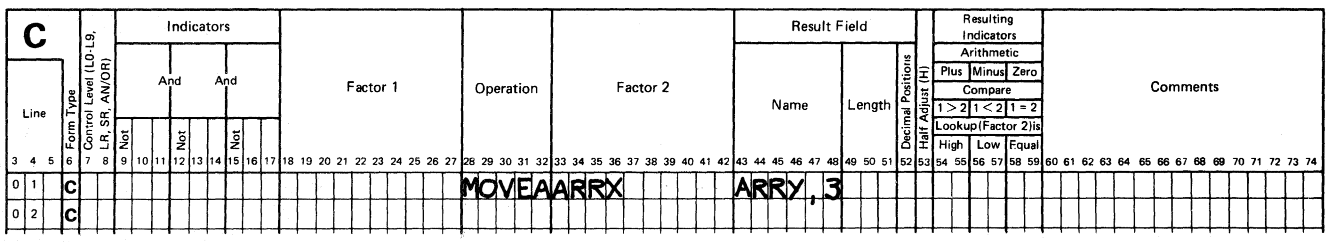
Figure 28-25 illustrates the use of the MOVEA operation.

Example: Alphameric array to alphameric array move. No indexing, different length arrays, same length elements.





Example: Alphameric array to alphameric array move. Index result field.



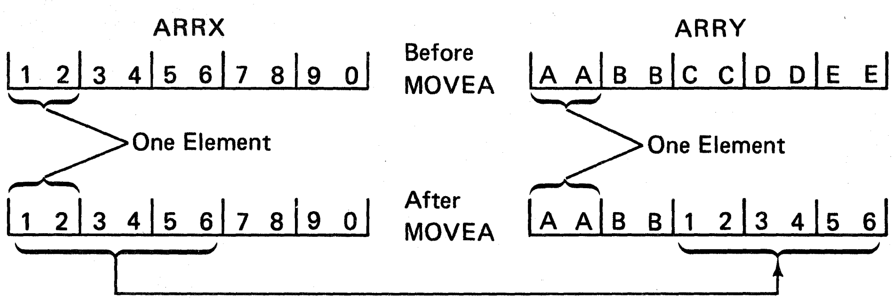
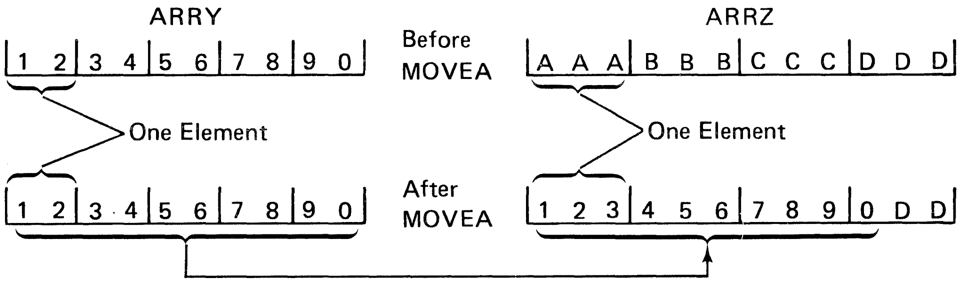


Figure 28-25 (Part 1 of 17). MOVEA Operation

Example: Alphameric array to alphameric array move. No indexing, different length array elements.

Line	Form Type	Factor 1	Operation	Factor 2	Result Field Name
01	C		MOVEA	ARRY	ARRZ
02	C				



Example: Alphameric array to alphameric array move. Index factor 2, different length array elements.

Line	Form Type	Factor 1	Operation	Factor 2	Result Field Name
01	C		MOVEA	ARRY,4	ARRZ
02	C				

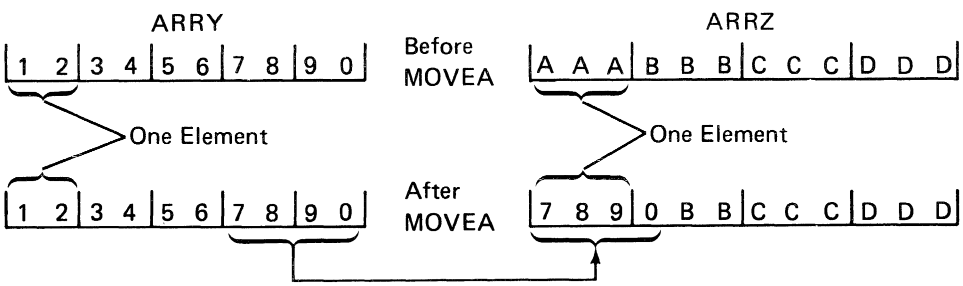
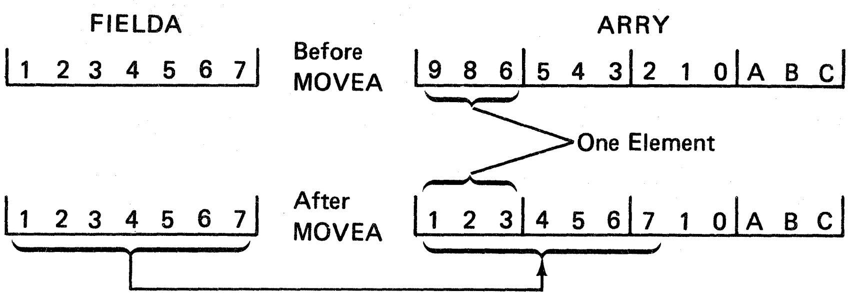


Figure 28-25 (Part 2 of 17). MOVEA Operation

Example: Alphameric array to alphameric array move. No indexing on array.

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators	Comments
01	C				MOVEA	FIELDA	ARRY					
02	C											



Example: Alphameric array to alphameric field move. Variable indexing.

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators	Comments
01	C				MOVEA	ARRX,N	FIELD					
02	C											

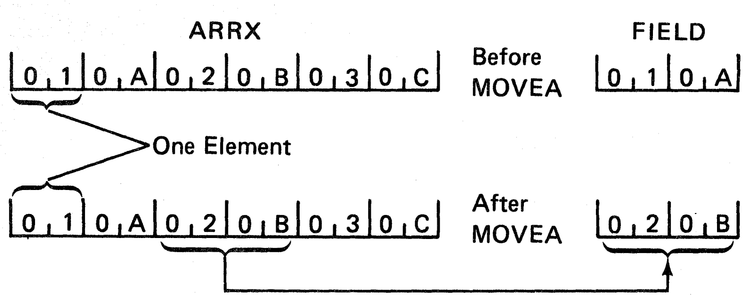
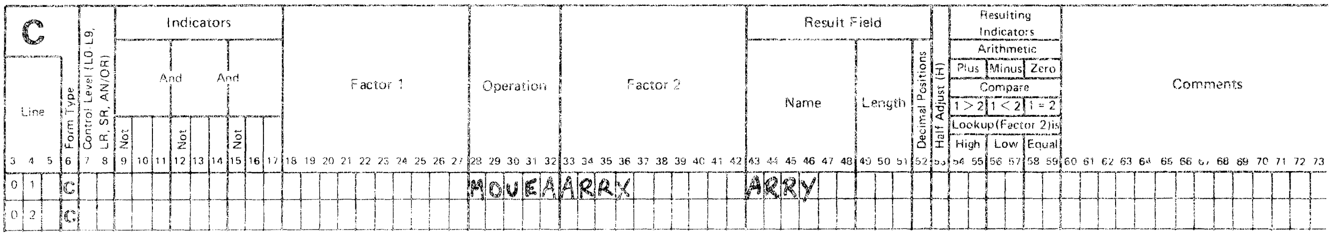
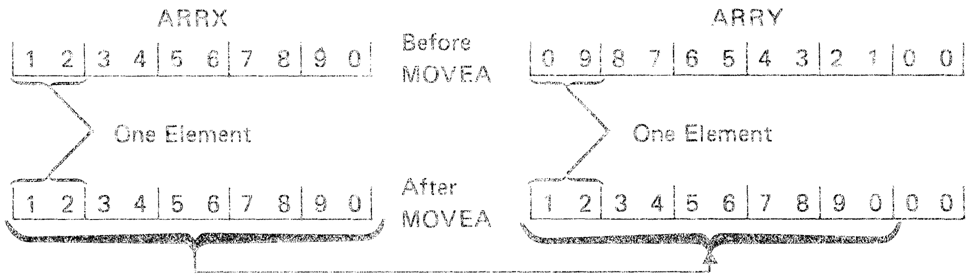


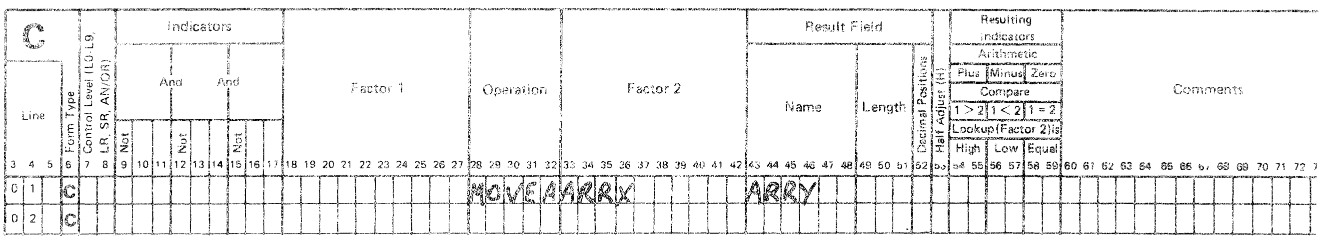
Figure 28-25 (Part 3 of 17). MOVEA Operation

Example: Numeric array to alphameric array move. No indexing, different length arrays, same length array elements.





Example: Alphameric array to numeric array. No indexing, different length arrays, same length array elements.



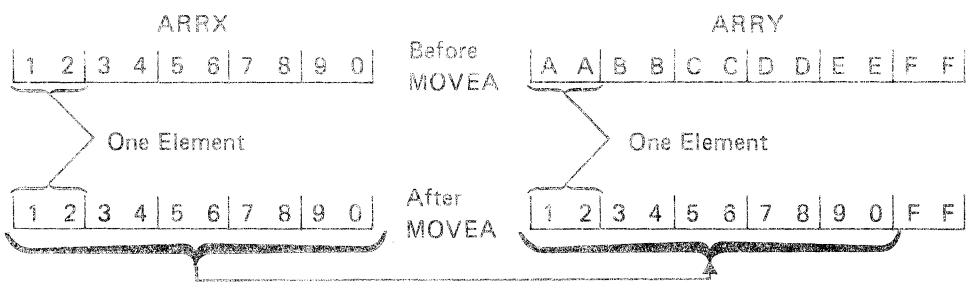
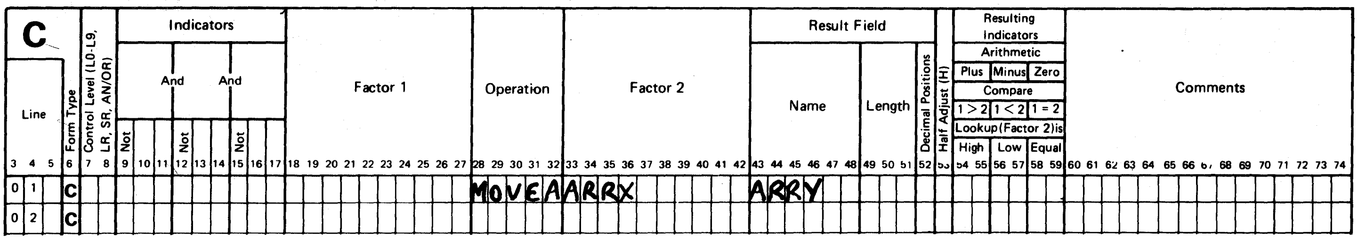
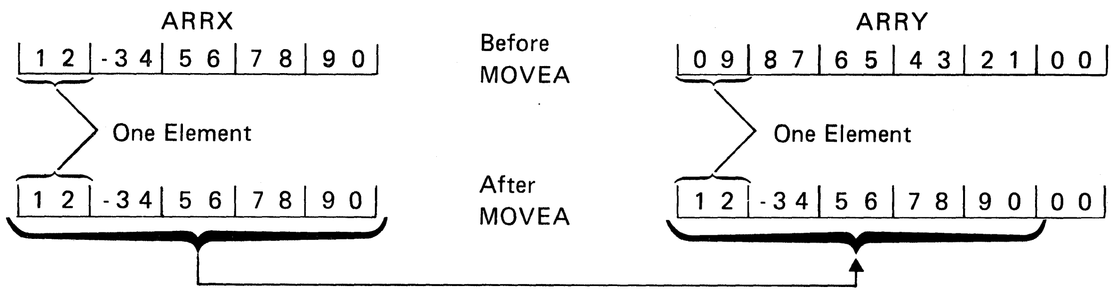


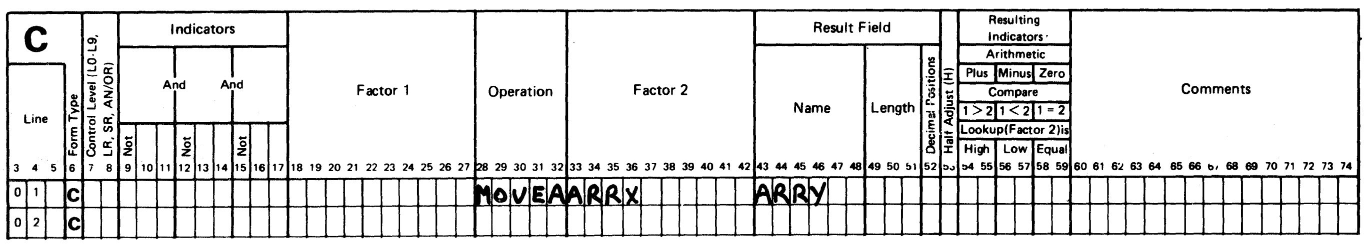
Figure 28-25 (Part 4 of 17). MOVEA Operation

Example: Numeric array to numeric array move. No indexing, different length arrays, same length array elements.





Example: Alphameric array to numeric array move. No indexing, different length arrays, same length array elements.



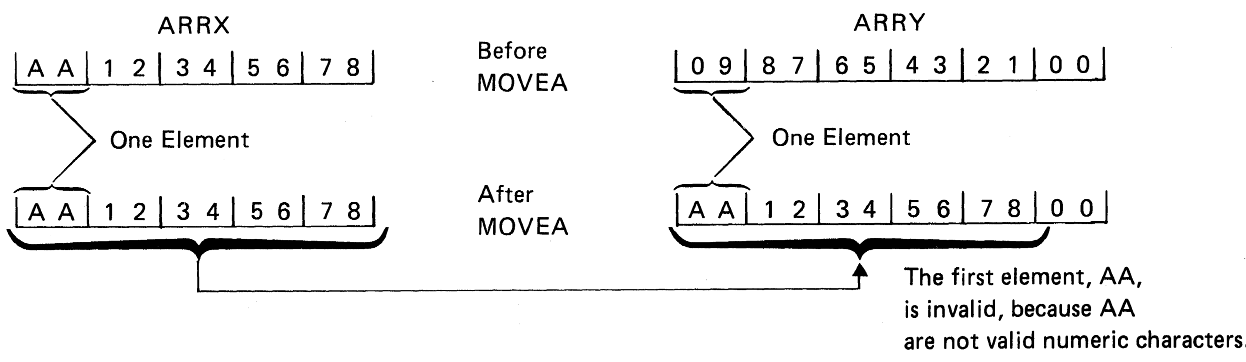
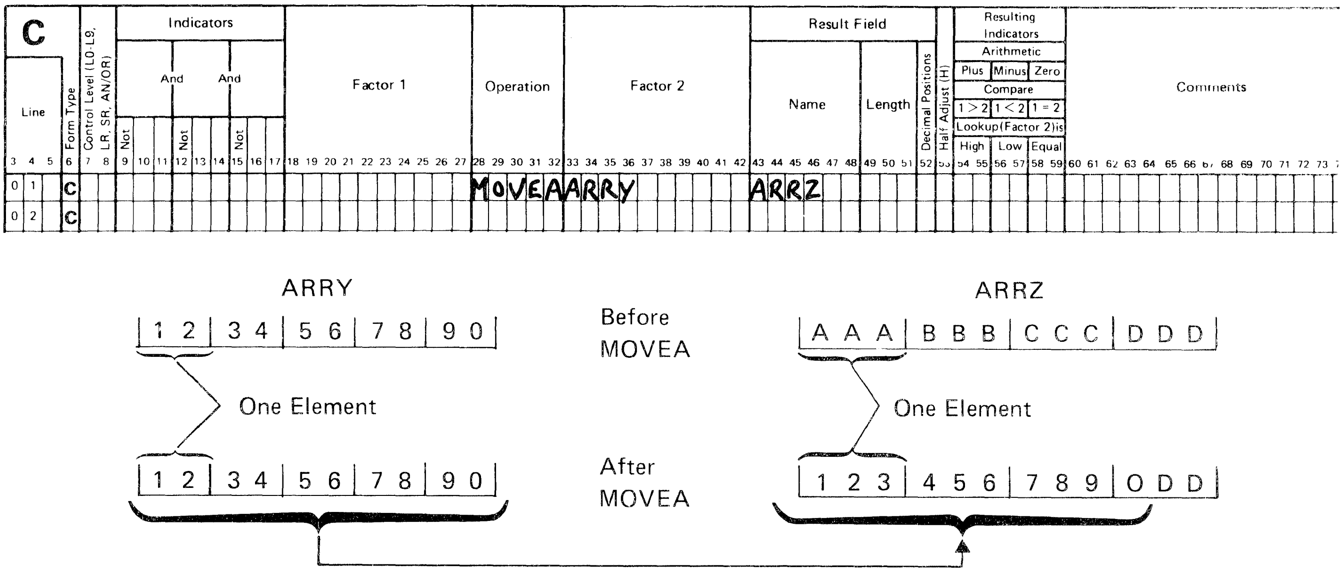


Figure 28-25 (Part 5 of 17). MOVEA Operation

Example: Numeric array to alphameric array move. No indexing, different length array elements.



Example: Alphameric array to numeric array move. No indexing, different length array elements.

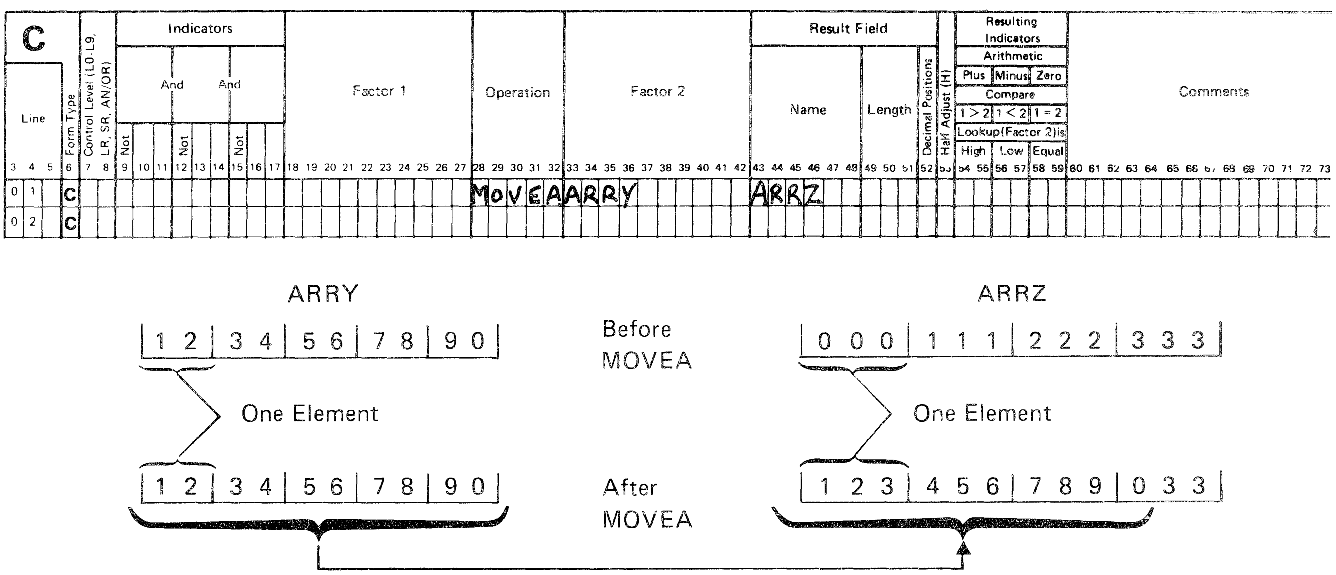


Figure 28-25 (Part 6 of 17). MOVEA Operation

Example: Numeric array to numeric array move. No indexing, different length array elements.

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators	Comments
01	C				MOVEA	ARRY	ARRZ					
02	C											

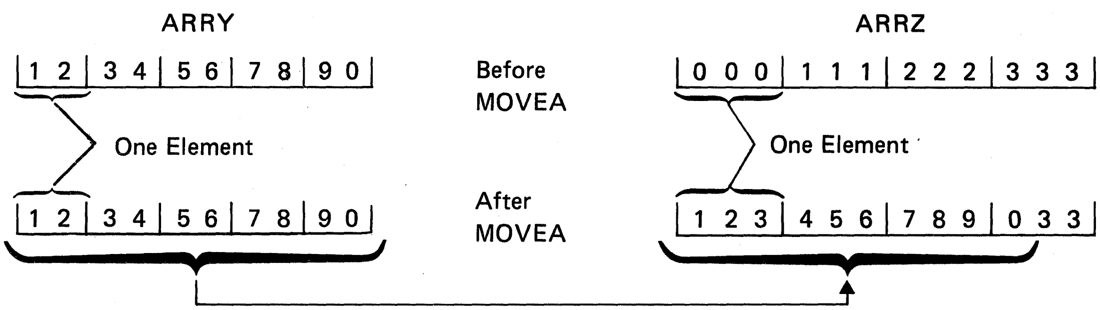
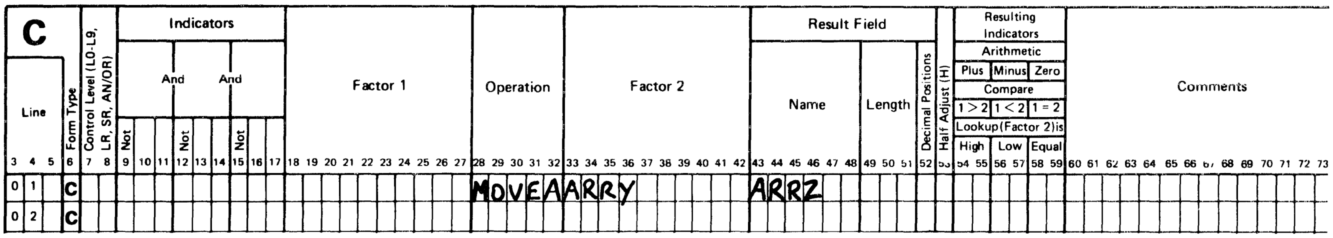


Figure 28-25 (Part 7 of 17). MOVEA Operation

Example: Numeric array to numeric array move. No indexing, different length array elements.



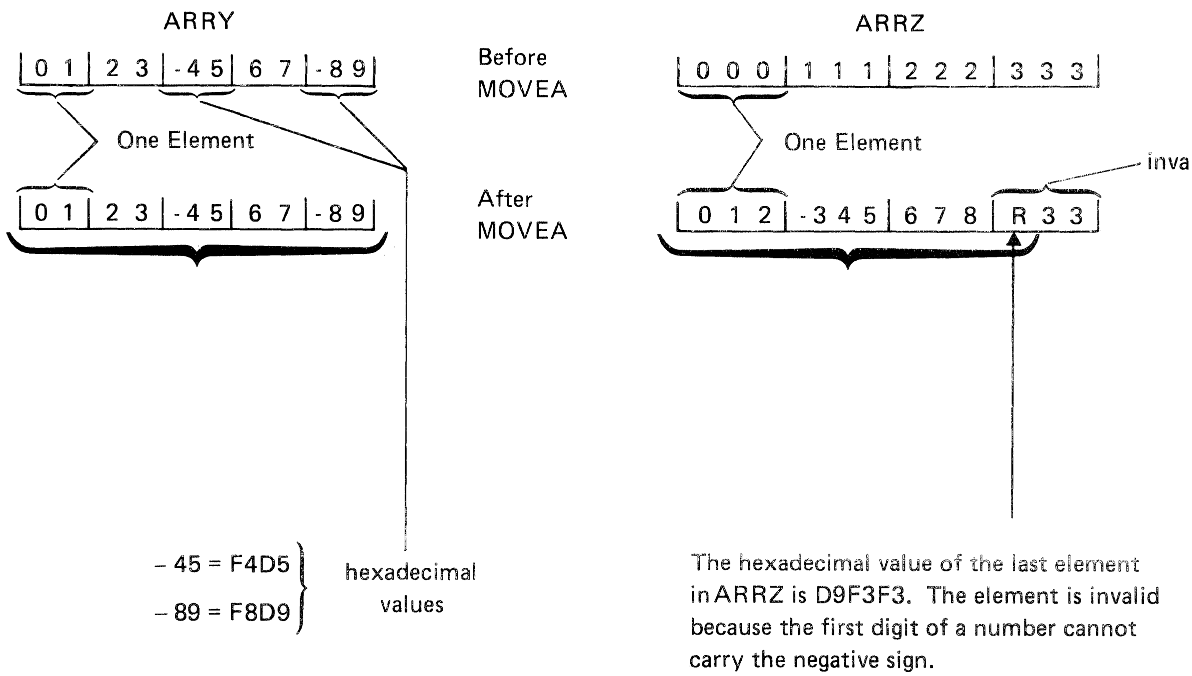
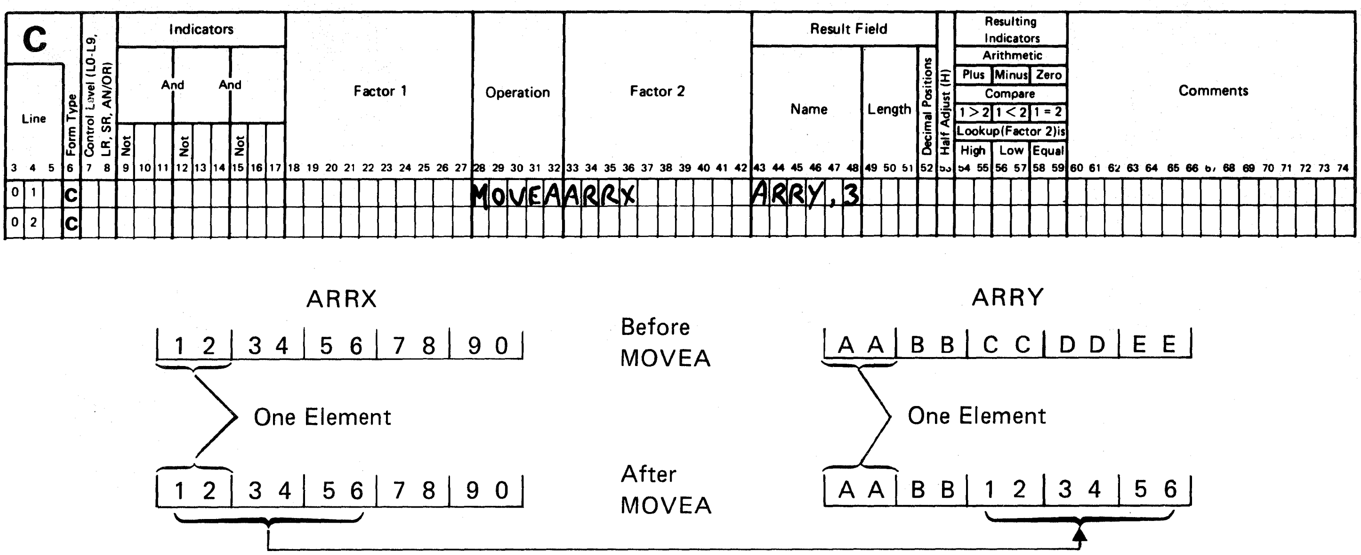


Figure 28-25 (Part 8 of 17). MOVEA Operation

Example: Numeric array to alphameric array move. Index result field, same length arrays and array elements.



Example: Alphameric array to numeric array move. Index result field, same length arrays and array elements.

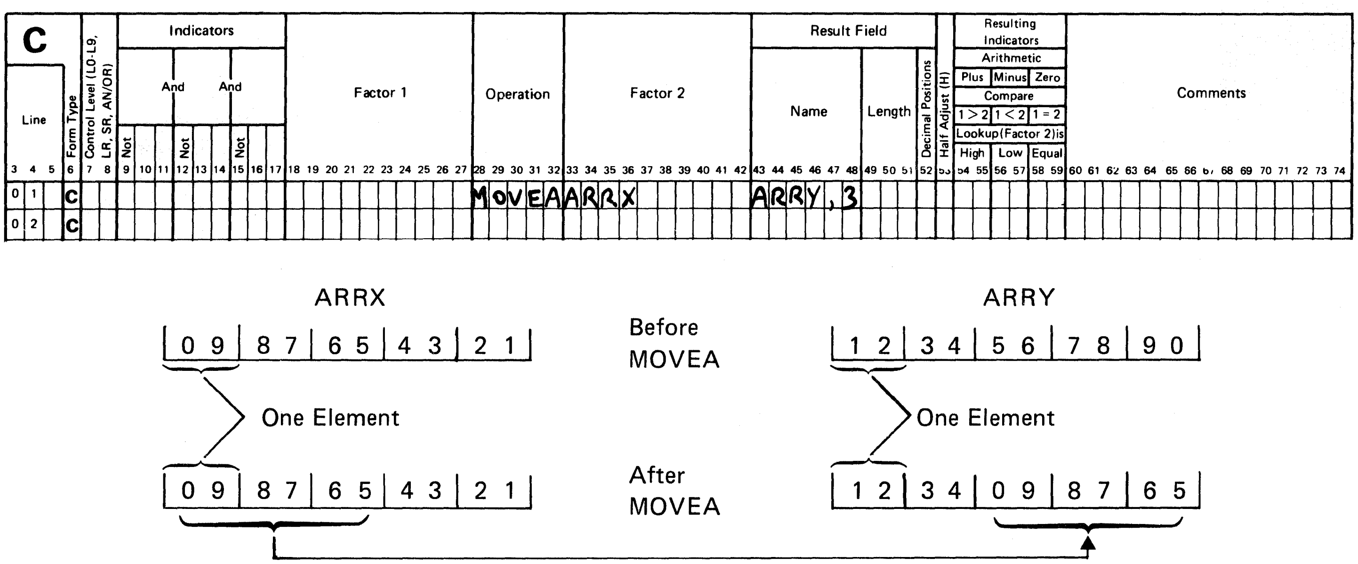


Figure 28-25 (Part 9 of 17). MOVEA Operation

Example: Numeric array to numeric array move. Index result field, same length arrays and array elements.

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators (Not, And, Not, And, Not)	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators (Plus/Minus/Zero; 1>2, 1<2, 1=2; High/Low/Equal)	Comments
01	C				MOVEA	ARRX	ARRY,3					
02	C											

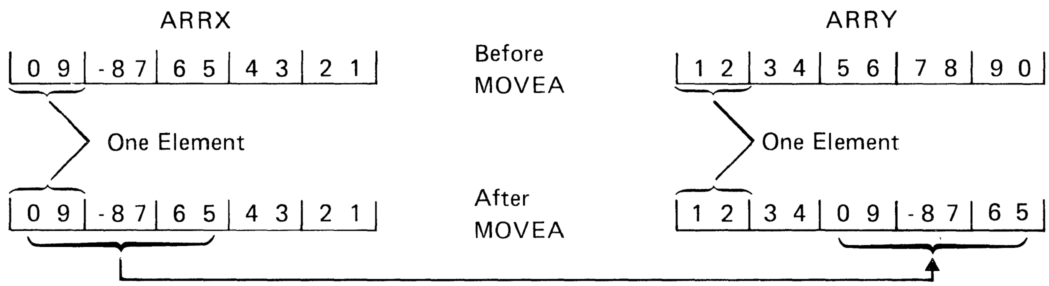
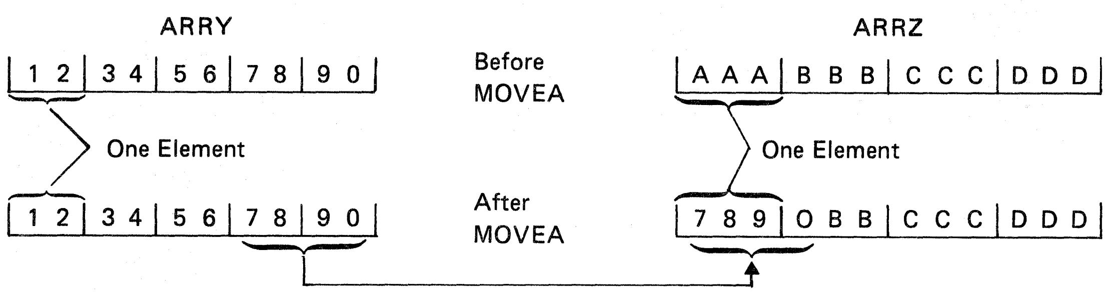


Figure 28-25 (Part 10 of 17). MOVEA Operation

Example: Numeric array to alphameric array move. Index factor 2, different length array elements.

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust	Resulting Indicators	Comments
01	C				MOVEA	ARRY,4	ARRZ					
02	C											



Example: Alphameric array to numeric array move. Index factor 2, different length array elements.

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust	Resulting Indicators	Comments
01	C				MOVEA	ARRY,4	ARRZ					
02	C											

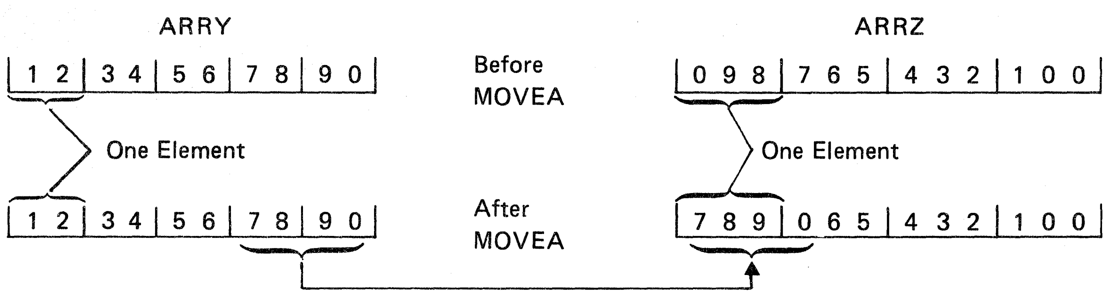
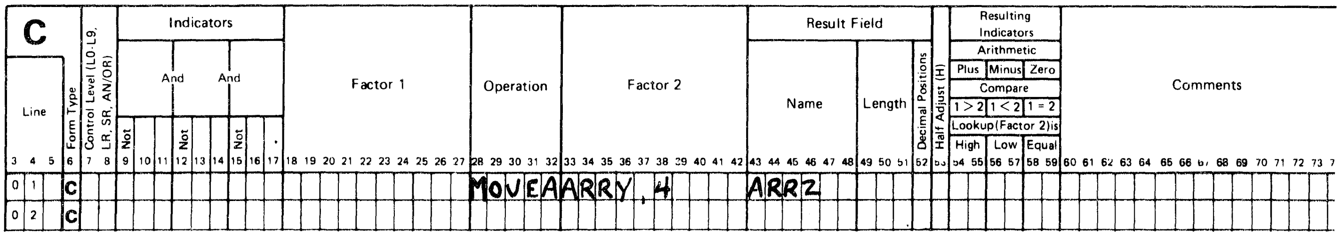
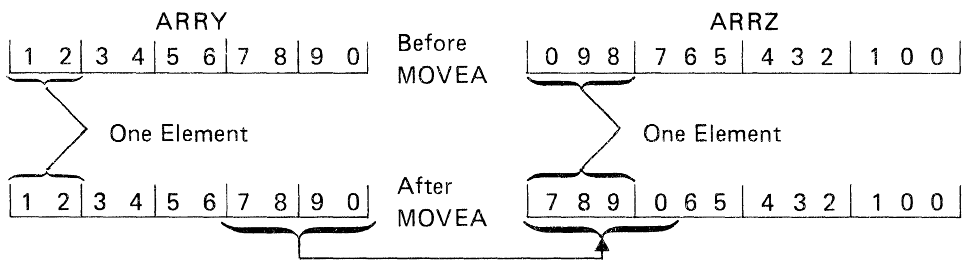


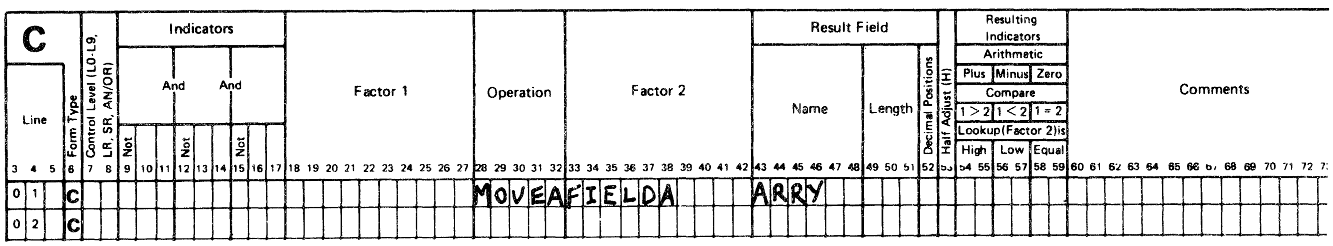
Figure 28-25 (Part 11 of 17). MOVEA Operation

Example: Numeric array to numeric array move. Index factor 2, different length array elements.





Example: Numeric field to alphameric array move. No indexing on array.



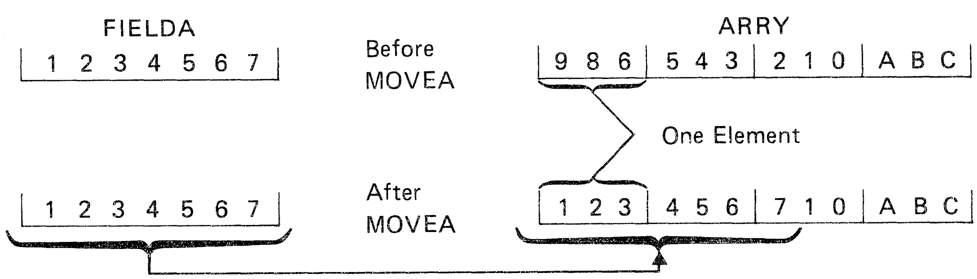
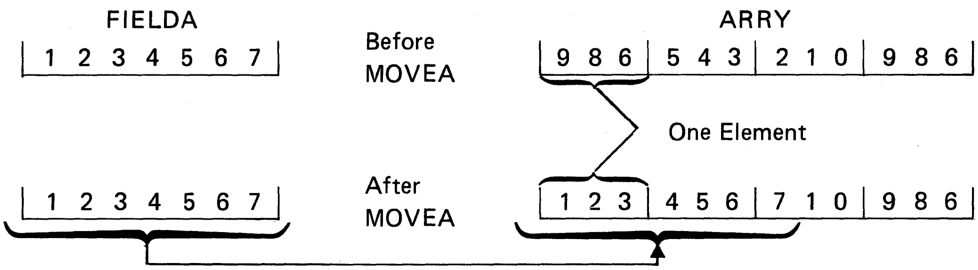


Figure 28-25 (Part 12 of 17). MOVEA Operation

Example: Alphameric field to numeric array move. No indexing on array.

Line	Form Type	Operation	Factor 2	Result Field Name
01	C	MOVEA	FIELDA	ARRY
02	C			



Example: Numeric field to numeric array move. No indexing on array.

Line	Form Type	Operation	Factor 2	Result Field Name
01	C	MOVEA	FIELDA	ARRY
02	C			

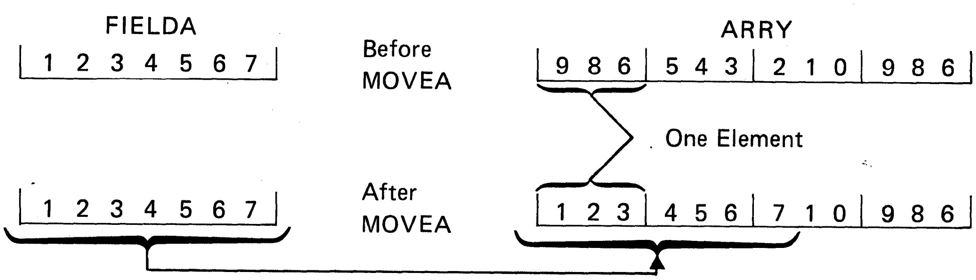


Figure 28-25 (Part 13 of 17). MOVEA Operation

Example: Numeric field to numeric array move. No indexing on array.

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators	Comments
01	C				MOVEA	FIELDA	ARRY					
02	C											

FIELDA: | - 1 2 3 4 5 6 | (the 6 carries the minus sign)

Before MOVEA — ARRY: | 9 8 6 | 5 4 3 | 2 1 0 | 9 8 6 |

One Element

After MOVEA — FIELDA: | - 1 2 3 4 5 6 |

ARRY: | 1 2 3 | - 4 5 6 | 2 1 0 | 9 8 6 |

(the 6 carries minus sign)

Example: Alphameric field to numeric array move. No indexing on array.

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators	Comments
01	C				MOVEA	FIELDA	ARRY					
02	C											

FIELDA: | 1 2 3 A B C 7 |

Before MOVEA — ARRY: | 9 8 6 | 5 4 3 | 2 1 0 | 9 8 6 |

One Element

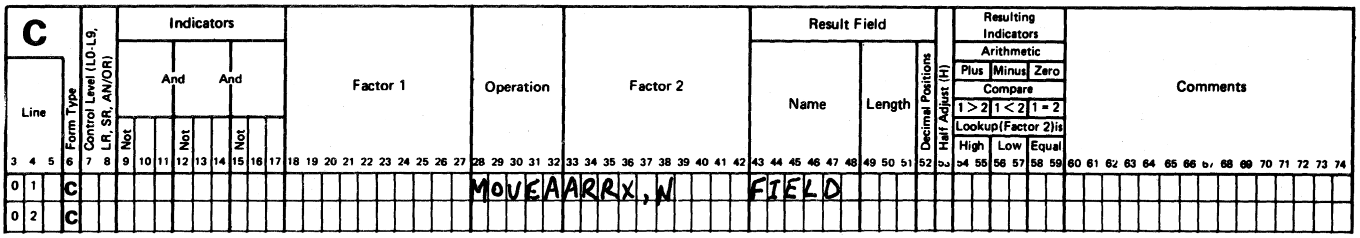
After MOVEA — FIELDA: | 1 2 3 A B C 7 |

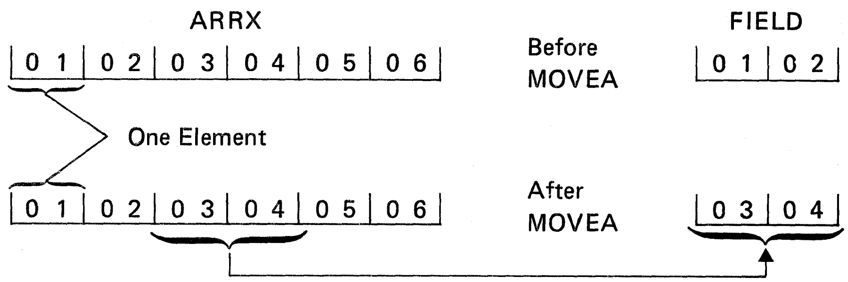
ARRY: | 1 2 3 | A B C | 7 1 0 | 9 8 6 |

The second element is invalid, because it is not valid numeric data.

Figure 28-25 (Part 14 of 17). MOVEA Operation

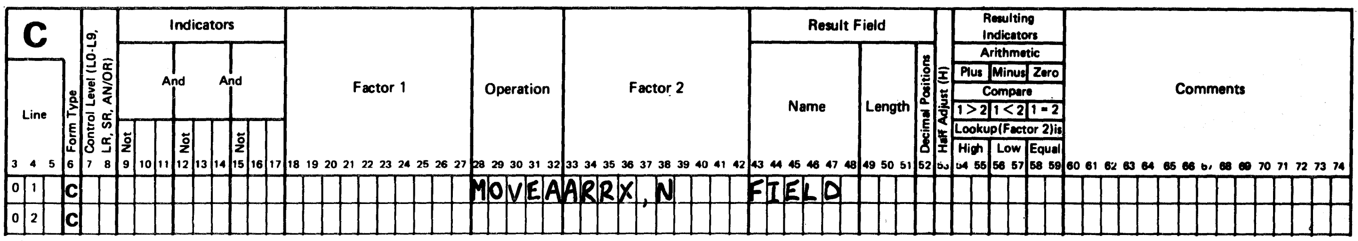
Example: Numeric array to alphameric field move. Variable indexing.

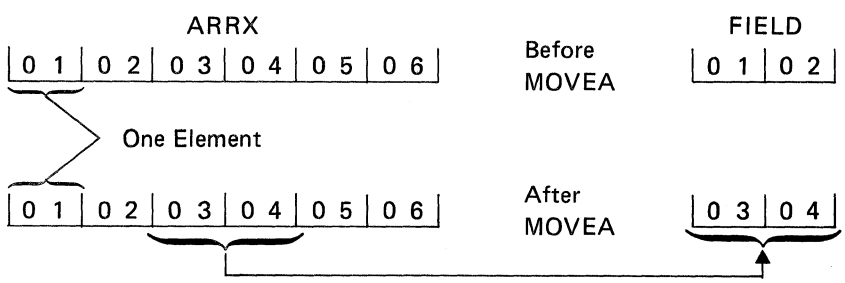




Note: N = 3

Example: Alphameric array to numeric field move. Variable indexing.

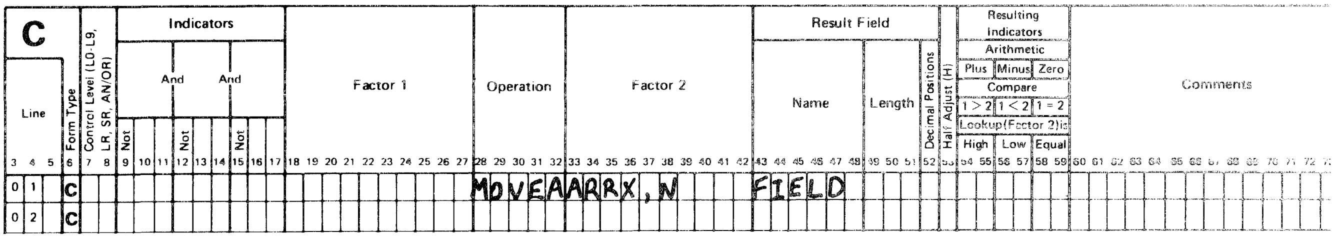


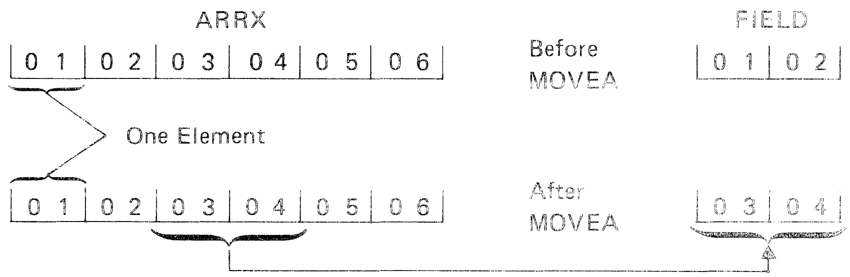


Note: N = 3

Figure 28-25 (Part 15 of 17). MOVEA Operation

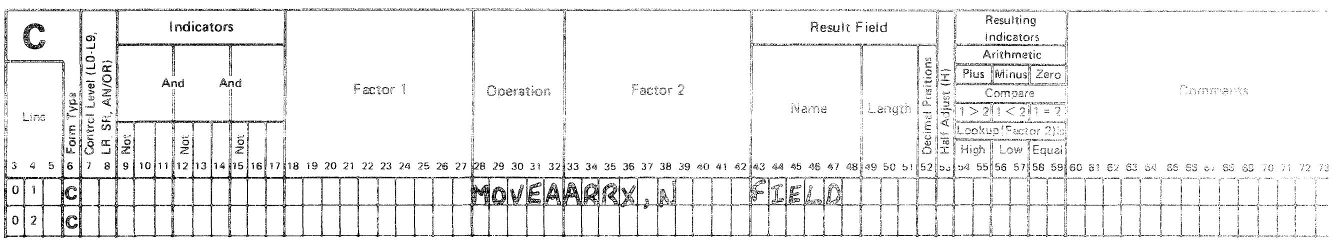
Example: Numeric array to numeric field move. Variable indexing.

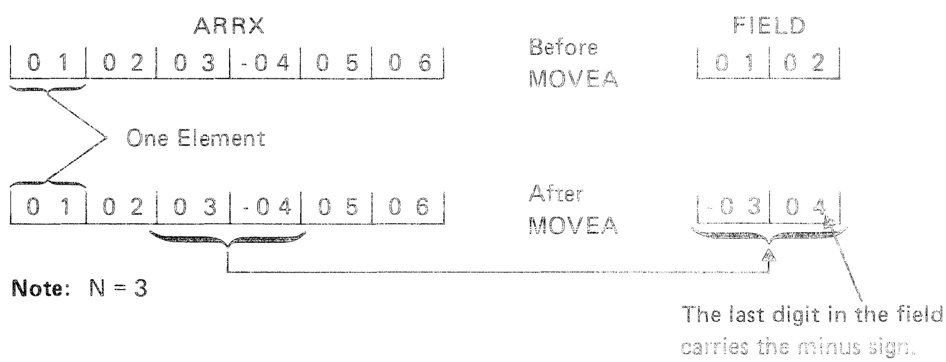




Note: N = 3

Example: Numeric array to numeric field move. Variable indexing.

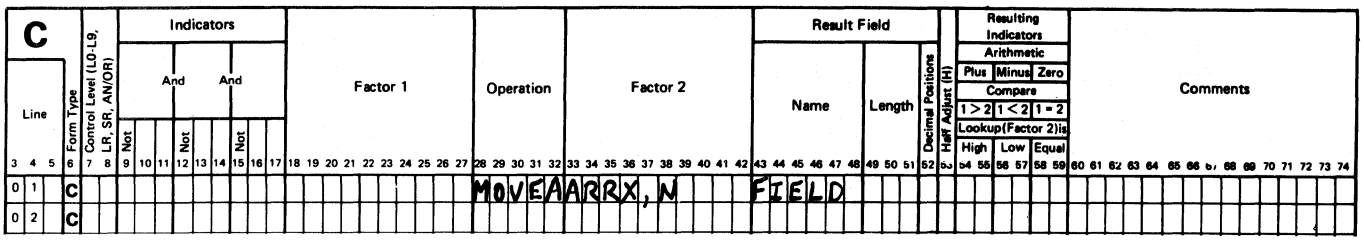




Note: N = 3

Figure 28-25 (Part 16 of 17). MOVEA Operation

Example: Alphameric array to numeric field move. Variable indexing.



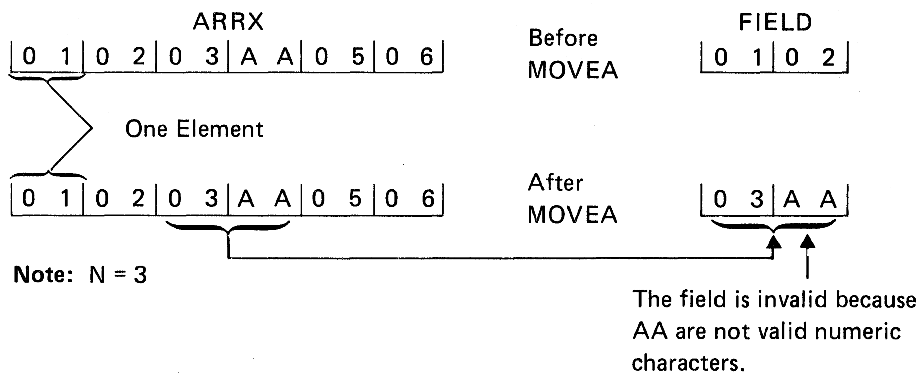


Figure 28-25 (Part 17 of 17). MOVEA Operation

MOVEL (Move Left)

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional	Optional	Blank	MOVEL	Required	Required	Blank	Blank	Blank

The MOVEL operation transfers characters from factor 2 to the leftmost positions in the result field. Moving begins with the leftmost character in factor 2. When a numeric field is moved into an alphameric field, both digit and zone portions of the rightmost character are transferred if that character is to be moved.

A summary of the rules for MOVEL operations for three conditions based on field lengths is as follows:

1. Factor 2 the same length as the result field:

   a. If factor 2 and the result field are numeric, the sign is moved with the rightmost position.
   b. If factor 2 is numeric and the result field is alphameric, the sign is moved with the rightmost position.
   c. If factor 2 is alphameric and the result field is numeric, a minus zone is moved into the rightmost position of the result field if the zone from the rightmost position of factor 2 is a D (characters J through R). However, if the zone from the rightmost position of factor 2 is not a D, a positive zone is moved into the rightmost position of the result field. Digit portions are converted to their corresponding numeric characters.
   d. If factor 2 and the result field are alphameric, all characters are moved.

2. Factor 2 longer than the result field:

   a. If factor 2 and the result field are numeric, the sign from the rightmost position of factor 2 is moved into the rightmost position of the result field.
   b. If factor 2 is numeric and the result field is alphameric, the result field contains only numeric characters.
   c. If factor 2 is alphameric and the result field is numeric, a minus zone is moved into the rightmost position of the result field if the zone from the rightmost position of factor 2 is a D (characters J through R). However, if the zone from the rightmost position of factor 2 is not a D, a positive zone is moved into the rightmost position of the result field. Other result field positions contain only numeric characters.
   d. If factor 2 and the result field are alphameric, only the number of characters needed to fill the result field are moved.

3. Factor 2 shorter than the result field:

   a. If factor 2 is either numeric or alphameric and the result field is numeric, the digit portion of factor 2 replaces the contents of the leftmost position of the result field. The sign in the rightmost position of the result field is not changed.
   b. If factor 2 is either numeric or alphameric and the result field is alphameric, the characters in factor 2 replace the equivalent number of the leftmost position in the result field. No change is made in the zone of the rightmost position of the result field.

The MOVEL operation is summarized in Figure 28-26.

Factor 2 and Result Field Same Length

	Factor 2		Result Field	
a. Numeric	7 8↓4 2 $\bar{5}$	Before MOVEL	5 6 7↓8 4	Numeric
	7 8↓4 2 $\bar{5}$	After MOVEL	7 8 4↓2 $\bar{5}$	
b. Numeric	7 8↓4 2 5	Before MOVEL	A K T 4 D	Alphameric
	7 8↓4 2 $\bar{5}$	After MOVEL	7 8 4 2 N	
c. Alphameric	P H 4 S N	Before MOVEL	5 6 7 8 4	Numeric
	P H 4 S N	After MOVEL	7 8 4 2 $\bar{5}$	
d. Alphameric	P H 4 S N	Before MOVEL	A K T 4 D	Alphameric
	P H 4 S N	After MOVEL	P H 4 S N	

Factor 2 Longer Than Result Field

	Factor 2		Result Field	
a. Numeric	0 0 0 0 0 8 4 2 $\bar{5}$	Before MOVEL	5↓6 7 8 4	Numeric
	0 0 0 0 0 8 4 2 $\bar{5}$	After MOVEL	0↓0 0 0 $\bar{0}$	
b. Numeric	9 0 3 1 7 8 4 2 $\bar{5}$	Before MOVEL	A K T 4 D	Alphameric
	9 0 3 1 7 8 4 2 $\bar{5}$	After MOVEL	9 0 3 1 7	
c. Alphameric	B R W C X H 4 S N	Before MOVEL	5 6 7 8↓4	Numeric
	B R W C X H 4 S N	After MOVEL	2 9 6 3↓$\bar{7}$	
d. Alphameric	B R W C X H 4 S N	Before MOVEL	A K T 4 D	Alphameric
	B R W C X H 4 S N	After MOVEL	B R W C X	

Factor 2 Shorter Than Result Field

	Factor 2		Result Field	
a. Numeric	7 8 4 2 $\bar{5}$	Before MOVEL	1↓3 0 9 4 3 2 1 0	Numeric
	7 8 4 2 $\bar{5}$	After MOVEL	7↓8 4 2 5 3 2 1 0	
a. Alphameric	C P T 5 N	Before MOVEL	1 3 0 9 4 3 2 1 0	Numeric
	C P T 5 N	After MOVEL	3 7 3 5 5 3 2 1 0	
b. Numeric	7 8 4 2 $\bar{5}$	Before MOVEL	B R W C X H 4 S A	Alphameric
	7 8 4 2 $\bar{5}$	After MOVEL	7 8 4 2 N H 4 S A	
b. Alphameric	C P T 5 N	Before MOVEL	B R W C X H 4 S A	Alphameric
	C P T 5 N	After MOVEL	C P T 5 N H 4 S A	

The arrow ↓ between numbers indicates a decimal point.

Figure 28-26. MOVEL Operations

MULT (Multiply)

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional	Optional	Optional	MULT	Required	Required	Optional	Optional	Optional

Factor 1 is multiplied by factor 2, and the product is placed in the result field. Factor 1 and factor 2 are not changed. If factor 1 is not present, the result field is multiplied by factor 2, and the product is placed in the result field. Be sure that the result field is large enough to hold the product. To determine the minimum length of the result field, use this rule: the length of the result field equals the length of factor 1 plus the length of factor 2.

MVR (Move Remainder)

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional	Optional	Blank	MVR	Blank	Required	Optional	Optional	Optional

The MVR operation moves the remainder from the previous divide operation to a separate field named as the result field. Factor 1 and factor 2 must not be used. This operation must immediately follow the divide operation.

The maximum length of the remainder (including decimal positions) is 15. The number of significant decimal positions is the greater of:

- The number of decimal positions in factor 1 of the previous divide operation.

- The sum of the decimal positions in factor 2 and the result field of the previous divide operation.

The maximum number of whole number positions in the remainder is equal to the whole number positions in factor 2 of the previous divide operation. Figure 28-27 shows the specification for a move remainder operation.

Line	Form Type	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions
01	C	14 N02	FIELDA	DIV	FIELDB	SAVE	6	0
02	C	14 N02		MVR		STORE	6	0
03	C							

Figure 28-27. Move Remainder Operation

NEXT (Next)

Indicators						Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Result Field	54-55	56-57	58-59
Optional	Optional	Required	NEXT	Required	Blank	Blank	Optional	Blank

The NEXT operation code forces the next input to the program to come from the device specified in factor 1. If NEXT is specified more than once between input operations, only the last operation is performed. The NEXT operation code can be used only for a WORKSTN file.

Note: For WORKSTN files, a device can be either a display station or an SSP-ICF session.

To use this operation, enter NEXT in columns 28 through 32. In factor 1, enter the name of a 2-character field that contains the device identification or a 2-character alphameric literal that is the device identification. In factor 2, enter the name of the WORKSTN file for which the operation is requested.

An indicator can be specified in columns 56 and 57. This indicator is set on if an exception or error occurs on the NEXT operation. If the INFSR subroutine is specified and columns 56 and 57 do not contain an indicator, the subroutine automatically receives control when an exception/error occurs. (For more information on the INFSR subroutine, see *Coding the INFSR Subroutine* in Chapter 6.) If the INFSR subroutine is not specified and columns 56 and 57 do not contain an indicator, the program halts when an exception or error occurs.

For more information on the NEXT operation code, see Chapter 6, *Using a WORKSTN File.*

POST (Post)

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional	Optional	Required	POST	Blank	Required	Blank	Optional	Blank

The POST operation allows you to retrieve status information for a specified display station that is using to a WORKSTN file. The status information is placed in the INFDS data structure that was specified in the result field. The program must contain the INFDS data structure for the WORKSTN file to use POST.

Factor 1 must contain a variable or an alphameric literal that identifies the display station whose status is being requested. The result field contains the name of the INFDS data structure in which this information is to be posted. Columns 56 and 57 can specify an indicator that is set on if an error occurs on the POST operation. An error occurs if the specified work station identifier is not using the file for which the INFDS data structure is specified.

If columns 56 and 57 do not specify an indicator but the program contains the INFSR subroutine, the subroutine automatically receives control when an error occurs. If the INFSR subroutine is not present and columns 56 and 57 do not contain an indicator, the program halts when an exception or error occurs. If the display station is not using the WORKSTN file, the device will be *not found*, and an error will occur on POST. (For more information on the INFSR subroutine, see *Coding the INFSR Subroutine* in Chapter 6.)

Columns 33 through 42, 49 through 55, and 58 and 59 must be blank for a POST operation.

READ (Read)

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional	Optional	Blank	READ	Required	Blank	Blank	Optional	Optional

The READ operation calls for immediate input from a demand or full-procedural file during the calculation phase of the program cycle. This operation differs from the FORCE operation because FORCE calls for certain input on the next program cycle, not on the present one.

The READ operation code must appear in columns 28 through 32. Factor 2 contains the name of the file from which a record should be read immediately. Full-procedural files must contain an indicator in columns 58 and 59. This indicator turns on when an end-of-file condition is reached for the demand file or the full-procedural file or for each READ operation after an end-of-file condition is reached. If the file in factor 2 is a demand file, the indicator in columns 58 and 59 does not turn off before the READ operation is performed. If the file is a full-procedural file, the indicator does turn off. If columns 58 and 59 are blank, a halt occurs on an end-of-file condition and on subsequent READ operations after the end-of-file condition is reached. Indicators can be specified in columns 7 through 17.

Note: In a READ operation, input occurs during calculation time and record-identifying indicators do not turn off.

An indicator can be specified in columns 56 and 57 if the READ operation is issued to a WORKSTN file. This indicator turns on if an exception condition occurs (that is, if the operator pressed one of the function control keys: Roll Up, Roll Down, Clear, Print, Record Backspace, or Help) or if an input error occurs. If columns 56 and 57 do not contain an indicator and either of these conditions occurs, the program halts unless the INFSR subroutine is specified. If the INFSR subroutine is specified, the subroutine automatically receives control, and an exception or error occurs. (For more information on the INFSR subroutine, see *Coding the INFSR Subroutine* in Chapter 6.)

The following columns must remain blank for a READ operation: columns 18 through 27 (factor 1), columns 43 through 48 (result field), columns 49 through 51 (field length), column 52 (decimal positions), column 53 (half-adjust), and columns 54 and 55 (resulting indicators).

If a READ operation is not successful, you must reposition the file by using either a SETLL or a CHAIN operation. If the file is not repositioned, all following READ, READE, and READP operations will fail.

The following files can appear as factor 2 in a READ operation:

- Sequential DISK files processed consecutively and specified as input or update files.
- Direct DISK files processed consecutively as input or update files.
- Indexed DISK files processed sequentially by key field and specified as input or update files.
- Indexed DISK files processed sequentially by limits and specified as input or update files.
- Sequential, direct, and indexed DISK files processed randomly and sequentially as input or update files.
- WORKSTN files.
- SPECIAL files.
- CONSOLE files.
- BSCA files.

READE (Read Equal Key)

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional	Optional	Required	READE	Required	Blank	Blank	Blank	Required

The READE operation retrieves the next sequential record from an indexed full-procedural DISK file (identified by an F in position 16 of the file description specifications) if the key field of the record matches factor 1.

Factor 1 identifies the record to retrieved. It is required, and it can be a field name, an array or table element, or a literal. Factor 1 must have the same field type and length as the key field of the file being read.

Factor 2 must contain the name of the file to be read as specified on the file description specification. This file must be full-procedural

A resulting indicator must be specified in positions 58 and 59 of the calculation specification. This indicator turns on if the next sequential record does not have a key field equal to factor 1, or if end of file occurs. If this indicator turns on, a record is not read by the program.

If a READE operation is not successful, you must reposition the file by using either a SETLL or a CHAIN operation. If the file is not repositioned, all following READ, READE, and READP operations will fail.

READP (Read Prior Record)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Optional	Optional	Blank	READP	Required	Blank	Blank	Blank	Required

The READP operation reads the prior record from a full-procedural DISK file (identified by an F in position 16 of the file description specifications). For example, if record X was just read using the READ operation code, READP reads the record prior to record X.

Factor 2 must contain the name of the file to be read as specified on the file description specifications. This file must be full-procedural.

Positions 58 and 59 must contain an indicator. This indicator turns on if no prior record exists in the file (beginning-of-file condition). It turns off before each READP operation. If a READP operation is not successful, you must reposition the file by using either a SETLL or CHAIN operation. If the file is not repositioned, all following READ, READE, and READP operations will fail.

REL (Release)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Optional	Optional	Required	REL	Required	Blank	Blank	Optional	Blank

The REL operation releases the device specified in factor 1 from the program. Either a requesting or acquired device can be released with the REL operation code. The specified device is released when the REL operation occurs during the calculations unless the device is the requester of a single-requester-terminal program. If the device specified in factor 1 is the requester of a single-requester- terminal program, the device is released at end of job, not when the operation code is read in the calculations. (If the device is a display station, it is no longer available to the program, but it is available for system log messages.)

Factor 2 must contain the filename for the WORKSTN file.

If an exception or error occurs on the attempt to release the device, the indicator specified in columns 56 and 57 turns on. If no indicator is specified, the program halts unless the INFSR subroutine is specified in the program. If the INFSR subroutine is specified, the INFSR subroutine automatically receives control when an exception or error occurs and no indicator is specified in columns 56 and 57.

When all devices are released from a primary WORKSTN file, the file goes to end of file, and RPG turns on the LR indicator. If the program containing the primary file is a never-ending program, the system operator must enter the STOP SYSTEM command before the WORKSTN file will go to end of file.

When all devices are released from a demand WORKSTN file and the program is not a never-ending program, the first READ operation after the last REL operation causes the READ end-of-file indicator to turn on (columns 58 and 59). You can then turn on the LR indicator, unless the LR indicator was specified as the end-of-file indicator. If the program containing the demand WORKSTN file is a never-ending program, the end-of-file indicator turns on when the system operator enters the STOP SYSTEM command. You can then turn on the LR indicator, unless the LR indicator was specified as the end-of-file indicator.

If RESTORE-NO is specified on the control language WORKSTN statement, a display format from the program may appear on the screen after the display station has been released. If RESTORE-YES is specified on the WORKSTN statement, the command display appears on the screen immediately when the display station is released.

For more information on the REL operation code, see Chapter 6, *Using a WORKSTN File*.

RLABL (RPG Label)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Blank	Blank	Blank	RLABL	Blank	Required	Blank	Blank	Blank

The RLABL operation allows the subroutine specified in an EXIT operation to refer to a field, data structure, array, table, or indicator defined in the RPG program. RLABL operations must be specified immediately after the EXIT operation that refers to the subroutine using the field, data structure, array, table, or indicator in the RLABL specification (see Figure 28-28).

The number of RLABL operations following the EXIT operation must correspond to the number of items referred to in the assembler subroutine. The order of the RLABL operations must correspond to the order of references in the assembler subroutine.

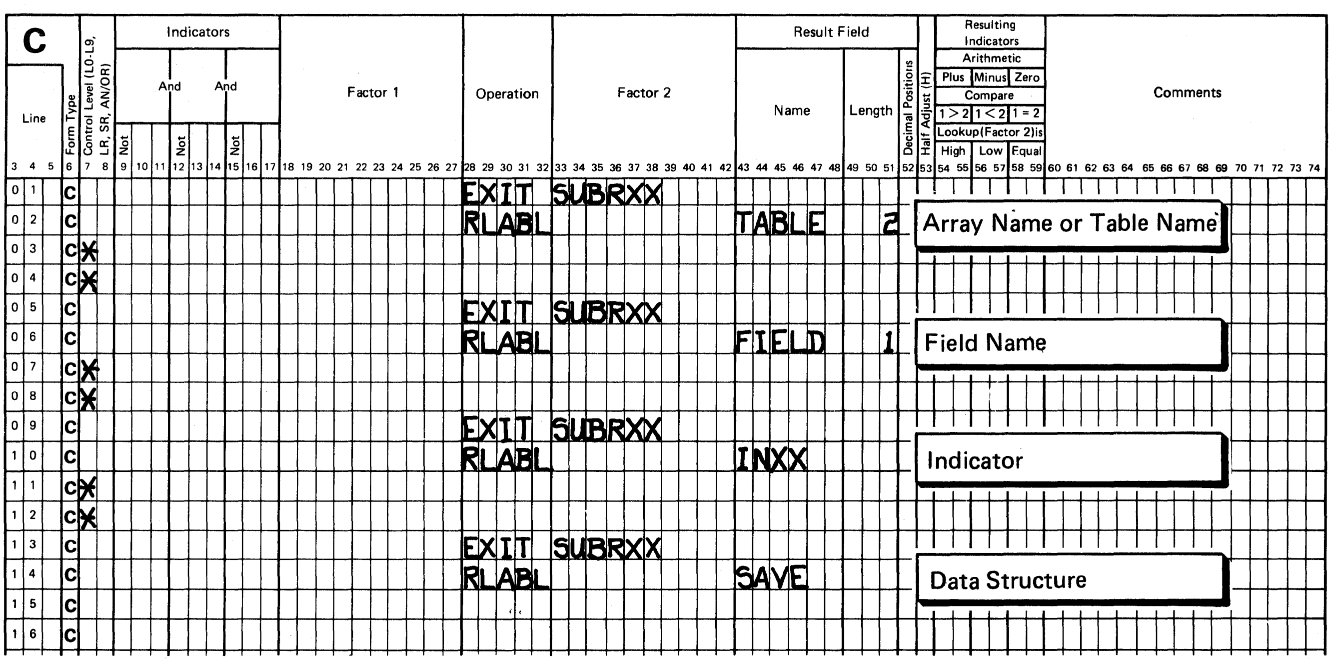


Figure 28-28. RPG Coding for RLABL Field Entries

The entries used with RLABL on the calculation specifications are as follows:

Columns	Entry
Operation (28-32)	RLABL
Result field 43-48	Field name, data structure name, array name, table name, or indicator (INxx, where xx is the indicator)
Field length (49-51)	Length of field (optional)
Decimal positions (52)	Decimal indication (optional)

Only RLABL operations specifying a field, array, or table name can have entries for field length (columns 49 through 51) and decimal positions (column 52).

The following columns must be blank for an RLABL operation: columns 7 and 8 (control level), columns 9 through 17 (indicators), columns 18 through 27 (factor 1), columns 33 through 42 (factor 2), column 53 (half adjust), and columns 54 through 59 (resulting indicators).

A name defined by a TAG, BEGSR, or ENDSR specification cannot be used in an RLABL specification.

Referring to an Indicator

When an indicator is specified in an RLABL operation, use the form INxx as the result field, where xx is the indicator to be transferred to the subroutine. For example, if the MR indicator is to be transferred to a subroutine, specify INMR as the result field for the RLABL operation.

When an indicator is specified in the RLABL operation, the RPG compiler creates the following parameters and passes them to the assembler-language subroutine:

```
B     SUBRxx
DC    XL1 '00'
DC    XL1 'Mask for the indicator'
DC    XL1 'Displacement to the indicator from XR1'
```

Referring to a Field

When a field name is specified in the RLABL operation, the RPG compiler creates the following parameters and passes them to the assembler-language subroutine:

```
B     SUBRxx
DC    IL1 'Field length-1'
DC    AL2 (rightmost address of field)
```

Referring to a Data Structure

When a data structure is specified in the RLABL operation, the RPG compiler creates the following parameters and passes them to the assembler-language subroutine:

```
B     SUBRxx
DC    XL3 'FFFFFF'
DC    IL2 'Data structure length-1'
DC    AL2 (leftmost address of data structure)
```

Referring to an Array or Table

The subroutine can refer to an array or table defined in the RPG program by using the control field created for that array or table. This control field is called the DTT (define the table), and one is created for each array or table built by the RPG program. The control field is in the following format:

Bytes	Meaning
0-1	Address of rightmost byte of the first entry
2-3	Address of rightmost byte of the last entry
4-5	Initialized to the address of the rightmost byte of first entry; used at object time for address of rightmost byte of the last looked-up entry
6-7	Length of an entry
8-13	Array name (arrays only)

The subroutine can obtain the data retrieved from the preceding LOKUP operation by using the address in bytes 4 and 5. To access the array or table itself, the address in bytes 0 and 1 must be used. Data which the subroutine uses is left unpacked.

When a table or array is specified in the RLABL operation, the RPG compiler creates the following parameters and passes them to the assembler subroutine:

B SUBRxx

DC IL1 'Entry length-1'

DC AL2 (leftmost address of the DTT)

See Figures 28-29 and 28-30 for examples of RPG linkage specifications.

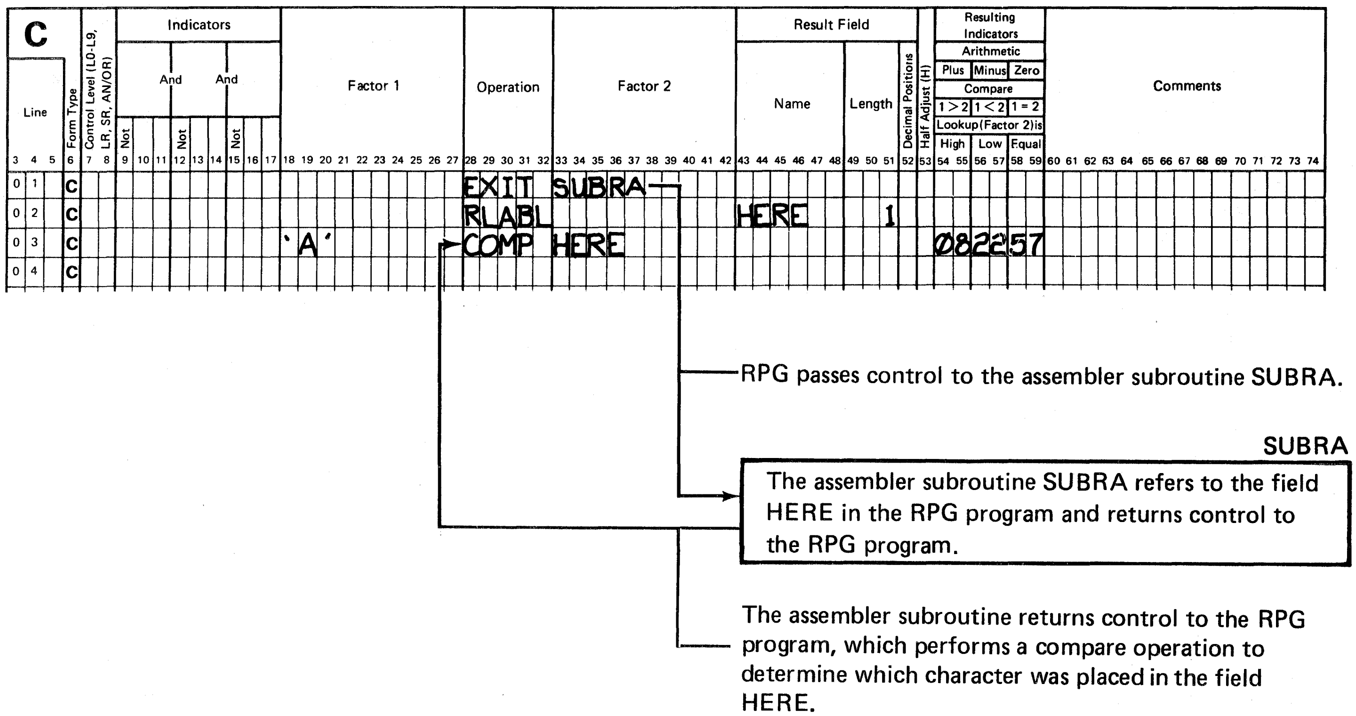
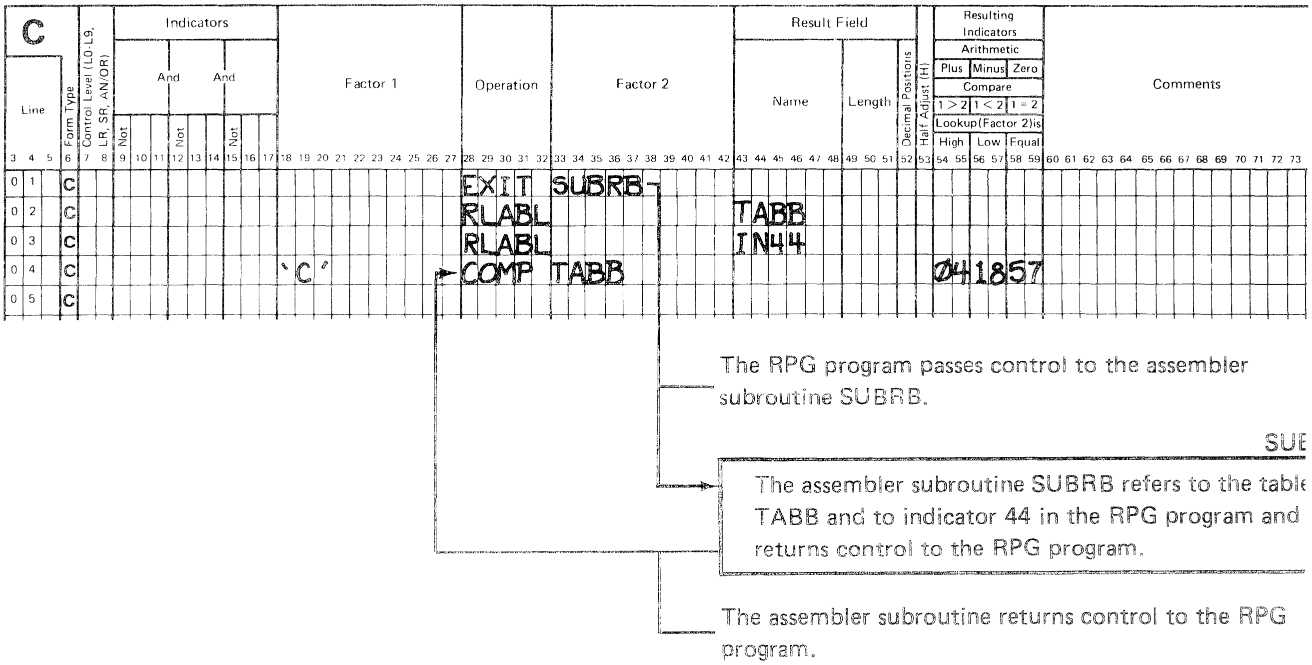


Figure 28-29. RPG Linkage to an Assembler Language Subroutine



The subroutine refers to both RLABL entries. It first tests the indicator. If the indicator is off, the subroutine returns control to the RPG program. If the indicator is on, the subroutine moves a character C into the last looked up entry in the table, TABB. When the subroutine returns control to the RPG program, the RPG program performs a compare operation to see whether the subroutine placed a C in TABB.

Figure 28-30. RPG Linkage to an Assembler Language Subroutine

Considerations for the Assembler-Language Programmer

To write an assembler-language subroutine that is linked to an RPG program, the assembler-language programmer must be aware of the following:

- The name of the subroutine must be the same as the name specified in factor 2 of the RPG EXIT operation.
- Upon entry to the assembler-language subroutine, the address recall register (ARR) contains a pointer to the parameters that represent the RPG fields to be referred to by the assembler-language subroutine. The return point to the RPG program is the first byte after the parameters.
- If the subroutine makes use of registers 1 and 2, the contents of these registers must be stored upon entry to, and restored before exit from, the subroutine.
- All external subroutines should be part of the root segment and not in overlays. You can do this by not specifying a category (CATG) statement in the assembler-language subroutine or by specifying a category of zero.

During compilation, the name of the library that contains the assembler-language subroutine can be specified. The input library is used when the subroutine library is not specified.

Message-Retrieving Subroutine (SUBR23)

The message-retrieving subroutine (SUBR23) allows you to retrieve messages from a user message member. After the message has been retrieved, it can be modified and written to an output file.

Linkage to SUBR23 is by the EXIT operation code, and input parameters are passed to SUBR23 by RLABL operation codes. To use SUBR23, specify EXIT in columns 28 to 31 and SUBR23 in columns 33 to 38. Four RLABL operation codes must be specified after the EXIT operation with the following result-field entries:

Result Field	Description
MIC number	Name of a 4-digit numeric field that contains the MIC (message identification code) of the text to be retrieved.
Text area	Name of the alphameric field or data structure into which the message text is read. The maximum length of a level-1 message is 75 characters and of a level-2 message is 225 characters.
Level	Name of a one-digit numeric field that designates the user message member level. A value of 1 in this field indicates a message level of 1; a value of 2 indicates a message level of 2.
Rcode	Name of a one-digit numeric field that contains the return codes. The return codes and their meanings are as follows: *Return Code* — *Meaning* 0 — Message was successfully retrieved with no truncation. 1 — Message was successfully retrieved; however, it was truncated because the length of the text area was less than the message length. 2 — Message was not found. 3 — Message level was invalid. 4 — An invalid MIC value was diagnosed. 5 — Message member was not found or message text length exceeds the level-1 maximum.

The text area, which is specified by the second RLABL operation, is blanked before each attempt to retrieve a message; therefore, a blank text area is returned to the user program when the return code value is 2 or greater. A total of 225 positions in the text area are blanked unless the text area is less than 225 characters in length.

SET (Set)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Optional	Optional	Optional	SET	Optional	Optional	Optional	Optional	Optional

The SET operation can be used only with input files assigned to the device KEYBORD, or with a CONSOLE file if the result field contains the word ERASE. Both KEYBORD and CONSOLE are specified in columns 40 through 46 of the File Description Specifications. All SET operations are directed to the display station that loaded the program.

The SET operation allows any one or any combination of the following:

- Command keys identified in columns 54 through 59 to be pressed
- The field, literal, or array or table element specified in factor 1 to be displayed on the display screen
- User messages (from USER1 message member) 0001 to 0099 to be displayed when numbers 01 to 99, respectively, are specified in the nn portion of the SETnn and KEYnn operation codes
- The buffer for a CONSOLE file to be blanked if ERASE is specified in the result field of the SET operation

Factor 1 can contain the constant, literal, field name, or table or array element to be displayed.

Factor 2 must contain the name of the CONSOLE file if ERASE is coded in the result field. For all other SET operations, leave factor 2 blank.

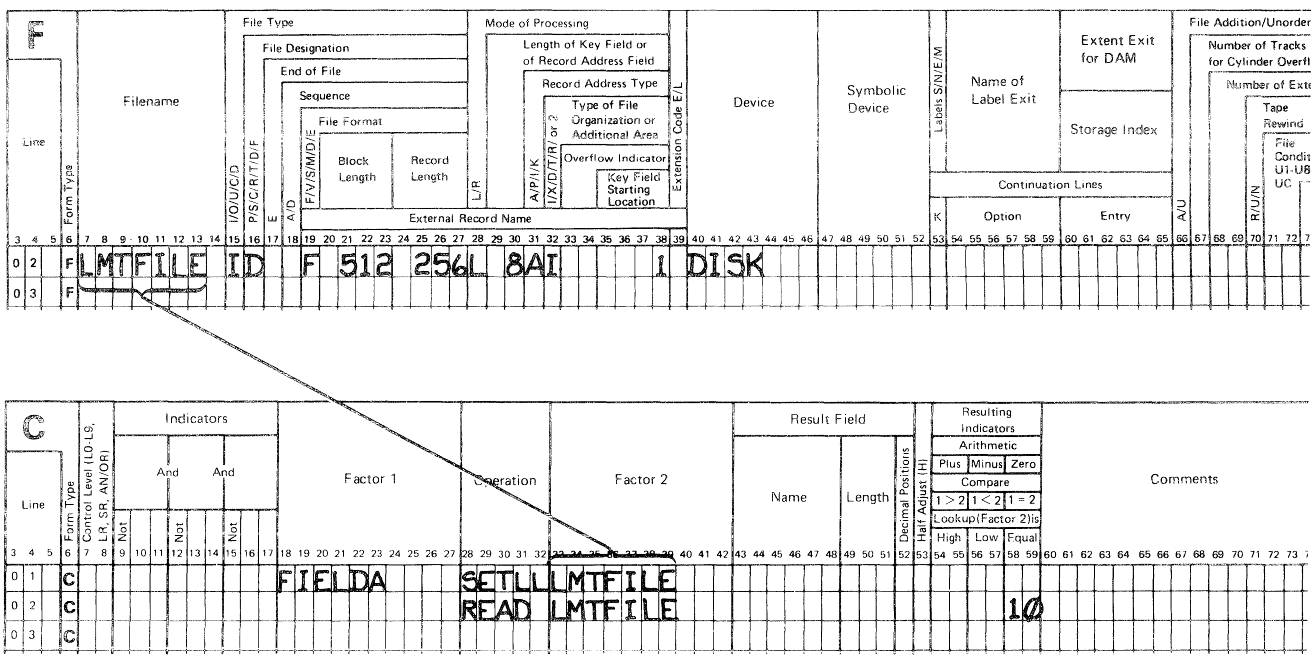
The result field must contain ERASE if the name of the CONSOLE file is coded in factor 2. For all other SET operations, leave the result field blank. See *Calculation Specifications for a SET Operation* in Chapter 9 for more information on coding the SET operation.

SETLL (Set Lower Limits Operation)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Optional	Optional	Required	SETLL	Required	Blank	Blank	Blank	Blank

The SETLL operation allows the lower limits for indexed full-procedural and indexed demand files being processed sequentially within limits to be set during calculations. Factor 1 must contain a field name or literal representing the value of the lower limit being set. The length of the field or literal must be equal to the total length of the key field specified for the file named in factor 2. Factor 2 must contain the name of the file for which the lower limit is to be set.

Figure 28-31 shows an example of SETLL coding.



FIELDA is defined on input specifications as an eight-position alphameric field.

Figure 28-31. SETLL Operation Code

Notes:

1. *When a lower limit is specified by SETLL, the end-of-file indicator specified for the READ operation to the file being processed is not set off by the RPG cycle.*

2. *If a READ operation is performed to the file prior to a SETLL operation, the record with the lowest key field in the file is fetched.*

SETOF (Set Off)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Optional	Optional	Blank	SETOF	Blank	Blank	One required		

The SETOF operation turns off any indicators specified in columns 54 through 59. At least one resulting indicator must be specified in columns 54 through 59.

SETON (Set On)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Optional	Optional	Blank	SETON	Blank	Blank	One required		

The SETON operation turns on any indicators specified in columns 54 through 59. At least one resulting indicator must be specified in columns 54 through 59.

SHTDN (Shutdown)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Optional	Optional	Blank	SHTDN	Blank	Blank	Required	Blank	Blank

The SHTDN operation turns on the resulting indicator specified in columns 54 and 55 if the system operator has requested shutdown. The indicator can then be used to condition ending the program in an orderly manner, such as printing some partial totals and going to normal end of job.

Columns 28 through 32 must contain SHTDN, and columns 54 and 55 must contain one of the following valid indicators: 01 through 99, L1 through L9, U1 through U8, H1 through H9, or LR.

SORTA (Sort an Array)

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional	Optional	Blank	SORTA	Required	Blank	Blank	Blank	Blank

The SORTA operation allows you to sequence the elements of an array while a program is running. You can ensure that the elements of the array are in the proper sequence for a LOKUP operation by performing a SORTA operation.

The array specified in factor 2 is sorted into the sequence specified in the extension specifications for the array. If no sequence is specified, the array is sorted into ascending sequence. The standard EBCDIC collating sequence is used for the SORTA operation. If an alternative collating sequence has been defined, it is not used. Related arrays, if existing, are not sorted. Only the array specified in factor 2 is sorted.

For examples of the SORTA operation, see Figure 28-32.

Note: Columns 18 through 27 (factor 1) and 43 through 59 (result field, half-adjust, and resulting indicators) must be blank if a SORTA operation is specified.

In Figure 28-32, the array ARY is sorted into ascending order because no entry is specified for sequence (columns 45) in the extension specifications. ARYA is sorted into ascending order because column 45 of the extension specifications contains A; ARYD is sorted into descending order because column 45 contains D.

Line	Form Type	Record Sequence of the Chaining File / Number of the Chaining Field / From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
01	E			ARY	2	10	3									
02	E			ARYA		15	5			A						
03	E			ARYD		10	4		0	D						
04	E															

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Plus / Minus / Zero	Comments
01	C											
02	C			Sort ARY and ARYA into ascending order.								
03	C											
04	C				SORTA	ARY						
05	C				SORTA	ARYA						
06	C											
07	C			Sort ARYD into descending order.								
08	C											
09	C				SORTA	ARYD						
10	C											

Figure 28-32. Example of SORTA Operation Code

SQRT (Square Root)

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional	Optional	Blank	SQRT	Required	Required	Blank	Blank	Blank

The SQRT operation derives the square root of the field named in factor 2. The square root of factor 2 is placed in the result field. Factor 1 is not used. An entire array can be used in a SQRT operation if factor 2 and the result field contain array names.

The number of decimal places in the result field can be either less than or greater than the number of decimal places in factor 2. However, the result field should not have less than half the number of decimal places in factor 2. The result of a SQRT operation is always half-adjusted.

If the value of the factor 2 field is negative, the job halts. The person using the display station can continue processing by responding to the error message. When processing continues, the result field is set to zero.

SUB (Subtract)

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional	Optional	Optional	SUB	Required	Required	Optional	Optional	Optional

Factor 2 is subtracted from factor 1. The difference is placed in the result field. Factor 1 and factor 2 are not changed by the operation. Subtracting a field from itself is a method of setting the result field to zeros. If factor 1 is not present, factor 2 is subtracted from the result field, and the difference is placed in the result field.

TAG (Tag)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Optional	Blank	Required	TAG	Blank	Blank	Blank	Blank	Blank

The TAG operation names the operation to which the program branches in the GOTO operation. If the TAG appears within a subroutine, the associated GOTO must appear within the same subroutine.

Factor 1 contains the label that must begin in column 18. The same label cannot be used for more than one TAG operation (or elsewhere as a subroutine name or ENDSR label).

Factor 2 and the result field are not used. No indicators can be entered in columns 9 through 17 for a TAG operation.

See Figures 28-20 and 28-21 for examples of the TAG operation.

TESTB (Test Bit)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Optional	Optional	Blank	TESTB	Required	Required	One required		

The TESTB operation compares the bits identified in factor 2 with the corresponding bits in the field named as the result field. The result field must be a one-position character field. Resulting indicators in columns 54 through 59 reflect the status of the result-field bits. Factor 2 is always a source of bits for the result field. The result field is the field in which corresponding bits are compared with the bits specified in factor 2.

Factor 2 can contain:

- *Bit numbers 0-7*: From 1 to 8 bits can be tested per operation. The bits to be tested are identified by the numbers 0 through 7 (0 is the leftmost bit). The bit numbers must be enclosed in apostrophes, and the entry must begin in column 33. For example, to test bits 0, 2, and 5, enter '025' in factor 2.

- *Field name*: The name of a one-position alphameric field, array element, or table element can be specified in factor 2. In this case, the bits that are on in the field, array element, or table element are tested in the result field; bits that are off are not tested.

See Figure 28-33 for a summary of TESTB operations.

Indicators assigned in columns 54 through 59 reflect the status of the result-field bits. At least one indicator must be assigned, and as many as three can be assigned for one operation. Two indicators can be the same for a TESTB operation, but not three. For TESTB operations, the resulting indicators turn on as follows:

Columns 54-55:

An indicator in these columns turns on if each bit specified in factor 2 or each bit that is on in the factor 2 field is off in the result field.

Columns 56-57:

An indicator in these columns turns on if the bits specified in factor 2 or the bits that are on in the factor 2 field are of mixed status (some on, some off) in the result field.

Columns 58-59:

An indicator in these columns turns on if each bit specified in factor 2 or each bit that is on in the factor 2 field is on in the result field.

Note: If the field in factor 2 has no bits on, then this indicator turns on.

The operation code TESTB must appear in columns 28 through 32. Conditioning indicators can be used in columns 7 through 17. At least one resulting indicator must be assigned in columns 54 through 59. As many as three resulting indicators can be assigned, but not more than two can be the same. Factor 1, decimal positions, and the half-adjust columns must be blank.

The following TESTB operation compares bits 0 and 7 with corresponding bits in the field named BITS. If bits 0 and 7 are off in the field named BITS, indicator 20 turns on. If bits 0 and 7 are of mixed status (one on, one off) in the field named BITS, indicator 21 turns on. If bits 0 and 7 are on in the field named BITS, indicator 22 turns on.

Line	Form Type	Factor 1	Operation	Factor 2	Result Field Name	Resulting Indicators High	Low	Equal
10	C		TESTB	'07'	BITS	20	21	22

The following operation compares the bits that are on in the field named ALPHA with corresponding bits in the field named BITS. If the bits that are on in the field named ALPHA are off in the field named BITS, indicator 20 turns on. If the bits that are on in the field named ALPHA are of mixed status (some on, some off) in the field named BITS, indicator 21 turns on. If the bits that are on in the field named ALPHA are on in the field named BITS, indicator 22 turns on.

Line	Form Type	Factor 1	Operation	Factor 2	Result Field Name	Resulting Indicators High	Low	Equal
	C		TESTB	ALPHA	BITS	20	21	22

The following operations use a one-position array element either as a source of bits or as a result field, or both. In the first operation, the bits that are on in the field named ALPHA are compared to corresponding bits in the array element ARR,NX. For example, assume that bits 1 and 4 are on in the field named ALPHA. If bits 1 and 4 are off in array element ARR,NX, indicator 20 turns on. If bits 1 and 4 are of mixed status (one on, one off) in array element ARR,NX, indicator 21 turns on. If bits 1 and 4 are on in array element ARR,NX, indicator 22 turns on.

Line	Form Type	Factor 1	Operation	Factor 2	Result Field Name	Resulting Indicators High	Low	Equal
12	C		TESTB	ALPHA	ARR,NX	20	21	22
14	C		TESTB	'24'	ARE,12	20	21	22
16	C		TESTB	ARE,12	ARR,NX	20	21	22

Figure 28-33. Summary of TESTB Operations

TESTZ (Test Zone)

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional	Optional	Blank	TESTZ	Blank	Required	One required		

The TESTZ operation tests the zone of the leftmost character in the result field. The result field must be alphameric because this operation can be done only on alphameric characters. At least one resulting indicator must be specified in columns 54 through 59. Resulting indicators turn on according to the results of the test. The characters &, A through I, and any other character with the same zone as the character A turn the plus indicator on. The characters - (minus), J through R, and any other character with the same zone as the character J turn the minus indicator on. Characters with any other zone turn the zero indicator on. Factor 1 and factor 2 are not used in this operation.

TIME (Time of Day)

Indicators					Result	Resulting Indicators		
7-8	9-17	Factor 1	Operation	Factor 2	Field	54-55	56-57	58-59
Optional	Optional	Blank	TIME	Blank	Required	Blank	Blank	Blank

The TIME operation accesses the system time of day and, if specified, the system date.

The system date accessed is not the same field as UDATE (the session date), so the system date can differ from UDATE. See the *System Reference* manual for a complete description of the system date and the control language DATE statement.

Columns 28 through 32 must contain the operation code TIME, and columns 43 through 48 (the result field) must specify the name of a numeric field with zero decimal positions into which the time of day or the time of day and the system date are written.

To access the time of day only, specify the result field as a 6-digit numeric field. To access both the time of day and the system date, specify the result field as a 12-digit numeric field. The time of day is always placed in positions 1 through 6 of the result field in the format hhmmss, where hh is hours, mm is minutes, and ss is seconds. If the system date is included, it is placed in positions 7 through 12 of the result field. The date format depends on the system date format and can be mmddyy, ddmmyy, or yymmdd.

XFOOT (Summing the Elements of an Array)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Optional	Optional	Blank	XFOOT	Required	Required	Optional	Optional	Optional

The XFOOT operation can be used only on numeric arrays. XFOOT adds the elements of the array together and places the sum into the field specified as the result field. Factor 1 is not used. Factor 2 contains the name of the array.

Z-ADD (Zero and Add)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Optional	Optional	Blank	Z-ADD	Required	Required	Optional	Optional	Optional

Factor 2 is added to a field of zeros. The sum is placed in the result field. Factor 1 is not used.

Z-SUB (Zero and Subtract)

Indicators		Factor 1	Operation	Factor 2	Result Field	Resulting Indicators		
7-8	9-17					54-55	56-57	58-59
Optional	Optional	Blank	Z-SUB	Required	Required	Optional	Optional	Optional

Factor 2 is subtracted from a field of zeros. The difference, which is actually the negative of factor 2, is placed in the result field. You can use this operation to change the sign of a field. Factor 1 is not used.

Chapter 29. Storage Dump of an RPG Program

Chapter 29. Storage Dump of an RPG Program

To help you debug a program, you can request a storage dump of the program. The storage dump is a printout showing the contents of fields, constants, indicators, work areas, and other areas of storage used by your program.

To request a storage dump, cancel your program at the system console and use the DUMP procedure. The DUMP procedure is described in the *System Reference* manual.

Figure 29-1 is the listing of a program that reads an indexed file and displays all records that have a key field (DEPTNO) equal to the department number entered at the display station. Figure 29-2 is a storage dump of the program shown in Figure 29-1 with each area of storage identified.

```
SOURCE PROGRAM NAME---------- RPGEXA        MODIFICATION REFERENCE NUMBER--- 000015
INPUT SOURCE LIBRARY--------- JAKLIB        MODIFICATION DATE--------------- 10/21/82
OUTPUT LIBRARY--------------- JAKLIB        MODIFICATION TIME--------------- 08.55
SUBROUTINE INPUT LIBRARY----- JAKLIB        COMPILER WORK FILES BLOCK SIZE-- 0040

LIST OF OPTIONS SPECIFIED FOR THIS COMPILE--

NODSM                                                    SOURCE OVERRIDES--
PRINT
NOXREF
00 REQUESTING DISPLAY STATIONS MAXIMUM
NONEP
NOHALT
REPLACE
LINK
NOOBJECT
GEN
```

Prolog

Figure 29-1 (Part 1 of 4). Sample Program Listing

```
H     024                                                           RPGEXA

FDISPLAY CP        373            WORKSTN
FEMPFILE IF    256 256  8AI     1 DISK

E                    AR         10 29

IDISPLAY NS  02   1 CA
I                                        2   9 DEPT
I        NS  01   1 C
IEMPFILE NS
I                                        1   8 DEPTNO
I                                       21  49 EMPNAM
INAMES       DS
I                                        1 290 AR

C   01                Z-ADD0100      MIC     40
C   01                Z-ADD1         LEVEL   10
C   01                EXIT SUBR23
C                     RLABL          MIC
C                     RLABL          ERR100 75
C                     RLABL          LEVEL
C                     RLABL          RCODE   10
C   01                Z-ADD0101      MIC
C   01                EXIT SUBR23
C                     RLABL          MIC
C                     RLABL          ERR101 75
C                     RLABL          LEVEL
C                     RLABL          RCODE
C   01                GOTO END
C   02 KG             GOTO END
C   02                Z-ADD*ZERO     I       30
C   02                MOVE *BLANK    AR
C   02      DEPT      CHAINEMPFILE              99
C   02 99             GOTO END
C           LOOP      TAG
C                     ADD  1         I
C           I         COMP 11                       21
C   21                GOTO END
C                     MOVE EMPNAM    AR,I
C                     READ EMPFILE                  90
C   90                GOTO END
C           DEPT      COMP DEPTNO                   20
C   20                GOTO LOOP
C           END       TAG

ODISPLAY D        01NKG
O       OR        02NKG
O                                   K6 'FORMAT'
O                         DEPT       8
O                 21      ERR100    83
O                 99      ERR101    83
O                         NAMES    373

* * * * * E N D   O F   S O U R C E * * * * *
```

Figure 29-1 (Part 2 of 4). Sample Program Listing

```
INDICATORS USED
        KG 01 02 20 21 90 99

            EXECUTION TIME TABLES AND ARRAYS

 STMT#    TABLE/  DEC   ENTRY  NUMBER OF    DTT       T/A
 DEFINED ARRAY    POS   LENGTH ENTRIES      DISP      DISP
  0003    AR               029    00010      0100      012A
                                                           Address of the rightmost byte of the first element in the a
  FIELD NAMES USED
 STMT#   NAME   DEC   LNG    DISP                   Address of the leftmost byte of the DTT.
  0010 NAMES          0290    010E— Data Structure. DISP is the address of the leftmost byte of the data structure.
  0005 DEPT           0008    0254
  0008 DEPTNO         0008    025C
  0009 EMPNAM         0029    0279
  0012 MIC       0    0004    0317
  0013 LEVEL     0    0001    0318 } Fields. DISP is the address of the rightmost byte of the field.
  0016 ERR100         0075    02C4
  0018 RCODE     0    0001    0319
  0022 ERR101         0075    030F
  0027 I         0    0003    031C

  LABELS USED
 STMT#   NAME     TYPE
  0031 LOOP       TAG
  0040 END        TAG

 ERROR NUMBER   STATEMENT NUMBER
   RPG-0097           0007
   RPG-0159

  ERROR SEVERITY                                     TEXT
RPG-0097    W     NO FIELD DESCRIBED FOR THIS OR PREVIOUS RECORD OR DATA STRUCTURE.  IF DATA
                  STRUCTURE, LENGTH DEFAULTS TO ONE.
RPG-0159    W     MISSING RECORD IDENTIFYING INDICATOR IN COLUMNS 19-20.

            * * * * * E N D   O F   C O M P I L E * * * * *
```

Figure 29-1 (Part 3 of 4). Sample Program Listing

```
          OVERLAY LINKAGE EDITOR STORAGE USAGE MAP

 START      OVERLAY   CATEGORY  NAME AND      CODE LENGTH
ADDRESS  NUMBER AREA             ENTRY    HEXADECIMAL DECIMAL

 0000                      0    RPGEXA          010E       270
 010E                      0    #FLDS           020F       527
 031D                      0    #ROOT2          0145       325
 0462                      0    #BUFF           040C      1036
 086E                      0    @PGTS           00A8       168
 0916                      0    #IPCR           0070       112
 0986                      0    #OPCR           0046        70
 09CC                      0    #CNST0          0006         6
 09D2                      0    #CNST1          000B        11
 09DD                      0    @PGDM           0019        25
 09F6                      0    SUBR23          019B       411
 0B91                      6    @PGTI           06DF      1759
 1270                      6    @PGTO           02F0       752
 1560                      6    @PGTD           023B       571
 179B                     86    #OPEN           00C5       197
 1860                    126    #INPUT          008A       138
 18B2                             @0992
 18B9                             @0999
 18E0                             @09C0
 18AA                             @098A
 18AE                             @098E
 18EA                     21    #IHK01          0008         8
 18F2                    126    #IFLD           0022        34
 190B                             @0A5A
 1914                    126    #DETC           0147       327
 1A5B                     21    #IHK02          0008         8
 1A63                     50    #CHN00          0050        80
 1A90                             @0D63
 1AB3                     11    @PGRI           0043        67
 1AF6                     15    @PGAA           00AD       173
 1BA3                     86    #CLOSE          0030        48
 1BB3                             @0C90
 1BD3                    126    #DEOUT          0069       105
 1C3C                     21    #OHK01          000C        12
 1C48                    126    #LROF           001D        29
 1C65                    126    #RCDID          0051        81
 1C8A                             @09EF
 1CB6                    126    #CNFLD          0026        38

SYS-3130 I RPGEXA MODULE'S MAIN STORAGE SIZE IS
           7388 DECIMAL
SYS-3131 I 0000 IS THE START CONTROL ADDRESS OF THIS MODULE
SYS-3134 I RPGEXA MODULE IS CATALOGED AS A LOAD MEMBER
           JAKLIB   IS THE LIBRARY NAME
               29 TOTAL NUMBER OF LIBRARY SECTORS
```

Figure 29-1 (Part 4 of 4). Sample Program Listing

```
IAR-80179B ARR-1486 XR1-800000 XR2-800000 WR4-5538 WR5-0009 WR6-00B0 WR7-8001 PMSR-1F02 DIR-80

ADDR      00       04       08       0C       10       14       18       1C
          Primary Work Area
000000X   C210179B 087B7BD4 E2C7D9D7 F1880000 00000000 00000000 00000000 00000000   *B......MSGRP1...................*
000020    00000000 00000000 00000000 00000000 00000000 00000000 00008007 0004087B   *................................*
000040    7BD4E2C7 D9D7F288 00000000 00000000 00000000 00000000 00000000 00000000   *.MSGRP2.........................*
000060    00000000 00000000 00000000 00000000 00000000 8008003E 7B7BD4E2 C7D9D700   *..........................MSGRP.*
                                              Constants
000080    00000000 00000000 00000000 00000000 40FFFF00 00010002 031D031D 00000000   *................ ...............*
                            Time of Compilation                  Date of Compilation
0000A0    0000F0F0 F0F0F0F0 00000919 C08718A3 C0871860 0000C3D9 5CB21021 00000000   *..000000...........-..CR*.......*
                Indicators                                                         Compiler Release Level
0000C0    00600000 00000000 00000000 00000000 00000000 00000000 00000000 F1000000   *.-..........................1...*
          Secondary Work Area
0000E0    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   *................................*
          DTT                                NAMES (Data Structure)
000100    012A022F 012A001D C1D94040 40400000 00000000 00000000 00000000 00000000   *........AR    ..................*

000120    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   *................................*
                                                             LEVEL    RCODE
          Fields                   DUPLICATE LINES SAME AS ABOVE
          ERR101                              *ZERO    MIC
000300    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00C22003   *.............................B..*
          Input/Output Control Block for WORKSTN File
000320    43036900 00003A00 017D0468 010005F0 1C8A0000 00000000 80000000 00000000   *.........'.....0................*
               Input/Output Control Block for DISK File
000340    000000D0 04031D03 F8000000 3A000100 04680200 076E0000 00000000 00008000   *........8.......................*
                              DTF for WORKSTN File
000360    00000000 00000000 00C00000 00800000 8003F880 05F00000 00000000 017DC4C9   *..................8..0.......'DI*

000380    E2D7D3C1 E8400000 00008000 00800000 80000000 00000000 00008000 C9000000   *SPLAY ......................I...*

0003A0    00000000 00000000 00000000 0000D9D7 C7C5E7C1 C6D40175 00008003 C98005E6   *..............RPGEXAFM......I..W*
                              DTF Extention for WORKSTN File
0003C0    07010000 00000000 00000000 0000086E 00000000 000005E6 00000000 00000000   *.......................W........*
                                                                DTF for DISK File
0003E0    00000000 00000000 00000000 00000000 00000000 00FF0000 A0000000 80000080   *................................*

000400    FFFF8007 6E0000C0 0200C001 00C5D4D7 C6C9D3C5 4080076E 000180FF FF000000   *.............EMPFILE ...........*

000420    00000080 FFFF0000 00000000 0000FFFF 00000000 00000000 00000000 00000000   *................................*
                                                                     DTF Key Field
000440    00000000 00000000 00080007 0001FFFF 00000000 00000000 00000000 00000000   *................................*
              Output Work Area
000460    00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   *................................*

                                   DUPLICATE LINES SAME AS ABOVE
```

Figure 29-2 (Part 1 of 8). Sample Storage Dump

Figure 29-2 (Part 2 of 8). Sample Storage Dump

```
IAR-80179B ARR-1486 XR1-800000 XR2-800000 WR4-5538 WR5-0009 WR6-00B0 WR7-8001 FMSR-1F02 DIR-80
                                                 End of Entry
                                                    End of Table
ADDR     00       04       08       0C       10       14       18       1C
                      WSID Table                 Logical Record Area
0005E0X  00000000 00000000 30000000 00FFFF00 00000000 00000000 00000000 00000000    *................................*

000600   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000    *................................*

                        DUPLICATE LINES SAME AS ABOVE
                                     Subroutine That Scans WORKSTN ID Table (@PGTS)
000860   00000000 00000000 00000000 00003408 08F63401 08F23402 08EE3501 08F61C04    *................6...2.......6..*

000880   08FF04C2 01086E5C 00448F6C 006A575D 01918AF2 01046C01 91567502 91BDFF00    *...B...*...%.......2..%.........*

0008A0   F281395D 018E8CF2 01109D01 018AF281 25B80002 F2901FF2 87259D01 018EF201    *2.....2.....2.....2..2......2.*

0008C0   157D008F F2811878 408FD010 43B84002 F21003D0 8743E202 00D0872F 75028A75    *.'..2... ..... .2.....S.........*

0008E0   01887402 040E0108 F60901C2 020000C2 010000C0 87000000 00404000 00000000    *........6..B...B.........  .....*

000900   00050902 09040906 0908090A 090C090E 09100912 09143408 09853401 0938E201    *..............................S.*
         Input Processing Control Routine (#IPCR)
000920   00750205 9C000E1B 9C010C12 BD1000F2 01037B10 1BC0FF00 00BD410D F2823CC0    *..............2...........2...*

000940   8100ACBD 440DF282 12F28129 BD480DF2 8123C202 00B4F401 0412407A 8000BDA0    *......2..2.....2..B...4... .....*

000960   00F20117 B8800FF2 9011BD83 0EF2810B F2870200 030E0109 850974D2 0200C201    *.2.....2.....2..2..........K..B.*
                  Output Processing Control Routine (#OPCR)
000980   0000C087 00003408 09CB3402 09987502 059C010C 0EC0FF00 000C7C05 E405E50C    *..........................U.V.*

0009A0   FF056705 68BD400D F28116BD 410DC081 00ACBDE0 00F28109 C20200B4 F4010412    *...... .2...........2..B...4...*
                                    Constants
0009C0   4075020E C2010000 C0870000 C6D6D9D4 C1E3F0F1 F0F0F1F0 F1F0F1F1 F1340809    * ...B.......FORMAT01001010111...*
         Subroutine That Calls Printer and Disk (@PGDM)   SUBR23
0009E0   F5BDE000 F28107F4 010C00F2 8704F401 0C08C087 00003401 0B543402 0B583408    *5...2..4...2..4.................*

000A00   0B5C3501 0B5C7502 022C030B 6A00D201 03C2020B 80BC0007 8C01060B 6D4D0202    *.*...*........K..B..............*

000A20   0B74F201 11D20104 0E010B5C 0B764D01 000B7AF2 870D1C00 0B7C008F 01060B7C    *..2..K.....*.......2............*

000A40   7DE100F2 020C9C00 07008E00 070B6EF2 8703BCE1 079E0106 020E010B 5C0B78D2    *'..2..........2............*..K*

000A60   01031C01 0A6C020C 000B7100 003DF10B 71BC0001 F2810E3D F20B71BC 0101F281    *.....%........1.....2...2.....2.*

000A80   043AFF0B 7DD20103 1C010B5E 02750102 7CF000B5 01062C00 0B700736 010B701C    *....'K...........0..............*

000AA0   010B7F01 7C40012C 000AAD07 5C000001 4C01010B 7F38FF0B 7D3BFF0B 7DF2900B    *..... ......*...........'...'2..*

000AC0   35010B5E 7CF300C0 870B51C2 010B677D F900F284 307DF000 F2822AF2 800F9801    *.....3.....B...'9.2..'0.2..2....*

000AE0   02003C87 0ADCD201 01C0870A CF980302 003C800A DCD20101 7DFF00F2 8116E202    *......K.............K..'..2..S.*
```

IAR-80179B ARR-1486 XR1-800000 XR2-800000 WR4-5538 WR5-0009 WR6-00B0 WR7-8001 PMSR-1F02 DIR-80

ADDR	00	04	08	0C	10	14	18	1C	
000B00X	01C0870A	CF3C800A	DC35010B	5E7CF400	C0870B51	C2020B80	F4010409	8D010C0B	*..............4......3...4.......*
000B20	60F2812D	35010B5E	8D010C0B	62F20107	7CF100C0	870B518D	010C0B64	F281088D	*-2............2...1..........2...*
000B40	010C0B66	F201077C	F500C087	08517CF2	00C20100	00C20200	00C08700	000000FF	*....2...5.......2.B...B..........*
000B60	FF000017	46174400	000000FF	00000100	0000FFFF	FF000400	0C00E100	00000000	*.................................*
000B80	08040000	8000004B	80FFFF00	00000000	Subroutine That Processes WORKSTN Input (@PGTI) 00340800	0B34010C	0734020C	037B0400	*.................................*
000BA0	35020C03	350153B8	207DC010	11AABC40	0DC0870C	0CC08710	15C2C087	0C854D01	*.........'...... ....3......*
000BC0	18122AF2	011E4D01	16122AF2	01167880	1DF21010	78401DF2	100AC087	0CEA7DF1	*...2.......2......2... .2......'1*
000BE0	25F28104	C0870D6E	C0870DC9	C0870ED3	8D400DF2	810A8D42	0DF28104	C0871156	*.2..........I...L. .2.....2......*
000C00	C2020000	C2C10000	C0870000	0F061256	125678FF	1D2C000C	56578501	564D0101	*B...B............................*
000C20	122AF281	260E0012	51122C78	2003F210	060E0012	52102CF2	5C0E78E0	02F29008	*..2............2.........2.....2..*
000C40	34011254	3CFF0C38	F287060E	00125012	2C781002	D20100C0	900C1D85	01533D00	*.........2............K..........*
000C60	1251F201	067A401D	F2870C0D	00125212	51F20103	7A801D4C	01181254	4C022012	*...2.... .2..........2...........*
000C80	52C0870B	3510010C	A2244D01	241224F2	814D1C01	0CA0060C	010CA512	2AC08700	*...................2..............*
000CA0	00C9C400	0C000D01	0CA5122A	F201075F	012424F2	F7223501	0CA55F01	04044D01	*.ID.........2......2..............*
000CC0	06122AF2	81C5C087	1015E285	0255AD01	0A1226F2	F1025C01	0CDE0470	87000035	*...2......S.........2.............*
000CE0	020C03B5	0153C087	028E1C01	0CF8040C	010CFB12	1AC08700	00404020	00003501	*..............8.......... ......*
000D00	0CFD9C01	50017880	02F21040	8C0915F4	010C047A	60028D00	35F25130	5F010101	*...........2.....4..........2.....*
000D20	788F0276	20C2FF01	04D4B501	[illegible]	122C4F00	[illegible]	[illegible]	812C5D00	*..............................2...*
000D40	1F20C001	0CFE7CF0	35F2871E	35020C03	9C015001	[illegible]	24BC0120	12247840	*.....5.0.2.......................*
000D60	02550155	F2100370	F1250087	05DE4C01	230C8C01	[illegible]	[illegible]	1220010D	*....2........................2....*
000D80	9E560E01	0D5B123E	38400000	F290254D	0116122A	[illegible]	[illegible]	[illegible]	*.........2.........2......2......*
000DA0	AC160E01	0D4D1240	8C015000	0C5F0116	16F28700	[illegible]	[illegible]	F09C0150	*....................1............*
000DC0	24F40101	04C03708	E870F035	[illegible]	022A2A61	[illegible]	[illegible]	40800135	*.1........1.0........3.........2.....*
000DE0	1233F201	3810010D	F0D62001	0CF4500C	0100F011	[illegible]	[illegible]	[illegible]	*..3......8.....................D....*

Figure 29-2 (Part 3 of 8). Sample Storage Dump

Figure 29-2 (Part 4 of 8). Sample Storage Dump

IAR-80179B ARR-1486 XR1-800000 XR2-800000 WR4-5538 WR5-0009 WR6-00B0 WR7-8001 PMSR-1F02 DIR-80

ADDR	00	04	08	0C	10	14	18	1C	
	@PGTI (Continued)								
000E00X	125C0DFD	0D01125C	122AC081	0F643501	0DFD6C01	0435B501	53C0870F	9E2C000E	*.*.....*..........%.............*
000E20	2635C087	1560003C	870E3ABD	1135C081	0F2D3DFF	0E26F201	93F28749	7C40115C	*....-.................2..2... .**
000E40	0610114C	04091245	B501532C	010E6850	8D013512	3BF2812D	1C010E66	060C010E	*.....................2..........*
000E60	6B122AC0	870000C9	C4000000	0D010E6B	122AC081	0F333501	0E6BC087	0FB13502	*,......ID....,..............,......*
000E80	0C03B501	537DFF2C	F281171C	010E962C	3501124F	7A000035	020C07BA	0400C087	*.....'..2.......................*
000EA0	0C004D01	2E122AF2	81221C01	0EB62E35	01124FC0	87000035	020C03B5	01537810	*.......2........................*
000EC0	1DF29008	C08711C7	C0870D6E	B50153C0	870BECBD	1135F281	543DFF0E	26F20180	*.2.....G..............2......2..*
000EE0	BD0235F2	011F8D01	20122AF2	01725F01	24248D01	10122AF2	8105C087	1015C3BC	*...2.......2...........2......C.*
000F00	420DF287	5BBD3435	F2812EBD	2435C081	1187BD80	35F2812D	BD8135F2	812DBD82	*..2.....2............2.....2....*
000F20	35F2812D	BD8335F2	812DF287	30BC420D	F2872DBC	860DF287	27BC850D	F28721BC	*.2.....2..2.....2.....2.....2...*
000F40	800DF287	1BBC870D	F28715BC	920DF287	0FBC930D	F28709BC	940DF287	03BC950D	*..2.....2.....2.....2.....2.....*
000F60	C0870BF0	1C010F72	060C010F	77122AC0	87000000	00400000	0C01125C	0F770D01	*...0................ .....*....*
000F80	0F77122A	C0810F33	35010F77	6C010150	7A80026C	010435B5	0153C087	0E193501	*............%......%............*
000FA0	125CC087	0FB13502	0C03B501	53C0870E	1D340810	144D0106	122A3502	0C03F281	*.*..........................2.*
000FC0	10792002	F21005C0	871015C9	C0871015	D9B50253	8D010C12	2A2C010F	F70C2C01	*....2......I....R...........7...*
000FE0	0FF30835	020C03F2	810E7920	02F21004	C0870000	C0870000	B502539C	0124018D	*.3.....2.....2..................*
001000	011C122A	F28107B5	021C9C01	00017B20	02C08700	00340811	55340111	4DB50253	*....2...........................*
001020	2C011258	222C0112	5A223501	11551C00	125F000E	01115512	2C350111	4D750106	*................................*
001040	B502103D	C2125FF2	01113502	0C032C01	12584E2C	01125A4E	B5010C0F	01125A12	*....B..2..........+.....+........*
001060	30F2046A	0C011258	125A3601	12333602	12333DC2	125FF281	073DC912	5FF2010A	*.2.................B..2...I..2..*
001080	7C40015C	FD0001F2	873A3DC3	125FF201	0ABC4001	ACFD0001	F28729D2	0101E202	*. .*...2...C..2... .....2..K..S.*
0010A0	013DD912	5FF20107	9CFE0000	F2870B3D	E2125FF2	01046CFE	0000D201	01E20201	*..R..2......2...S..2..%...K..S..*
0010C0	C087105B	E20202D2	0102C087	105B0F01	1258122E	36011258	36021258	0C001101	*....S..K........................*
0010E0	12580C00	11191258	3DC2125F	F281073D	C9125FF2	01117C40	013DFF12	58F2814A	*..........B..2...I..2... .....2..*

IAR-B0179B ARR-1486 XR1-800000 XR2-800000 WR4-5538 WR5-0009 WR6-00B0 WR7-8001 FMSR-1F02 DIR-80

ADDR	00	04	08	0C	10	14	18	1C	
	@PGTI (Continued)								
001100X	5C000001	F287433D	C3125FF2	0111BC40	013DFF12	58F28132	AC000001	F2872B0E	**...2...C..2... .....2......2...*
001120	01125812	2C0C0011	3912580C	00114712	583DD912	5FF20107	9C000101	F2870B3D	*.................R..2......2...*
001140	E2125FF2	01046C00	0101C201	00003502	0C03C087	00003501	124F1C01	1249AF35	*S..2..%...B....................*
001160	010C07C2	02124EF4	01041240	35020C03	3501125C	7DF000F2	8206BCE8	1EF28703	*...B..+4... ....*'O.2...Y.2..*
001180	BC0A1EF4	010C0435	01125CBD	E81EF281	0A7B8F02	5F010101	7A20025F	010404B5	*...4......*.Y.2................*
0011A0	01536C00	2735C087	08A03408	11E97880	1DF21007	BC890DC0	870BF0BB	207DC087	*..%..........Z...2........O..'..*
0011C0	11EA9059	F2870A34	0811E9C0	8711EA90	60BC061D	BC051EBC	48208C01	23124FF4	*....2.....Z.....-..............4*
0011E0	010C04BC	001DC087	00003401	120F3402	12133408	1217C202	12183501	12170E01	*.....................B.........*
001200	1217122E	9C010301	F4010409	C2010000	C2020000	C0870000	10040000	8000004B	*........4...B...B..............*
001220	80FFFF00	00000000	00000000	01000200	FF0300FD	F0F0F0F0	18011003	08000200	*...................0000........*
001240	01D9C5C1	C440C087	0000C087	116C0000	00000000	00000000	00000000	00000000	*.READ ....%................*
					Subroutine That Processes WORKSTN Output (@PGTO)				
001260	12601262	12641266	1268126A	126C126E	34081498	34011494	34021490	BC400DB5	*.-..........%.............. ..*
001280	010CB601	4E1C0715	2F077C40	075C0606	07B50153	1C0112E5	064D011C	151CF281	*....+...... .*.........V......2.*
0012A0	1275011C	340112AC	8C015000	00B50153	F287119C	0150248D	0150151C	F201058C	*................2...........2...*
0012C0	0150151E	2C0112E7	500C0112	EA151C8D	0150151E	F201073C	C012E8F2	87043C00	*.......X..............2.....Y2....*
0012E0	12E8C087	0000C9C4	0000000D	0112EA15	1CF20108	3CDD13CE	C08713C2	350112EA	*.Y....ID..........2.........B....*
001300	782002F2	1003F287	2B788002	F210083C	EE13CEC0	8713C24D	0106151C	C0011499	*...2..2.....2..........B.........*
001320	8502538D	0108151C	2C011330	08C00100	007B2002	35021490	8C074C15	2FBD404C	*............................. .*
001340	F20107A0	06484CF2	8159BC07	1EAC0123	0CAC0120	4EB50153	BC01304D	0112151C	*2......2.............+...........*
001360	F281291C	01136D12	0C011541	00000D01	15411522	F2811588	03301541	F28703BF	*2..................2.......2...*
001380	F6300700	15401520	C002137F	F4010C04	350112EA	6C010435	2C0013CE	358D2435	*6.... ......4.......%...........*
0013A0	F2810DBD	010EF201	198C0A1E	F4010C04	350112E4	5F010101	7B8F027A	20025F01	*2.....2.....4....................*
0013C0	0404B501	534C0027	13CEC087	1560003C	8713DB3D	FF13CEF2	01B3F287	0D4C0711	*.............-.........2..2......*
0013E0	152F4C04	091527B5	01534D01	2E151CF2	81201C01	140A2E35	0114943C	8714A2C0	*...-..........2...............*

Figure 29-2 (Part 5 of 8). Sample Storage Dump

```
IAR-80179B ARR-1486 XR1-800000 XR2-800000 WR4-5538 WR5-0009 WR6-00B0 WR7-8001 PMSR-1F02 DIR-80

ADDR     00       04       08       0C       10       14       18       1C
         @PGTO (Continued)
001400X  87149935 011539C0 87000035 021490B5 01537DEE 27F2812A BD8035F2 812ABD81   *..................'..2.....2....*
001420   35F2812A BD8335F2 812A0D01 12EA151C F28109BD 2435F281 27F2871E BC830DF2   *.2.....2........2.....2..2.....2*
001440   874BBC84 0DF28745 BC870DF2 8715BC92 0DF2870F BC940DF2 8709BC95 0DF28703   *.....2.....2.....2.....2.....2..*
001460   BC960D35 0115391C 011533AF 35011494 C2021538 F4010412 40350214 90BC400D   *...............B...4... ...... .*
001480   BDF04FF2 8207BCE8 1EF4010C 04C20200 00C20100 00C08700 00340815 12340115   *.0.2...Y.4...B...B..............*
0014A0   0AF28013 B502532C 01153B22 2C01153D 22750106 F287113C 8014A21C 01153B0C   *.2..................2...........*
0014C0   1C01153D 0C75010E 0F01153D 1514F204 180C0115 3B153D36 0115167C 40015CFD   *.............2.............. .*.*
0014E0   0001D201 02C08714 C80F0115 3B151A36 01153B7C 40013DFF 153BF281 0A0C0015   *..K.....H........... .....2.....*
001500   04153B5C 000001C2 01000035 021490C0 87000000 FF00FD00 01000200 00404008   *...*...B....................   .*
001520   F1F0F0E6 D9C9E3C5 40404040 40404040 C0870000 C0871479 00000000 00000000   *100WRITE        ................*
001540   00001542 15441546 1548154A 154C154E 15501552 15541556 1558155A 155C155E   *...............+.............*..*
         Subroutine That Updates WORKSTN Return Codes (@PGTD)
001560   3408168C 34021688 3501168C 1C001735 007C0000 B5015307 04173A17 3A3D0217   *................................*
001580   35F28410 BDF04FF2 8211BD10 34F2019D 3CAA1735 C20216F2 F28742BD 081EF281   *.2...0.2.....2......B..22.....2.*
0015A0   06BD011E F20186BD 002AF281 80BDF12A F2817A3C F2173A2C 0017352A 38301735   *....2.....2...1.2...2...........*
0015C0   3D3C1735 F2146238 30173539 4017353D BC1735F2 1453C202 17267A10 1DE20202   *....2....... ......2..B......S..*
0015E0   8D000017 35C08215 DDF28104 C2021716 BDFF01F2 81152C00 160301C2 02173F2C   *..........2..B......2.......B....*
001600   03173A00 35021688 F2872235 0216882C 00173534 BD0335F2 0107C202 1716F287   *........2.............2..B...2.*
001620   04C20217 21C08715 DDC08716 980D0317 3A173EF2 040A3501 168C7CFF 00B50153   *.B..................2...........*
001640   4D011416 E1F28137 7501144C 0404173A 2803166B 34280216 6F342803 16733528   *.....2..................,.......*
001660   02167735 C20216E4 6C001900 6C001800 6C001700 6C001600 3502168C BC80020E   *....B..U%...%...%...%...........*
001680   01168C16 E3C20200 00C08700 00010204 08102040 80000174 2C001695 2AC20216   *....TB.............. .........B..*
0016A0   8DD20128 BD3C08F2 0404AF00 080ABC00 47BD3908 F2820CBC 0147BD41 08F28203   *.K.....2............2.........2..*
0016C0   BC0247AF 000809BB F808AC00 4408AC00 46007A00 00350216 88B50153 C087162D   *........8.......................*
0016E0   00000001 F0F1F2F3 F4F5F6F7 F8F9C1C2 C3C4C5C6 03FF04FF 08231107 1817241B   *....0123456789ABCDEF............*
```

Figure 29-2 (Part 6 of 8). Sample Storage Dump

```
                IAR-80179B ARR-1486 XR1-800000 XR2-800000 WR4-5538 WR5-0009 WR6-00B0 WR7-8001 PMSR-1F02 DIR-80

ADDR      00       04       08       0C       10       14       18       1C
          @PGTD (Continued)
001700X   2803340B 381F800F 810F821F 830F9913 AA2FDD1F EE13FF27 002B0127 0327081B     *................................*
001720    1033FF11 271227FF BD43F347 F43FF53B F637F84B FF00F040 404040F0 F0F9F9F0     *..........3.4.5.6.8...0    00990*
001740    F0F0F0F0 F0F1F1F1 F2F0F1F1 F2F5F1F1 F2F6F1F1 F2F7F1F1 F2F7F5F1 F2F8F1F1     *00000111201125112611271127512811*
001760    F2F8F5F1 F2F9F9F1 F3F1F1F1 F3F2F1F1 F3F3F1F1 F1F2F1F1 F1F2F2F1 F1F2F3F1     *28512991311132113311121112211231*
001780    F1F2F4F1 F1F2F5F1 F1F2F617 8B178D17 8F179117 93179517 971799C2 020004F4     *12411251126................B...4*
          Open Mainline Routine (#OPEN)
0017A0    010401AD 06077AE2 0236F281 04F40104 0FE20204 F4010401 AD060740 E20236F2     *.......S..2..4...S..4...... S..2*
0017C0    8104F401 040FC201 0000C202 17D95F18 DADAF401 040FF287 04000400 C57A01C2     *..4...B...B..R....4...2.....E..B*
0017E0    7A40C375 01997502 05F40104 0C750205 F4010402 1C0017FA 10380000 C5F2900D     *. C......4......4...........E2..*
001800    7BC00079 1000F290 043C8718 1D4D0103 0099F281 07750103 C08717F4 F28009C2     *......2...........2........42..B*
001820    0200B4F4 01041211 C2010000 3C40030F 0C00030E 030F0CFF 030D030E 0CFF020D     *...4....B.... ..................*
001840    020E3CF0 031C0C0B 031B031C 3C4005E5 0C7C05E4 05E50CFF 05670568 C0871BD3     *...0......... .V...U.V.........L*
          Input Mainline Routine (#INPUT)
001860    7B06C97B F1C27B9F C37BFFC4 7A40C378 20C3C010 18A00C01 188A18E7 750299B8     *..I.1B..C..D. C..C.........X....*
001880    1000F210 150C0118 D500001E 01188A97 B9C100B9 8001C010 18CFB880 00F29006     *..2.....N........A...........2..*
0018A0    7AFFC47A E0C3C087 1CB6C087 1C48C087 1C483C31 18CEF287 043C3018 CEB80100     *..D..C................2.........*
0018C0    BB8025E2 0100C202 00B4F401 04120074 08E4C087 0916B880 00F21004 C0871C65     *...S..B...4......U.......2......*
                                Input Hook (#IHK01) Input Fields Routine (#IFLD)
0018E0    BA010075 10E418E9 18EAC201 0B91C210 09167502 9BB90218 F2100C2C 0119061E     *.....U.Z..B...B.........2.......*
                                                     Detail Calculations Routine (#DETC)
001900    B50212C0 870000C0 8719142C 07025408 C0871907 7802C9F2 900C0403 031709D5     *......................I2.......N*
001920    04000318 09D67802 C9F29010 C08709F6 0303174A 02C40003 18000319 7802C9F2     *.....O..I2.....6.....D........I2*
001940    90060403 031709DA 7802C9F2 9010C087 09F60303 174A030F 00031800 03197802     *..........I2.....6..............*
001960    C9F29004 C0871A57 7804C9F2 90587840 C6F29004 C0871A57 0402031C 0313C202     *I2........I2... F2............B.*
001980    012A8C1C 00024CE2 021CE202 01340200 010D0100 010103C0 041982C2 020343BC     *.......S..S................B....*
0019A0    801BBE80 00850205 0C070461 02547B08 D5C20203 42C0871A 5BF28706 7A08D5F2     *.........../....NB.......2....N2*
0019C0    8704C087 1A907804 C97808D5 F29004C0 871A5706 20031C09 D644D210 031C47E1     *........I..N2...........O.K.....*
0019E0    1009DCC0 871AB381 20CB0078 20CBF290 04C0871A 57C0871A F6010103 1C021A03     *..............2.........6.......*
```

Figure 29-2 (Part 7 of 8). Sample Storage Dump

```
IAR-80179B ARR-1486 XR1-800000 XR2-800000 WR4-5538 WR5-0009 WR6-00B0 WR7-8001 FMSR-1F02 DIR-80

ADDR      00       04       08       0C       10       14       18       1C
          #DETC (Continued)
001A00X   0C1C012A 0279C202 0343BC83 1B7B04D4 B88000F2 100EC202 0343C087 1A5BB880   *......B........M...2..B..........*
001A20    00F29006 7A04D4F2 870BC087 1A90F287 04C0871A 167804D4 F29004C0 871A570D   *.2....M2......2........M2.......*
                                                                  Input Hook (#IHK02)
001A40    07025402 5CC0871A B38110CB 007810CB F29004C0 8719D3C0 871BD3C2 0109DDC2   *....*............2.....L...LB...B*
              Chain Code Routine (#CHN00)
001A60    10091634 081A6AC2 0200002C 05036105 2C011A77 017A0000 35020355 35100361   *.......B....../................/*
001A80    D202B4C2 010343F4 01041230 C0870000 34081A8F B50212C0 871A6300 0080001A   *K..B...4.........................*
                                                 Subroutine That Sets Resulting Indicators (@PGRI)
001AA0    A5C0871A 802C0702 5C072C1C 027930C0 871A8C74 08FF7402 FD7502FF 7404FBF2   *........*.......................2*
001AC0    1000B9FF 00F21022 7504FB2C 001AD100 F200043C 7B1ADC2C 011ADE02 7A00003C   *.....2.......J.2................*
                                                        Subroutine That Resolves Array Index (@PGAA)
001AE0    7A1ADCE2 0203C087 1ABFE202 017402FF 7502FD75 10FF7402 FFC2021A F9B608A9   *...S......S..............B..9...*
001B00    B4080E4C 07FC0000 9C0132F8 9C019AF8 98032FF9 9C0059F9 980198F9 9C019FFB   *...........8...8...9...9...9....*
001B20    7502F66C 01F60644 00F40000 F204516C 01FD005F 01FDF6F2 87045E01 FDF65700   *..6%.6...4..2..%......62.....6..*
001B40    F495C022 1B3AF2A0 371F001B 5295F282 285400F4 F3F28121 5C01F8F6 5E01F6F6   *4.....2.......2....432..*.86..66*
001B60    F2A01D5E 01F6F6F2 A0165E01 F6F8F2A0 0F5E01F6 F6C0871B 3E9D0102 FDF20215   *2....662....682....66........2..*
001B80    6C01FD00 6C05F80C D202B4F4 0104121D 14000000 951C0100 00FD7502 FF35101B   *%...%.8.K..4.....................*
              Close Mainline Routine (#CLOSE)
001BA0    070007C2 0100007A 20C37808 D97A08D9 F2100FC2 021BBEF4 01040FF2 87048004   *...B.....C..R..R2..B...4...2....*
                                                 Detail Output Routine (#DEOUT)
001BC0    00C53501 00997502 05F40104 03F40104 040000C2 0204683C 00037779 40C67802   *.E.......4...4.....B........ F..*
001BE0    C9F2100D 3C000377 7940C678 04C9F290 47C20201 00360203 2B8C057A 09D1C202   *I2....... F..I2..B...........JB.*
001C00    00003602 032B8C07 07025478 20CBF290 058C4A52 02C47808 D5F29005 8C4A5203   *..............2......D..N2......*
                                                                       Output Hook (#OHK01)
001C20    0FC20201 00360203 2B0CFF05 BA020D8C 2174022F C0871C3C C0871860 C2021270   *.B.........................-B...*
                            LR and Overflow Processing Routine (#LROF)
001C40    C201031D C2100986 7820C3C0 101BB37B FFD77BFF D87A02C2 7804C2F2 10037B02   *B...B.....C......P..Q..B..B2....*
                     Record Identification Routine (#RCDID)
001C60    C2C08718 F2B50112 B5101434 081C72C2 0100009C 0A220AF2 8708C087 18B2C087   *B...2..........B.......2........*
001C80    18B9C201 0000C087 18E00C02 00C803F3 7DC100F2 010BC087 1C6B0204 C9800019   *..B..........H.3'A.2........I...*
                                                        Control Fields Processing Routine (#CNFLD)
001CA0    0B7D4000 F2010BC0 871C6B00 02C98000 0000C087 1C7E7820 C3F21015 3502009B   *.' .2.......I..........C2......*
001CC0    BB01002C 011CCA1A 7A000078 80D9F290 04C08718 AA7A80D9 C08718AE FFC1D5C3   *.............R2........R.....ANC*
001CE0    C540D4C1 C9D5E3C5 D5C1D5C3 C540D4C1 C9D5E3C5 D5C1D5C3 C540D4C1 C9D5E3C5   *E MAINTENANCE MAINTENANCE MAINTE*
001D00X   00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   *................................*

                    DUPLICATE LINES SAME AS ABOVE
```

Figure 29-2 (Part 8 of 8). Sample Storage Dump

Chapter 30. Differences between RPG on System/36 and RPG on System/34

Chapter 30. Differences between RPG on System/36 and RPG on System/34

If you convert RPG programs from IBM System/34 to System/36, you should be aware of the following differences between the way RPG programs work on the two systems.

Updating Past the End of the File

System/36

If you try to update a shared or unshared sequential DISK file past the end of the file, the update is ignored and error message RPG-9043, TRIED RECORD UPDATE BEFORE INPUT FOR FILE, is displayed.

System/34

If you try to update an *unshared* sequential DISK file past the end of the file, the update is ignored but no error message is displayed.

If you try to update a *shared* sequential DISK file past the end of the file, the update is ignored and error message RPG-9043, TRIED RECORD UPDATE BEFORE INPUT FOR FILE, is displayed.

Creating a Direct File That Does Not Allow Deletions

System/36

If you chain past the area allocated for the file, the indicator in columns 54 and 55 of the CHAIN operation turns on and error message RPG-9035, NO RECORD FOUND ON GET OPERATION FOR FILE, is displayed. Then, if you try to continue to create the file by writing additional records to it, the additional records overlay the last record written to the file. That is, the additional records are written on top of each other in the same space as the preceding record.

System/34

If you chain past the area allocated for the file, the indicator in columns 54 and 55 of the CHAIN operation turns on and error message RPG-9035, NO RECORD FOUND ON GET OPERATION FOR FILE, is displayed. Then, if you try to continue to create the file by writing an additional record to it, no additional record is written and error message RPG-9038, FILE IS FULL, is displayed.

Chapter 31. Using Ideographic Data

Chapter 31. Using Ideographic Data

RPG can process ideographic data when you use the ideographic version of the SSP and the ideographic hardware devices that the ideographic version supports. Display stations with ideographic capability are supported by the WORKSTN file only.

Ideographic support allows the RPG compiler to process IBM-supplied or your user-defined ideographic character sets. Very little error checking is performed on ideographic data. Ideographic data is transparent to the RPG II compiler. You must be sure that the ideographic data is processed properly by your program.

Ideographic characters can be present in literals, constants, fields, tables, and arrays. The transparent literal option must be specified in column 57 of the control specification if transparent literals or constants are present in the program. (For more information on the transparent literal option, see *Column 57 (Transparent Literal)* in Chapter 20.) A field, table, or array containing ideographic data is considered to contain alphameric data by the RPG compiler. No error checking occurs for ideographic data in fields, tables, or arrays.

Ideographic data has a 2-byte representation, rather than a 1-byte representation as in the EBCDIC character set. This can cause the RPG operation codes that process data 1 byte at a time (COMP, MOVE, and so on) to produce incorrect results. In addition, ideographic data is enclosed by the shift-out (S/O) control character (hex 0E) and the shift-in (S/I) control character (hex 0F). These control characters must be taken into consideration when an operation that processes ideographic data is performed. (For more information on considerations that apply to processing ideographic data, see *Processing Considerations* in this chapter.)

Specifying Ideographic Data

Ideographic Literals and Constants

Ideographic characters can be specified as a literal in factor 1 or factor 2 or the calculation specifications. Ideographic characters can be specified in the constant or edit word section of the output specifications (columns 45 through 70). Ideographic literals and constants must begin with an apostrophe immediately followed by the S/O control character. Ideographic literals and constants must end with the S/I control character immediately followed by the ending apostrophe.

Note: When ideographic literals or constants are processed by RPG, the S/O and S/I control characters are considered to be part of the literal or constant data. When the constant is displayed or printed on an ideographic device, these control characters appear as blanks.

When an ideographic literal or constant is used, the transparent literal option must be specified in column 57 of the control specification. When this option is specified, the compiler checks for literals or constants that begin with an apostrophe followed by the S/O control character. If a literal or constant is found that begins with an apostrophe followed by the S/O control character, the compiler checks to see if the literal or constant is valid. A literal or constant is not valid if:

- A second S/O control character is found before the S/I control characters.
- An odd number of 1-byte characters are found between the S/O and S/I control characters.
- The S/I control character is not immediately followed by the ending apostrophe.

An invalid transparent literal or constant is rechecked to see if it is a valid alphameric literal or constant. If the literal or constant is a valid transparent literal or constant, it is not checked for embedded apostrophes.

Any ideographic character can be entered in an ideographic literal or constant. Each ideographic character has a 2-byte hex representation. (An ideographic blank also occupies 2 bytes.) Because each character occupies 2 bytes in storage, ideographic constants can only be from 1 to 11 characters long (this also allows for the control characters), and literals can only be up to three characters long (this also allows for the control characters).

Note: An ideographic literal or constant can be composed only of ideographic data. Mixing ideographic and EBCDIC data in the same constant causes the literal or constant to be checked as alphameric.

Ideographic Fields, Tables, and Arrays

Ideographic characters can be present in fields, tables, and arrays. The RPG compiler does not recognize these characters as ideographic. The compiler treats ideographic characters as alphameric. Ideographic fields, tables, and arrays must therefore conform to the rules for alphameric fields, tables, and arrays.

When ideographic data is present in a field, table, or array, the data must be enclosed in the S/O and S/I control characters. These control characters are considered to be part of the field, table element, or array element. Therefore, when the length of the field, table element, or array element is defined, space must be left for the control characters. For example, if you want to define a field so that it can contain four ideographic characters, you must specify a field length of 10 (tow positions for each ideographic character, and one position for each control character). If you do not specify a large enough length, the field, table element, or array element is truncated, causing one of the control characters to be lost.

You must also consider the control characters when the field, table element, or array element is processed. For example, if a field is being printed or displayed on an ideographic device, the control characters appear as blanks. If blank after (column 39 of the output specifications contains a B) is specified for a field, the control characters are also blanked out and must be reconstructed if the field is to still contain ideographic data.

Note: When a field, table, or array contains ideographic data, it should contain only ideographic data. Mixing ideographic and EBCDIC data in the same field, table, or array can cause incorrect results.

Ideographic Comments

Ideographic characters can be entered as comments in source statements. If you enter ideographic characters as comments, but do not enclose them by control characters they are not displayed as ideographic characters. The source statements that allow comments are the extension specifications (columns 58 through 74) and the calculation specifications (columns 60 through 74). Ideographic characters can also be specified on a comment line (column 7 contains an asterisk).

Processing Considerations

Ideographic data can produce incorrect results when used with certain RPG operation codes. Since ideographic data has a 2-byte hex representation, operations that compare data byte by byte are not meaningful unless they check for an equal condition. Care must also be taken when ideographic data is moved. If the lengths of the data being moved and the area that the data is being moved to are not correctly specified, the S/O or S/I control characters can be lost.

A number of RPG operations and functions operate by comparing data 1 byte at a time. The COMP and LOKUP operations compare for high, low, and equal conditions. These operations compare the 1-byte EBCDIC values that correspond to the data that is present and produce a result based on the standard 1-byte collating sequence. Because of this, the only valid test when ideographic data is being processed is for an equal condition. If all the bytes in a field are equal to all the bytes in another field, the fields are equal whether they contain ideographic or EBCDIC data.

Match fields and sequence checking are also invalid for ideographic data. Match fields cause data from different records to be compared, 1 byte at a time. This produces incorrect results for ideographic data. Sequence checking compares data in different fields to see if the fields are in ascending or descending order. This comparison is done 1 byte at a time and therefore produces incorrect results for ideographic data.

The SETLL operation is another 1-byte comparison operation that cannot be used with ideographic data. This operation causes the key of each recored to be compared with a lower limit value. If the key of the record is higher than the lower limit, the record is selected for processing. As this comparison is carried out using 1-byte EBCDIC values, the SETLL operation can produce incorrect results when used with ideographic data.

RPG allows you to define an alternate collating sequence for EBCDIC data. In other words, you can redefine the order in which 1-byte segments of data will be sorted. This is meaningless for ideographic data.

Care must be taken when the various move operations (MOVE, MOVEA, MOVEL) are used with ideographic data. The length of the field, table element, or array element that the ideographic data is being moved to must be defined as being exactly the same length as the literal, field, table element, or array element being moved. If the lengths are not the same, the data will not be recognized as ideographic. For example, if the field that the data is being moved to is shorter than the length of the ideographic data, the data is truncated, causing one of the control characters to be lost. If the field that the data is being moved to is longer than the ideographic data, one of the control characters will be embedded in the field. This causes the control character to be considered part of the data.

Moving Ideographic Data and Deleting Control Characters (SUBR40)

SUBR40 is a move and edit routine that moves the contents of one field to another field. If the S/O and S/I control characters are found as the first and last characters in the field, SUBR40 deletes them.

SUBRO40 is called as shown in Figure 31-1.

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 of | Program Identification | Programmer | Date

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators (Not, And, Not, And, Not)	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators (Arithmetic Plus Minus Zero; Compare 1 > 2 1 < 2 1 = 2; Lookup(Factor 2)is High Low Equal)	Comments
01	C				EXIT	SUBR4Ø						
02	C				RLABL		EMPNO	1Ø				SENDING FIELD
03	C				RLABL		SOCSEC	8				RECEIVING FIELD
04	C				RLABL		RETCDE	1				RETURN CODES
05	C				RLABL		SNDLEN	3	Ø			SENDING LENGTH
06	C				RLABL		RECLEN	3	Ø			RECEIVING LEN
07	C											

Figure 31-1. Calling SUBR40

If you want the receiving field to contain all the data that was present in the sending field, you must specify a length for the receiving field that is two positions less than the length of the sending field. This allows two positions for each ideographic character (or one for each EBCDIC character) while deleting the S/O and S/I control characters (and the two positions they occupied). If you specify a receiving field longer than the sending field minus two positions, all the data from the sending field is moved and the receiving field is padded on the right with blanks (1-byte EBCDIC blanks). If the receiving field is shorter than the sending field minus two positions, the data being moved is truncated on the right.

Five RLABL fields must be specified when SUBR40 is called. The first two specify the sending and receiving fields for the move. The third field is where the return codes are written to indicate the status of the move operation . The fourth and fifth fields must be loaded with the lengths of the sending and receiving fields. These are the lengths of the fields specified on the first two RLABLs for the call to SUBR40 (in Figure 31-1, you would need to load the lengths of EMPNO an SOCSEC). The return code field must be defined as a one-position alphameric field; the length fields must be defined as three-position numeric fields with zero decimal positions.

SUBR40 produces return codes to indicate the status of the move operation. The following list contains these return codes and their meanings:

Return Code	Explanation
0	Data moved; no errors.
1	Data moved; padding occurred.
2	Data moved; truncation occurred.
3	Data moved; S/O and S/I control characters were not found.
4	Data not moved. Either an odd field length was found, a length of zero was found, the length was greater than 256, or an invalid character was found in the field length.

If more than one return code can be issued, only the highest return code is returned.

Moving Ideographic Data and Adding Control Characters (SUBR41)

SUBR41 is a move and edit routine that moves the contents of one field into another field. If the S/O and S/I control characters are not found in the first and last positions of the field, SUBR41 adds them to the field when it is moved.

SUBR41 is called as shown in Figure 31-2.

IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

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

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators	Comments
01	C				EXIT	SUBR41						
02	C				RLABL		SOCSEC	8				
03	C				RLABL		EMPNO	1Ø				
04	C				RLABL		RETCDE	1				
05	C				RLABL		SNDLEN	3	Ø			
06	C				RLABL		RECLEN	3	Ø			
07	C											

Figure 31-2. Calling SUBR41

If you want the receiving field to contain all the data that is in the sending field, you must specify the length of the receiving field to be two positions longer than the length of the sending field (to hold the S/O and S/I control characters). If you specify a receiving field that is longer than the sending field plus two, the data is padded on the right when it is moved into the receiving field. If the receiving field is shorter than the sending field plus two, the data is truncated on the right when it is moved. If the receiving field is specified either longer or shorter than the sending field plus two positions, the S/I control character is still placed in the correct position (the rightmost position).

Five RLABL fields must be specified when SUBR41 is called. The first two specify the sending and receiving fields for the move. The third field is where the return codes are written to indicate the status of the move operation. The fourth and fifth fields must be loaded with the lengths of the sending and receiving fields. These are the lengths of the fields specified on the first two RLABLs for the call to SUBR41 (in Figure 31-2, you would need to load the lengths of SOCSEC and EMPNO). The return code field must be defined as a one-position alphameric field; the length fields must be defined as three-position numeric fields with zero decimal positions.

SUBR41 produces return codes to indicate the status of the move. The following list contains these return codes and their meanings:

Return Code	Explanation
0	Data moved; no errors.
1	Data moved; padding occurred to left of S/I control character.
2	data moved; data truncated to left of S/I control character.
3	Data moved; S/O and S/I already present.
4	Data not moved. Either odd field length found, length of zero found, length greater than 256, or invalid character found in field length.

If more than one return code can be issued, only the highest return code is issued.

Ideographic Device Support

The *SIZE, *MODE, *INP, AND *OUT keywords identify subfields in the INFDS data structure that contain values for ideographic devices. For more information on these keywords, see *Coding the INFDS Data Structure* in Chapter 6.

Messages

The RPG displayed messages (both compile time and execution time) are displayed in either the standard character set or the ideographic character set. The messages are displayed in an ideographic character set if ideographic support was requested when the user signed on.

The RPG compiler messages are printed in either the standard character set or an ideographic character set. The messages are printed in an ideographic character set if ideographic support was requested when the user signed on.

Chapter 32. Problem Determination

Chapter 32. Problem Determination

If a problem occurs while you are using RPG, the cause of the problem may not be obvious. An error in your application or in system operation could have caused the problem. The problem determination procedure in this chapter can help you solve or circumvent the problem. If you need more information, refer to the following publications before you contact your service representative:

- *IBM System/36 System Problem Determination - 5360* (SC21-7919) if you use a System/36 Unit 5360
- *IBM System/36 System Problem Determination - 5362* (SC21-9063) if you use a System/36 System Unit 5362
- *System Problem Determination - 5364* (SC21-9375) if you use a System/36 Unit 5364.

How to Use this Procedure

This procedure is arranged in a sequence of questions that you can answer with a *Yes* or *No.* Based on your answer, you are directed to another question or to a recommendation for action.

Start at the beginning of the procedure and follow the question-and-answer sequence, answering each question to which you are directed based on your previous answer. If the problem is a condition that requires more detailed procedures, you are referred to those procedures.

Identifying and solving RPG Problems

When an RPG problem occurs, you can use the following series of questions to pinpoint its possible cause:

1 **Did you receive a message indicating that an operator needs to do something to a device such as a printer or a display station?**

No **Yes**

Take the actions indicated by the message and save any automatic dumps printed as a result of the message. If the action requires operator intervention, call your system operator. If the action prompts you to call for help, contact your service representative.

When you examine a message for indicated actions, check the following:

- Second-level message text, which describes the message in more detail. To get the second-level message text, press the Help key.
- Some messages contain a number of options for possible recovery actions. These options are explained in Chapter 1 of the *RPG II Messages* manual (SC21-7940).

If you still cannot solve your problem after fully examining the message, contact your service representative.

2 **If you are having problems communicating with the system, are other system users having problems communicating with the system as well?**

No **Yes**

Call your system operator and describe the problem. Have your operator use the procedures in the appropriate *System Problem Determination* manual referred to at the beginning of this chapter.

3 **Is this the first time ever the job or subroutine was run?**

Yes **No**

You may have a system problem. Call your system operator, describe your problem, and have the operator use the procedures in the appropriate *System Problem Determination* manual referred to at the beginning of this chapter.

4 **Have changes been made to the user program since the last time it ran successfully?**

No Yes

Read on, but consider what has been changed. For example: have operating procedures changed? are new device files being used? or have program changes been applied recently? A good starting point for problem determination is a changed item.

5 **Are you having a problem not caused by the programming, such as spooled output that is not produced, or a device that is not working?**

No Yes

Confirm that the device was started. If it was, you may have a system problem. Call your system operator and have the operator use the appropriate procedures as described in the corresponding *System Problem Determination* manual referred to at the beginning of this chapter.

6 **Are you using the current release of SSP?**

To answer this question, enter the SSP STATUS SESSION (or D S) operator command, and press the Roll Up key three times. In the upper left corner of the fourth screen the current release of SSP is displayed.

Yes No

Install the current release of SSP.

7 **Have all IBM-supplied PTFs (Program Temporary Fixes) you have received that apply to the current release of SSP been installed?**

Yes No

Install the PTFs you have received that have not yet been applied.

8 **Are you using the current release of RPG? The release number is printed on the first line of the source listing for any RPG program listed.**

Yes No

Install the current release of RPG and compile or run the program again.

9 **Have all IBM-supplied PTFs that apply to the current release of RPG been installed?**

Yes No

Install the PTFs you have received that have not yet been applied, and run the program again.

10 **Have any non-IBM changes been made to RPG or to SSP?**

No **Yes**

If RPG has been changed, install its current release and the PTFs, and run the program again. If SSP has been changed, install its current release and the PTFs.

11 **Was the output incorrect or not produced?**

No **Yes**

Check if:

- The program is in an infinite loop. Cancel the job. Apply the DEBUG operation in the program to locate the loop. Correct the problems in the program. Run the program again.

- The output data has incorrect values. Use the test data and walk through the program by hand to determine what the values are supposed to be. Compare the two sets of values and determine the statements in which the value of a data item can be changed. Correct the problems in the program. Run the program again.

12 **Did the RPG user program have an exception/error?**

No **Yes**

Two kinds of exceptions/errors can occur: program and file. Examples of program exception/errors are division by zero, use of an incorrect index, and use of incorrect data items in an arithmetic operation. Examples of file exception/errors are undefined record types and device errors.

You may begin investigating the problem by using a DEBUG operation as described in Chapter 4.

If you cannot solve or circumvent the problem, contact your service representative.

13 Was a printout produced?

Yes — No

Check if:

- Your display station is assigned to another printer. Use the SSP STATUS SESSION (or D S) operator command to find the printer ID of the session printer for your display station. Compare the printer ID of the printer you are using to the printer ID of session printer. Either get your printouts from the current session printer, or change the printer identification of the spool file by using the SSP SET procedure.

- The program contains a conditional statement that controls whether or not something is printed. Check the compiler listing to find the conditional statements that control printing. Determine the data values that satisfy the conditions and compare these values to those used when the program ran. If you need a printout, you can either use data values for which printing is performed, set an indicator on to control printing, or change the conditional statement to let printing be performed for the data values you are using. Run the program again.

14 Has the Input Inhibited indicator stayed on longer than expected?

No — Yes

Some possible causes are:

- You are attempting to access a data file currently being used by another program.

- There is a message at the system console that requires action by your system operator.

- Your program is in an infinite loop.

Press the Error/Reset key. If the indicator does not turn off and you believe the program is in an infinite loop, press the Attn key. You will get a menu of inquiry options for an interrupted job.

If you still cannot solve your problem, contact your service representative.

15 **Is the Attention key working?**

Yes **No**

Check if the program has performed an ON ATTN IGNORE statement. Look for an ON ATTN IGNORE statement on the printout of the program. If you need to stop the program, ask the system operator to cancel the job. Sign on again and continue using the system.

If you do not need to stop the program, wait until the program ends or an ON ATTN SYSTEM statement is performed. The Attn key request will then be processed.

If this creates a problem, you may want to change the program by removing the ON ATTN IGNORE statement or by putting an ON ATTN SYSTEM statement right after the statement(s) that require the ON ATTN IGNORE condition.

If after using this procedure you or your system operator have not solved the problem, please consult the *System Problem Determination* manual for your system unit referred to at the beginning of this chapter. If your problem occurred when you compiled your program, refer also to the *Solving Problems that Occur at Compilation Time* section in Chapter 3.

Contacting Your Service Representative

If you cannot solve your problem using the problem determination procedures listed in this chapter, in Chapter 3, and in the appropriate *System Problem Determination* manual referred to at the beginning of this chapter, you may want to contact your service representative. Before contacting your service representative, prepare the following:

- For compile time problems:
  - A diskette copy of the RPG user source program
  - A diskette copy of the user compile procedure
  - A diskette copy of the user source copy members
  - A diskette copy of the user assembler subroutines
  - A listing of the source compilation
  - A task dump at the time of the failure
  - Run the APAR procedure documented in *IBM System/36 System Problem Determination - 5360* or in *IBM System/36 System Problem Determination - 5362* and in *System Reference*, and include the entire history file.
- For execution time problems, provide the above required information as well as:
  - A diskette copy of the user load module
  - A diskette copy of the user files
  - A diskette copy of the user display screens
  - A diskette copy of the user execution procedure

The procedures for obtaining the above information are explained in the *System Problem Determination* manual for your system unit.

Glossary

$SOURCE file. The file from which program products read their input statements in order to process them.

$WORK file. The file used by some program products for processing their input statements.

#LIBRARY. The library, provided with the system, that contains the System Support Program Product. See *system library*.

access method. The way that records in files are referred to by the system. The reference can be consecutive (records are referred to one after another in the order in which they appear in the file), or it can be random (the individual records can be referred to in any order).

acknowledgment character (ACK). In binary synchronous communications, a transmission control character sent as a positive response to a data transmission.

ACK0. In binary synchronous communications, the even-numbered positive acknowledgment character. See *acknowledgment character*.

ACK1. In binary synchronous communications, the odd-numbered positive acknowledgment character. See *acknowledgment character*.

acquire. To assign a display station or session to a program.

adapter. See *communications adapter*.

address. A name, label, or number that identifies a location in storage, a device in a network, or any other data source.

address output file. Record address file, containing relative numbers, that is produced by a sort program.

addrout file. See *address output file*.

allocate. To assign a resource, such as a disk file or a diskette file, to perform a specific task.

alphabetic character. Any one of the letters A through Z (uppercase and lowercase), or any one of the special characters #, $, and @.

alphameric. Consisting of letters, numbers, and often other symbols, such as punctuation marks and mathematical symbols.

alternative collating sequence. A user-defined collating sequence that replaces the standard EBCDIC collating sequence.

alternative index. An index that is built after an indexed file is created and that provides a different order for reading or writing records in the file. Contrast with *primary index*.

application. (1) A particular business task, such as inventory control or accounts receivable. (2) A group of related programs that apply to a particular business area, such as the Inventory Control or the Accounts Receivable application.

application program. A program used to perform an application or part of an application.

array. A named set of data items, all of which are the same type, arranged in a pattern (for example, columns and rows).

array element. A single data item in an array.

ascending key sequence. The arrangement of data in order from the lowest value of the key field to the highest value of the key field. Contrast with *descending key sequence*.

assembler language. A symbolic programming language in which the set of instructions includes the instructions of the machine and whose data structures correspond directly to the storage and registers of the machine.

asynchronous transmission. In data communications, a method of transmission in which the bits included in a character or block of characters occur during a specific time interval.

However, the start of each character or block of characters can occur at any time during this interval. Contrast with *synchronous transmission*.

auto report. An RPG option that simplifies the defining of formats for printed reports and that allows the previously written statements to be included in new programs.

autoanswer. In data communications, the ability of a station to receive a call over a switched line without operator action. Contrast with *manual answer*.

autocall. In data communications, the ability of a station to place a call over a switched line without operator action. Contrast with *manual call*.

autolink. A part of the overlay linkage editor that automatically resolves external references by searching the library for the appropriate object program.

batch BSC. The SSP support that provides data communications with BSC computers and devices via the RPG T specification or the assembler $DTFB macroinstruction.

batch processing. A processing method in which a program or programs process records with little or no operator action. Contrast with *interactive processing*.

binary. (1) Pertaining to a system of numbers to the base two; the binary digits are 0 and 1. (2) Involving a choice of two conditions, such as on-off or yes-no.

binary synchronous communications (BSC). A form of communications line control that uses transmission control characters to control the transfer of data over a communications line. Compare with *synchronous data link control*.

bit. Either of the binary digits 0 or 1. See also *byte*.

block. (1) A group of records that is recorded or processed as a unit. Same as *physical record*. (2) Ten sectors (2560 bytes) of disk storage. (3) In data communications, a group of records that is recorded, processed, or sent as a unit.

branch instruction. An instruction that changes the sequence in which the instructions in a computer program are performed. The sequence of instructions continues at the address specified in the branch instruction.

buffer. (1) A temporary storage unit, especially one that accepts information at one rate and delivers it at another rate. (2) An area of storage, temporarily reserved for performing input or output, into which data is read or from which data is written.

byte. The amount of storage required to represent one character; a byte is 8 bits.

call. (1) To activate a program or procedure at its entry point. Compare with *load*. (2) In data communications, the action necessary in making a connection between two stations on a switched line.

cancel. To end a task before it is completed.

chained file. An input, output, or update disk file from which records can be read randomly.

character. A letter, digit, or other symbol.

code. (1) Instructions for the computer. (2) To write instructions for the computer. Same as *program*. (3) A representation of a condition, such as an error code.

collating sequence. The sequence in which characters are ordered within the computer for sorting, combining, or comparing.

combined file. A file used as both an input and an output file.

command display. A display that allows an operator to display and send messages, and use control commands and procedure commands to start and control jobs. Contrast with *standby display*. See also *console display* and *subconsole display*.

command key indicator. Coded as KA through KL, and KQ through KY. An indicator that is set on when an operator presses the corresponding command key.

command keys. The 12 keys on the top row of the display station keyboard that are used with the Cmd key (and optionally the Shift key) to request up to 24 different actions defined for program products and user programs. Compare with *character key* and *function key*.

comment. Words or statements in a program or procedure that serve as documentation rather than as instructions.

communications adapter. A hardware feature that enables a computer or device to become a part of a data communications network.

compilation time. The time during which a source program is translated from a high-level language to a machine language program.

compile. To translate a program written in a high-level programming language into a machine language program.

compile-time array. An array that is built into the source program and that becomes a permanent part of the compiled program. Contrast with *execution-time array* and *preexecution-time array*.

compile-time table. A table that is built into the source program and that becomes a permanent part of the compiled program. Contrast with *preexecution-time table*.

compiler. A program that translates instructions written in a high-level programming language into machine language.

compression. (1) A technique for removing strings of duplicate characters and for removing trailing blanks before transmitting data. (2) In data communications, a technique for removing strings of duplicate characters.

conditioning. The use of indicators to control when calculations or output operations are done

conditioning indicator. An indicator used to indicate when calculations are done or which attributes apply to a format or format field.

configuration. The group of machines, devices, and programs that make up a data processing system. See also *system configuration*.

configure. (1) To describe (to the system) the devices, optional features, and program products installed on a system. (2) To describe to SSP-ICF both the communication facilities connected to System/36 and the attributes of the subsystem and remote system.

consecutive processing. The processing of records in the order in which they exist in a file. Same as *sequential processing*. See also *random processing*.

console display. A display that can be requested only at the system console. From a console display an operator can display, send, and reply to messages and use all control commands.

constant. A data item with a value that does not change. Contrast with *variable*.

continuation line. A line of a source statement into which characters are entered when the source statement cannot be contained on the previous line or lines.

control break. A change in the contents of a control field.

control field. A field that identifies a record's relationship to other records (such as a part number in an inventory record). Control fields are compared from record to record to determine when certain operations are to be performed.

control-level indicator. An indicator used to specify certain fields as control fields and to control which operations are performed at total and detail time in the RPG program cycle.

control station. The primary or controlling computer on a multipoint line. The control station controls the sending and receiving of data.

control storage. Storage in the computer that contains the programs used to control input and output operations and the use of main storage. Contrast with *main storage*.

conversational file. A BSC file that allows sending data characters as a response instead of using acknowledgment characters.

creation date. The program date at the time a file is created. See also *program date*, *session date*, and *system date*.

current library. The first library searched for any required members. The current library can be specified during sign-on or while running programs and procedures.

current record. The record that is currently available to the program.

cursor. A movable symbol (such as an underline) on a display, usually used to indicate to the operator where to type the next character.

data communications. The transmission of data between computers and/or remote devices (usually over a long distance).

data display station. A display station from which an operator can only enter data. A data display station is acquired and controlled by a program. Contrast with *command display station*.

data link escape (DLE) character. In BSC, a transmission control character usually used in transparent text mode to indicate that the next character is a transmission control character.

data management. The System Support Program Product support that processes a request to read or write data.

data terminal equipment (DTE). The data processing unit that uses communications lines.

data type. A category that identifies the mathematical qualities and internal representation of data.

debug. To detect, locate, and remove mistakes from a program.

decimal. Pertaining to a system of numbers to the base ten; decimal digits range from 0 through 9.

default value. A value stored in the system that is used when no other value is specified.

define-the-file (DTF). A control block containing information that is passed between data management routines and users of the data management routines.

delete-capable file. A file from which records can be logically removed without compressing the file.

delta position. The position in the edit word that corresponds to the leftmost position in the data field.

demand file. A file that can be specified as an input, update, or combined file and that is used with the READ or KEY operation code.

descending key sequence. The arrangement of data in order from the highest value of the key field to the lowest value of the key field. Contrast with *ascending key sequence.*

detail record. A record that contains the daily activities or transactions of a business. For example, the items on a customer order are typically stored in detail records. Contrast with *header record.*

detail time. A portion of the RPG program cycle during which calculation and output operations for specified fields are performed for each record read.

development support utility (DSU). A program product that can be used to create, edit, remove, view, or print source members and procedure members. DSU has a full screen editor that allows to edit an entire screen of data at a time. For the RPG programmer, DSU can check RPG II or Autoreport statements for syntax errors as the statements are entered, or all at once after the statements have been entered. DSU can co-reside with SEU, and requires no conversion of data.

diagnosed-source file. A library member containing source statements and associated error messages.

diagnosed-source member. See *diagnosed-source file.*

direct file. (1) A disk file in which records are referenced by the relative record number. Contrast with *indexed file* and *sequential file.*(2) See *relative file.*

disk. A storage device made of one or more flat, circular plates with magnetic surfaces on which information can be stored.

disk file. A set of related records on disk that are treated as a unit.

diskette. A thin, flexible magnetic plate that is permanently sealed in a protective cover. It can be used to store information copied from the disk.

display. (1) A visual presentation of information on a display screen. (2) To show information on the display screen.

display format. Data that defines (or describes) a display.

display layout sheet. A form used to plan the location of data on the display.

display screen. The part of the display station on which information is displayed.

display station. A device that includes a keyboard from which an operator can send information to the system and a display screen on which an operator can see the information sent or receive information from the system.

DLE. See *data link escape character.*

do group. A group of operations that can be performed repeatedly and is delimited and controlled by a DO, DOUxx, DOWxx, or IFxx operation and an associated END operation.

DSU. See *development support utility.*

DTE. See *data terminal equipment.*

DTF. See *define-the-file.*

dump. (1) To copy the contents of all or part of storage, usually to an output device. (2) Data that has been dumped.

EBCDIC. See *extended binary-coded decimal interchange code.*

EBCDIC character. Any one of the symbols included in the 8-bit EBCDIC set.

edit. (1) To modify the form or format of data; for example, to insert or remove characters such as for dates or decimal points. (2) To check the accuracy of information that has been entered, and to indicate if an error is found.

edit code. A number or letter indicating that editing should be done according to a defined pattern.

element. The smallest unit of data in a table or array.

embedded blanks. Blanks that are surrounded by any other characters.

end-of-text (ETX) character. In binary synchronous communications, the transmission control character used to end a logical set of records that began with the start-of-text character.

end-of-transmission (EOT) character. In binary synchronous communications, the transmission control character usually used to end communications.

end-of-transmission-block (ETB) character. In binary synchronous communications, the transmission control the start-of-text character.

enter. To type in information on a keyboard and press the Enter key in order to send the information to the computer.

EOT. See *end-of-transmission character.*

ETB. See *end-of-transmission-block character.*

ETX. See *end-of-text character.*

EXCEPT group name. A name used in the place of indicators to identify a record or group of records written at exception output time.

execution-time array. An array that is loaded after the program begins. Contrast with *compile-time array* and *preexecution-time array.*

extended binary-coded decimal interchange code (EBCDIC). A set of 256 eight-bit characters.

external indicators. Indicators that can be set by another program before a program is run or changed while a program is running. The external indicators are U1 through U8.

feature. A programming or hardware option, usually available at an extra cost.

fetch routine. A routine that finds a segment on disk and loads the segment into storage.

field. One or more characters of related information (such as a name or an amount).

file. A set of related records treated as a unit.

file name. The name used by a program to identify a file. See also *label.*

first-page indicator. Coded as 1P. An indicator that specifies which lines (such as headings) should be printed on the first page only.

fixed currency symbol. A currency symbol that appears in a specified position of an edited field. Contrast with *floating currency symbol.*

floating currency symbol. A currency symbol that appears immediately to the left of the leftmost position that is used in an edited field. Contrast with *fixed currency symbol.*

format. (1) A defined arrangement of such things as characters, fields, and lines, usually used for displays, printouts, or files. (2) To arrange such things as characters, fields, and lines.

full-procedural file. A DISK file that can be processed both randomly and sequentially.

function key. A keyboard key that requests an action but does not display or print a character. The cursor movement and Help keys are examples of function keys. Compare with *command keys* and *character key.*

group indication. The printing of control information for only the first record of a group of records containing identical control information.

half-adjust. A method of rounding off a number by adjusting the last significant digit.

hardware. The equipment, as opposed to the programming, of a system.

header record. A record that contains information, such as customer name and customer address, that is common to following detail records. Contrast with *detail record.*

hex. See *hexadecimal.*

hexadecimal. Pertaining to a system of numbers to the base sixteen; hexadecimal digits range from 0 (zero) through 9 (nine) and A (ten) through F (fifteen).

host system. The primary or controlling computer in the communications network. See also *control station.*

I/O. See *input/output.*

ID. Identification.

ideographic data. Two-byte characters enclosed by the shift-out (S/O) control character (hex 0E) and the shift-in (S/I) control character (hex 0F).

index. (1) A table containing the key value and location of each record in an indexed file. (2) A computer storage position or register, the contents of which identify a particular element in a set of elements.

indexed file. A file in which the key and the position of each record is recorded in a separate portion of the file called an index. Contrast with *direct file* and *sequential file.*

indicator. An internal switch that communicates a condition between parts of a program or procedure.

informational message. A message that provides information to the operator, but does not require a response.

initialize. To prepare for use. For example, to initialize a diskette.

input. Data to be processed.

input/output (I/O). Pertaining to either input or output, or both.

inquiry. (1) A request for information in storage. (2) A request that puts a display station into inquiry mode. (3) In data communications, a request for information from another system.

inquiry mode. A mode during which the job currently running from a display station is interrupted so that other work can be done. The operator puts the display station in inquiry mode by pressing the Attn key.

inquiry program. (1) A program that allows an operator to get information from a disk file. (2) A program that runs while the system is in inquiry mode.

interactive. Pertains to activity involving requests and replies as, for example, between an operator and a program or between two programs.

Interactive Communications Feature (SSP-ICF). A feature of the System Support Program Product that allows a program to interactively communicate with another program or system.

interactive processing. A processing method in which each operator action causes a response from the program or the system. Contrast with *batch processing.*

intermediate block check. In binary synchronous communications, an option that permits checking each record, instead of checking the contents of the total buffer, when large buffers of data are received.

intermediate-text-block (ITB) character. In binary synchronous communications, the transmission control character used to indicate the end of a section of data to be checked. See *intermediate block check.*

interrupt. (1) To temporarily stop a process. (2) In data communications, to take an action at a receiving station that causes the sending station to end a transmission.

ITB. See *intermediate-text-block character.*

job. (1) A unit of work to be done by a system. (2) One or more related procedures or programs grouped into a procedure.

job queue. A list, on disk, of jobs waiting to be processed by the system.

job step. A unit of work represented by a single program or a procedure that contains a single program. A job consists of one or more job steps.

K-byte. 1024 bytes.

key. One or more characters used to identify the record and establish the record's order within an indexed file.

label. The name that identifies a statement.

level indicator. Two characters (L0 through L9 and LR) that control calculation and output processing during total time.

library. (1) A named area on disk that can contain programs and related information (not files). A library consists of different sections, called library members. (2) The set of publications for a system.

library member. A named collection of records or statements in a library. The types of library members are *load member*, *procedure member*, *source member*, and *subroutine member*.

limits file. A file that contains upper and lower values of the record keys that can be used to read from an indexed file.

link-editing. To combine, by the overlay linkage editor, a number of load members and/or subroutine members into one program.

linkage. The coding that passes control and parameters between two routines.

literal. A symbol or a quantity in a source program that is itself data, rather than a reference to data.

load. (1) To move data or programs into storage. (2) To place a diskette into a diskette drive or a magazine into a diskette magazine drive. (3) To insert paper into a printer.

load member. A library member that contains information in a form that the system can use directly, such as a display format. Contrast with *source member*.

load module. A program in a form that can be loaded into main storage and run. The load module is the output of the overlay linkage editor.

local. Pertaining to a device that is directly connected to your system without the use of a communications line. Contrast with *remote*.

local data area. A 512 byte area on disk that can be used to pass information between jobs and job steps during a session. A separate local data area exists for each command display station.

location name. In interactive communications, the identifying name associated with a particular remote system or device.

look-ahead field. A field that allows the program to look at information in a field on the next record that is available for processing in any input or update file.

loop. A sequence of instructions that is performed repeatedly until an ending condition is reached.

machine language. A language that can be used directly by a computer without intermediate processing.

main storage. The part of the processing unit where programs are run. Contrast with *control storage*.

manual answer. In data communications, the operator actions required to receive a call over a switched line. Contrast with *autoanswer*.

manual call. In data communications, the operator actions required to place a call over a switched line. Contrast with *autocall*.

master file. A collection of permanent information, such as a customer address file.

match fields. When processing more than one file with RPG, fields that are compared to determine whether operations should be done.

match level. The value identified by the match field indicators M1 through M9. The match level identifies match fields.

member. See *library member*.

memory resident overlay (MRO). An option that allows a program to request that its overlays remain in main storage.

menu. A displayed list of items from which an operator can make a selection.

message. Information sent to an operator or programmer from a program. A message can be either displayed or printed.

message identification. A field in the display or printout of a message that directs the user to the description of the message in a message guide or a reference manual. This field consists of up to four alphabetic characters, followed by a dash, followed by the message identification code.

message identification code (MIC). A four-digit number that identifies a record in a message member. This number can be part of the message identification.

message member. A library member that defines the text of each message and its associated message identification code.

MIC. See *message identification code*.

mode. A method of operation. For an example, see *enter/update mode*.

MRO. See *memory resident overlay*.

MRT program. See *multiple requester terminal program.*

multiple requester terminal (MRT) program. A program that can process requests from more than one display station or SSP-ICF session at the same time using a single copy of the program. Contrast with *single requester terminal (SRT) program.*

NEP. See *never-ending program.*

never-ending program (NEP). A long-running program that does not share system resources, except for shared files and the spool file.

noncontiguous key. A key made up of characters or character strings which occur in separate areas of a record.

nonswitched line. A connection between computers or devices that does not have to be established by dialing. Contrast with *switched line.*

not-found indicator. An indicator that is set on when the specified record cannot be found.

numeric. Pertaining to any of the digits 0 through 9.

object module. A set of instructions in machine language. The object module is produced by a compiler from a subroutine or source program and can be input to the overlay linkage editor.

OCL. See *operation control language.*

offline. Neither controlled directly by, nor communicating with, the computer, or both. Contrast with *online.*

OLE. See *overlay linkage editor.*

online. Being controlled directly by, or directly communicating with, the computer, or both. Contrast with *offline.*

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

operation control language (OCL). A language used to identify a job and its processing requirements to the System Support Program Product.

output. The result of processing data.

overflow indicator. An indicator that signifies that the last line on a page has been printed or skipped.

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

overlay. (1) To write over (and therefore destroy) an existing file. (2) A program segment that is loaded into main storage and replaces all or part of a previously loaded program segment.

overlay linkage editor (OLE). The part of the System Support Program Product that combines object programs to produce code that can be run and allows the user to determine overlays for programs.

overlay region. A continuous area of main storage in which segments can be loaded independently of other regions.

override. (1) A parameter or value that replaces a previous parameter or value. (2) To replace a parameter or value.

packed decimal format. A format in which each byte (except the rightmost byte) within a field represents two numeric digits. The rightmost byte contains one digit and the sign. For example, the decimal value + 123 is represented as 0001 0010 0011 1111. Contrast with *zoned decimal format.*

packed key. An index key in packed decimal format.

parameter. A value supplied to a procedure or program that either is used as input or controls the actions of the procedure or program.

physical file. An indexed file containing data for which one or more alternative indexes have been created.

point-to-point line. A communications line that connects a single remote station to a computer.

poll. To execute a polling sequence.

polling. A method for determining whether each of the stations sharing a communications line has data to send.

preexecution-time array. An array that is loaded at the same time as the user program, before the program begins. Contrast with *compile-time array* and *execution-time array.*

preexecution-time table. A table that is loaded at the same time as the user program, before the program begins. Contrast with *compile-time table.*

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

primary index. The index that is built when an indexed file is created. Contrast with *alternative index.*

printout. Information from the computer that is produced by a printer.

priority. The relative ranking of items. For example, a job with high priority in the job queue will be run before one with medium or low priority.

problem determination. The process of identifying why the system is not working. Often this process identifies programs, equipment, data communications facilities, or user errors as the source of the problem.

procedure. A set of related operation control language statements (and, possibly, utility control statements and procedure control expressions) that cause a specific program or set of programs to be performed.

processing unit. The part of the system unit that performs instructions and contains main storage.

program. (1) A sequence of instructions for a computer. See *source program* and *load module.* (2) To write a sequence of instructions for a computer. Same as *code.*

program cycle. In RPG, the series of operations performed by the computer for each record read.

program date. The date associated with a program (job step). See also *creation date, session date*, and *system date.*

program product. A licensed program for which a fee is charged.

prompt. A displayed request for information or operator action.

queue. A line or list formed by items waiting to be processed.

random access. An access method in which records can be read from, written to, or removed from a file in any order.

random by key. A processing method for chained files in which record keys identify records to be processed.

random by relative record number. A processing method for chained files in which relative record numbers identify the records to be processed.

random processing. The processing of records in an order other than the order that they exist in a file. See also *consecutive processing* and *sequential processing.*

record. A collection of fields that is treated as a unit.

record address file. An input file that indicates to a program which records are to be read from a disk file, and the order in which these records are to be read from the disk file.

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

record-identifying indicator. An indicator that identifies the record just read.

record type. The classification of records in a file.

region. The amount of main storage available for a program. See also *job region* and *step region.*

relative record number. A number that specifies the location of a record in relation to the beginning of the file.

remote. Pertaining to a system or device that is connected to your system through a communications line. Contrast with *local.*

requester. A display station or interactive communications session that requests a program to be run.

resulting indicator. An indicator that is set depending on the result of an operation.

return code. In data communications, a value generated by the system or subsystem that is returned to a program to indicate the results of an operation issued by that program.

reverse-interrupt character (RVI). In binary synchronous communications, a request by the receiving station to the sending station to stop sending and receive a message.

root segment. The first segment of an program with overlays. The root segment remains in main storage at all times while the program is being run.

routine. A set of statements in a program that causes the system to perform an operation or a series of related operations.

run. To cause a program, utility, or other machine function to be performed.

scratch file. A file, usually used as a work file, that exists until the program that uses it ends.

screen design aid (SDA). The part of the Utilities Program Product that helps the user design, create, and maintain displays and menus. Additionally, SDA can generate specifications for RPG and WSU work station programs.

SDA. See *screen design aid.*

search word. Data used to find a match in a table or array.

secondary file. Any input file other than the primary file.

sequential access. An access method in which records are read from, written to, or removed from a file based on the logical order of the records in the file.

sequential by key. A method of indexed file processing in which records are read or written in the order of the record keys.

sequential file. A file in which records occur in the order in which they were entered. Contrast with *direct file* and *indexed file.*

sequential processing. The processing of records in the order in which they exist in a file. Same as *consecutive processing* See also *random processing.*

session. (1) The logical connection by which a System/36 program or device can communicate with a program or device at a remote location. (2) The length of time that starts when an operator signs on the system and ends when the operator signs off the system.

session date. The date associated with a session. See also creation date, *program date*, and *system date*

session library. The library specified, or assigned as a default, when signing on or while running a program.

SEU . See *source entry utility.*

shift-in (S/I) control character:. A character that indicates the end of a string of ideographic characters. The shift-in control character is represented by hex OF.

shift-out (S/O) control character:. A character that indicates the start of a string of ideographic characters. The shift-out control character is represented by hex OE.

single requester terminal (SRT) program. A program that can process requests from only one display station or SSP-ICF session from each copy of the program. Contrast with *multiple requester terminal program.*

SNA. See *systems network architecture.*

source entry utility (SEU). The part of the Utilities Program Product used by the operator to enter and update source and procedure members.

source member. A library member that contains information in the form in which it was entered, such as RPG specifications. Contrast with *load member.*

source program. A set of instructions that are written in a programming language and that must be translated to machine language before the program can be run.

special character. A character other than an alphabetic or numeric character. For example; *, +, and % are special characters.

specification sheets. Forms on which a program is coded and described.

spool file. A disk file that contains output that has been saved for later printing.

spool-writer. The part of the System Support Program Product that prints output that has been saved in the spool file.

spooling. The part of the System Support Program Product that saves output on disk for later printing.

SRT program. See *single requester terminal program.*

SSP. See *System Support Program Product.*

SSP-ICF. See *Interactive Communications Feature.*

standby display. A display that allows an operator to enter data only. When a standby display appears, the display station can be acquired by a program. Contrast with *command display.*

start-of-text (STX) character. In binary synchronous communications, a transmission control character used to begin a logical set of records that will be ended by the end-of-text character or end-of-transmission-block character.

storage usage map. An overlay linkage editor printout that shows the names and storage locations of routines that make up the load member.

STX character. See *start-of-text character*.

subconsole. A display station that controls a printer or printers.

subconsole display. A display that can be requested only from a command display that appears on a subconsole. From a subconsole display an operator can display and send messages, and enter all control commands except those that can be entered only at the system console. See also *console display*.

subroutine. A group of instructions that can be called by another program or subroutine.

switched line. In data communications, a connection between computers or devices that is established by dialing. Contrast with *nonswitched line*.

synchronous. Occurring in a regular or predictable sequence.

synchronous data link control (SDLC). A form of communications line control that uses commands to control the transfer of data over a communications line. Compare with *binary synchronous communications*.

synchronous transmission. In data communications, a method of transmission in which the sending and receiving of characters is controlled by timing signals. Contrast with *asynchronous transmission*.

system. The computer and its associated devices and programs.

system configuration. A process that specifies the machines, devices, and programs that form a particular data processing system.

system console. A display station from which an operator can keep track of and control system operation.

system date. The date assigned by the system operator during the initial program load procedure. See also *creation date*, *program date* and *session date*.

system library. The library, provided with the system, that contains the System Support Program Product and is named #LIBRARY.

System Support Program Product (SSP). A group of licensed programs that manage the running of other programs and the operation of associated devices, such as the display station and printer. The SSP also contains utility programs that perform common tasks, such as copying information from diskette to disk.

systems network architecture (SNA). A set of rules for controlling the transfer of information in a data communications network.

table. (ANSI) A collection of data in which each item is uniquely identified by a label, by its position relative to the other items, or by some other means.

temporary-text-delay (TTD) character. A BSC transmission control character that indicates to the receiving station that there is a temporary delay in the transmission of data.

total operations. Calculation and output operations performed only after a group of records has been processed.

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

transaction. (1) An item of business. The handling of customer orders and customer billing are examples of transactions. (2) In interactive communications, the communication between the application program and a specific item (usually another application program) at the remote system.

transaction file. A file containing data, such as customer orders, that is usually used only with a master file.

transmission control characters. In data communications, special characters that are included in a message to control communication over a data link. For example, the sending station and the receiving station use transmission control characters to exchange information; the receiving station uses transmission control characters to indicate errors in data it receives.

transparent data. Data that can contain any hexadecimal value.

transparent literal. a literal (or constant) that begins with an apostrophe followed immediately by the shift-out (S/O) control character (hex 0E), and up to 3 ideographic characters for a literal or up to 11 ideographic characters for a constant, and ends with the shift-in (S/I) control character (hex 0F) followed immediately by an apostrophe.

transparent text mode. A mode that allows BSC to send and receive messages containing any of the 256 character combinations in hexadecimal, including transmission control characters.

tributary station. In data communications, a secondary device on a multipoint line.

truncate. To shorten a field or statement to a specified length.

TTD character. See *temporary-text-delay character*.

unique. The only one.

update file. A disk file from which a program reads a record, updates fields in the record, and writes the record back into the location it came from.

Utilities Program Product. A program product that contains the data file utility (DFU), the source entry utility (SEU), the work station utility (WSU), and the screen design aid (SDA).

utility program. A System Support Program Product program that allows you to perform a common task, such as copying information from diskette to disk.

valid. (1) Allowed. (2) True, in conforming to an appropriate standard or authority.

variable. A name used to represent a data item whose value can change while the program is running. Contrast with *constant*.

work file. A file that is used for temporary storage of data being processed.

work station. A device that lets people transmit information to or receive information from a computer; for example, a display station or printer.

work station utility (WSU). The part of the Utilities Program Product that helps you to write programs for data entry, editing, and inquiry.

World Trade. (1) Pertains to the distinction between the US and the rest of the world. (2) Pertains to the combination of:

- IBM World Trade Americas/Far East Corporation
- IBM World Trade Europe/Middle East/Africa Corporation

WSU. See *work station utility*.

X.21. In data communications, a specification of the CCITT that defines the connection of data terminal equipment to an X.21 (public data) network.

X.21 feature. The feature that allows System/36 to be connected to an X.21 network.

zero suppression. The substitution of blanks for leading zeros in a number. For example, 00057 becomes   57 when using zero suppression.

zoned decimal format. A format for representing numbers in which the digit is contained in bits 4 through 7 and the sign is contained in bits 0 through 3 of the rightmost byte; bits 0 through 3 of all other bytes contain 1s (hex F). For example, in zoned decimal format, the decimal value of + 123 is represented as 1111 0001 1111 0010 1111 0011. Contrast with *packed decimal format*.

Index

Special Characters

B

C

D

E

F

G

H

I

K

L

M

N

O

R

S

U

W

X

Y

Z

READER'S COMMENT FORM

Please use this form only to identify publication errors or to request changes in publications. Direct any requests for additional publications, technical questions about IBM systems, changes in IBM programming support, and so on, to your IBM representative or to your nearest IBM branch office. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

☐ If your comment does not need a reply (for example, pointing out a typing error) check this box and do not include your name and address below. If your comment is applicable, we will include it in the next revision of the manual.

☐ If you would like a reply, check this box. Be sure to print your name and address below.

Page number(s): **Comment(s):**

Please contact your nearest IBM branch office to request additional publications.

Name ____________________

Company or Organization ____________________

Address ____________________

____

City State Zip Code

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

SC21-9006-4

Cut Along Line

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

NO POSTAGE
NECESSARY IF
MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

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

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Information Development
Department 532
Rochester, Minnesota 55901

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

IBM

International Business Machines Corporation

IBM System/36: Programming with RPG II (File No. S36-28) Printed in U.S.A. SC21-9006-4

System/36:
Programming with RPG II

SC21-9006-4

READER'S COMMENT FORM

Please use this form only to identify publication errors or to request changes in publications. Direct any requests for additional publications, technical questions about IBM systems, changes in IBM programming support, and so on, to your IBM representative or to your nearest IBM branch office. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

☐ If your comment does not need a reply (for example, pointing out a typing error) check this box and do not include your name and address below. If your comment is applicable, we will include it in the next revision of the manual.

☐ If you would like a reply, check this box. Be sure to print your name and address below.

Page number(s): Comment(s):

Please contact your nearest IBM branch office to request additional publications.

Name ____________________

Company or Organization ____________________

Address ____________________

____

City State Zip Code

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

SC21-9006-4

Cut Along Line

Fold and tape
Please do not staple
Fold and tape

NO POSTAGE
NECESSARY IF
MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

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

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Information Development
Department 532
Rochester, Minnesota 55901

Fold and tape
Please do not staple
Fold and tape

IBM

International Business Machines Corporation

System/36:
Programming with RPG II

SC21-9006-4

READER'S COMMENT FORM

Please use this form only to identify publication errors or to request changes in publications. Direct any requests for additional publications, technical questions about IBM systems, changes in IBM programming support, and so on, to your IBM representative or to your nearest IBM branch office. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

☐ If your comment does not need a reply (for example, pointing out a typing error) check this box and **do not** include your name and address below. If your comment is applicable, we will include it in the next revision of the manual.

☐ If you would like a reply, check this box. Be sure to print your name and address below.

Page number(s): Comment(s):

Please contact your nearest IBM branch office to request additional publications.

Name ______________________________

Company or
Organization ______________________________

Address ______________________________

____
City State Zip Code

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

SC21-9006-4

Cut Along Line

Fold and tape

Please do not staple

Fold and tape

NO POSTAGE
NECESSARY IF
MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

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

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Information Development
Department 532
Rochester, Minnesota 55901

Fold and tape

Please do not staple

Fold and tape

IBM

International Business Machines Corporation

IBM System/36: Programming with RPG II (File No. S36-28) Printed in U.S.A. SC21-9006-4

What Is Your Opinion of This Manual?

Your comments can help us produce better manuals. Please take a few minutes to evaluate this manual as soon as you become familiar with it. Circle Y (Yes) or N (No) for each question that applies. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

FINDING INFORMATION

Y N Is the table of contents helpful?
What would make it more helpful?

Y N Is the index complete?
List specific terms that are missing.

Y N Are the chapter titles and other headings meaningful?
What would make them more meaningful?

Y N Is information organized appropriately?
What would improve the organization?

Y N Does the manual refer you to the appropriate places for more information?
List specific references that are wrong or missing.

UNDERSTANDING INFORMATION

Y N Is the purpose of this manual clear?
What would make it clearer?

Y N Is the information explained clearly?
Which topics are unclear?

Y N Are the examples clear?
Which examples are unclear?

Y N Are examples provided where they are needed?
Where should examples be added or deleted?

Y N Are terms defined clearly?
Which terms are unclear?

Y N Are terms used consistently?
Which terms are inconsistent?

Y N Are too many abbreviations and acronyms used?
Which ones are not understandable?

Y N Are the illustrations clear?
Which ones are unclear?

USING INFORMATION

Y N Does the information apply to your situation?
Which topics do not apply?

Y N Is the information accurate?
What information is inaccurate?

Y N Is the information complete?
What information is missing?

Y N Is only necessary information included?
What information is unnecessary?

Y N Are the examples useful models?
What would make them more useful?

Y N Is the format of the manual (shape, size, color) effective?
What would make the format more effective?

OTHER COMMENTS

Use the space below for any other opinions about this manual or about the entire set of manuals for this system.

YOUR BACKGROUND

What is your job title?

What is your primary job responsibility?

How many years have you used computers?

Which programming languages do you use?

How many times per month do you use this manual?

Your name

Company name

Street address

City, State, ZIP

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

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

NO POSTAGE
NECESSARY IF
MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

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

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Development Laboratory
Information Development, Department 532
Rochester, Minnesota 55901

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

IBM

International Business Machines Corporation

Cut Along Line

IBM®

International Business Machines Corporation

Programming with RPG II

Contents

File Number
S36-28

Order Number
SC21-9006-4

Part Number
59X3996

Printed in U.S.A.

SC21-9006-04