

# 5280



SC21-7804-1  
S5280-28

## **IBM 5280 Distributed Data System**

**DE/RPG User's Guide**

Program No. 5708-DE1



SC21-7804-1  
S5280-28

# **IBM 5280 Distributed Data System**

**DE/RPG User's Guide**

Program No. 5708-DE1

## Preface

This manual is intended as a guide to programming techniques that facilitate your use of DE/RPG. The *IBM 5280 Introduction to DE/RPG* should be read first. For details about the syntax of the language and specific details about its functions, see the *IBM 5280 DE/RPG Reference Manual*.

This manual has seven parts. Chapters form specific descriptions within each part. It is recommended that the first time you use this manual, you read it from front to back. Thereafter, you may want to use the index or table of contents to locate a specific subject.

Program listings have been included to illustrate various functions. Symbolic displays and diskettes have been provided to demonstrate what appears on the displays and what appears on the diskette data set.

**Note:** This manual follows the convention that *he* means *he* or *she*.

### Second Edition (June 1981)

This is a major revision and obsoletes SC21-7804-0 and incorporates SN21-8195. Because the changes and additions are extensive, this manual should be reviewed in its entirety.

Changes are periodically made to the information herein; these changes will be reported in technical newsletters or in new editions of this publication.

This publication is for planning purposes only. The information herein is subject to change before the products described become available. Also, this publication contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Use this publication only for the purposes stated in the *Preface*. It is possible that this material might contain reference to, or information about, IBM products (machines and programs), programming or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

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

This publication could contain technical inaccuracies or typographical errors. Use the Reader's Comment Form at the back of this publication to make comments about this publication. If the form has been removed, address your comments to IBM Corporation, Information Design and Development, Department 997, 11400 Burnet Road, Austin, Texas 78758. IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

<b>INTRODUCTION</b> . . . . .	<b>v</b>	<b>CHAPTER 5. FORMATTING TECHNIQUES FOR THE DISPLAYS</b> . . . . .	<b>73</b>
How to Use the IBM 5280 DE/RPG Manuals . . . . .	v	Using Different Display Formats For Entry and Review Modes . . . . .	77
How to Read the Listings in this Manual . . . . .	vi	Using the Position Columns to Create Prompting Text . . . . .	79
<b>PART 1. CREATING DATA SETS</b> . . . . .	<b>1</b>	Using the Display Attributes to Facilitate Entry . . . . .	82
<b>CHAPTER 1. CHARACTERISTICS OF DATA-ENTRY PROGRAMS THAT USE TRANSACTION FILES</b> . . . . .	<b>13</b>	Using EDTCDE With Display Formats . . . . .	86
Modes of Operation Available for Programs Using Transaction Files . . . . .	16	Using CLRL and SLNO to Display Multiple Records . . . . .	88
Manual Selection of an Operating Mode . . . . .	16	<b>CHAPTER 6. FORMATTING TECHNIQUES FOR THE DISKETTE DATA SET</b> . . . . .	<b>91</b>
Automatic Selection of the Correct Format for the Manually Selected Mode . . . . .	18	Merging Fields From Multiple Records . . . . .	92
Key-Initiated Modes . . . . .	20	Positioning Fields Within a Data Set . . . . .	95
Program Functions Used with Transaction Files . . . . .	22	Converting Numeric Data . . . . .	97
Special Job Functions for Use with Transaction Files . . . . .	22	<b>PART 3. USING INDICATORS</b> . . . . .	<b>99</b>
Manually Copying Records for Programs Using Transaction Files . . . . .	22	<b>CHAPTER 7. USING INDICATORS FOR DATA-ENTRY PROGRAMS</b> . . . . .	<b>101</b>
How the Usage Column Entries Affect Programs Using Transaction Files . . . . .	25	<b>CHAPTER 8. USING INDICATORS FOR APPLICATION PROGRAMS</b> . . . . .	<b>105</b>
<b>CHAPTER 2. CHARACTERISTICS OF APPLICATION PROGRAMS THAT USE SUBROUTINES ON THE C-SPECIFICATIONS</b> . . . . .	<b>27</b>	Operations that Set Indicators and Operations Conditioned by Indicators . . . . .	106
Execute Mode—The Only Mode Available For Use With Application Programs . . . . .	31	A Sample of Using Indicators With I/O Operations . . . . .	107
Organizing Data in Data Sets Using a Background Program . . . . .	32	A Sample of Using Indicators with Arithmetic Operations . . . . .	110
Controlling Data Set Update Through Subroutines . . . . .	35	A Sample of Using Indicators with Branching Operations . . . . .	113
Controlling the Input/Output Devices . . . . .	37	Complex Use of Indicators on the C-Specifications . . . . .	114
Controlling the Display . . . . .	37	<b>PART 4. ACCESS METHODS</b> . . . . .	<b>117</b>
Controlling the Diskette . . . . .	40	<b>CHAPTER 9. SEQUENTIAL ACCESS METHODS</b> . . . . .	<b>123</b>
Accessing Communications . . . . .	42	Sequential Access of Nonkeyed and Nonindexed Data Sets . . . . .	124
Combining Data-Entry and Background Programs . . . . .	43	Sequential Access of Keyed and Nonindexed Data Sets . . . . .	126
<b>PART 2. FORMATTING DATA FOR DISPLAYS AND DISKETTES</b> . . . . .	<b>47</b>	Sequential Access of Indexed Data Sets . . . . .	127
<b>CHAPTER 3. CONTROLLING DISPLAY AND DISKETTE FORMATS VIA THE Z-SPECIFICATIONS</b> . . . . .	<b>49</b>	<b>CHAPTER 10. DIRECT ACCESS METHODS</b> . . . . .	<b>129</b>
Samples Showing Various Types of Entry Format Specifications . . . . .	50	Direct Access for Nonkeyed and Nonindexed Data Sets . . . . .	129
Samples Showing Various Types of Review Format Specifications . . . . .	54	Direct Access For Keyed and Indexed Data Sets . . . . .	130
Using the Z-Specification to Control Diskette Formats For a Transaction File . . . . .	61	<b>PART 5. DATA TABLES</b> . . . . .	<b>135</b>
<b>CHAPTER 4. CONTROLLING DISPLAY AND DISKETTE FORMATS VIA THE C-SPECIFICATIONS</b> . . . . .	<b>65</b>	<b>CHAPTER 11. USING DATA TABLES IN DATA-ENTRY PROGRAMS</b> . . . . .	<b>137</b>
Calling Subroutines From the Z-Specification . . . . .	67	Considerations About Declaring Indexes . . . . .	137
Calling a Subroutine From the A-Specification . . . . .	70	Arranging Data in Tables . . . . .	139
		Arranging Tables in Data Sets . . . . .	141
		<b>CHAPTER 12. USING ARRAYS IN APPLICATION PROGRAMS</b> . . . . .	<b>143</b>
		Using LOKUP on the C-Specifications . . . . .	143
		Using ARRAYNAME,INDEX as Fields . . . . .	144
		Using the MOVEA Operation . . . . .	145

<b>PART 6. PRINTING</b> . . . . .	147
<b>CHAPTER 13. UNFORMATTED PRINTING FOR DATA-ENTRY PROGRAMS</b> . . . . .	149
<b>CHAPTER 14. FORMATTED PRINTING FOR APPLICATION PROGRAMS</b> . . . . .	151
Designing a Print Data Set to be Used with an Existing Form . . . . .	154
Designing a Print Data Set That Designs the Form to be Used . . . . .	162
<b>PART 7. USING CALCULATIONS</b> . . . . .	167
Using Named Fields in Calculations . . . . .	168
Using Counters in Calculations . . . . .	170
<b>PART 8. CHAINING FROM JOB TO JOB</b> . . . . .	173
Using EOJ on the Z-Specification . . . . .	173
Using a Constant Name to Select the Next Program . . . . .	174
Using a Variable Name to Select the Next Program . . . . .	175
<b>GLOSSARY</b> . . . . .	177
<b>APPENDIX A. A-, Z-, AND C-SPECIFICATION FORMS</b> . . . . .	179
<b>INDEX</b> . . . . .	185

Before beginning this manual you should have an understanding of the following:

- The relationship of the various DE/RPG manuals
- The listings used as samples in this manual

Each of these topics is described in the following text.

### HOW TO USE THE IBM 5280 DE/RPG MANUALS

Three manuals have been provided to describe the DE/RPG (data entry with RPG subroutines) program product. The following list indicates the audience for which each manual was written, an overview of the manual contents, and a set of objectives for each manual. Review this list before beginning to use this manual to determine which manual provides background necessary for your understanding of the subject.

*IBM 5280 Introduction to DE/RPG, SC21-7803.* The audience for this manual is anyone who is either unfamiliar with data entry or with this program product and who needs to understand the data-entry functions of the program product. The manual takes the reader through the entire process of creating a simple data-entry program (beginning with designing the displays to be used and concluding with using the program to enter data). A second data-entry program is included to help the reader understand some of the more advanced data-entry functions provided. The most complex subject that is included in the manual is the use of tables. The objective of the manual is to familiarize an inexperienced user with some basic data-entry functions and operations provided by DE/RPG and to provide the user with enough information to allow him to create a basic DE/RPG program. The manual does not contain any information about the calculation specification operations available with DE/RPG.

*IBM 5280 DE/RPG User's Guide, SC21-7804.* The audience for this manual is a person who has some knowledge of either DE/RPG or of programming concepts (preferably one who has used another program language). The manual describes characteristics of the language and illustrates how its functions can be applied to typical business applications. The topics progress from the simple to the complex. The objective of the manual is to provide the user with an understanding of how DE/RPG can be used to solve business-related problems. Since some of the more powerful functions provided with DE/RPG are the result of combining operations, another objective of the manual is to describe some of these combined operations and to show how they can be used.

The *User's Guide* has been organized to allow you to find the specific topic you need. There are seven parts within the manual; each part is marked by a title page. When you read the manual for the first time, read through all the parts to understand the organization of the program product and of the manual. Once you are familiar with the product, you can refer directly to the specific descriptions you need. The *User's Guide* is written to present topics related to the DE/RPG program product; therefore, it does not include information about operating the system. For this type of information, see the *IBM 5280 DE/RPG Reference Manual*, SC21-7787, the *IBM 5280 Operator's Guide*, GA21-9364, and the *IBM 5280 System Concepts manual*, GA21-9352.

*IBM DE/RPG Reference Manual*, SC21-7787. The audience for this manual is all users of the DE/RPG program product. The content of the manual includes detailed descriptions of the characteristics, functions, and operations provided by the program product. The objective of the manual is to provide all the information required for the user of the program product to code and debug programs.

## HOW TO READ THE LISTINGS IN THIS MANUAL

To help you learn how to effectively use DE/RPG, this manual includes actual program listings. These listings merge statements from the Z-, A-, and C-specifications. They do not include column numbers or column heading information. If you need to determine the location of a specific entry on the listings, use the back of the Z- and A- specifications included in Appendix A to locate the column number and then match that column number on the listing with the appropriate description on the back of the specifications. The top markings on the backs of the specifications match the characters on the listings. The bottom markings on the backs of the specifications should not be used for reading the listings in this manual.

The following example illustrates how to use the listings and specifications in the manner described:

1. Place the top of the Z-specification (back side) under the Z-specification line in question. Align the leftmost side of the listing with the leftmost character mark on the specification.
2. Locate the column number that identifies the character(s) you want to identify (in this example, the characters are H2) **1**.
3. Match the column number with a column number heading on the back side of the specification **2**. The description explains the entry.

**Note;** The C-specification does not have descriptive information. To determine the meaning for a particular column entry, find the column number in the manner previously described and use the *DE/RPG Reference Manual* to locate the column number heading and its description.

The backs of the Z- and A-specifications are included in case you do not have pads of specifications available. If you need to use these forms, tear them out of the manual.

```

00001Z*****
00002Z* PROGRAM 77.
00003Z*****
00004ZJ COUNTX                               TFILE(DISKET)
00005Z H2EXMPLE           1E
00006Z                   R                   H2

```



- |   |  |
|---|--|
| <p>1-5 Identifies the source statement order.</p> <p>6 Identifies the type of source statement.</p> <p>7 Names the type of source statement:<br/>         *—User comment<br/>         J—Job specification<br/>         blank—Format specification</p> <p>8-9 The identification associated with this format:<br/>         1 through 9—A single numeric character ID.<br/>         A0 through Z9—A two-character ID consisting of an alphabetic character followed by a numeric character.</p> <p>10-17 The name used to:<br/>         — identify the job (J in column 7).<br/>         — identify the format or subroutine (blank in column 7).<br/>         These columns are not used if column 21 contains an R.</p> <p>18-19 Reserved.</p> <p><b>Note:</b> Columns 20-54 are not used if column 7 contains a J.</p> <p>20 Specifies the number of times the format is repeated before the next format is used:<br/>         1 through 9—Repeat the format for the specified number of times unless the SEL FMT or NEXT FMT key is pressed.<br/>         blank or N—Repeat the format until the SEL FMT or NEXT FMT key is pressed.</p> <p>21 Specifies how the format is used:<br/>         E—(Entry) used to enter and display data.<br/>         R—(Review) used to select a format for scan, update, or verify of existing records.</p> <p>22-37 Used for logical selection of a format. Multiple tests are allowed. In enter mode, the format selected is used to format the <i>next</i> record entered. In review mode, the format selected is used to display the <i>current</i> record.</p> <p>22 In review mode (column 21 contains an R), an A specifies the <i>ending</i> of two characters in the data record to create a unique record identifier.</p> <p>23-30 *POSnnnn identifies the position in the data record to be tested, where nnnn is a numeric value from 1 to 1024.</p> | <p>31-32 Reserved.</p> <p>33-34 The characters EQ or blank when a character to test for is specified in position 35-37.</p> <p>35-37 Specifies the character that controls format selection if it matches the character in the data record.</p> <p>38-44 Reserved.</p> <p><b>2</b> 45-46 Specifies the identification of the format used for the entry or display of the next record. If columns 22-37 are specified, the format is selected when a match occurs. If columns 22-37 are not specified in enter mode (E in column 21), the format is selected when the repeat count (column 21) is met or the NEXT FMT key is pressed. If columns 22-37 are not specified in review mode (R in column 21), the format is selected if no previous match occurs.</p> <p>47-54 Reserved.</p> <p>55-80 Keywords that specify information used for jobs or formats:<br/> <b>JOB specifications (J in column 7):</b><br/>         CFILE (data set)—Includes the COPY function in the job. The parameter data set is the data set name from which records will be copied.<br/>         DATE(*DMY/*YMD)—The format of the date available in UDATE. The default is *MDY, where M = month, D = day, and Y = year.<br/>         EDITC(cuptd)—Five characters that define the editing control for output fields, where:<br/>         — cu is a two-character currency symbol (default = b\$).<br/>         — p is the decimal point character (default = .).<br/>         — t is the thousand separator character (default = ,).<br/>         — d is the date separator character (default = /).<br/>         The system default for this option is b\$.:/ if EDITC is not specified.<br/>         ENTRATR (attr...)—Specifies the attributes that are applied to all input/both fields only when the fields are being entered, where attr is:<br/>         BL (blink)<br/>         CS (column separators)<br/>         HI (high intensity)<br/>         ND (nondisplay)<br/>         RI (reverse image)<br/>         UL (underline)<br/>         A combination of attributes can also be used.<br/>         EXITATR (attr...)—Specifies the attributes that are applied to all input/both fields after the fields have been entered. See the ENTRATR for a description of the attr parameter.</p> |
|---|--|





## Part 1. Creating Data Sets

This part of the manual contains general information about creating diskette data sets using DE/RPG; it consists of two chapters:

- Chapter 1. Characteristics of Data Entry Programs That Use Transaction Files
- Chapter 2. Characteristics of Application Programs That Use Subroutines on the C-Specifications

DE/RPG uses two basic kinds of programs to create data sets: (1) data-entry programs and (2) application programs. Application programs can further be subdivided into interactive and noninteractive programs.

Data-entry programs are primarily for transcribing data from an existing source into data sets on the diskette; they normally have limited preprocessing requirements. This type of program allows interactive entry and provides data checks and edits required for accurate entry. Data-entry programs also provide operator-controlled keyboard functions that enhance the entry process.

Application programs are primarily for processing data. This type of program does not generally require extensive operator entry. It often uses existing data as input and performs automatic operations against the data with limited or no operator interaction. Application programs normally process data rather than create it. Interactive application programs can be used when a combination of data-entry and processing functions is required.

The samples in Figure 1-1 illustrate the two basic program types. Sample A is a data-entry program, Sample B is an interactive application program, and Sample C is a noninteractive application program.

The displays and diskette data sets that result from using the programs are also illustrated in the figure. Numeric keys **1** indicate the parts of the program that produce the matching displays or data sets.

#### Sample A

```

00001Z*****
00002Z* PROGRAM 1. FIGURE 1-1 SAMPLE A IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ SAMPLE1 TFILE(EXDATAST) 5
00005Z A1BEGIN 1E A2
00006Z A2END 1E A1
00007Z R *POS1 'B' A1
00008Z R *POS1 'E' A2
00009A F INPUT 37 DEVICE(CRT) DSPSIZ(6 80)
00010A R BEGIN
00011A FLD1 1 I 1 INSERT('B')
00012A FLD2 30 I 2 PMT(CUSTOMER NAME)
00013A FLD3 6 I 2 PMT(NUMBER)
00014A R END
00015A FLD4 1 I 3 INSERT('E')
00016A FLD5 6 2I 4 PMT(PAYMENT)
00017A FLD6 9 2I 4 PMT(OUTSTANDING BALANCE)
00018A F EXDATAST 37 DEVICE(DISK X'4000')
00019

```

#### Sample B

```

00001Z*****
00002Z* PROGRAM 2. FIGURE 1-1 SAMPLE B IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ SAMPLE1
00005Z A1SUBCON 1E A1
00006A F INPUT 37 DEVICE(CRT) DSPSIZ(6 80)
00007A R BEG
00008A FLD1 1 I 1 INSERT('B')
00009A FLD2 30 I 2 PMT(CUSTOMER NAME)
00010A FLD3 6 I 2 PMT(NUMBER)
00011A R ENDING
00012A FLD4 1 I 3 INSERT('E')
00013A FLD5 6 2I 4 PMT(PAYMENT)
00014A FLD6 9 2I 4 PMT(OUTSTANDING BALANCE)
00015A F EXDATAST 37 DEVICE(DISK X'4000')
00016A R BEGIN
00017A FLD2 1
00018A FLD3 31
00019A FLD1 37
00020A R END
00021A FLD6
00022A FLD4
00023A FLD5
00024C SUBCON BEGSR
00025C EXFMTBEG
00026C EXFMTENDING
00027C WRITEBEGIN 6
00028C WRITEEND
00029C ENDSR

```

Figure 1-1 (Part 1 of 4). Program Types

Sample C

```

00001Z*****
00002Z* PROGRAM 3. FIGURE 1-1 SAMPLE C IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ NONINT
00005Z A1OUT          1E                      EOJ
00006A          F MAST1          37          DEVICE(DISK X'4000')
00007A          R BEG
00008A          FLD1             1
00009A          FLD2            30
00010A          FLD3             6
00011A          F MAST2          16          DEVICE(DISK X'4000')
00012A          R ENDING
00013A          FLD4             1
00014A          FLD5             6
00015A          FLD6             9
00016A          F EXDATAST       37          DEVICE(DISK X'4000')
00017A          R BEGIN
00018A          FLD2
00019A          FLD3
00020A          FLD1
00021A          R END
00022A          FLD6
00023A          FLD4
00024A          FLD5
00025C          OUT             BEGSR
00026C          LOOP            TAG
00027C          READ BEG                      01
00028C          READ ENDING                  02
00029C          N01  N02          WRITEBEGIN 6
00030C          N01  N02          WRITEEND
00031C          N01  N02          GOTO LOOP
00032C          ENDSR

```

Figure 1-1 (Part 2 of 4). Program Types

**1** 0 0001      A 40  
CUSTOMER NAME  
B.....

**2** 0 0001      A 40  
NUMBER  
B\*\*\*\*\*.....

**3** 0 0001      A 40  
PAYMENT  
E.....

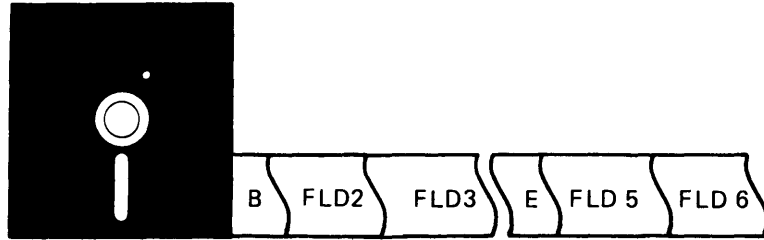
**4** 0 0001      A 40  
OUTANDING BALANCE  
E\*\*\*\*\*.....

\*indicates data

... indicates the length of an unfilled field

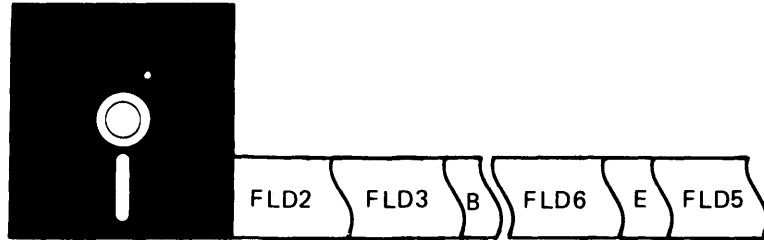
Figure 1-1 (Part 3 of 4). Program Types

5



Data Set Resulting from the Program in Sample A

6



Data Set Resulting from the Programs in Sample B and Sample C

Figure 1-1 (Part 4 of 4). Program Types

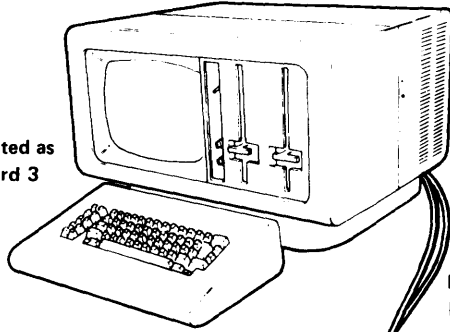
The data sets that result from the programs in Figure 1-1 contain the same fields but in a different sequence. The sequence of the fields in the data sets is not an important difference in the programs. It is possible for the program in Sample A to produce a data set exactly like that produced by the program in Sample B or Sample C. The important difference in the programs is the way that they are executed by the operator. Although the programs produce the same displays (except for Sample C which has no display) and similar data sets, their execution is different.

This part of the manual describes the differences in program types offered by DE/RPG. To understand these differences, you must first understand some basic information about the way the IBM 5280 system works.

The IBM 5280 system uses partitioned areas of user storage for the execution of DE/RPG programs. Figure 1-2 illustrates this storage area division.

The IBM 5281 Data Station

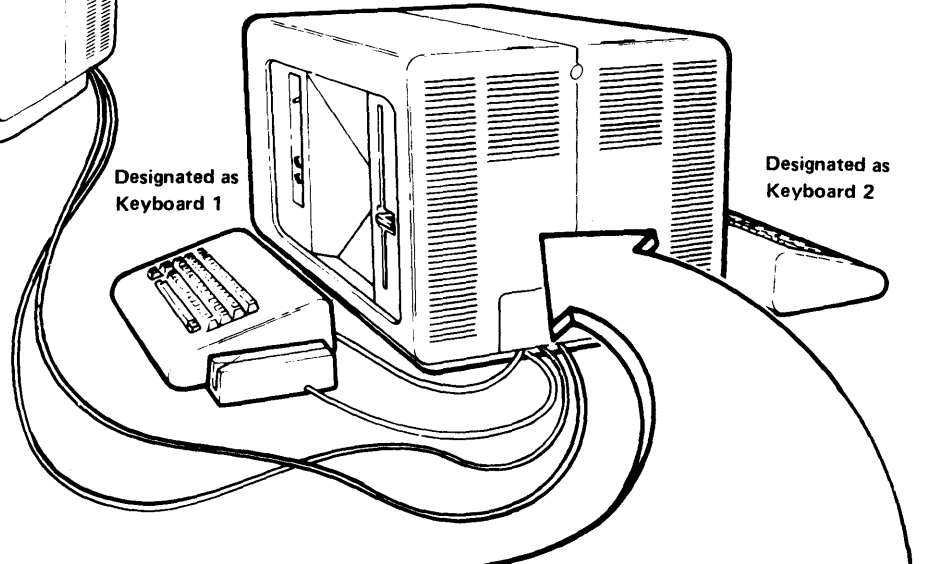
Designated as  
Keyboard 3



The IBM 5286 Dual Programmable  
Data Station

Designated as  
Keyboard 1

Designated as  
Keyboard 2



Partitioned Main Storage  
for the IBM 5286

Foreground Partitions

Common Area 15 K Bytes
First partition linked to Keyboard 1 10 K Bytes <i>A data-entry program requiring operator entries is in this partition.</i>
Second partition linked to Keyboard 2 11 K Bytes <i>A data-entry program requiring operator entries is in this partition.</i>
Third partition linked to Keyboard 3 12 K Bytes <i>An interactive application program is in this partition. It requires operator interaction.</i>
Background partition 16 K Bytes <i>A noninteractive application program is in this partition. It requires only limited operator interaction.</i>

Figure 1-2. A Sample Partition Division of Main Storage



There are two kinds of partitions: foreground and background. Each data station has its own foreground partition. The background partitions are not associated with a particular data station; they can be used by any operator using any data station in the system.

All programs that require extensive operator interaction should be used in a foreground partition. The foreground partition has control of the keyboard and display until it temporarily relinquishes its control to a background partition upon request from a program in the background partition. Control returns to the foreground partition once the requirements of the background partition have been met.

Programs that can execute without keyboard input can be loaded into a background partition and can be executed concurrently with a program in a foreground partition. A background partition is an area of user storage that is attached to the keyboard/display only when keyboard input is needed by the program and only as allowed by the operator of the data station. Application programs may be executed in either foreground or background partitions depending on whether or not they will require operator attention during the execution of the program.

Figure 1-3 illustrates the differences between program types in DE/RPG.

**Interactive Data-Entry Program with no Subroutines on the C-Specification**

- I/O control is accomplished through TFILE on the Z-specification.
- Keyboard functions such as AUXST, AUXDUP, and CHECK(AD) and CHECK(AS) are available.
- All keyboard modes are operational (enter, update, verify, and rerun).
- Records are automatically written into a transaction data set as they are completed (unless writing is suppressed).
- Data set organization is always sequential as entered.
- Limited resequencing of fields in a record is available.

**Interactive Application Program with Subroutines on the C-Specification**

- I/O control is accomplished through EXFMT, READ, CHAIN, and WRITE on the C-specification.
- Keyboard functions such as AUXST, AUXDUP, and CHECK(AD) and CHECK(AS) are available.
- The execute mode is operational.
- Records are written into a named data set only when specified by a WRITE (or UPDAT) on the C-specification.
- Data set organization can be sequential as entered, or by key, or by index.
- Complex resequencing of fields in multiple records is available.

**Noninteractive Application Program with Subroutines on the C-Specification**

- I/O control is accomplished through READ, CHAIN, and WRITE on the C-specification.
- Not applicable.
- Not applicable.
- Records are written into a named data set only when specified by a WRITE (or UPDAT) on the C-specification.
- Data set organization can be sequential by key, or by index.
- Complex resequencing of fields in multiple records is available.

**Figure 1-3 (Part 1 of 3). Comparison of Functions Based on Program Type**

<b>Interactive Data-Entry Program with no Subroutines on the C-Specification</b>	<b>Interactive Application Program with Subroutines on the C-Specification</b>	<b>Noninteractive Application Program with Subroutines on the C-Specification</b>
<ul style="list-style-type: none"> <li>• Program termination is through the use of the End of Job key or by EOJ on the entry format line.</li> </ul>	<ul style="list-style-type: none"> <li>• Program termination is conditioned in the subroutine and controlled by EOJ on the entry format line.</li> </ul>	<ul style="list-style-type: none"> <li>• Program termination is conditioned in the subroutine and controlled by EOJ on the entry format line.</li> </ul>
<ul style="list-style-type: none"> <li>• Display format control is accomplished by testing and the next format ID on the Z-specification.</li> </ul>	<ul style="list-style-type: none"> <li>• Display format control is accomplished by EXFMT and conditioning indicators on the C-specification.</li> </ul>	<ul style="list-style-type: none"> <li>• Not applicable.</li> </ul>
<ul style="list-style-type: none"> <li>• CLRL is allowed to retain data from a previous record on the display.</li> </ul>	<ul style="list-style-type: none"> <li>• CLRL is not allowed.</li> </ul>	<ul style="list-style-type: none"> <li>• Not applicable.</li> </ul>
<ul style="list-style-type: none"> <li>• EDTCDE for the display is not allowed.</li> </ul>	<ul style="list-style-type: none"> <li>• EDTCDE for the display is allowed to provide punctuation editing (output fields only).</li> </ul>	<ul style="list-style-type: none"> <li>• Not applicable.</li> </ul>
<ul style="list-style-type: none"> <li>• CFILE for manual copying of records is available.</li> </ul>	<ul style="list-style-type: none"> <li>• CFILE is not available.</li> </ul>	<ul style="list-style-type: none"> <li>• Not applicable.</li> </ul>
<ul style="list-style-type: none"> <li>• PRTFILE for manual unformatted printing of records is available.</li> </ul>	<ul style="list-style-type: none"> <li>• PRTFILE is not available.</li> </ul>	<ul style="list-style-type: none"> <li>• PRTFILE is not available.</li> </ul>
<ul style="list-style-type: none"> <li>• No formatted printing is allowed.</li> </ul>	<ul style="list-style-type: none"> <li>• Formatted printing is allowed.</li> </ul>	<ul style="list-style-type: none"> <li>• Formatted printing is allowed.</li> </ul>
<ul style="list-style-type: none"> <li>• No communications support is provided.</li> </ul>	<ul style="list-style-type: none"> <li>• Communications is provided for remote sending and receiving of data.</li> </ul>	<ul style="list-style-type: none"> <li>• Communications is provided for remote sending and receiving of data.</li> </ul>

**Figure 1-3 (Part 2 of 3). Comparison of Functions Based on Program Type**

**Interactive Data-Entry Program with no Subroutines on the C-Specification**

- Indicator control is limited to conditioning error messages and CHECK(BY).
- Updating records is accomplished manually through the update mode.
- Deleting records is accomplished manually through the key sequence.
- Inserting records is accomplished manually through the key sequence.
- The RANGET, XCHK, LOOK, and SUBST table functions are available.

**Interactive Application Program with Subroutines on the C-Specification**

- Indicator control can condition a variety of operations on the C-specification.
- Updating records is accomplished through the UPDAT statement on the C-specification.
- Deleting records is accomplished automatically through the DELET statement on the C-specification.
- Not applicable.
- The RANGET, XCHK, LOOK, SUBST, and LOKUP table functions are available.
- The LOKUP table function and the use of the *arrayname, index* combination as a variable field is available.

**Noninteractive Application Program with Subroutines on the C-Specification**

- Indicator control can condition a variety of operations on the C-specification.
- Updating records is accomplished through the UPDAT statement on the C-specification.
- Deleting records is accomplished automatically through the DELET statement on the C-specification.
- Not applicable.
- The LOKUP table function and the use of the *arrayname, index* combination as a variable field is available.
- \*\* When tables are used on the C-specification they are considered RPG arrays and not RPG tables.

**Figure 1-3 (Part 3 of 3). Comparison of Functions Based on Program Types**



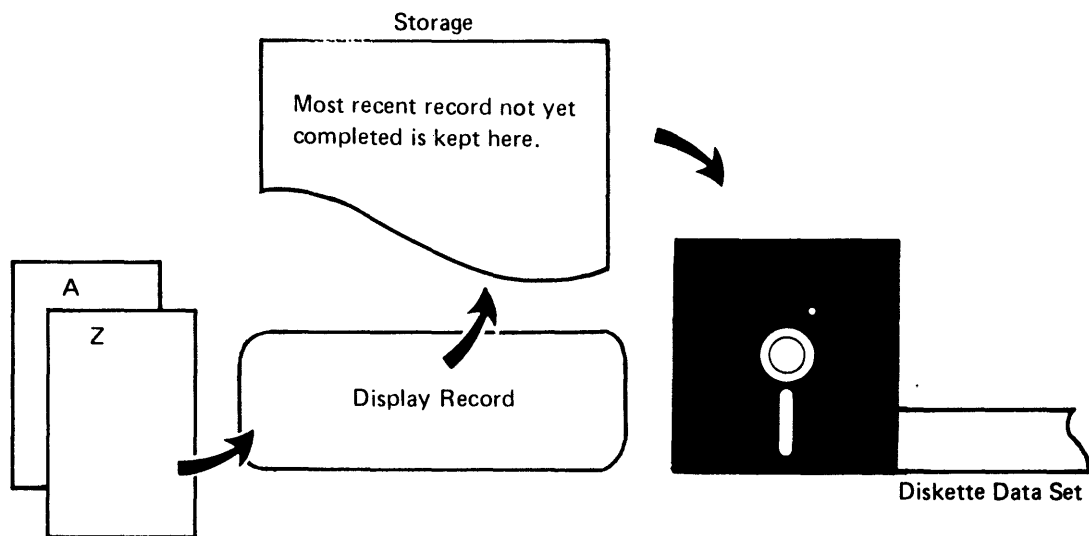
## Chapter 1. Characteristics of Data-Entry Programs that Use Transaction Files

Programs that are created exclusively for data-entry purposes are created by using transaction files. Transaction files include the use of a special keyword (TFILE) on the Z-specification; the TFILE keyword evokes data-entry functions not otherwise available to the operator. The sample in Figure 1-4 illustrates the steps involved in a program that uses a transaction file.

```

00001Z*****
00002Z* PROGRAM 4. FIGURE 1-4 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ SAMPLE1 TFILE(EXDATAST) 1
00005Z A1BEGIN 2 1E A2
00006Z A2END 5 1E A1
00007Z R *POS1 'B' A1
00008Z R *POS1 'E' A2
00009A F INPUT 37 DEVICE(CRT) DSPSIZ(6 80)
00010A 3 R BEGIN
00011A FLD1 1 I INSERT('B')
00012A FLD2 30 I PMT(CUSTOMER NAME)
00013A FLD3 6 I PMT(NUMBER)
00014A 6 R END
00015A FLD4 1 I INSERT('E')
00016A FLD5 6 2I PMT(PAYMENT)
00017A FLD6 9 2I PMT(OUTSTANDING BALANCE)
00018A 4 F EXDATAST 37 DEVICE(DISK X'4000')
00019

```



Program Using a Transaction File

1. Specify a name for the transaction file to use. 1
2. Specify the first entry format to be used for the display. 2  
The record for this must be defined on the A-specification. 3
3. When the record is complete, it is automatically written in the transaction data set. 4  
The data set name used with TFILE must match a file line on the A-specification.
4. Specify the second entry format to be used for the display. 5  
The record for this format must be defined on the A-specification. 6
5. When the record is completed, it is automatically written in the transaction data set. 4  
The name on the file description line for the diskette must match the name used with TFILE.
6. The program is complete. The operator can choose either format by a key sequence (FMT SEL and ID number). The program can be terminated if the operator uses the End of Job key.

**Note:** If the operator wants to verify or update a record or add to the data set, the correct format is automatically selected.

Figure 1-4. Characteristics of a Data-Entry Program

Programs using transaction files have the following characteristics:

- The enter, verify, update, and rerun modes and numerous key-initiated functions such as auto record advance are valid.
- All programmed functions which are related to the keyboard are valid (this includes keywords such as AUXDUP and AUXST and keyboard functions such as auto dup, and auto skip).
- Writing records in the diskette data set is automatically accomplished at the completion of each entry format unless the writing of the record is suppressed. Part 2 describes this topic in detail.
- Special job functions such as manually copying records from other data sets (CFILE) and unformatted printing (PRTFILE) are available. Part 6 provides detailed information about unformatted printing.
- Limited record reformatting is available. You can only reformat fields within a single record, and you must use all fields in the original record. Part 2 describes this topic in detail.
- Automatic format chaining and operator-controlled format selection is available. Part 2 describes this topic in detail.
- All input/output control for the transaction data set is automatic (in other words, the operator cannot control the input/output devices through the transaction file).

Whenever a transaction file is used to create a data set, the result is a data set that contains sequentially written records. Updating the data set is accomplished when the operator initiates the update mode through the keyboard. The status line on row 1 of the display informs the operator of the current mode.



## **MODES OF OPERATION AVAILABLE FOR PROGRAMS USING TRANSACTION FILES**

The basic modes that are available for data-entry programs are enter, update, verify, and rerun. Within the data-entry program, the entry format statement and the review format statement on the Z-specification specify the record arrangement to use for the modes. The entry format determines the record arrangement to use during the enter mode. The review format determines the record arrangement to use during the update, verify, and rerun modes.

The enter mode of operation allows an operator to provide initial data entries into the record or to add new records to the data set using the entry format display as a guide. The update mode allows the operator to change data in a record. The verify mode helps the operator check the accuracy of the initial entry by allowing him to reenter the data without seeing the initial entry; upon the completion of the verified entry, the initial entry is displayed and an error is flagged if the entries do not match. The operator is given the choice of actions to take: either to correct the initial entry or to accept it. The rerun mode automatically cycles the program through its automatic functions and calculations. One use for this mode is to update all references to a field that has been duplicated and now needs to be corrected. By correcting the initial entry of the field and then using the rerun mode, all fields in which the initial entry had been duplicated now contain the corrected value.

### **Manual Selection of an Operating Mode**

The operator selects the initial operating mode through a keyboard entry in response to the mode selection menu. The format to be used for the modes can be selected either automatically through the next-format sequence defined on the Z-specification or by manual operator selection.

The following text illustrates how an operator selects a mode for a data-entry program and how the appropriate format is then selected either by the program or by the operator.

Once a data-entry program has been written, compiled, and brought into a foreground partition, the following display appears.

**Note:** Values that were present in the program appear on the display. The displays show sample values.

Name of DE/RPG  
Object Program

```
0 0001      A 21 40
Program name: SYSSEP
Device address: 4400
Partition number: ---

Press ENTER
```

05-00

When the Enter key is pressed, the mode selection menu appears.

```
0 0001      D 01 40
Select Initial Data Entry Mode
Options are:
  1. Enter-New/REPLACE  3. Verify      5. Rerun
  2. Update              4. Enter-ADD

Select Option: 1 Press ENTER
```

06-81

Once the mode has been selected, the display advances to the data set open prompt.

```
Q 0030      N 01 F0      E
Enter data for data set open
Data set name:  MASTER
Device address:  4400

Press ENTER
```

06-82

When the values for the display fields have been accepted or supplied and the Enter key has been pressed, the file is automatically opened. The operation of the program named in the first display takes control.

## Automatic Selection of the Correct Format for the Manually Selected Mode

The entry and review statements on the Z-specification determine the format. The sample in Figure 1-5 illustrates the coding required for the entry and review modes.

```
00001Z*****
00002Z* PROGRAM 5. FIGURE 1-5 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ SAMPLE TFIL(EXP1)
00005 1A1ONE 1E A2 SLNO(3)
00006 A2TWO 1E A1 SLNO(3)
00007Z R *POS31 '0' A1 2
00008Z R *POS31 'T' A2
00009A F DISP1 31 DEVICE(CRT) DSPSIZ(6 80)
00010A R ONE
00011A FLD1 15 I001001DSPATR(UL)
00012A FLD2 15 I002001DSPATR(UL)
00013A FLD3 1 I003001INSERT('0')
00014A R TWO
00015A FLD4 10 I001001DSPATR(UL)
00016A FLD5 10 I001030DSPATR(UL)
00017A FLD6 10 I001050DSPATR(UL)
00018A FLD7 1 I001080INSERT('T')
00019A F EXP1 31 DEVICE(DISK X'4000')
00020
00021
00022
00023
00024
00025
```

Figure 1-5. A Program Illustrating the Specification of the Entry and Review Modes

Formats ONE and TWO (with format IDs A1 and A2 respectively) are specified in the entry format statements for the enter mode 1; format IDs A1 and A2 are specified in the review format statements 2 for the update, rerun, and verify modes. Whenever the operator selects option 1 or option 5 on the mode selection menu, format ONE is displayed first.

The following display illustrates the entry format named ONE.

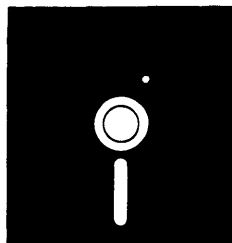
O 0001      A 40  
 -----  
 -----  
 O **1**

Format ONE is displayed once, then the next entry format (TWO) is displayed.

O 0001      A 40  
 -----  
 -----  
 -----  
**2** T

When the data for format TWO has been entered, format ONE is redisplayed. This process of displaying format ONE and then format TWO continues until the operator stops the job by using the End of Job key.

Whenever the operator selects one of the review mode options, format ONE or TWO is automatically selected for the display based on the entry in the test position of the current diskette record. If an O **1** is in the test position, format ONE is displayed for the rerun, update, or verify modes. If a T **2** is in the test position, format TWO is displayed for the rerun, update, or verify modes.



When this is the current record, the format for ONE is used.

When this is the current record, the format for TWO is used.



**Note:** The review format and the entry in the usage column on the A-specification for the field determine whether the fields in the record are displayed. The operator manually selects the mode in which to operate, and the program automatically selects the appropriate format for the mode.

## KEY-INITIATED MODES

In addition to the basic data-entry modes (enter, update, verify, and rerun), several key-initiated modes are available for programs using transaction files. The *IBM 5280 DE/RPG Reference Manual* contains detailed descriptions of all keys and modes. Briefly, the key-initiated modes offered by DE/RPG are:

- Update search (U-S)
- Update insert (U-I)
- Verify correct (V-C)
- Verify insert (V-I)
- Verify search (V-S)
- Verify display (V-D)
- Copy (C)
- Copy search (C-S)
- Copy transfer (C-T)
- Rerun display (R-D)
- Print (P)

The designations in parentheses are the codes used to represent the modes in positions 35 through 37 of the status line. The designations for the four major modes are enter (E), update (U), verify (V), and rerun (R).

The following list names the keys that are active during data-entry programs and indicates in which modes they can be used:

Function	Modes														
	E	C	C-S	C-T	P	R	R-D	U	U-I	U-S	V	V-C	V-D	V-I	V-S
Attention	X	X	X			X	X	X	X	X	X	X	X	X	X
Auto Duplicate/Skip	X	X			X			X	X		X				
Auto Enter	X	X			X			X			X	X			
Cancel			X		X				X	X				X	X
Character Advance	X	X						X	X		X	X		X	
Character Backspace	X	X						X	X		X	X		X	
Character Delete	X	X						X	X			X		X	
Character Insert	X	X						X	X			X		X	
Clear Screen	X	X						X	X						
Cursor Down	X	X						X	X			X		X	
Cursor Left	X	X						X	X		X	X		X	
Cursor Up	X	X						X	X		X	X		X	
Cursor Right	X	X						X	X	X			X	X	
Duplicate	X	X						X	X		X	X		X	
Edit Release	X	X					X	X	X		X	X		X	
End of Job	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Erase Input	X	X			X			X	X		X	X		X	
Field Advance	X	X						X	X		X	X		X	
Field Backspace	X	X						X	X		X	X		X	
Field Correct											X				
Field Exit	X	X						X	X		X	X		X	
Field Exit Minus	X	X						X	X		X	X		X	X
Help	X	X						X	X			X		X	
Hexadecimal	X	X						X	X		X	X		X	
Home (Record Backspace)	X	X	X					X	X	X	X	X			X
Mark Field	X	X					X	X	X		X	X		X	
New Line	X	X						X	X			X			
Next Format	X	X						X	X		X	X		X	
Page Forward		X						X			X				
Print	X	X					X	X	X		X	X	X		
Record Advance	X	X	X		X			X	X	X	X	X		X	X
Record Backspace (Home)	X	X	X					X		X	X	X			X
Record Correct											X				
Record Delete	X							X	X		X				
Record Display											X		X		
Record Insert								X			X				
Record Transfer		X	X												
Reset	X	X					X	X	X		X	X		X	
Review File	X														
Search Content		X						X			X				
Search End-of-Data		X						X			X				
Search Relative Record		X						X			X				
Search Sequential Content		X						X			X				
Select Format	X	X						X	X		X	X		X	
Skip	X	X						X	X		X	X		X	
System Request	X	X	X			X	X	X	X	X	X	X	X	X	X

To compare the keys that are active during the execution of a data-entry program with keys that are active during the execution of an application program, see the topic *Execute Mode—The Only Mode Available for Use with Application Programs* in Chapter 2.

## PROGRAM FUNCTIONS USED WITH TRANSACTION FILES

Program functions that facilitate entry are available to transaction files. These functions use keywords such as AUXDUP, AUXST, CHECK(AD), and CHECK(AS). These functions rely upon operator controlled switches. For example, when the current field statement contains AUXDUP(named field), the named field is automatically duplicated into the current field when the Auto Dup function is active. These program functions provide additional keyboard control for the data-entry operator.

## SPECIAL JOB FUNCTIONS FOR USE WITH TRANSACTION FILES

Two job functions can be specified for programs using transaction files by including the CFILE and PRTFILE keywords. Each of these functions must be used in conjunction with the TFILE keyword in the job specification statement.

### Manually Copying Records for Programs Using Transaction Files

CFILE allows the operator to manually perform copy functions using the search and copy keys. The function provided by CFILE is not to be confused with the copy utility. The sample in Figure 1-6 illustrates a program that uses the CFILE function. CFILE **1** names the data set that is to be copied. When CFILE is used, the data set must be named in a file description statement **2** for the diskette. Both data sets (the one being copied from and the one being copied to) must have the same record length.

```

00001Z*****
00002Z* PROGRAM 6. FIGURE 1-6 IN THE DE/RPG USER'S GUIDE *
00003Z***** 1
00004ZJ INVOICE TFILE(INTERBIL) CFILE(CUS+
00004Z TMAST)
00005Z A1INSTRUC 1E A2 WRITE(*NO)
00006Z A2PURHCAS NE A1 SLNO(3)
00007Z R A2
00008A F EXMP 80 DEVICE(CRT) DSPSIZ(6 80)
00009A R INSTRUC
00010A 0002001'FIND THE CUSTMAST RECORD THAT MATC+
00010A HES THE ORDER. USE CUSTMAS DATA SE+
00010A T.'
00011A 1 I001001CHECK(FE) PMT(USE THE FIELD EXIT KE+
00011A Y TO CONTINUE)
00012A R PURHCAS
00013A 1 I001001INSERT('T')
00014A 0002001'ITEM'
00015A 0003001'PRICE'
00016A 0004001'QUANT'
00017A 6 I002010
00018A 5 2I003010
00019A 4 0I004010
00020A F INTERBIL 80 DEVICE(DISK X'4000')
00021A 2 F CUSTMAST 80 DEVICE(DISK X'4000')
00022
00023

```

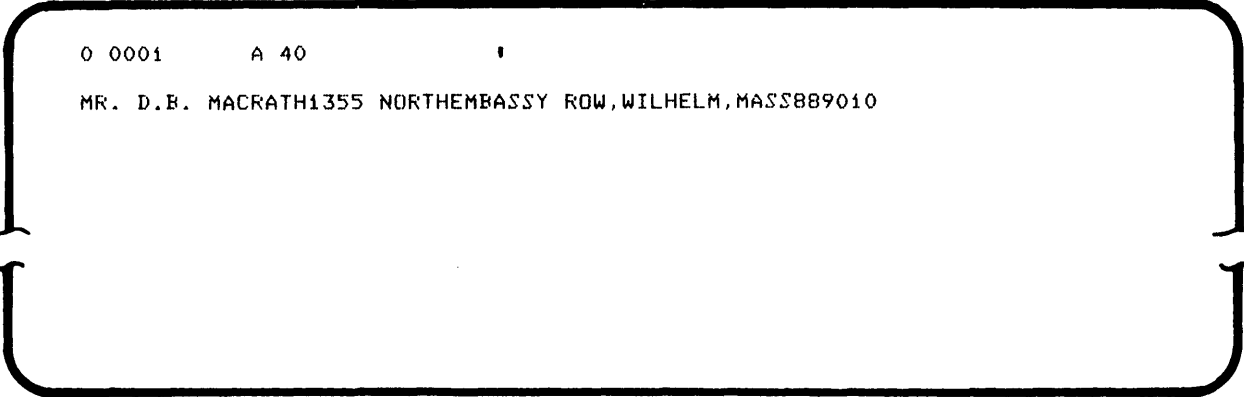
Figure 1-6. A Program Illustrating the Use of the Copy Function

In order to use the copy-related keys, the program must be in the enter mode; that is, the operator must have selected option 1 Enter-NEW/REPLACE or 5 Enter-ADD from the mode selection menu. Copy-related keys are keys that are operational when the copy function is in effect. The copy-related keys are Review Second Data Set, Transfer Record, and Return to Transaction Data Set.

Once in an enter mode, the operator initiates the manual copy function by using the Review Second Data Set key. The process involved in manually copying records using the copy keys is given in the following list.

**Note:** The search keys can also be used to identify the desired record to be copied. The *IBM 5280 Operator's Guide* contains a detailed description of the available operations for the copy mode.

1. After the operator has pressed the Review Second Data Set key, a display appears requesting the name of the data set from which to copy records. Using the sample in Figure 1-6 as an example, the operator enters CUSTMAST.
2. The records (starting with the first one in the named data set) are shown using display format 0. For example, the first record in the CUSTMAST data set might appear as follows:



```
0 0001      A 40      '
MR. D.B. MACRATH1355 NORTHEMBASSY ROW,WILHELM,MASS889010
```

If this is the record the operator wants to copy, he can change any data on the record before copying it. When changes are made, the updates and changes do not appear on the original record (in the data set being copied from); the updated data only appears in the copied record.



3. By pressing the Transfer Record key, the record is copied into the receiving data set. The data set is extended to accept the copied record.
4. The next sequentially encountered record in the data set being copied from is displayed. If the operator does not want the record copied, he presses the Enter key to display the next record in the data set. When the operator wants to terminate the manual copy operation, he presses the Return to Transaction Data Set key and the program returns to the enter mode.

The normal operation of the program resumes. The next format is automatically displayed and processing continues. The sample in Figure 1-7 illustrates the data set that might result from using the sample in Figure 1-6.

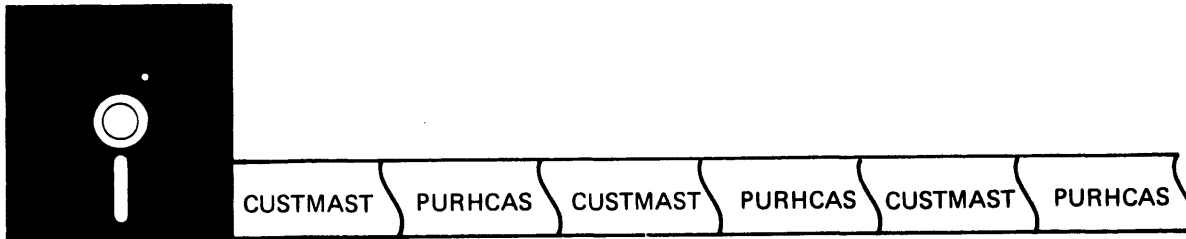


Figure 1-7. Data Set Resulting from Using the Program in Figure 1-6.

This process is not necessarily the most desirable way to merge records from two different data sets. It is simply an available technique for programs using transaction files. Consider the impact this function has on the review operating modes. For example, unless the record being copied and the records in the receiving data set have the same display format, the operator has difficulty recognizing which record type is being displayed. Formatting techniques for overcoming this situation are described in Part 2.

## HOW THE USAGE COLUMN ENTRIES AFFECT PROGRAMS USING TRANSACTION FILES

The usage entry is in column 38 on the A-specification. Its purpose is to specify how fields are to be processed. Four entries are possible in this column for programs using transaction files: I for input, O for output, B for both input and output, and W for work space.

Fields designated as I in the enter mode initially display as blanks until the operator enters data. In the update mode, the existing data is displayed and the operator is allowed to alter the data. In the verify mode, the field is initially blank until the operator enters the verify data; then, the characters are displayed and any errors (mismatches) are indicated. All input fields are included in records written to the diskette data set.

Fields designated as O can be used for literal messages and are to be excluded from the diskette data set. Literals are displayed immediately in all modes. All output fields are excluded from records that are written in the diskette data set.

Fields designated as B are treated as input fields except during execute mode. During execute mode, the initial value of the field is displayed at the start of the format. The operator may change the data in a key-entered field or accept the data as displayed. All fields described as input and output (B in the Usage column) must be named.

Fields designated as W are neither displayed nor included in the diskette data set. This type of field is normally used to hold intermediate calculation results or to declare a variable to be used in another operation.

This completes the description of the characteristics of data-entry programs using transaction files. The next chapter describes the characteristics of application programs.



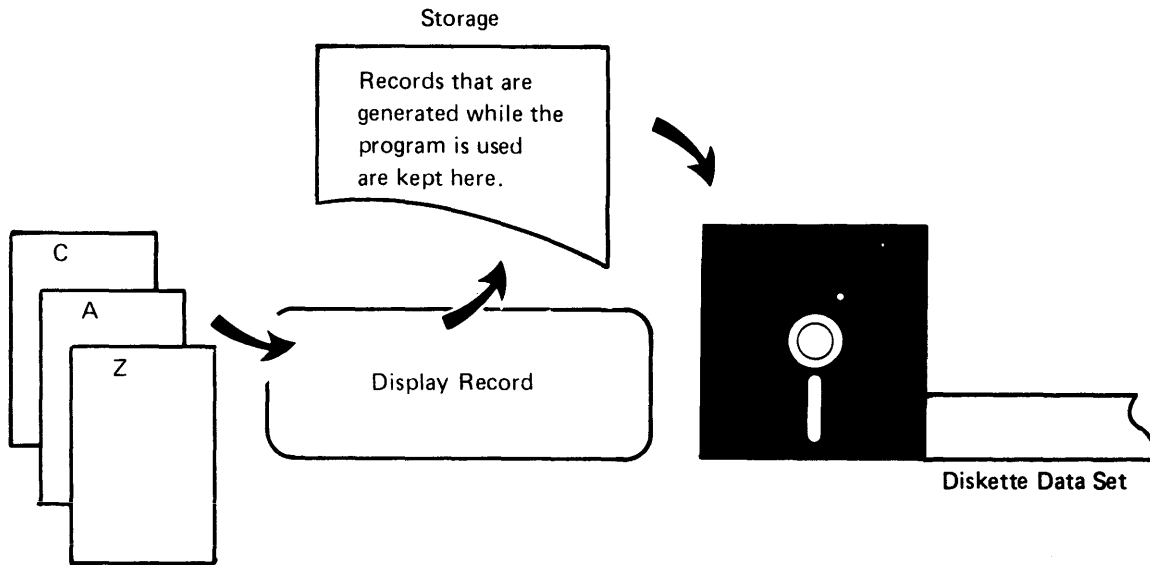
## **Chapter 2. Characteristics of Application Programs that Use Subroutines on the C-Specifications**

In general, application programs that use subroutines on the C-specifications do not use transaction files. The path this type of program takes is much different from that taken by programs that use transaction files. All input/output operations are controlled by the programmer; no default display or diskette operations are available. The sample in Figure 2-1 indicates the steps involved in using a simple application program.

```

00001Z*****
00002Z* PROGRAM 7. FIGURE 2-1 IN THE DE/RPG USER'S GUIDE
00003Z*****
000(1) J SAMPLE1
000(2) A1SUBCON          1E          A1
00006A          F INPUT          37          DEVICE(CRT) DSPSIZ(6 80)
00007A          4 R  BEG
00008A          FLD1             1  I        INSERT('B')
00009A          FLD2             30 I        PMT(CUSTOMER NAME)
00010A          FLD3             6  I        PMT(NUMBER)
00011A          6 R  ENDING
00012A          FLD4             1  I        INSERT('E')
00013A          FLD5             6  2I       PMT(PAYMENT)
00014A          FLD6             9  2I       PMT(OUTSTANDING BALANCE)
00015A          F  EXDATAST      37          DEVICE(DISK X'4000')
00016A          8 R  BEGIN
00017A          FLD2             1
00018A          FLD3             31
00019A          FLD1             37
00020A          10 R END
00021A          FLD6
00022A          FLD4
00023A          FLD5
00024C          2  SUBCON          BEGSR
00025C          EXFMTBEG          3
00026C          EXFMTENDING      5
00027C          WRITEBEGIN      7
00028C          WRITEEND          9
00029C          ENDSR

```



Program Using a Subroutine

Figure 2-1 (Part 1 of 2). Characteristics of an Application Program

1. Call the subroutine. **1**
2. Define the subroutine. **2**
3. Execute the format for the display to the first record. **3** This record must be defined on the A-specification. **4**
4. Execute the format for the display of the next record. **5** This record must be defined on the A-specification. **6**
5. Write the first record in the diskette data set. **7** The data set name is defined by the name on the file description line for the diskette on the A-specification. **8** The description of the record follows the data set name.
6. Write the next record in the diskette data set. **9** The description of the record follows the data set. **10**
7. The program is complete. The preceding cycle is continued until the operator uses the End of Job key.

**Note:** The operator cannot use the verify or update mode with this program. If the operator wants to update a record, more code must be added to this program or another program must be used.

**Figure 2-1 (Part 2 of 2). Characteristics of an Application Program**

The characteristics of an application program controlled through subroutines are:

- Display sequence and diskette data set formatting is totally controlled by the subroutines. Part 2 contains detailed information about this topic.
- Data-entry functions, including key-initiated modes (such as search) are not available. The only available mode of operation is the execute mode.
- A variety of methods for accessing existing data sets is available. Part 4 contains detailed information about this topic.
- Formatted printing is available. Chapter 14 contains detailed information about this topic.
- A variety of ways of arranging data on diskette is available. Records can be organized by key fields, and data sets can be indexed or sequential.
- Data set update must be controlled from the subroutine.
- All input/output control for the display, diskette, printer, and communications is controlled by subroutines on the C-specifications.
- Programs can operate without operator interaction.

## **EXECUTE MODE—THE ONLY MODE AVAILABLE FOR USE WITH APPLICATION PROGRAMS**

Execute mode is automatically selected whenever the program calls a subroutine that includes an EXFMT statement on the C-specification. This mode inhibits automatic functions provided by the data-entry operating modes and programmed keyboard functions.

During execute mode and whenever an EXFMT is performed from the subroutine to a format described on the A-specification, the following function keys are active in the application program:

- Attention
- Auto Enter
- Character Advance
- Character Backspace
- Character Delete
- Character Insert
- Duplicate  
(Works only with AUXDUP)
- Edit Release
- Erase Input
- Field Advance
- Field Correct
- Field Exit
- Field Exit Minus
- Help
- Hexadecimal
- Record Advance
- System Request
- Skip



## **ORGANIZING DATA IN DATA SETS USING A BACKGROUND PROGRAM**

The formatting characteristics of data sets that use subroutines on the C-specifications are described in detail in Part 2. Specifically, the two subjects described in Part 2 are: (1) using application programs to create data sets containing key fields and (2) using application programs to create indexed data sets.

A key field is a specially designated field that can be used to organize data on the diskette in sequence according to the value of the key field or to automatically identify the correct record (by key field) in a data set during a read operation from a subroutine. Part 4 contains information about using key fields. Any field that satisfies the following requirements can be a key field:

1. The field must exist in a data set on the diskette.
2. The values of the fields designated as keys must be in an ascending order within the data set unless an index data set is used.

The sample in Figure 2-2 illustrates a program that creates a data set with key fields.

```

00001Z*****
00002Z* PROGRAM B. FIGURE 2-2 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ MASTNAME
00005Z V1HELLO 1E EDJ
00006A F INVTEMP 15 DEVICE(DISK X'4000')
00007A R ITEMINV
00008A ITEM# 6
00009A PRICE 5
00010A ONHAND 4
00011A F INVMAST 15 DEVICE(DISK X'4000')
00012A R ITMAS
00013A 1 K ITEM# 6 1
00014A PRICE 5
00015A ONHAND 4
00016C HELLO BEGSR
00017C NOW TAG
00018C 2 READ ITEMINV 3 01
00019C N01 WRITEITMAS 4 03
00020C N01 N03 GOTO NOW
00021C ENDSR
00022C
00023C
00024
00025
00026
00027
00028

```

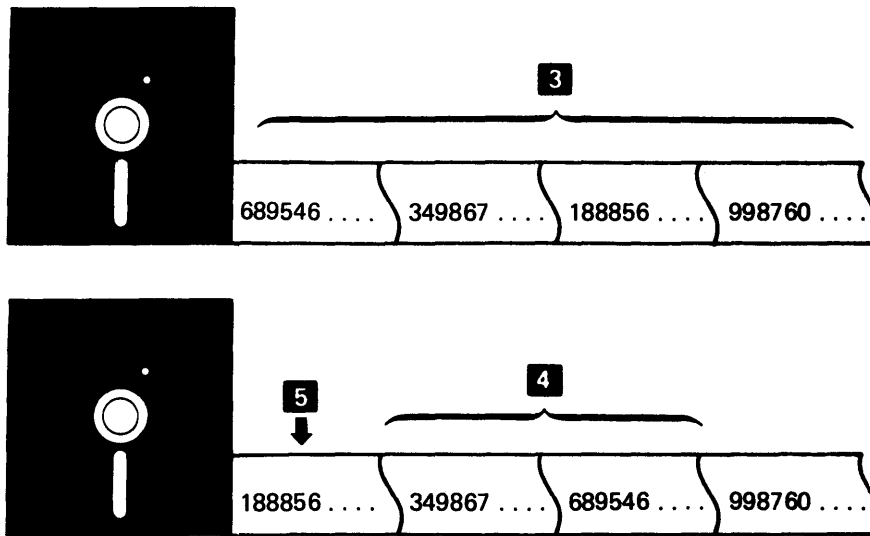


Figure 2-2. A Program that Creates a Data Set with Key Fields

Notice that the key field has a K in the name type column **1**. The program in Figure 2-2 makes each item number field in the data set a key field. The program reads each record **2** from the original data set **3** and determines where to place it in the new data set **4** according to its key value. Each time the program reads a new key field, the records (following the new record) must be rewritten to reorder the records according to their key values. The result is a data set that contains records ordered by their key fields **5**.

This process takes time, however, because records must be rewritten with the entry of each new key field. By using an index file along with the key field, you can avoid rewriting all the records after each entry. The sample in Figure 2-3 illustrates a program that uses an index.

```

00001Z*****
00002Z* PROGRAM 9. FIGURE 2-3 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ MASTMAKE
00005Z V1HELLO 1E EOJ
00006A F INV 16 DEVICE(DISK D1)
00007A R ITEMINV
00008A ITEM# 6
00009A PRICE 5
00010A ONHAND 5
00011A F INVMAST 16 DEVICE(DISK D1) INDEX(HOLD) 1
00012A R ITMAST
00013A K ITEM# 6
00014A PRICE 5
00015A ONHAND 5
00016C HELLO BEGSR
00017C NOW TAG
00018C READ ITEMINV 01
00019C N01 WRITEITMAST 0203
00020C N01N02N03 GOTD NOW
00021C ENDSR
00022C
00023C
00024C
00025C

```

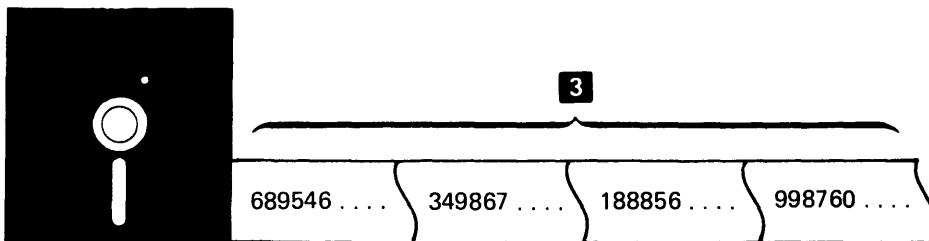
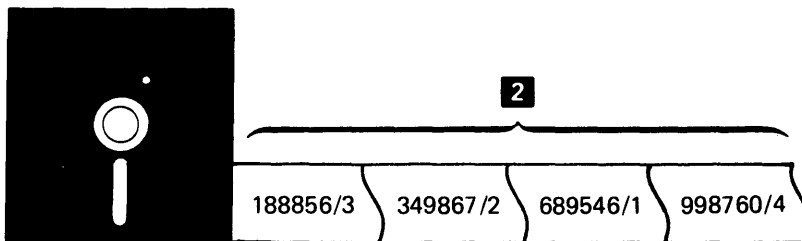
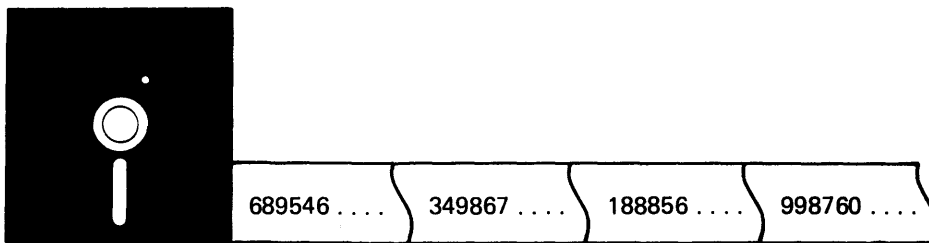


Figure 2-3. A Program that Uses an Index Data Set

Notice that a new piece of code is required: the INDEX keyword and its parameter **1**. When you use the index function with a key field, records are written in the data set exactly as they are entered, but they can be accessed as if they were entered in key sequence. This is accomplished by the index file **2** which holds the key fields (in sequence) and their relative record location in the resulting data set **3**.

Key fields and indexed data sets are only available for programs that use subroutines to create data sets or for the Sort/Merge Program Product.

## **CONTROLLING DATA SET UPDATE THROUGH SUBROUTINES**

The UPDAT statement in a subroutine is the only way a data set can be updated as it is created by an application program. The sample in Figure 2-4 illustrates an update operation.

```

00001Z*****
00002Z* PROGRAM 10. FIGURE 2-4 IN THE DE/RPG USER'S GUIDE
00003Z*****
00004ZJ  UPDATMAS
00005Z  Y1GOFIRS      1E                      EOJ
00006A      F INPUT          65                DEVICE(CRT) DSPSIZ(6 80)
00007A      R CHANGE
00008A                      0 1 1'CUSTOMER NAME'
00009A                      0 2 1'ADDRESS'
00010A                      0 3 1'CUSTOMER#'
00011A      CUSTNA          30 B 1 20
00012A      ADDR           30 B 2 20
00013A      CUSNUM          5  B 3 20
00014A      F CUSMAST       65                DEVICE(DISK D1)
00015A      R HEAD
00016A      CUSTNA          30
00017A      ADDR           30
00018A      CUSNUM          5
00019C      GOFIRS        BEGSR
00020C      LOOP          TAG
00021C                      READ HEAD 1
00022C      N05           EXFMTCHANGE 2
00023C      N05           UPDATHEAD 3
00024C      N05           GOTO LOOP
00025C                      ENDSR
                                05

```

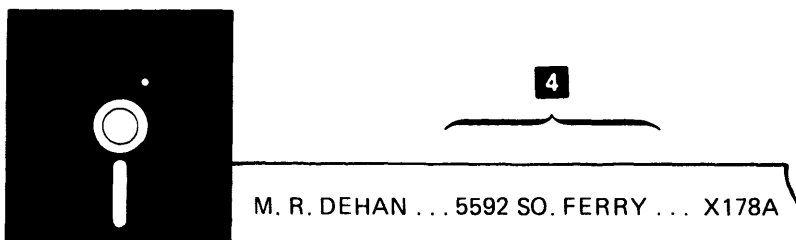
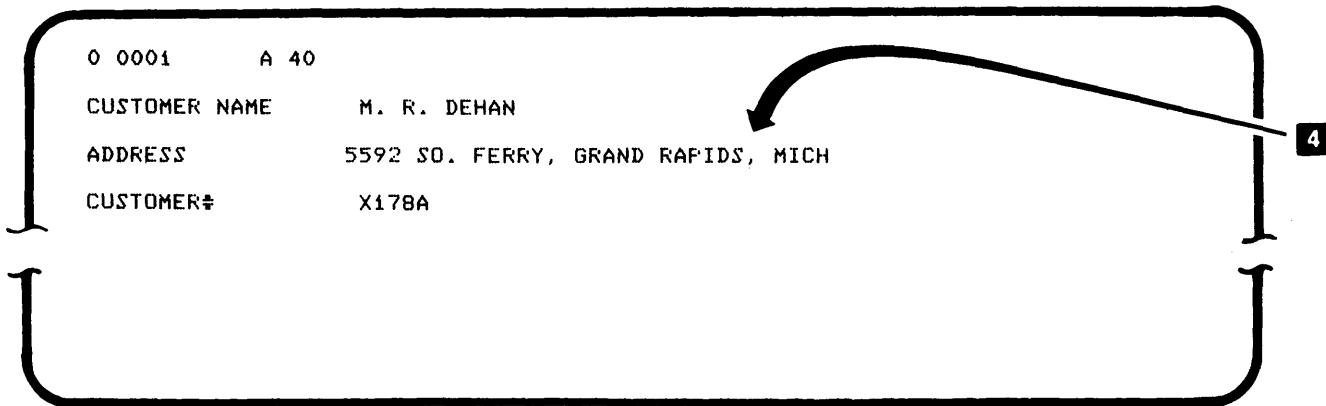
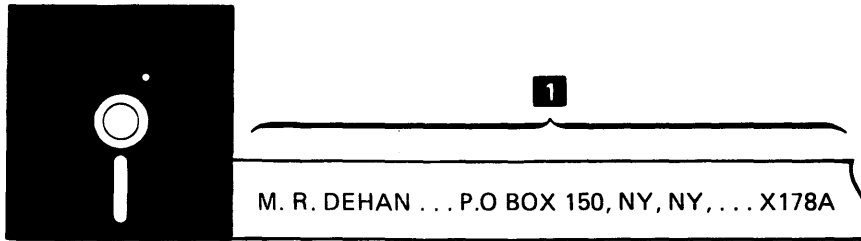


Figure 2-4. An Application Program that Includes an Update Operation

Notice that the record that is to be updated must first be read from its data set **1**. Next, the record must be written to the display **2** so the operator can see its contents. All fields that are to be changed in the record must be fields that are described with either a B (both) or I (input) usage entry. The update operation **3** must be described in the subroutine to cause the operator's changes **4**.

Programs that use subroutines do not have automatic verify or rerun capabilities. These capabilities can be provided by program operations that compare entries and set indicators. See Part 3 for a description of using indicators.

## **CONTROLLING THE INPUT/OUTPUT DEVICES**

All input/output devices (display, diskette, printer, and communications) must be specifically controlled from the subroutine. Control for the printer is described in Chapter 14.

### **Controlling the Display**

There are two ways to display data: (1) reference it through an entry format on the Z-specification and (2) specify it through an EXFMT or WRITE operation in the subroutine. If the program is exclusively a background program (does not require any interaction), the first method is not available.

The EXFMT operation in an interactive application program allows you to display fields and allows the operator to change the fields. All usage entries (I, O, W, and B) are valid. The WRITE operation allows you only to display the fields; it does not allow the operator to change them. Only the output (O) usage entry is allowed.

The samples in Figure 2-5 illustrate these techniques.

Sample A

```

00001Z*****
00002Z* PROGRAM 11. FIGURE 2-5 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ EMPBACK
00005Z A2AT          1E                      EOJ
00006A          F LOOK          95          DEVICE(CRT) DSPSIZ(6 80)
00007A          5 R DISP
00008A          0                      'THE PROGRAM IS DONE'
00009A          1 I                      CHECK(FE)
00010A          F TEMPITM       95          DEVICE(DISK X'4000')
00011A          R ITTRANS
00012A          ITEM#          6
00013A          DESC           80
00014A          COST           9
00015A          F MASTITEM       95          DEVICE(DISK X'4000')
00016A          R ITMMAS
00017A          K ITEM#          6          1
00018A          DESC           80
00019A          COST           9
00020C          AT              BEGSR
00021C          BRANCH          TAG
00022C          READ ITTRANS    2          05 3
00023C          N05             4 WRITEITMMAS          01
00024C          N05 N01        GOTO BRANCH
00025C          05             1 EXFMTDISP
00026C          ENDSR
00027

```

Sample B

```

00001Z*****
00002Z* PROGRAM 12. FIGURE 2-5 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ PUTTOGET
00005Z L1MIX          1E                      EOJ
00006A          F TEMP          22          DEVICE(DISK D1) INDEX(KEEP)
00007A          R ITEMINV
00008A          K CUSTN          4          1
00009A          ITEM#          6
00010A          PRICE          5
00011A          AMT            6
00012A          F MERGE         65          DEVICE(DISK D1)
00013A          R CUSTMAS
00014A          CUSNAM          30
00015A          ADDR            30
00016A          CUST#          4
00017A          F BILLTOT       82          DEVICE(DISK D1)
00018A          R BILL
00019A          CUSNAM          30
00020A          ADDR            30
00021A          CUST#          4
00022A          ITEM#          6
00023A          PRICE          5
00024A          AMT            6
00025A          F INPUT         1          DEVICE(CRT) DSPSIZ(6 80)
00026A          R MSG
00027A          0                      'BE PATIENT. I AM READING RECORDS.'
00028C          MIX            1 BEGSR
00029C          1 BRANCH        1 WRITEMSG
00030C          BRANCH          TAG
00031C          01             READ CUSTMAS          01
00032C          CUST#          GOTO END
00033C          02             CHAINITEMINV         02
00034C          N03             GOTO BRANCH
00035C          WRITEBILL          03
00036C          GOTO BRANCH
00037C          END            TAG
00038C          ENDSR

```

Figure 2-5. Programs that Control the Display

In Sample A, the program executes the format named DISP **1** on the display when all records have been read from the data set. The content of this record cannot be altered by the operator. When the last record is read **2**, an indicator is turned on **3**. The last record is written in the new data set **4**, and the DISP format is displayed **5**.

In Sample B, the program writes a message on the display while the program is being executed.

Part 3 contains detailed information about using indicators.



## Controlling the Diskette

All diskette operations must be controlled from the subroutines. Operations that read data from diskette and write data to diskette are available. There are basically two ways of reading data from diskette: (1) sequentially (READ) or (2) directly (CHAIN). Information about writing on diskette has been provided earlier in this chapter and is also described in Part 2. Part 4 describes additional details about reading from a diskette. The sample in Figure 2-6 illustrates the coding that controls diskette operations in subroutines.

```

00001Z*****
00002Z* PROGRAM 13. FIGURE 2-6 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ PUTTOGET
00005Z L1MIX          1E                      EOJ
00006A          F TEMP          22          DEVICE(DISK D1) INDEX(KEEP) 6
00007A          R ITEMINV
00008A          K CUST#          4          1
00009A          ITEM#          6
00010A          PRICE          5
00011A          AMT            6
00012A          F MERGE        65          DEVICE(DISK D1)
00013A          R CUSTMAS
00014A          CUSNAM          30
00015A          ADDR           30
00016A          CUST#          4
00017A          F BILLTOT      82          DEVICE(DISK D1)
00018A          R BILL
00019A          CUSNAM          30
00020A          ADDR           30
00021A          6 CUST#          4
00022A          6 ITEM#          6
00023A          PRICE          5
00024A          AMT            6
00025C          MIX            BEGSR
00026C          BRANCH         TAG
00027C          2 CUST#          READ CUSTMAS 1          01
00028C          CHAINITEMINV 3          02
00029C          4 WRITEBILL 5          03
00030C          N01N02N03      GOTO BRANCH
00031C          ENDSR

```

Figure 2-6 (Part 1 of 2). A Program that Controls the Diskette

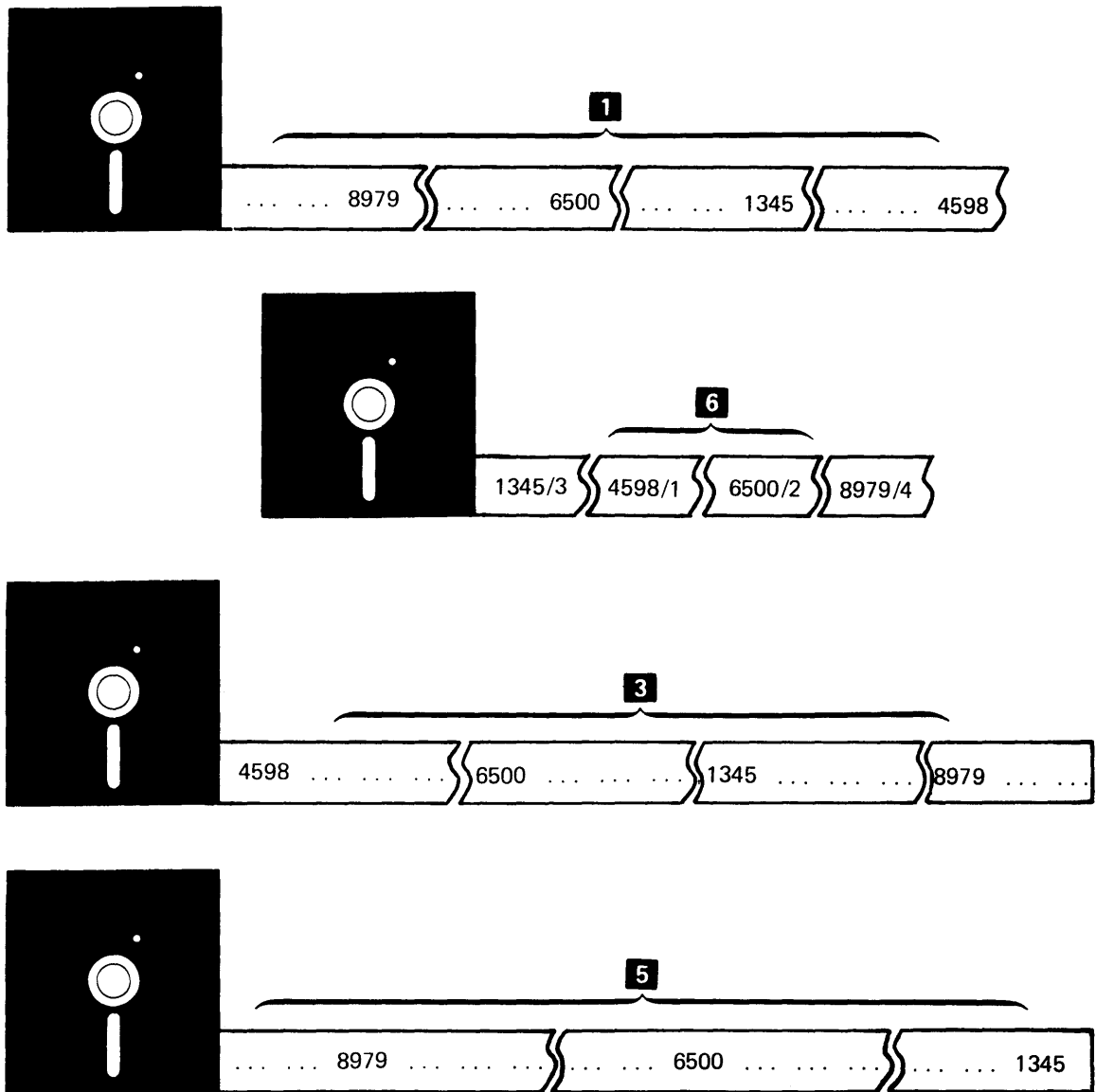


Figure 2-6 (Part 2 of 2). A Program that Controls the Diskette

The initial organization of the data set determines the type of read operation that can be used. For example, the READ operation can be performed against all types of data sets. The CHAIN by key field operation, however, can be performed only against data sets organized in ascending key sequence or against indexed data sets.

In the sample, the first operation in the subroutine sequentially reads **1** records from a data set one at a time. The second operation uses a field **2** from the first record (as a reference for the key field in the second data set **3**) to determine which record in the second data set is to be read.

The third operation writes **4** a new record in the new data set **5**. This new record contains fields from the two previous records. The *DE/RPG Reference Manual* contains details about the restrictions placed on using the READ and CHAIN operations.

## Accessing Communications

DE/RPG accesses communications through a communications file statement description. The sample in Figure 2-7 illustrates a program that uses communications.

```
00001Z*****
00002Z* PROGRAM 14. FIGURE 2-7 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ COMEXP
00005Z Z1SUBEX          1E                      EOJ
00006A          F ITFILE          45          DEVICE(DISK D1)
00007A          R ITEMTRAN
00008A          ITEM#            6
00009A          COST              9
00010A          NUMSOD            15
00011A          ONHAND            15
00012A          F ITEMUP          45          DEVICE(COMM) 1
00013A          R ITINVEN
00014A          ITEM#            6
00015A          COST              9
00016A          NUMSOD            15
00017A          ONHAND            15
00018C          SUBEX          BEGSR
00019C          2 OPEN ITEMUP
00020C          HERE          TAG
00021C          4 READ ITEMTRAN          05 6
00022C          N05          5 WRITEITINVEN
00023C          N05          GOTO HERE
00024C          3 CLOSEITEMUP
00025C          ENDSR
```

Figure 2-7. A Program that Uses Communications

Communications control is provided through the use of a communications access method which is part of the Communications Utilities Program Product. The COMM or COMM3270 device keyword 1 allows you to specify that the program is to use communications facilities. To use a data set with communications, you must first open the file 2 and then close 3 it when you are through. Open and close operations for all other input/output devices are automatically provided by DE/RPG.

This program is reading a record from an existing data set 4. It is then sending the contents of the record 5 over the communications network to the host system. Each record in the data set is read. When the end of the data set is reached, an indicator is turned on 6 and the file is closed. Section 3 contains additional information about using indicators.

The *IBM 5280 Communications Utilities Reference Manual*, SC34-0247, contains detailed information about the communications support provided by the IBM 5280.

The *IBM 5280-3270 Emulation Reference Manual*, SC34-0384, contains detailed information about the IBM 3270 Emulation.

## COMBINING DATA-ENTRY AND BACKGROUND PROGRAMS

DE/RPG is extremely flexible. The choice of program type is yours. Two considerations should guide your choice of program type: (1) the requirement for operator involvement and (2) the desired data set organization.

Often, a combination of the two basic program types is desirable. The sample in Figure 2-8 illustrates a combination program.

```

00001Z*****
00002Z* PROGRAM 15. FIGURE 2-8 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ COMBINA                                     TFILE(BILLMST) 1
00005Z X1FIND          1E                               X2          WRITE(*NO)
00006Z X2TOGETH       1E                               X1          WRITE(BILL) 5
00007A          F GET          75          DEVICE(CRT) DSPSIZ(6 80)
00008A          R FIND
00009A
00010A          2 NUMBER          5          0002001 'CUSTOMER NUMBER'
00011A          R TOGETH          I002017CHECK(DR) EXSR(GOGET)
00012A
00013A
00014A          CUSTN          30          0001001 'CUSTOMER NAME'
00015A          ADRES          30          0002001 'ADDRESS'
00016A          ITEMN          6          I002017INSERT(CUSNA)
00017A          PRICE          6          I002017INSERT(ADDR)
00018A          F BILLMST       75          0I003001PMT(ENTER ITEM NUMBER) 4
00019A          5 R BILL          2I004001PMT(ENTER COST)
00020A          ITEMN
00021A          PRICE
00022A          CUSTN
00023A          ADRES
00024A          F CUSMAST       65          DEVICE(DISK D1)
00025A          R LOOKSE
00026A          CUSNA          30
00027A          ADDR          30
00028A          K NUMBER          5
00029C          GOGET          BEGSR
00030C          NUMBER          3 CHAINLOOKSE          0102
00031C          ENDSR

```

Figure 2-8 (Part 1 of 2). A Program that Combines Data-Entry and Application Program Types

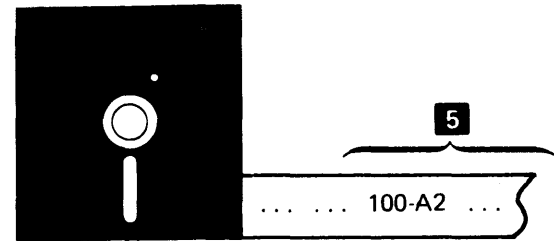
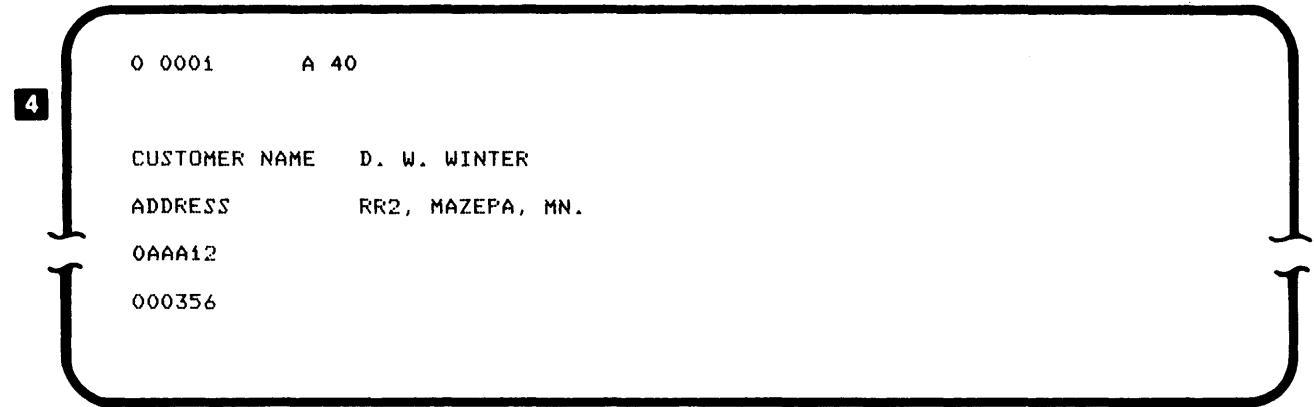
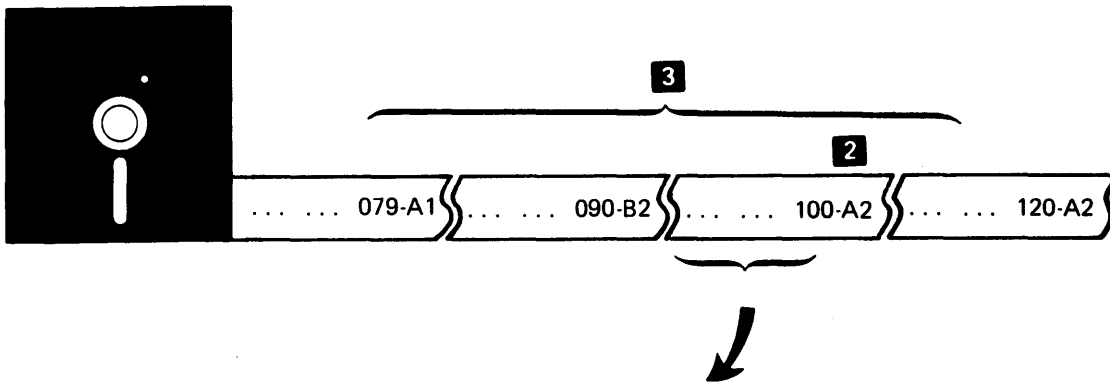
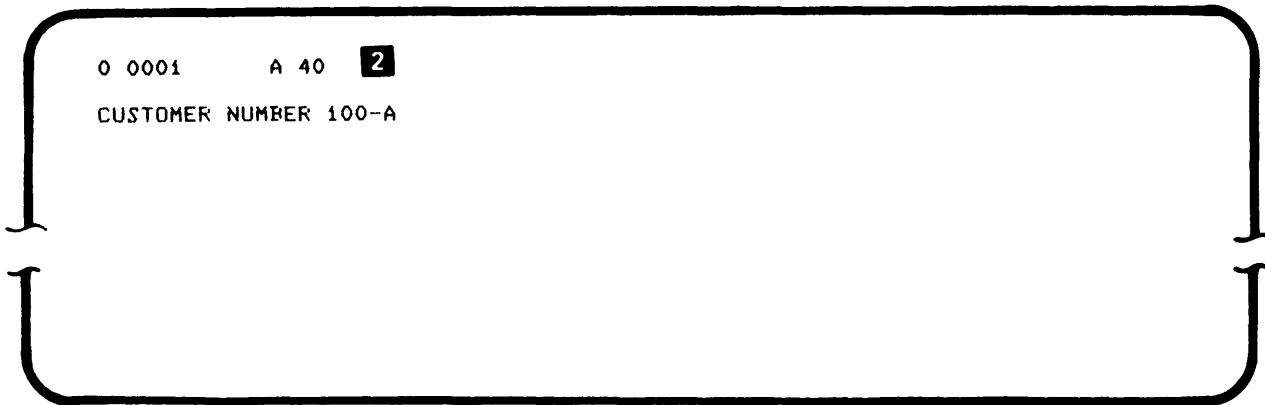


Figure 2-8 (Part 2 of 2). A Program that Combines Data-Entry and Application Program Types

In Figure 2-8, the program is basically a data-entry program that uses a transaction file **1**. The program is using a subroutine to select the correct record from another data set and appropriate data from it.

In the program, the operator is being prompted for a customer number **2**. DE/RPG then uses this number to find the correct record in an existing data set **3**. It inserts the data into a display record. Next, the operator is prompted for information about the transaction **4**. When this format is complete, a reformatted record **5** is written in the transaction data set.

The program has all the advantages of a data-entry program and it has the additional advantage of automatically supplying information from a master data set which is typically available only with application programs.

This concludes the topic describing the program types available with DE/RPG. Part 2 describes formatting characteristics of both program types.



## Part 2. Formatting Data for Displays and Diskettes

This part of the manual contains information about controlling the sequence of formats and about the contents of the formats. It consists of four chapters:

- Chapter 3. Controlling Display and Diskette Formats Via the Z-Specifications
- Chapter 4. Controlling Display and Diskette Formats Via the C-Specifications
- Chapter 5. Formatting Techniques for the Display
- Chapter 6. Formatting Techniques for the Diskette Data Set

Formatting is organizing data for an I/O device such as the display, diskette, or printer. This part of the manual describes format specifications and descriptions for the displays and diskettes. Part 6 describes format specifications and descriptions for the printer. To understand the relationships of display and diskette formats, read the entire four chapters in this part.

In DE/RPG, format control is through the statement types on the Z- and A-specifications or through operations on C-specifications. The format statements (both entry and review) on the Z-specification and the operations (EXFMT, WRITE, READ, and CHAIN) on the C-specification control the use of the formats. Specifically, statements on the Z-specification and operations on the C-specification determine whether or not the formats described on the A-specification are displayed, written in the diskette data set, printed, or copied. The record statements on the A-specification describe the contents of the display and diskette formats.





## Chapter 3. Controlling Display and Diskette Formats via the Z-Specifications

All display formats for this type of data set must be specified in format statements on the Z-specification and described in record statements on the A-specification. An entry format statement must be included to automatically display the format in the enter mode. A review format statement must be included to automatically display the format in the verify, rerun, and update modes. Manual format selection with the Sel Fmt key is always available with transaction files.

This chapter contains descriptions for specifying:

- A single entry format
- Two entry formats that are chained (automatically follow one another in the selection)
- The entry format based on a field test of data in the previous record
- An entry format controlled by the Next Fmt key

## SAMPLES SHOWING VARIOUS TYPES OF ENTRY FORMAT SPECIFICATIONS

The simplest specification for an entry mode display format is shown in the following sample in Figure 3-1. There is one format (named IDENT with the ID A0). The format is used once **1**. Because the next format ID field is blank **2**, the format does not automatically advance.

```

00001Z*****
00002Z* PROGRAM 16.  FIGURE 3-1  IN THE DE/RPG USER'S GUIDE  *
00003Z*****
00004ZJ  INQUIRE                               TFILE(LOOK)
00005Z  A0IDENT 11E                               2
00006A      F INPUT           36                DEVICE(CRT) DSPSIZ(6 80)
00007A      R IDENT
00008A      FLD1           30X I          FMT(ENTER CUSTOMER'S NAME)
00009A      FLD2           6D I          FMT(ENTER CUSTOMER ID) CHECK(DR)
00010A      F LOOK           36                DEVICE(DISK D1)
    
```

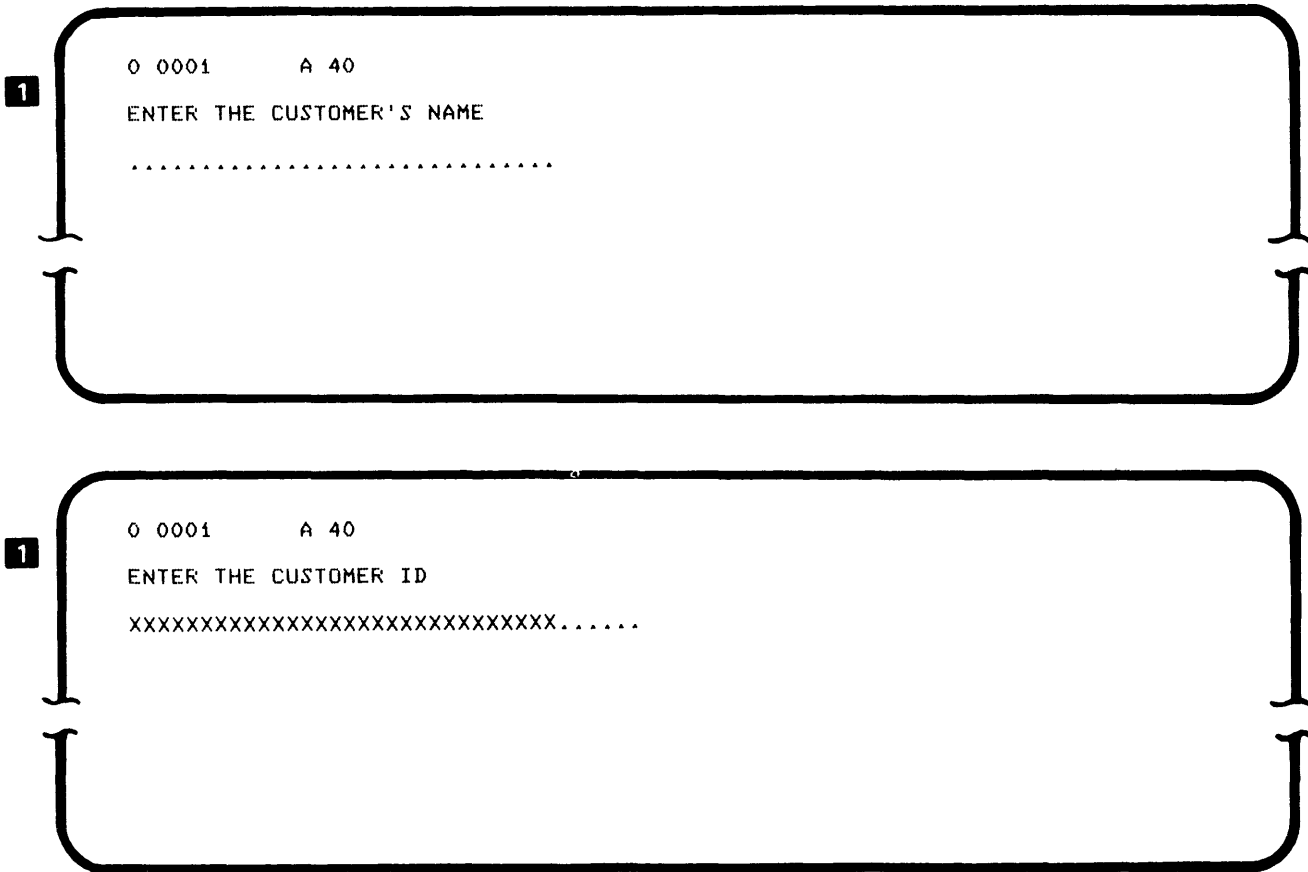


Figure 3-1. A Sample Program with Simple Format Selection

The sample in Figure 3-2 shows a slightly more complex use of specifying display formats. As illustrated by this sample, the format named IDENT with format ID A0 is used first **1**. It is used once, and then the next format **2** is displayed. The next format **3** is named MENU and has the ID A1. This format is also used once. When the record for the format is complete and has been advanced, the format named IDENT with ID A0 is automatically displayed **4**. The operator can stop this process by using the End of Job key to terminate the job.

```

00001Z*****
00002Z* PROGRAM 17. FIGURE 3-2 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004Z*****
00005J INQUIRE TFIL(LOOK)
0000 1 A0IDENT 1E A1 2
0000 3 A1MENU 1E A0 4 SLNO(3)
0000 09 F INPUT 36 DEVICE(CRT) DSPSIZ(6 80)
0000 09A R IDENT
00010A FLD1 30X I PMT(ENTER THE CUSTOMER'S NAME)
00011A FLD2 6D I PMT(ENTER CUSTOMER ID)
00012A CHECK(DR)
00013A R MENU
00014A 0001001'1 ENTER ITEM# AND PRICE DATA'
00015A 0002001'2 ENTER SHIPTO DATA'
00016A FLD3 1 I003001PMT(SELECT A NUMBER)
00017A F LOOK 36 DEVICE(DISK D1)

```

Figure 3-2 (Part 1 of 2). A Sample Program with Complex Format Selection

**1** 0 0001      A 40  
 ENTER THE CUSTOMER'S NAME  
 .....

**1** 0 0001      A 40  
 ENTER THE CUSTOMER ID  
 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.....

**3** SELECT A NUMBER  
 1 ENTER ITEM# AND PRICE DATA  
 2 ENTER SHIPTO DATA  
 -

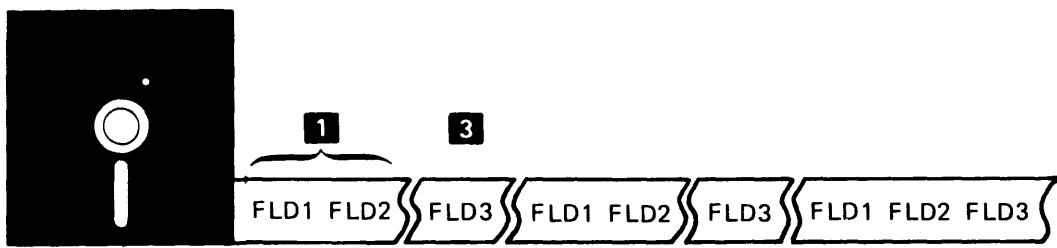


Figure 3-2 (Part 2 of 2). A Sample Program with Complex Format Selection

A more complex format selection is shown in Figure 3-3. The format selection is based on a test performed against a specified position in the previous record. Look at the sample that follows. The first format is IDENT with a format ID of A0; the second format is MENU. Notice that the selection of the next format (either ITEMINF or SHIPINF) is based upon the presence of a 1 or 2 in position 1 of the MENU record.

If a 1 is in position 1 of the MENU record, the format named ITEMINF is selected. The N 2 in the repeat field for the ITEMINF format statement means that the format will continue to be displayed until the operator selects another format. If the operator uses the next format function, the IDENT format with ID A0 is automatically selected. If a 2 is in position 1 of the MENU record, the format named SHIPINF is selected. The 1 in the Repeat field for the SHIPINF format statement means that the format is used only once. The next format that is automatically selected is IDENT with ID A0.

```

00001Z*****
00002Z* PROGRAM 18.  FIGURE 3-3    IN THE DE/RPG USER'S GUIDE  *
00003Z*****
00004ZJ  INQUIRE                                TFILE(LOOK)
00005Z  A0IDENT          1E                      A1
00006Z  A1MENU           1E *POS0001 1 '1'      A2          SLNO(3)
00007Z           E *POS0001 2 '2'      A3
00008Z  A2ITEMINF       2NE                      A0
00009Z  A3SHIPINF       1E                      A0
00010A           F INPUT              61          DEVICE(CRT)  DSPSIZ(6 80)
00011A           R IDENT
00012A           FLD1              30X I          PMT(ENTER THE CUSTOMER'S NAME)
00013A           FLD2              6D I          PMT(ENTER THE CUSTOMER ID) CHECK(DR)
00014A           R MENU
00015A                                     0001001'1 ENTER ITEM# AND PRICE DATA'
00016A                                     0002001'2 ENTER SHIPTO DATA'
00017A           FLD3              1            I003001PMT(SELECT A NUMBER) CHECK(DR)
00018A           R ITEMINF
00019A           FLD4              6 I          PMT(ENTER ITEM#)
00020A           FLD5              5 2I         PMT(ENTER PRICE)
00021A           FLD6              3 0I         PMT(ENTER QUANTITY)
00022A           FLD7              1 I          INSERT('I')
00023A           R SHIPINF
00024A           FLD8              30X I          PMT(ENTER NAME)
00025A           FLD9              30 I          PMT(ENTER ADDRESS)
00026A           FLD10             1 I          INSERT('S')
00027A           F LOOK              61          DEVICE(DISK D1)

```

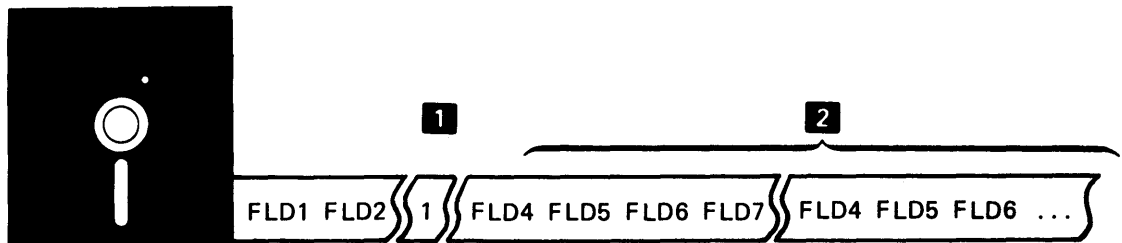


Figure 3-3. A Sample Program Showing the Use of a Simple Test to Select the Entry Format

These samples illustrate a variety of ways to specify display entry formats. The following text describes ways to use review format specifications.

## SAMPLES SHOWING VARIOUS TYPES OF REVIEW FORMAT SPECIFICATIONS

A review format is not required; if you use a review format, the simplest way to specify the review format is to simply include a review statement with no test position or character to test column entries. The sample in Figure 3-4 shows you how to do this. When you do not include a test position, you are not controlling the format that is used for review purposes. The first format is always selected in this sample. Whether or not the data being reviewed fits the format that is specified is not considered by DE/RPG. If a field is not long enough for the data, the data is simply truncated. Only the leftmost positions (as many as fit in the review format field) remain. Also, DE/RPG only takes the first ID specified. Any format IDs included in the next format columns after the first untested ID are never used. Therefore, it is always a good coding habit to include a test position and character to test entry in the records that are written in the diskette data set. Because the test position **1** in the review statement is blank, format ID A0 **2** is selected regardless of the record type that is being reviewed. If the IDENT record was the last record written, then the IDENT record is reviewed in the IDENT format. DE/RPG does not advance to the next review statement; it continues to use format A0 for all records that are read. Therefore, if the next record that is read is a MENU record, it is read with an IDENT format. Although this process would cause no problems in this sample program because the first field of the IDENT format is longer than the only field for the MENU record, it is still an improper coding practice.

```

00001Z*****
00002Z* PROGRAM 19. FIGURE 3-4 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ INQUIRE TFILE(LOOK)
00005Z AIDENT 1E A1
00006Z A1MENU 1E 1 A0
00007Z R 2 A0
00008Z R A1
00009A F PRE 5 DEVICE(CRT) DSPSIZ(6 80)
00010A R IDENT
00011A FLD1 1 I INSERT('1')
00012A FLD2 1 I INSERT('2')
00013A 0 'THIS IS FORMAT IDENT'
00014A R MENU
00015A 0 'THIS IS THE MENU FORMAT'
00016A F LOOK 5 DEVICE(DISK D1)
00017

```

Figure 3-4. A Sample Program Showing the Invalid Use of Multiple Review Formats

A good way to code a review statement is shown in the next sample. This sample demonstrates the use of the test position and character to test columns on the Z-specification. Notice that the samples being used in this topic consist of the programs written in the entry mode topic.

The sample in Figure 3-5 is taking two records (ITEMINF and SHIPINF) from the program and writing review statements that allow DE/RPG to automatically select the appropriate formats for displaying the records **1**. The other formats in the program do not have review statements in this sample.

```

00001 *****
00002 * PROGRAM 20, FIGURE 3-5 FROM THE DE/RPG USER'S GUIDE *
00003 *****
00004ZJ INQUIRE TFIL(LOOK)
00005Z AOIDENT 1E A1
00006Z A1MENU 1E *POS0001 '1' 1 A2 SLNO(3)
00007Z E *POS0001 '2' 1 A3
00008Z A2ITEMINF NE A0
00009Z A3SHIPINF 1E A0
00010Z 4 R *POS15 'I' 2 A2
00011Z 5 R *POS0061 'S' 3 A3
00012A F INPUT 61 DEVICE(CRT) DSPSIZ(6 80)
00013A R IDENT
00014A FLD1 30X I PMT(ENTER THE CUSTOMER'S NAME)
00015A FLD2 6D I PMT(ENTER THE CUSTOMER ID) CHECK(DR)
00016A R MENU
00017A 0001001'1 ENTER ITEM# AND PRICE DATA'
00018A 0002001'2 ENTER THE SHIPTO DATA'
00019A FLD3 1 I003001CHECK(DR)
00020A R ITEMINF
00021A FLD4 6 I PMT(ENTER ITEM#)
00022A FLD5 5 2I PMT(ENTER PRICE)
00023A FLD6 3 0I PMT(ENTER QUANTITY)
00024A FLD7 1 I INSERT('I')
00025A R SHIPINF
00026A FLD8 30X I PMT(ENTER NAME)
00027A FLD9 30 I PMT(ENTER ADDRESS)
00028A FLD10 1 I INSERT('S')
00029A F LOOK 61 DEVICE(DISK D1)

```

Figure 3-5. A Sample Program Showing the Valid Use of Multiple Review Formats

When the ITEMINF and SHIPINF records were initially written, they contained record markers that were included to identify them during the review modes. These record markers were I **2** for the ITEMINF record and S **3** for the SHIPINF record. They were supplied by the INSERT(' ') operation.



These record markers are now used to direct DE/RPG in selecting the appropriate format for the review mode. As the sample indicates, the review statement for the ITEMINF format contains a \*POS15  in the Test Position column and an I in the Character to Test columns. The 15 is determined in the following way:

Add the lengths of the fields preceding the test character field

$$6 + 5 + 3 = 14$$

Add 1 to determine the position for the test character

$$14 + 1 = 15$$

The \*POS61  test position for the SHIPINF record is determined in the same way.

The result is that DE/RPG always selects the ITEMINF format whenever an I is encountered in position 15 of the current diskette record and the SHIPINF format whenever an S is encountered in position 61 of the current diskette record.

This sample does not show you the optimal way to specify review formats; the length of the fields in the ITEMINF and SHIPINF records are not the same. Therefore, position 15 of the SHIPINF record could contain an I (because this would be a position within the customer name field of the SHIPINF record), and therefore, the ITEMINF format could be selected for reviewing a SHIPINF record. The sample in Figure 3-6 shows you how to correct this type of problem.

```

00001 *****
00002 * PROGRAM 21.  FIGURE 3-6      FROM THE DE/RPG USER'S GUIDE *
00003 *****
00004ZJ  INQUIRE                                TFILE(LOOK)
00005Z  AOIDENT      1E                          A1
00006Z  A1MENU      1E *POS0001    '1'          A2      SLNO(3)
00007Z      E *POS0001    '2'          A3
00008Z  A2ITEMINF   NE                          A0
00009Z  A3SHIPINF   1E                          A0
00010Z      2 R *POS0061    'I'          A2
00011Z      R *POS0061    'S'          A3
00012A      F INPUT      61                DEVICE(CRT) DSPSIZ(6 80)
00013A      R IDENT
00014A      FLD1      30X I      PMT(ENTER THE CUSTOMER'S NAME)
00015A      FLD2      6D I      PMT(ENTER THE CUSTOMER ID) CHECK(DR)
00016A      R MENU
00017A                                0001001'1 ENTER ITEM# AND PRICE DATA'
00018A                                0002001'2 ENTER THE SHIPTO DATA'
00019A      FLD3      1      I003001CHECK(DR)
00020A      R ITEMINF
00021A      FLD4      6 I      PMT(ENTER ITEM#)
00022A      FLD5      5 2I     PMT(ENTER PRICE)
00023A      FLD6      3 0I     PMT(ENTER QUANTITY)
00024A      FLD7      1 46 I   CHECK(BY)
00025A      FLDA      1 I     INSERT('I')
00026A      R SHIPINF
00027A      FLD8      30X I     PMT(ENTER NAME)
00028A      FLD9      30 I     PMT(ENTER ADDRESS)
00029A      FLD10     1        INSERT('S')
00030A      F LOOK      61      DEVICE(DISK D1)
00031

```

Figure 3-6. A Program Showing a Suggested Technique for Multiple Review Format Selection

Take the previous sample and add a field that extends the record marker for the ITEMINF record to the corresponding position of the record marker in the SHIPINF record. Determine the length of this field in the following way:

Subtract the position of the ITEMINF record marker from the position of the SHIPINF record marker

$$61 - 15 = 46$$

The length of this new field should be 46. The CHECK(BY) operation provides a field that consists of blanks and which cannot be altered by the operator. Look at the sample 1. Now the review statement for both format selections contains a \*POS61 test position 2.

To illustrate the most complex review mode format selection, the sample uses a new program. The entire program is not shown; only those parts that are directly related to the complex review format selection are provided. See the sample in Figure 3-7.

```

00001Z*****
00002Z* PROGRAM 22. FIGURE 3-7 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ RESEARCH TFILE(CENSUS)
00005Z BOGENERAL 1E *POS0001 'M' B1
00006Z 1 E *POS0001 'F' B2
00007Z B1MALE 1E *POS0002 'S' B4
00008Z E *POS0002 'M' B5
00009Z B2FEMALE 1E *POS0002 'S' B6
00010Z E *POS0002 'M' B7
00011Z B4MASISTAT NE B0
00012Z B5MAMARSTA NE B0
00013Z B6FESISTAT NE B0
00014Z B7FEMARSTA NE B0
00015Z R *POS0001 'F'
00016Z RA*POS0002 'S' B6
00017Z R 0
00018A F EX 5 DEVICE(CRT) DSPSIZ(6 80)
00019A R GENERAL
00020A 0 'THIS IS THE GENERAL RECORD'
00021A FLD1 1 I PMT(S=SINGLE, M=MARRIED, USE ENTER)
00022A R MALE
00023A 0 'THIS IS THE MALE RECORD'
00024A FLD2 1 I INSERT('M')
00025A FLD3 1 I PMT(S=SINGLE, M=MARRIED, USE ENTER)
00026A R FEMALE
00027A 0 'THIS IS THE FEMALE RECORD'
00028A FLD4 1 I INSERT('F')
00029A FLD5 1 I PMT(S=SINGLE, M=MARRIED, USE ENTER)
00030A R MASISTAT
00031A 0 'THIS IS THE MASISTAT RECORD'
00032A FLD6 1 I PMT(USE ENTER, NEXT IS GENERAL)
00033A R MAMARSTA
00034A 0 'THIS IS THE MAMARSTA RECORD'
00035A FLD7 1 I PMT(USE ENTER, NEXT IS GENERAL)
00036A R FESISTAT
00037A 0 'THIS IS FESISTAT'
00038A FLD8 1 I PMT(USE ENTER, GENERAL NEXT)
00039A R FEMARSTA
00040A 0 'THIS IS THE FEMARSTA RECORD'
00041A FLD9 1 I PMT(USE ENTER, GENERAL NEXT)
00042A F CENSUS 5 DEVICE(DISK D1)

```

Figure 3-7 (Part 1 of 2). A Program Showing Complex Use of Multiple Format Selection

2

```
0 0001      A 40
F=FEMALE  M=MALE      USE ENTER
THIS IS THE GENERAL RECORDM 3
```

4

```
0 0001      A 40
S=SINGLE, M=MARRIED, USE ENTER
THIS IS THE MALE RECORDM
```

5

```
0 0001      A 40
MM
OWNS 5-ROOM HOME IN SECTOR 16      973-48-295
PROFESSIONAL
36 YEARS OF AGE
```

**Note:** The program literals indicate the type of record that would be displayed. The sample displays show the type of information that might be used for this kind of application.

**Figure 3-7 (Part 2 of 2). A Program Showing Complex Use of Multiple Format Selection**

The technique being demonstrated is the selection of the correct format to use for records that use two record markers to select the appropriate format for review mode. The process is called ANDing. Before the format can be selected, both test conditions **1** must be true.

The sample program that demonstrates format selection through ANDing test conditions uses an application common to population census applications. The program allows an operator to enter data into record types in a way that this data can be separated and identified later. The first record that an operator sees is the one named GENERAL. The first display **2** demonstrates the type of information required by this record type. Depending on the entry in the first position of this record (F for female and M for male), DE/RPG selects either the MALE or FEMALE format. In this sample, the selected format is MALE **3**. The operator enters data of the type shown by the second sample display **4**. If the second field of this record is an M, DE/RPG automatically selects the format called MAMARSTA (for male married statistics). The type of data the operator enters into this record is indicated by the third display **5**. During the review mode, the MAMARSTA format is selected whenever DE/RPG recognizes an M in position one of the current diskette record and an M in position two. As you notice in the third display, this is the identifying sequence used by this record type. This concludes the example of ANDing test conditions to select a format for the review mode.

The various ways formats can be selected for the review mode have been demonstrated in this topic. The next topic in this chapter describes the ways in which you can determine the contents of the diskette formats.

## USING THE Z-SPECIFICATION TO CONTROL DISKETTE FORMATS FOR A TRANSACTION FILE

The default format for diskette records written by using a transaction file is identical to the format specified for the display. The first sample program used in this chapter demonstrates this default. Look at the sample in Figure 3-8. Notice that the detailed view of the diskette record **1** contains fields in the order in which they were entered by the operator using the display format **2**.

To change the default for writing the diskette record created by a transaction file, you must specify a different diskette record to be used in the program. The record you specify must contain all data fields contained in the entry display format and it must not contain fields from any other record. The order of the fields is the part of the record that can be changed.

```

00001Z*****
00002Z* PROGRAM 23. FIGURE 3-8 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ INQUIRE TFILE(LOOK)
00005Z AIDENT 1E A1
00006Z A1MENU 1E *PDS0001 '1' A2 SLNO(3)
00007Z E *PDS0001 '2' A3
00008Z A2ITEMINF NE A0
00009Z A3SHIPINF 1E A0
00010Z R *PDS0061 'I' A2
00011Z R *PDS0061 'S' A3
00012A F INPUT 61 DEVICE(CRT) DSPSIZ(6 80)
00013A R IDENT
00014A FLD1 30X I PMT(ENTER THE CUSTOMER'S NAME)
00015A FLD2 6D I PMT(ENTER THE CUSTOMER ID) CHECK(DR)
00016A R MENU
00017A
00018A 0001001'1 ENTER ITEM# AND PRICE DATA'
00019A 0002001'2 ENTER THE SHIPTO DATA'
00020A FLD3 1 I003001CHECK(DR)
00021A R ITEMINF
00022A FLD4 6 I PMT(ENTER ITEM#)
00023A FLD5 5 2I PMT(ENTER PRICE)
00024A FLD6 3 0I PMT(ENTER QUANTITY)
00025A FLD7 46 I CHECK(BY)
00026A FLDA 1 I INSERT('I')
00027A R SHIPINF
00028A FLD8 30X I PMT(ENTER NAME)
00029A FLD9 30 I PMT(ENTER ADDRESS)
00030A FLD10 1 I INSERT('S')
00030A F LOOK 61 DEVICE(DISK D1)
  
```

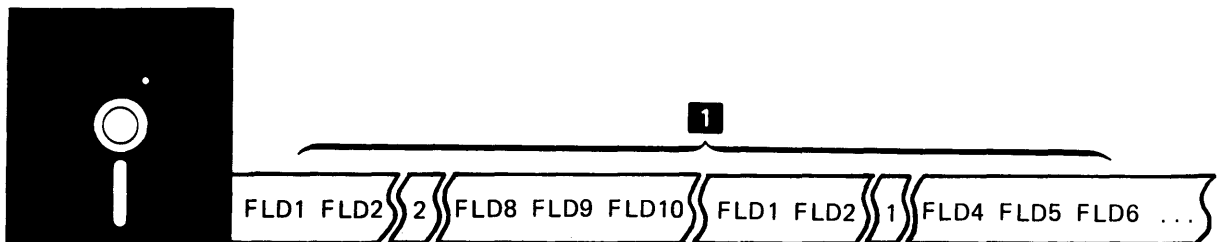


Figure 3-8. A Sample Program Showing Diskette Data Set Reformatting

The sample in Figure 3-9 demonstrates how this specification can be made. Although the program is the same one used in the previous sample, notice that there is now a record following the file statement for the diskette **1**. Also notice the keyword WRITE followed by the new record name on the entry format line **2**. As the sample illustrates, the program reorganizes the sequence of the fields in the diskette record **3** as specified by the format named as the WRITE parameter.

```

00001Z*****
00002Z* PROGRAM 24. FIGURE 3-9 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ INQUIRE TFILE(LOOK)
00005Z A0IDENT 1E A1
00006Z A1MENU 1E *POS0001 '1' A2
00007Z E *POS0001 '2' A3
00008Z A2ITEMINF NE A0 WRITE(ITEM) 2
00009Z A3SHIPINF 1E A0
00010Z R *POS0061 'I' A2
00011Z R *POS0061 'S' A3
00012A F INPUT 61 DEVICE(CRT) DSPSIZ(6 80)
00013A R IDENT
00014A FLD1 30X I PMT(ENTER THE CUSTOMER'S NAME)
00015A FLD2 6D I PMT(ENTER THE CUSTOMER ID) CHECK(DR)
00016A R MENU
00017A 0001001'1 ENTER ITEM# AND PRICE DATA'
00018A 0002001'2 ENTER THE SHIPTO DATA'
00019A FLD3 1 I003001CHECK(DR)
00020A R ITEMINF
00021A FLD4 6 I PMT(ENTER ITEM#)
00022A FLD5 5 2I PMT(ENTER PRICE)
00023A FLD6 3 0I PMT(ENTER QUANTITY)
00024A FLD7 46 I CHECK(BY)
00025A FLDA 1 I INSERT('I')
00026A R SHIPINF
00027A FLD8 30X I PMT(ENTER NAME)
00028A FLD9 30 I PMT(ENTER ADDRESS)
00029A FLD10 1 INSERT('S')
00030A F LOOK 61 DEVICE(DISK D1)
00031A 1 R ITEM
00032A FLD7 46
00033A FLD6 3 0
00034A FLD4 6
00035A FLD5 5 2
00036A FLDA 1
00037

```



Figure 3-9. A Sample Program Showing Simple Reformatting for a Data-Entry Program

To write a record that uses a different format from the one used for the display, both these conditions must exist: (1) a new record must be created within the diskette file and (2) the WRITE keyword followed by the record name must appear in the statement for the entry mode display format.

When the entry mode display and diskette formats are not identical, you must make a decision about which to use for displaying the record in the review mode. Your choice consists of using either the initial entry format or of using the diskette format. If you select the diskette format for the review mode, you must include a description of this format within the CRT file.

One other possibility exists concerning the diskette format. So far, you have seen that you can use the default format that is identical to the display format and that you can specify a separate format. The last possibility for data sets created by transaction files is the suppression of writing a diskette record. This is done with the WRITE(\*NO) operation **1**. A possible application is the MENU record shown in previous samples. This record must be displayed, but the entry is not required in the diskette data set. Look at the following sample. Note that FLD3 is now missing from the data set **2**.



```

00001Z*****
00002Z* PROGRAM 25. FIGURE 3-10 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ INQUIRE TFILE(LOOK)
00005Z A0IDENT 1E A1
00006Z A1MENU 1E *POS0001 '1' A2 1 WRITE(*NO) SLNO(3)
00007Z E *POS0001 '2' A3
00008Z A2ITEMINF NE A0 WRITE(ITEM)
00009Z A3SHIPINF 1E A0
00010Z R *POS0061 'I' A2
00011Z R *POS0061 'S' A3
00012A F INPUT 61 DEVICE(CRT) DSPSIZ(6 80)
00013A R IDENT
00014A FLD1 30X I PMT(ENTER THE CUSTOMER'S NAME)
00015A FLD2 6D I PMT(ENTER THE CUSTOMER ID) CHECK(DR)
00016A R MENU
00017A 0001001'1 ENTER ITEM# AND PRICE DATA'
00018A 0002001'2 ENTER THE SHIPTO DATA'
00019A FLD3 1 IOU3001CHECK(DR)
00020A R ITEMINF
00021A FLD4 6 I PMT(ENTER ITEM#)
00022A FLD5 5 2I PMT(ENTER PRICE)
00023A FLD6 3 0I PMT(ENTER QUANTITY)
00024A FLD7 46 I CHECK(BY)
00025A FLDA 1 I INSERT('I')
00026A R SHIPINF
00027A FLD8 30X I PMT(ENTER NAME)
00028A FLD9 30 I PMT(ENTER ADDRESS)
00029A FLD10 1 INSERT('S')
00030A F LOOK 61 DEVICE(DISK D1)
00031A R ITEM
00032A FLD7 46
00033A FLD6 3 0
00034A FLD4 6
00035A FLD5 5 2
00036A FLDA 1
00037

```

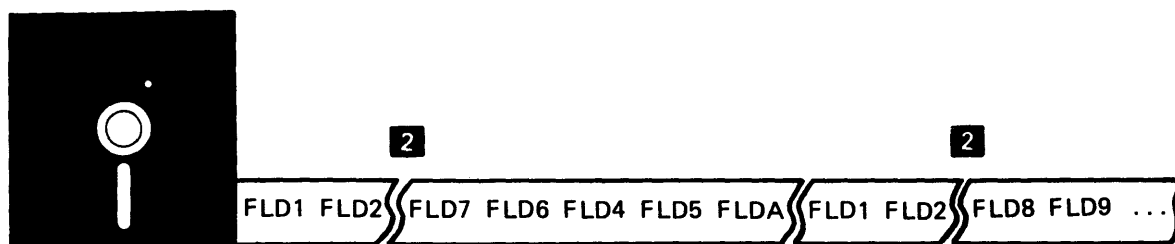


Figure 3-10. A Sample Program Showing How a Record Can Be Displayed but Not Written in the Diskette Data Set

This concludes the topic describing the control of diskette formats for data sets created by a transaction file. The next topic describes controlling display formats and diskette formats through subroutines on the C-specification.

## Chapter 4. Controlling Display and Diskette Formats via the C-Specifications

All format control through subroutines on the C-specification must be explicit. In other words, you must specify the operation to be performed on the format, such as reading it from or writing it to a diskette data set or placing data on the display.

Subroutines on the C-specification can be accessed in one of two ways: (1) by naming the subroutine in an entry format statement on the Z-specification or (2) by using the EXSR keyword and subroutine name on the A-specification. The samples in Figure 4-1 demonstrate the ways in which subroutines can be called.

Sample A illustrates calling the subroutine from the Z-specification. Sample B illustrates calling the subroutine from the A-specification.

**Sample A**

```

00001Z*****
00002Z* PROGRAM 26. FIGURE 4-1 SAMPLE A IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004Z.1 SAMPLE1
00005.1 C1MAST          1E                      EOJ
00006A          F INPUT          62          DEVICE(CRT) DSPSIZ(6 80)
00007A          R HEADER
00008A          FLD1          30   I   3   1PMT(ENTER NAME)
00009A          FLD2          30   I   4   1PMT(ENTER ADDRESS)
00010A          FLDX          1   I   5   1PMT(IF THIS IS THE LAST RECORD TO +
00010A          /                                     ENTER, PLACE AN X HERE) DSPATR(RI)
00011A          FLD3          1   I   5   79INSERT('H')
00012A          F OUTPUT          62          DEVICE(DISK D1)
00013A          R HEAD
00014A          FLD3
00015A          FLD1
00016A          FLD2
00017C          1 MAST          BEGSR
00018C          RETURN          TAG
00019C          EXFMHEADER
00020C          FLDX          COMP 'X'          01
00021C          WRITEHEAD
00022C          N01          GOTO RETURN
00023C          ENDSR
00024
00025

```

Figure 4-1 (Part 1 of 2). A Sample Program Showing a Call to a Subroutine from the Z-Specification

Sample B

```

00001Z*****
00002Z* PROGRAM 27. FIGURE 4-1 SAMPLE B IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ  SAMFLEX                                TFILE(INTERMED)
00005Z  E1HEADER          1E                      E2
00006Z  E2DETAIL          1E                      E1
00007A          F INPUT          67              DEVICE(CRT) DSPSIZ(6 80)
00008A          R HEADER
00009A          FLD1           30  I             PMT(ENTER CUSTOMER NAME)
00010A          FLD2           6  I             PMT(ENTER CUSTOMER#)
00011A          FLD3           30  I             PMT(ENTER ADDRESS)
00012A          FLD4           1  I             INSERT('H')
00013A          R DETAIL
00014A          FLD5           6  I             PMT(ENTER ITEM#)
00015A          FLD6           30  I             PMT(ENTER DESCRIPTION)
00016A          FLD7           5  I             PMT(ENTER PURCHASE)
00017A          FLD8           5  I             PMT(ENTER PRICE) EXSR(OUT) 2
00018A          F FINAL          46              DEVICE(DISK D1)
00019A          R DET
00020A          K FLD5           6
00021A          FLD6           30
00022A          FLD7           5
00023A          FLD8           5
00024A          F INTERMED       67              DEVICE(DISK D1)
00025
00026C          2 OUT          BEGSR
00027C          WRITEDET
00028C          ENDSR
00029

```

Figure 4-1 (Part 2 of 2). A Sample Program Showing a Call to a Subroutine from the A-Specification

## CALLING SUBROUTINES FROM THE Z-SPECIFICATION

A subroutine named in an entry format on the Z-specification is executed, as are all other entry formats, when it is selected based on a previous test condition. When the subroutine is called, DE/RPG exits to the C-specification subroutine description. This description must begin with a BEGSR operation which names the subroutine (the name must be the same as the name specified on the Z-specification). Calling a subroutine in an entry format statement allows you to perform complex reformatting of data that has been entered via a single entry format specification. The following sample in Figure 4-2 shows you how this facility allows you to take data from a single entry format and reformat it to create three separate data sets from the one display record. The transaction data set is the fourth data set; it is in the order of entry.

```

00001Z*****
00002Z* PROGRAM 28. FIGURE 4-2 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ  MANY                                TFILE(OUT)
00005Z  A1FIRST      1E                        A2
00006Z  A2OUTPUT    1E                        A1
00007A          F INPUT      180                DEVICE(CRT) DSPSIZ(6 80)
00008A          1 R FIRST
00009A          CUSN         20      I      FMT(ENTER THE CUSTOMER'S NAME)
00010A          ADDR        40      I      FMT(ENTER THE ADDRESS)
00011A          COMP        30      I      FMT(ENTER THE NAME OF THE COMPANY)
00012A          PUR         80      I      FMT(ENTER THE PURCHASE)
00013A          DATE         6      I      FMT(ENTER THE DATE)
00014A          F CUSNMAST   66                DEVICE(DISK D1)
00015A          R ONE
00016A          CUSN         20
00017A          ADDR        40
00018A          DATE         6
00019A          F COMFMAST  36                DEVICE(DISK D1)
00020A          R TWO
00021A          COMP
00022A          DATE
00023A          F ORDMAS    106                DEVICE(DISK D1)
00024A          R THREE
00025A          DATE
00026A          PUR
00027A          CUSN
00028A          F OUT       180                DEVICE(DISK D1)
00029C          OUTPUT      BEGSR
00030C          WRITEONE   2
00031C          WRITETWO  3
00032C          WRITETHREE 4
00033C          ENDSR

```

Figure 4-2 (Part 1 of 2). A Sample Program Showing Complex Reformatting

0 0001      A 40

MARGARET RUTHERFERD 1533 RAGNOLLA DRIVE, WASHBURN, MARYLAND APEX DESIGNS ESQ

300 FAIR ROOMOR FANTS, LOT 576

032880

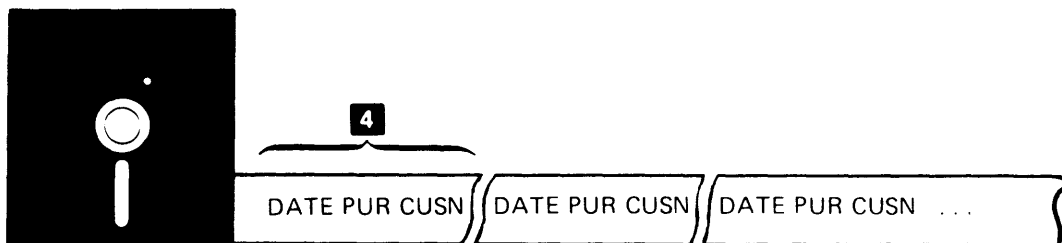
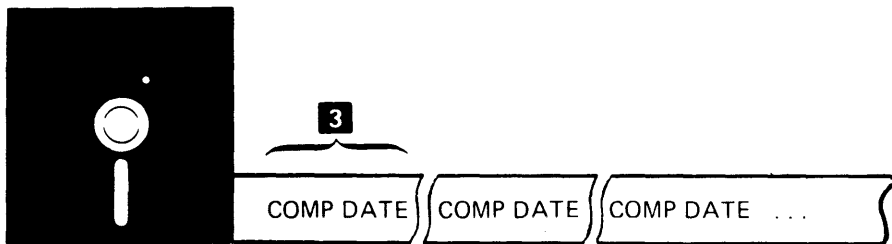
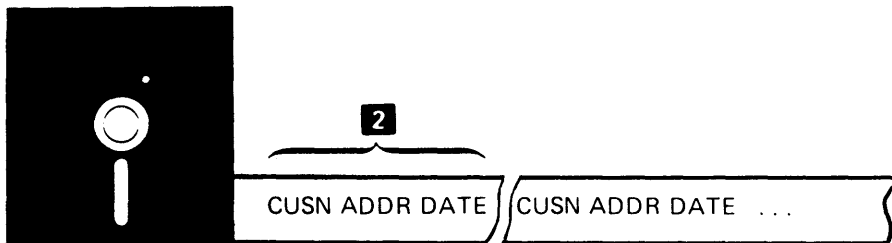
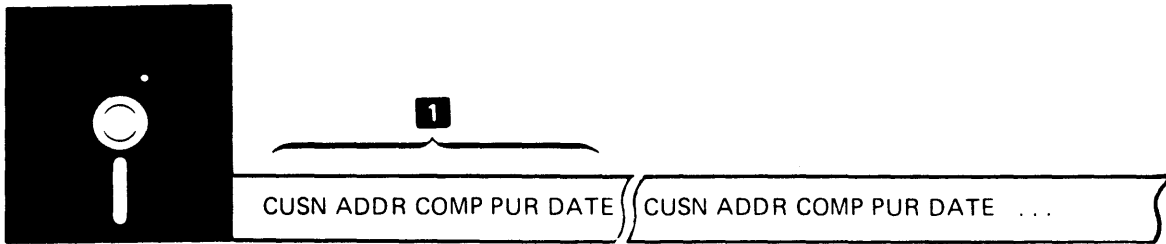


Figure 4-2 (Part 2 of 2). A Sample Program Showing Complex Reformatting

This program illustrates the combination of using a transaction file and explicit write operations to create data sets. Each time the operator completes the entry using the display format named FIRST **1**, a default record is written in the data set named by the TFILE function and three reformatted records **2**, **3**, and **4** are written in three separate data sets as determined by the subroutine on the C-specification.

In addition to the complex reformatting capabilities shown in the previous sample, the entire interactive program can be controlled by the subroutine on the C-specification. The sample in Figure 4-3 demonstrates how this can be accomplished. The first entry format specified is the name of a subroutine on the C-specification **1**. As you look at the subroutine contents you can see that it contains an operation named EXFMT **2** followed by a name that is repeated on the A-specification. This operation allows you to tell DE/RPG when to display a format that the operator uses to enter data. Once the data is entered, control returns to the subroutine and the data can be reformatted for writing to a diskette data set or written as it occurs in the display format **3**. It is possible to loop (or cycle) through the process again and again **4**.

```

00001Z*****
00002Z* PROGRAM 29. FIGURE 4-3 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ SAMPLE
00005Z 1 DISTART 1E EDJ
00006A F INPUT 41 DEVICE(CRT) DSPSIZ(6 80)
00007A R ITEMID
00008A FLD1 6 I PMT(ENTER ITEM#)
00009A FLD2 30 I PMT(ENTER DESCRIPTION)
00010A FLD3 4 0I PMT(ENTER # SOLD)
00011A FLDX 1 1 PMT(TO END, ENTER X)
00012A F MASITEM 41 DEVICE(DISK D1)
00013A R ITEM
00014A FLD1 6
00015A FLD2 30
00016A FLD3 4 0
00017C START BEGSR
00018C RETURN TAG
00019C EXFMTITEMID 2
00020C FLDX 3 COMP 'X' 01
00021C N01 3 WRITEITEM
00022C N01 3 GOTO RETURN 4
00023C ENDSR
00024
00025
00026

```

Figure 4-3. A Sample Program Showing Program Control from the C-Specification

When the subroutine controls the operation of the programs, you can use the End of Job key to terminate the program. In this sample, another technique for ending the job is used. The operator enters an X in a designated field when he wants to terminate the program. This X tells the program to go to termination.

## CALLING A SUBROUTINE FROM THE A-SPECIFICATION

The second way in which you can call a subroutine on the C-specification is with the EXSR keyword on the A-specification.

This operation allows you to control the format selection from the keyboard but still use the C-specification to perform such operations as writing a record in the diskette data set. The sample in Figure 4-4 illustrates the sequence that occurs when you use the exit from the A-specification to call a subroutine that writes a diskette record.

```

00001Z*****
00002Z* PROGRAM 30. FIGURE 4-4 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00007' SAMPLEX TFILE(INTERMED)
0000 1 E1HEADER 1E E2
0000 2 E2DETAIL 1E E1
0000 H F INPUT 67 DEVICE(CRT) DSPSIZ(6 80)
00008A R HEADER
00009A FLD1 30 I FMT(ENTER CUSTOMER NAME)
00010A FLD2 6 I FMT(ENTER CUSTOMER#)
00011A FLD3 30 I FMT(ENTER ADDRESS)
00012A FLD4 1 I INSERT('H')
00013A R DETAIL
00014A FLD5 6 I FMT(ENTER ITEM#)
00015A FLD6 30 I FMT(ENTER DESCRIPTION)
00016A FLD7 5 I FMT(ENTER PURCHASE)
00017A FLD8 5 I FMT(ENTER PRICE) EXSR(OUT) 4
00018A F FINAL 46 DEVICE(DISK D1)
00019A R DET
00020A 6 K FLD5 6
00021A FLD6 30
00022A FLD7 5
00023A FLD8 5
00024A F INTERMED 67 DEVICE(DISK D1)
00025
00026C OUT BEGSR
00027C 5 WRITEDET
00028C ENDSR
00029

```

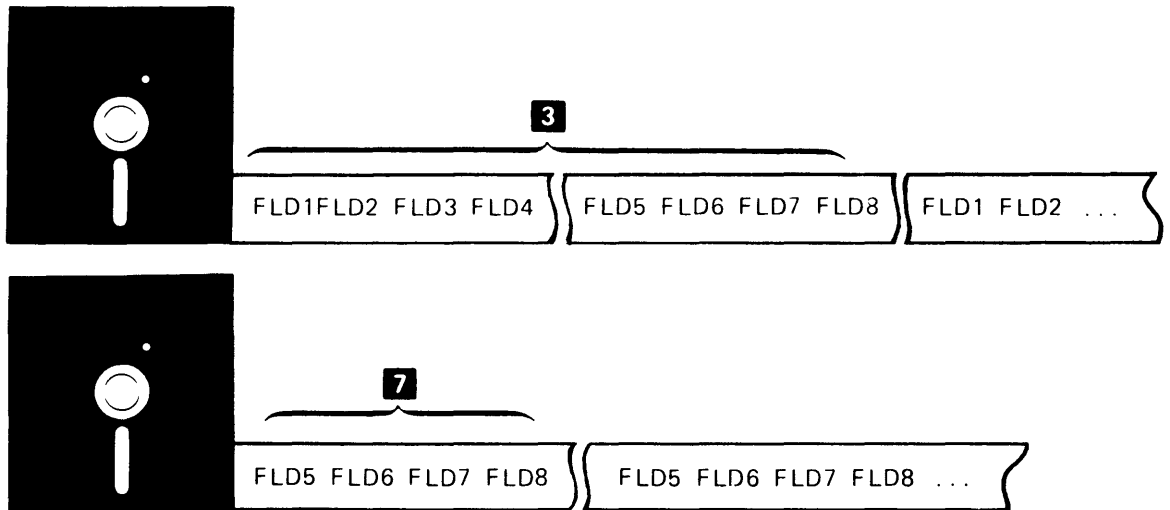


Figure 4-4. A Sample Program Showing Display Control from the A-Specification and Diskette Data Set Control from the C-Specification

The sample program displays a HEADER format **1** as specified by the entry statement on the Z-specification. After the operator completes the entries for one HEADER record, the DETAIL format is automatically displayed **2**. At the completion of each format, a record is written in the transaction data set **3**. At the conclusion of the DETAIL format **4**, the subroutine writes a DET record **5** in key sequence **6** in the diskette data set **7**. When the DETAIL format has been completed, the HEADER format is redisplayed. This process is repeated for the new customer. The process continues until the operator uses the End of Job key. In this example, the End of Job key works because a transaction file is being used.



This concludes the description of controlling formats through subroutines on the C-specification. Chapter 5 contains topics describing the control of the contents of the display, and Chapter 6 contains topics describing the control of the contents of diskette records.

## **Chapter 5. Formatting Techniques for the Displays**

**This chapter describes characteristics of the display formats. A variety of ways to design displays for various applications are provided.**

The default display format consists of fields strung together with no overlap or spacing other than the spacing resulting from unused field positions or display attributes. The sample in Figure 5-1 demonstrates the default format. Notice that the position columns of the A-specification are unused.

```

00001Z*****
00002Z* PROGRAM 31. FIGURE 5-1 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ OVMMASTER TFILE(OVERMAST)
00005Z A1FIRST 1E A1
00006Z R A1
00007A F OVERALL 17 DEVICE(CRT) DSPSIZ(6 80)
00008A FIRST
00009A 1 FLD1 5C I SHIFT(XXXDD) DSPATR(CS)
00010A 2 FLD2 3 I DSPATR(CS)
00011A FLD3 3 4 I DSPATR(CS)
00012A 4 FLD4 5 I DSPATR(CS)
00013A 0 'FLD1'
00014A 0 'FLD2'
00015A 0 'FLD3'
00016A 0 'FLD4'
00017A F OVERMAST 17 DEVICE(DISK D1)

```

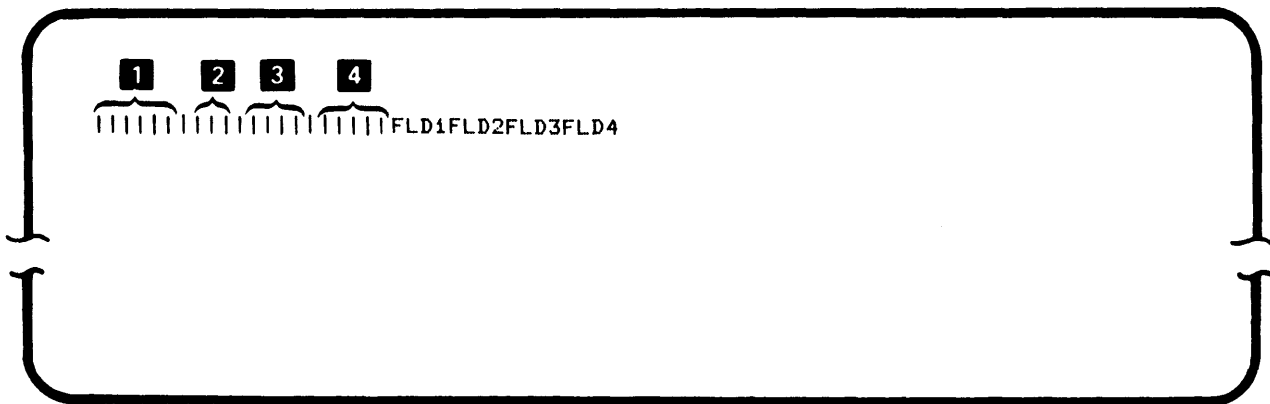


Figure 5-1. A Sample Program Showing the Default Display Format with the Use of Display Attributes

The unformatted display shown in the sample in Figure 5-1 is not format 0 which is referred to as the default format for data shown in some of the modes (such as copy and insert) described in the *DE/RPG Reference Manual*. Format 0 consists of single-character fields that are strung together across the display. The sample in Figure 5-1 illustrates a user-generated format that consists of fields with lengths specified by the person writing the program.

During the update mode, the preceding sample program allows an operator to alter data. All changes in the data are tested against the edits and checks initially imposed on the fields. If an operator tries to enter 1AA23, for example, in FLD1, an error occurs because the new data does not match the pattern specified for the field (alpha, alpha, alpha, digit, digit).

One method of providing flexibility to the program so that edits and checks can be suspended during the update mode is to specify format 0 in the review statement. See the sample in Figure 5-2.

```

00001Z*****
00002Z* PROGRAM 32. FIGURE 5-2 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ OVMASTER TFILE(OVERMAST)
00005Z A1FIRST 1E A1
00006Z R 0
00007A F OVERALL 17 DEVICE(CRT) DSPSIZ(6 80)
00008A R FIRST
00009A FLD1 5C I SHIFT(XXXDD) DSPATR(CS)
00010A FLD2 3 I DSPATR(CS)
00011A FLD3 4 I DSPATR(CS)
00012A FLD4 5 I DSPATR(CS)
00013A F OVERMAST 17 DEVICE(DISK D1)
00014

```

Figure 5-2. A Sample Program Showing the Use of Format 0 to Suppress Edits and Checks

When format 0 is specified in a review statement, each data character forms a field. None of the original edits and checks are enforced. The operator is free to alter the data as needed. When the diskette record is an exact replica of the display record, the display appears the same as the diskette contents.

If you specify format 0 in a review statement of a program, be aware that it displays the diskette contents. If the diskette record has been reformatted, the field order may be different from that used in the initial entry; and, the operator may not be able to identify the fields that are displayed. No literals or prompts can be used with format 0.

The sample in Figure 5-3 shows a formatted display. It takes the first sample and uses the position columns. Notice how the fields are placed exactly where you want them to be in order to present an attractive display.

```

00001Z*****
00002Z* PROGRAM 33. FIGURE 5-3 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ OVMASTER                                TFILE(OVERMAST)
00005Z A1FIRST          1E                        A1
00006Z                R                          A1
00007A          F OVERALL          17            DEVICE(CRT) DSPSIZ(6 80)
00008A          R FIRST
00009A          FLD1              5C  I  1  2SHIFT(XXXDD) DSPATR(CS)
00010A          FLD2              3   I  2  1DSPATR(CS)
00011A          FLD3              4   I  2  20DSPATR(CS)
00012A          FLD4              5   I  3  1DSPATR(CS)
00013A                0  1  10'FLD1'
00014A                0  2  10'FLD2'
00015A                0  2  25'FLD3'
00016A                0  3  10'FLD4'
00017A          F OVERMAST        17            DEVICE(DISK D1)
00018
00019

```

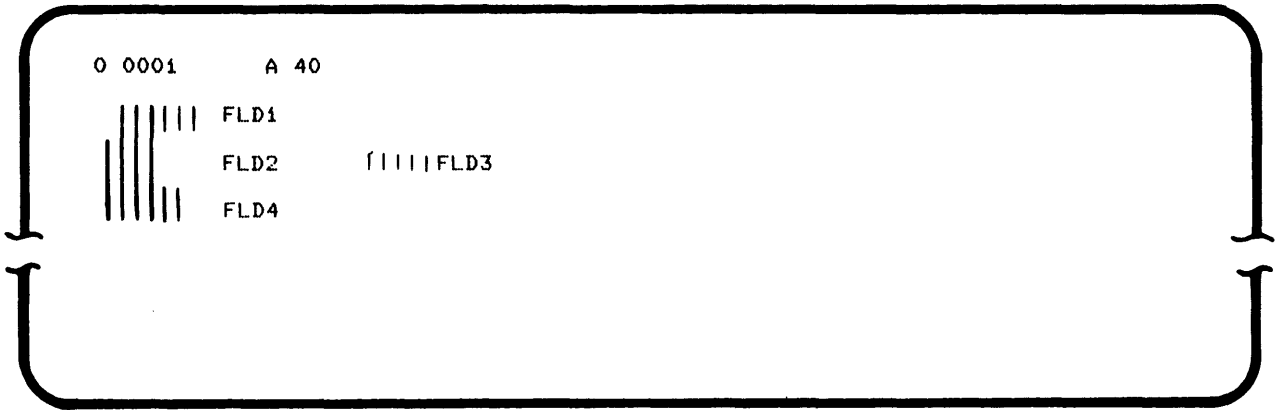


Figure 5-3. A Sample Program Showing a Formatted Display with Display Attributes

In the position columns on the A-specification, the entries in the first three columns specify the row and the second three columns specify the column on the display. If you are not using the Source Entry Program provided with the DE/RPG compiler to enter your program in preparation for a compilation, ensure that the numbers entered in the position columns are right-adjusted with either zero or blank fill.

The positioning of fields on the display does not affect their positioning on the diskette. Other than the sequence for the transaction data set being a replica of the display sequence, the display formats and the diskette formats are independent of one another.

## **USING DIFFERENT DISPLAY FORMATS FOR ENTRY AND REVIEW MODES**

When an experienced operator is entering the same type of data every day, a highly prompted display is unnecessary. The operator does not need to see a display filled with prompting information to guide the entry. The operator simply wants to enter the data and proceed to the next record.

Consider, however, the IBM 5280 used by an operator who only occasionally uses it to enter data into the system. This operator could be an individual in a remote location. The main occupation of the operator may be shipping parts, but occasionally there is a need to use the IBM 5280 to update inventory. A highly prompted and formatted display is not only desirable but necessary to guide the entry.

It is possible that the operator for whom you are writing the program has the need for both types of displays. For the initial entry, he requires a highly prompted, formatted display. For review purposes, he requires only the data fields without formatting. By specifying a separate review format display, this can be accomplished. See the sample in Figure 5-4.

```

00001Z*****
00002Z* PROGRAM 34. FIGURE 5-4 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ INVENT TFILE(OUTPUTX)
00005E1FIRST NE E1 SLNO(2)
00006
00007Z E2SECOND 1E 3
00008Z R E2
00009A F INPUT 11 DEVICE(CRT) DSFSIZ(6 80)
00010A R FIRST
00011A 0001001'THIS IS THE INVENTORY PROGRAM. IT +
00011A IS TO BE USED ONLY WHEN A PURCHASE'
00012A 0002001'HAS BEEN MADE. AT THIS TIME, ENTE+
00012A R THE INFORMATION THAT IS REQUESTED+
00012A ,
00013A 0003001'AND THEN PRESS THE REC ADV KEY. YO+
00013A U WILL SEE A MENU THAT WILL GIVE YO+
00013A U'
00014A 0004001'THE CHOICE OF CONTINUING OR OF END+
00014A ING THE JOB.'
00015A 0005001'PLACE THE ITEM# HERE:'
00016A FLD1 6 I005022
00017A 0005034'PLACE THE PRICE HERE:'
00018A FLD2 5 2I005056
00019A 0005062'PRESS REC ADV'
00020A R SECOND
00021A FLD1 6 I
00022A FLD2 5 2I
00023A F OUTPUTX 11 DEVICE(DISK D1)
00024
00025
00026

```

**4**

```

O 0001 A 40
THIS IS THE INVENTORY PROGRAM. IT IS TO BE USED ONLY WHEN A PURCHASE
HAS BEEN MADE. AT THIS TIME, ENTER THE INFORMATION THAT IS REQUESTED
AND THEN PRESS THE REC ADV KEY. YOU WILL SEE A MENU THAT WILL GIVE YOU
THE CHOICE OF CONTINUING OR OF ENDING THE JOB.
PLACE THE ITEM# HERE:00AAA1 PLACE THE PRICE HERE: 09950 PRESS REC ADV

```

**5**

```

O 0001 A 40
00AAA109950

```

Figure 5-4. A Sample Program Showing Different Displays for the Entry and Review Modes

The entry format used for this program is named FIRST and its ID is E1 **1**. Because the next format ID column contains E1 **2**, the next format, which defines the format to be used for the review mode is not displayed during the entry process; it is manually selected. Before a format can be used for the review mode, it must be defined as an entry mode format **3**.

Notice that SLNO(2) is used. Because this is the default value, using SLNO does not affect the location of the literals on the display. In other words, SLNO in this program does not accomplish anything. The use of SLNO is described later in this chapter.

When you use literals, check the length of the message to ensure that it does not exceed 80 characters. This prevents unnecessary word divisions that make the message difficult to read.

The fully prompted display is only used for initial entry **4**. An unprompted display is used for the review mode **5**. The format statements control this sequence.

## USING THE POSITION COLUMNS TO CREATE PROMPTING TEXT

One of the most useful functions available in providing programs to be used by inexperienced operators is the writing of extensive prompting information on the display. The position columns on the A-specification allow you to design displays that satisfy this requirement. By using the position columns and literals, you can create menus.

The sample program in Figure 5-5 displays the menu; it does not use the operator's entry to select a format. To have the entry control the selection of a format, you must use indicators and the C-specification, or test position and character to test entries on the Z-specification. Chapter 3 contains information about using the test position and character to test to select a format, and Chapter 8 contains information about using indicators to select formats. It is also possible to select programs from menus. To do this, you would include EOJ('programname' device \*PASS) on the entry formats; this would automatically load the appropriate program next. The *DE/RPG Reference Manual* contains additional information about this keyword and its parameters.



```

00001Z*****
00002Z* PROGRAM 35. FIGURE 5-5 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ EXAMPLE TFILE(OUTPUT)
00005Z A1FIRST 1E
00006A F INPUT 80 DEVICE(CRT) DSPSIZ(12 80)
00007A R FIRST
00008A 0001001'ENTER ONE OF THE FOLLOWING NUMBERS-
00008A IN THE FIELD WHERE THE CURSOR IS L-
00008A OCATED'
00009A 0003005'1 SELECT INVENTORY'
00010A 0004005'2 SELECT SOLD TO DATE'
00011A 0005005'3 SELECT TRACE'
00012A 0006005'4 SELECT CUSTOMER MASTER'
00013A 0007005'5 SELECT BILLING'
00014A 1 I010076CHECK(FE)
00015A F OUTPUT 80 DEVICE(DISK D1)

```

```

0 0001 A 40
ENTER ONE OF THE FOLLOWING NUMBERS IN THE FIELD WHERE THE CURSOR IS LOCATED _
1 SELECT INVENTORY
2 SELECT SOLD TO DATE
3 SELECT TRACE
4 SELECT CUSTOMER MASTER
5 SELECT BILLING

```

Figure 5-5. A Sample Program Showing the Design for a Menu

In addition to menus, you can design fill-in-the-blank entries. The sample in Figure 5-6 illustrates this.

```
00001Z*****
00002Z* PROGRAM 36.  FIGURE 5-6 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ EX TFILE(OUTPUT)
00005Z A1FIRST 1E A1
00006A F INPUT 55 DEVICE(CRT) DSPSIZ(6 80)
00007A R FIRST
00008A 0002001 'CUSTOMER NAME:'
00009A 0003001 'STREET ADDRESS:'
00010A 0004001 'CITY:'
00011A 0005001 'STATE:'
00012A 20 I002017PMT(ENTER THE FOLLOWING INFORMATION)
00013A 15 I003017
00014A 10 I004017
00015A 10 I005017
00016A F OUTPUT 55 DEVICE(DISK D1)
```

```
0 0001 A 40
ENTER THE FOLLOWING INFORMATION
CUSTOMER NAME:
STREET ADDRESS:
CITY:
STATE:
```

Figure 5-6. A Sample Program Showing a Design for a Display with Fill-in-the-Blank Entries

As in the menu, the prompts used to guide the operator are not written to the diskette data set. This requires no special action on your part; prompts are never written to the diskette data set.

## USING THE DISPLAY ATTRIBUTES TO FACILITATE ENTRY

If you choose to use the default format, you can use the column separator display attributes to separate the entry fields. The sample program in Figure 5-7 uses the display attributes on a field basis **1** with no field positioning. The display looks as if it contains one long field containing column separators. As you begin to enter data, you see that fields have been defined and that blank spaces indicate the field boundaries.

```

00001Z*****
00002Z* PROGRAM 37. FIGURE 5-7 FROM THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ EXAMPLE TFILE(OUTPUT)
00005Z A1FIRST 1E A1
00006A F INPUT 59 DEVICE(CRT) DSPSIZ(6 80)
00007A R FIRST
00008A 1 FLD1 21 I PMT(ENTER THE CUSOMTER'S NAME, ADDR+
00008A RESS, CITY, AND STATE) DSPATR(CS)
00009A 2 FLD2 16 I DSPATR(CS)
00010A 3 FLD3 11 I DSPATR(CS)
00011A 4 FLD4 11 I DSPATR(CS)
00012A F OUTPUT 59 DEVICE(DISK D1)
    
```

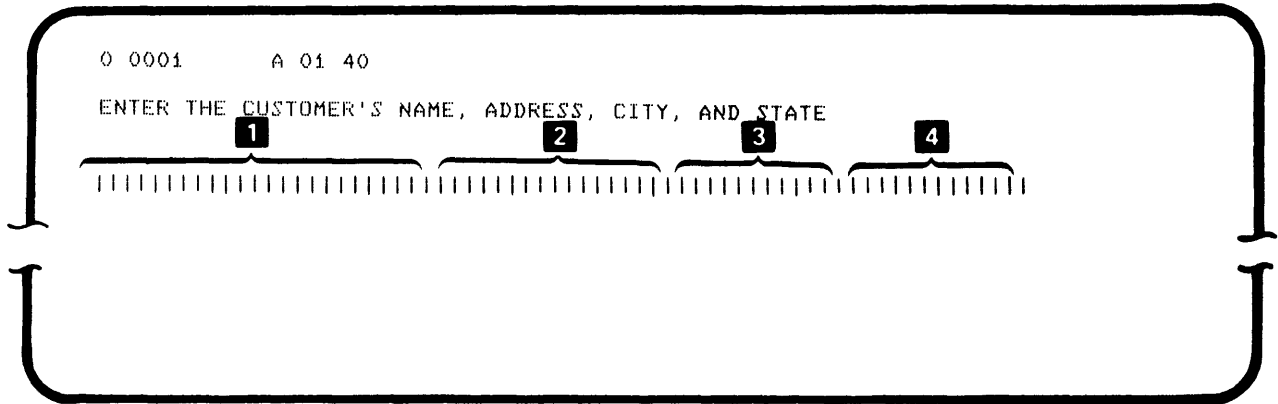


Figure 5-7. A Sample Program Showing the Use of Display Attributes with Unformatted Fields

The sample in Figure 5-8 illustrates the use of reverse image with positioned fields.

```

00001Z*****
00002Z* PROGRAM 38. FIGURE 5-8 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ SAMPLE TFILE(OUTPUT)
00005Z A1FIRST 1E
00006A F INPUT 55 DEVICE(CRT) DSPSIZ(6 80)
00007A R FIRST
00008A
00009A FLD1 20 0001001'CUSTOMER''S NAME:'
00010A 0002001'STREET ADDRESS:'
00011A FLD2 15 1002018DSPATR(RI)
00012A 0003001'CITY:'
00013A FLD3 10 1003008DSPATR(RI)
00014A 0004001'STATE:'
00015A FLD4 10 1004008DSPATR(RI)
00016A F OUTPUT 55 DEVICE(DISK D1)

```

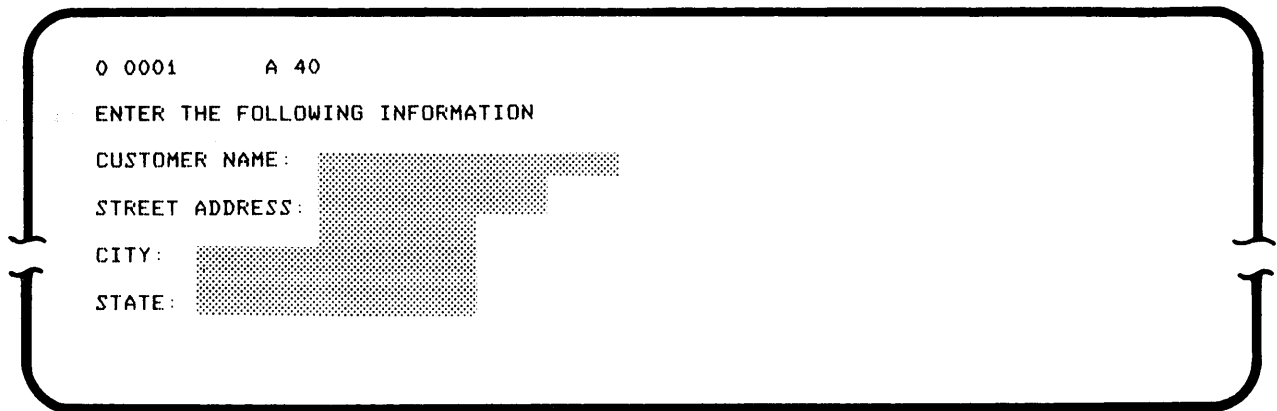


Figure 5-8. A Sample Program Showing the Use of the Reverse Image Display Attribute

As shown by the sample in Figure 5-9, you can use the no display attribute to provide security:

```

00001Z*****
00002Z* PROGRAM 39. FIGURE 5-9 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ SAMPLE TFILE(OUTPUT)
00005Z A1FIRST 1E A1
00006A F INPUT 24 DEVICE(CRT) DSPSIZ(6 80)
00007A R FIRST
00008A GROSS 6 21002002PMT(ENTER EMPLOYEE'S #)
00009A SUBST(EMPT GROSST) DSPATR(ND)
00010A HOURS 3 11003001PMT(ENTER HOURS WORKED)
00011A ALL 10 21004001INSERT(GROSS*HOURS) DSPATR(ND)
00012A F OUTPUT 24 DEVICE(DISK D1)
00013A F REFTAB 12 NUMENT(3)
00014A T EMPT 6 0
00015A T GROSST 6 0
**CTDATA REFTAB
659613250000
459294350000
853026360000

```

Figure 5-9. A Sample Program Showing the Use of the No Display Attribute

All the previous samples in this chapter have demonstrated using field attributes for each field on the display. It is possible to specify field attributes to be shown only on the current field. The attributes remain displayed only as long as the field is not exited. The samples in Figure 5-10 illustrate this technique.

In Sample A of Figure 5-10, the ENTRATR is specified with no EXITATR. The resulting display shows the current field in reverse image. All other fields (except those with field attributes defined) are normal displays. In Sample B of Figure 5-10, an EXITATR is provided. The resulting display shows reverse image only for the current field. The entire display contains column separators.

**Sample A**

```

00001Z*****
00002Z* PROGRAM 40. FIGURE 5-10 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ ATTR                                TFILE(ALL)
00005Z                                      ENTRATR(RI)
00006Z A1ONLY          NE                      A1
00007Z                  R                      A1
00008A          F INPUT          100          DEVICE(CRT) DSPSIZ(6 80)
00009A          R ONLY
00010A          FLD1             10   I   2   2PMT(ENTER A NAME)
00011A          FLD2             8    I   2  20PMT(ENTER A TELEPHONE NUMBER)
00012A          FLD3            15   I   2  40PMT(ENTER THE CREDIT CARD NUMBER)
00013A          FLD4            20   I   3   2PMT(ENTER THE CUSTOMER NAME)
00014A          DSPATR(CS)
00015A          F ALL            100          DEVICE(DISK D1)

```

Figure 5-10 (Part 1 of 2). A Sample Program Showing the Use of the ENTRATR Attribute without the EXITATR Attribute

Sample B

```
Z*****
Z* PROGRAM 41. FIGURE 5-10 SAMPLE B IN THE DE/RPG USER'S GUIDE *
Z*****
00001ZJ ATTR                                TFILE(ALL)
00002Z                                      ENTRATR(RI) EXITATR(CS)
00003Z A1ONLY          NE                    A1
00004Z                  R                    A1
00005A                  F INPUT             100      DEVICE(CRT) DSPSIZ(6 80)
00006A                  R ONLY
00007A                  FLD1                10      I 2 2PMT(ENTER A NAME)
00008A                  FLD2                8       I 2 20PMT(ENTER A TELEPHONE NUMBER)
00009A                  FLD3                15      I 2 40PMT(ENTER THE CREDIT CARD NUMBER)
00010A                  FLD4                20      I 3 2PMT(ENTER THE CUSTOMER NAME)
00011A                  F ALL               100      DEVICE(DISK D1)
```

Figure 5-10 (Part 2 of 2). A Sample Program Showing the Use of the ENTRATR Attribute with the EXITATR Attribute

## USING EDTCDE WITH DISPLAY FORMATS

The sample in Figure 5-11 illustrates the use of EDTCDE for the display. EDTCDE was intended primarily for use with printed data sets to provide appropriate punctuation. Therefore, its use with the display involves some extra manipulation. An EDTCDE to the display can only be performed by using a WRITE operation to the display from a subroutine. All fields in the edited record must be O usage. The sample in Figure 5-11 illustrates a way that you can write a program with input (I-usage) fields and still use EDTCDE.

```
00001Z*****
00002Z* PROGRAM 42. FIGURE 5-11 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ EDITEX TFILE(MINE)
00005Z A1TRANS 1E A2 CLRL(*ND)
00006Z A2CALCS 1E A1
00007Z R A1
00008A F INPUT 18 DEVICE(CRT) DSPSIZ(6 80)
00009A R TRANS
00010A FIRST 9 2I 2 30PMT(ENTER THE AMOUNT OF THE BALANCE)
00011A SECOND 9 2I 3 30PMT(ENTER THE AMOUNT OF THE PAYMENT)
00012A R PATT
00013A FIRST 9 20 4 30EDTCDE(K '$')
00014A SECOND 9 20 5 30EDTCDE(K '$')
00015A F MINE 18 DEVICE(DISK D1)
00016C CALCS BEGSR
00017C WRITEPATT
00018C ENDSR
```

Figure 5-11 (Part 1 of 2). A Sample Program Showing Currency Edition on the Display

0 0001      N 09 40 000007      A1 E

ENTER THE AMOUNT OF THE BALANCE

-

0 0001      N 09 40 000008      A1 E

ENTER THE AMOUNT OF THE BALANCE

55555555

55555555

\$5,555,555.55

\$5,555,555.55

Figure 5-11 (Part 2 of 2). A Sample Program Showing Currency Editing on the Display



## USING CLRL AND SLNO TO DISPLAY MULTIPLE RECORDS

The sample in Figure 5-12 illustrates a program that uses the CLRL and SLNO keywords to display multiple records simultaneously.

This program is presenting the operator with a format that allows him to enter name and address information. When the operator completes the information and uses the Rec Adv key, it appends the second record, containing a request for the driver's license and telephone number, to the first record. The first record continues to be displayed. When the second record is completed and the operator presses the Enter key, the process starts again.

```

00001Z*****
00002Z* PROGRAM 43. FIGURE 5-12 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ CLEAR TFILE(WHEE)
00005Z A1FIRST 1E A2 CLRL(2)
00006Z A2SECOND 1E A1 SLNO(4)
00007A F INPUT 20 I DEVICE(CRT) DSPSIZ(6 80)
00008A R FIRST
00009A FLD1 10 I 1 20DSPATR(CS)
00010A FLD2 10 I 2 20DSPATR(CS)
00011A 0 1 2'NAME' DSPATR(BL HI)
00012A 0 2 2'ADDRESS' DSPATR(BL HI)
00013A R SECOND
00014A FLD3 10 I 1 30DSPATR(CS)
00015A FLD4 10 I 2 30DSPATR(CS)
00016A 0 1 2'DRIVER'S LICENSE NO.'
00017A DSPATR(BL HI)
00018A 0 2 2'TELEPHONE NUMBER' DSPATR(BL HI)
00019A F WHEE 20 DEVICE(DISK D1)
00020

```

```

0 0001      A 40
NAME        BEV MACHAR
ADDRESS     ROCHESTER

```

Figure 5-12 (Part 1 of 2). A Sample Program Showing the Use of CLRL and SLNO on the Display

```
0 0001      A 40
NAME          BEU MACHAR
ADDRESS       ROCHESTER
DRIVER'S LICENSE NO.  M-78-99013
TELEPHONE NUMBER 1-612-5532
```

```
0 0001      A 40
NAME          |
ADDRESS       |
DRIVER'S LICENSE NO.  M-78-99013
TELEPHONE NUMBER 1-612-5532
```

```
0 0001      A 40
NAME          NEW REMET
ADDRESS       HAMMOND
DRIVER'S LICENSE NO. |
TELEPHONE NUMBER |
```

Figure 5-12 (Part 2 of 2). A Sample Program Showing the Use of CLRL and SLNO on the Display



## Chapter 6. Formatting Techniques for the Diskette Data Set

The method used to write the record into the diskette data set determines what is required to format the diskette data set. There are basically two ways to write a record into a diskette data set: (1) implicitly for transaction data sets (a record is automatically written in a transaction data set when completed and WRITE(\*NO) is not specified) and (2) explicitly for application programs (records are written as specified in the subroutines).

You learned some reformatting techniques in Chapter 5. This chapter provides two additional techniques for reformatting diskette data sets.

1. Merging fields from multiple records into a single diskette record
2. Positioning fields within a diskette data set

## MERGING FIELDS FROM MULTIPLE RECORDS

It is possible to merge fields from multiple records in data sets created by either the transaction file or by subroutines. Both techniques are illustrated in this chapter. The first one that is presented is for data sets created by subroutines.

In the sample program in Figure 6-1, the HEADER record 1 and the DETAIL record 2 are completed before the subroutine is called 3. When the exit to the subroutine is made, two operations are performed: (1) two fields from the DETAIL record are multiplied and a new field is created to hold the contents 4; (2) a new record, which consists of fields from the HEADER record and the DETAIL record and the new field from the subroutine, is written in another diskette data set 5. The entries in the repeat columns on the Z-specification determine the sequence of the formats 6.

```

00001Z*****
00002Z* PROGRAM 44. FIGURE 6-1 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004Z! SMPLEX                                TFILE(EXMP)
00005 1E1HEADER 6 1E                          E2
00006 2E2DETAIL NE                             E1
00007 F INPUT 67                               DEVICE(CRT) DSFSIZ(6 80)
00008A R HEADER
00009A FLD1 30 I PMT(ENTER CUSTOMER NAME)
00010A FLD2 6 I PMT(ENTER CUSTOMER#)
00011A FLD3 30 I PMT(ENTER ADDRESS)
00012A FLD4 1 I INSERT('H')
00013A R DETAIL
00014A FLD5 6 PMT(ENTER ITEM#)
00015A FLD6 30 PMT(ENTER DESCRIPTION)
00016A FLD7 5 0I PMT(ENTER QUANT PURCHASED)
00017A FLDB 5 2I PMT(ENTER PRICE) EXSR(OUT) 3
00018A F FINAL 55 DEVICE(DISK D1)
00019A R OUTPUT
00020A FLD1
00021A FLD5
00022A FLD7
00023A FLDB
00024A FLD9
00025A F EXMP 67 DEVICE(DISK D1)
00026C OUT REGSR
00027C FLD7 MULT FLDB 4 92
00028C WRITEOUTPUT 5
00029C ENDSR

```

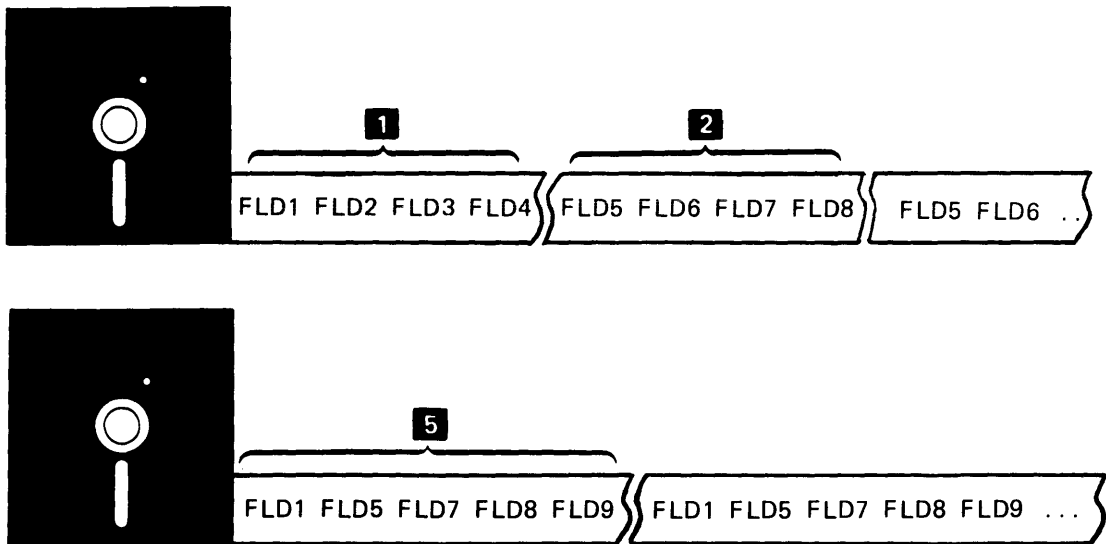


Figure 6-1. A Sample Program Showing the Merging of Fields from Multiple Records in a Data-Entry Program

The sample program in Figure 6-2 illustrates merging fields from different records into one diskette record using a transaction file. The two records that are used to provide operator entry (HEAD1 and DET1) **1** are used once each. After these have been completed, the record that merges fields from the preceding records is accessed **2**. The fields are supplied in the last record by the insert operation **3**. The operator sees the field contents, but he is not able to alter them in this record. If the automatic record advance function is inactive, the record remains on the display until the operator presses the Enter key.

```

00001Z*****
00002Z* PROGRAM 45. FIGURE 6-2 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ TRY TFILE(FINAL)
00005 F1HEAD1 1E F2 WRITE(*NO)
00006 1 F2DET1 1E F3 WRITE(*NO)
00007 2 F3ALL 1E F1
00008A F INPUT 28 DEVICE(CRT) DSPSIZ(6 80)
00009A R HEAD1
00010A FLD1 5 2I PMT(ENTER FLD1) DSPATR(CS)
00011A FLD2 3 I PMT(ENTER FLD2) DSPATR(RI)
00012A FLD3 4 I PMT(ENTER FLD3) DSPATR(CS)
00013A R DET1
00014A FLD4 6 I PMT(ENTER FLD 4) DSPATR(RI)
00015A FLD5 10 I PMT(ENTER FLD5) DSPATR(CS)
00016A R ALL
00017A 10 I INSERT(FLD5)
00018A 5 2I INSERT(FLD1)
00019A 3 I 3 INSERT(FLD2)
00020A 4 I INSERT(FLD3)
00021A 6 I INSERT(FLD4)
00022A F FINAL 28 DEVICE(DISK D1)

```

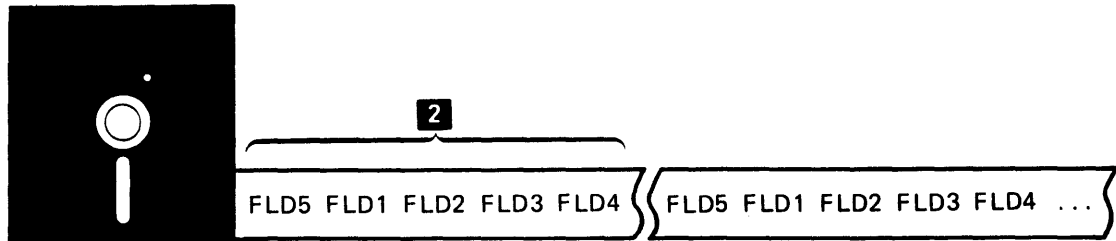


Figure 6-2. A Sample Showing the Merging of Fields from Multiple Records in a Data-Entry Program without a C-Specification

It is difficult to verify this diskette record for three reasons: (1) the fields are not in the order in which they were entered, (2) the fields are now in one record rather than in the initial two records, and (3) the fields were supplied for the diskette record by an automatic function (INSERT) which the operator cannot alter.

In addition to merging fields from various records, it is also possible to drop fields by simply not including them in the new record.

## **POSITIONING FIELDS WITHIN A DATA SET**

Earlier in this chapter, a description of padding a diskette record (with the CHECK(BY) operation) was provided. One of the reasons for padding a record is to ensure that a record marker is in a certain position in the diskette record.

Using the position columns for a reformatted record, it is possible to place each field where you want it in a diskette record. The following sample program in Figure 6-3 illustrates how this is accomplished.



```

00001Z*****
00002Z* PROGRAM 46. FIGURE 6-3 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ INQUIRE TFIL(LOOK)
00005Z AOIDENT 1E A1
00006Z A1MENU 1E *PDS0001 '1' A2 SLNO(3) WRITE(*NO)
00007Z E *PDS0001 '2' A3
00008Z A2ITEMINF NE A0 WRITE(ITEM)
00009Z A3SHIPINF 1E A0
00010Z R *PDS0061 'I' A2
00011Z R *PDS0061 'S' A3
00012A F INPUT 61 DEVICE(CRT) DSPSIZ(6 80)
00013A R IDENT
00014A FLD1 30X I PMT(ENTER THE CUSTOMER'S NAME)
00015A FLD2 6D I PMT(ENTER THE CUSTOMER ID)
00016A R MENU
00017A 0001001'1 ENTER ITEM# AND PRICE DATA'
00018A 0002001'2 ENTER SHIPTO DATA'
00019A FLD3 1 I003001CHECK(DR)
00020A R ITEMINF
00021A FLD4 6 I PMT(ENTER ITEM#)
00022A FLD5 5 2I PMT(ENTER PRICE)
00023A FLD6 3 0I PMT(ENTER QUANTITY)
00024A FLDA 1 I INSERT('I')
00025A R SHIPINF
00026A FLD8 30 I PMT(ENTER NAME)
00027A FLD9 30 I PMT(ENTER ADDRESS)
00028A FLD10 1 I INSERT('S')
00029A F LOOK 61 DEVICE(DISK D1)
00030A R ITEM
00031A FLD6 3 0 010
00032A 2 FLD5 5 2 020 1
00033A FLD4 6 030
00034A FLDA 1 061

```

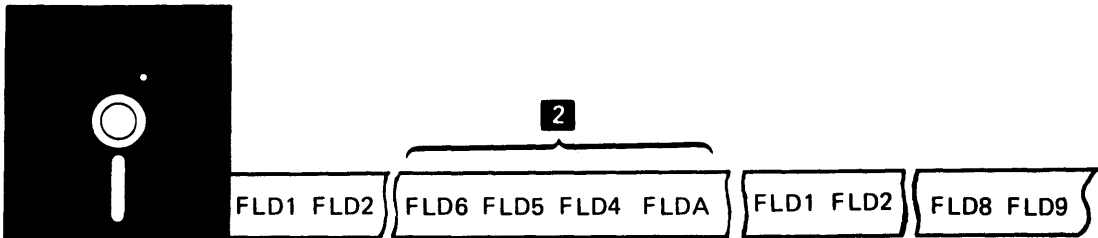


Figure 6-3. A Sample Program Showing the Placing of Fields as Specified in a Diskette Data Set

This sample program is one that was used earlier in the manual. Previously the CHECK(BY) operation had been used to position the record mark (I) in the proper location. This time you reposition the fields to place the record mark in the correct location.

Notice that the reformatted field descriptions use the position columns **1**. When an entry is contained in a reformatted diskette field description, the result is the placement of the field in the record beginning at the position number. For example, FLD5 begins in position 20 **2** of the diskette record.

If the position entry overlaps the field lengths, the field positions are overlaid. For example, if FLD6 were 20 positions long, began in position 10, and was followed by FLD5 which started in position 20, the last 10 positions of FLD6 would be overlaid by the first 10 positions of FLD5.

This concludes the chapter on formatting. Related topics are *Access Methods* in Part 4 and *Creating Data Sets* in Part 2.

## CONVERTING NUMERIC DATA

### Numeric Data Formats

Numeric data on a diskette file may be in zoned decimal, packed decimal, or binary format. To have DE/RPG convert data from one format to another format, use the Data Type column on a field description line.

Figure 6-4 shows how a record containing packed and binary data can be converted to a record containing only zoned fields. Chapter 10 of the *IBM 5280 DE/RPG Reference Manual* contains detailed descriptions of zoned decimal, packed decimal, and binary data formats.

```

00001Z*****
00002Z* PROGRAM 82. FIGURE 6-4 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ EXPAND
00005Z X1MAIN          1E                                EOJ
00006A                F CRT                          80      DEVICE(CRT)
00007A                R MSGGO
00008A                R MSGEND                        0 2    2'EXPAND PROGRAM IN PROCESS'
00009A                R MSGEND                        0 2    1'EXPAND PROGRAM COMPLETE'
00010A                F IN80                          80      DEVICE(DISK D1)
00011A                R INREC
00012A                DATA1                          10 0
00013A                DATA2                          9F 0
00014A                DATA3                          5B 0
00015A                F OUT80                          80      DEVICE(DISK D2)
00016A                R OUTREC
00017A                DATA1
00018A                DATA2
00019A                DATA3
00020A                MAIN                            BEGSR
00021C                EXFMTMSGGO
00022C                TAG
00023C                LOOP                            READ INREC
00024C                01                             GOTO DONE
00025C                WRITEOUTREC                    01
00026C                GOTO LOOP
00027C                GOTO LOOP
00028C                DONE                            TAG
00029C                EXFMTMSGEND
00030C                ENDSR

```

Figure 6-4. A Sample Program Showing How to Convert Numeric Data

The input data records have the following format:

Bytes 1-10	Data1	Zoned
Bytes 11-15	Data2	Packed
Bytes 16-19	Data3	Binary
Bytes 20-80	unused	

The output data records have the following format:

Bytes 1-10	Data1	Zoned
Bytes 11-19	Data2	Zoned
Bytes 20-24	Data3	Zoned
Bytes 25-80	unused	



t

t

## Part 3. Using Indicators

This part of the manual contains information about using indicators to condition operations for data-entry and application programs. It consists of two chapters:

- Chapter 7. Using Indicators in Data-Entry Programs
- Chapter 8. Using Indicators in Application Programs

In general terms, indicators are flags that condition program functions. In DE/RPG, indicators are denoted by numbers 01 through 99; they allow you to program automatically conditioned operations. Indicators can be both set and used on either the A- or C-specifications. As the name indicator implies, they have two conditions: off and on. To be set on, a test on a conditioning operation must be successfully made or a SETON operation must be specified. A conditioning failing operation or a SETOF operation sets the indicator off.

The condition of the indicator (off or on) determines the actions that can be accomplished against the field using the indicator. To understand how this works, look at the following sample.

```
00001Z*****
00002Z* PROGRAM 47. FIGURE 7-1 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ INDIC
00005Z A1START          NE          A1          WRITE(*NO)
00006A          F INPUT          20          DEVICE(CRT) DSPSIZ(6 80)
00007A          R START
00008A          GROSS            5 2I       PMT(ENTER THE GROSS AMOUNT)
00009A          COMP(GT 800.00 01)
00010A          TEMP            5 2I       AUXDUP(GROSS)
00011A 01          CHECK(BY)
00012A          CALC            5 2I       INSERT(TEMP*1.5)
00013
```

Figure 7-1. A Sample Program Showing Setting an Indicator on the A-Specification

Suppose that every time the pay amount for an employee exceeds \$800.00, you want to multiply the gross amount by 1.5. You would use a compare function as shown in the sample to determine when the gross exceeds 800.00. When the gross does exceed 800.00, indicator 01 is turned on. When the Auto Dup function is active, the gross amount is duplicated in a field which is multiplied by 1.5; otherwise, it is not duplicated and the amount in the field named CALC is 0.

This sample is for a data-entry program. Additional samples for data-entry and application programs are included in the following chapters.

## Chapter 7. Using Indicators for Data-Entry Programs

There are two uses for indicators: conditioning the indicators (setting them on or off) and conditioning an operation with the indicators. In other words, to use indicators effectively, there must be some action that either sets them on or off and there must be a succeeding action that is performed or not performed based on the condition (on or off) of the indicator.

The act of setting indicators on or off is called conditioning. The actions that affect the state of the indicators are called conditioning operations.

Indicators on the A-specification can only be set on by a successful compare operation or by using the SETON keyword. The sample in Figure 7-2 illustrates one method for setting an indicator on and another for setting it off.

```

00001Z*****
00002Z* PROGRAM 48. FIGURE 7-2 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ INDEX
00005Z HOREC1          1E                      WRITE(*NO) SLNO(2)
00006A          F DISPEX          6          DEVICE(CRT) DSPSIZ(6 80)
00007A          R REC1
00008A          D001001 'GROSS AMOUNT PAID'
00009A          GROSS          6 01001 2 COMP(GT 800 01) 3
00010 1 01          ERROR(01 'GROSS IS TOO LARGE') 4
00011A          *RTN          5 SETOF(01)
00012

```

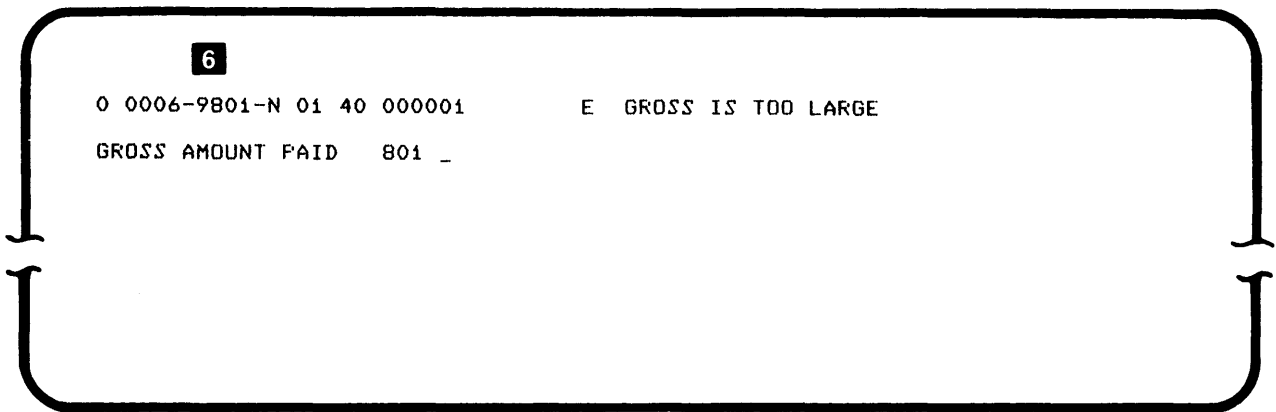


Figure 7-2. A Sample Program Showing the Use of Indicators to Write an Error Message

In this sample, the indicator 1 is being set on whenever the amount an employee earns is greater than 800. The operation that conditions the indicator is a compare 2 similar to the one in Figure 7-1. Whenever the indicator is set on 3, it conditions an operation that sets an error condition 4. Once the error condition has been reset, the indicator is set off by the SETOF keyword 5.

The program in Figure 7-2 allows you to determine that you do not want the entry to exceed a specified amount. In this sample, the amount is 800. Every time an entry exceeding 800 occurs, the operator sees a 4-digit error code (98xx) flashing on the status line 6. To display the text for the error message, he must press the Help key while the error is flashing. The text for the error message appears approximately midway on the status line.

Only two operations can be conditioned by indicators on the A-specification: bypassing a field and writing an error message. The sample in Figure 7-1 illustrates a program that uses an indicator in a data-entry program to condition a field and the sample in Figure 7-2 illustrates a program that uses an indicator to write an error message.

The sample in Figure 7-3 illustrates a typical payroll data-entry program in which indicators might be used effectively both to bypass fields and to write errors.

```

00001Z*****
00002Z* PROGRAM 49. FIGURE 7-3 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ INDICAT TFILE(PAYEX)
00005Z Z1RECEX NE Z1
00006Z R Z1
00007A F DISPLAY 60 DEVICE(CRT) DSPSIZ(6 80)
00008A R RECEX
00009A 0001001'TOTAL HOURS:'
00010A 0001020'OVERTIME HOURS:'
00011A 0001041'SICK HOURS:'
00012A 0001059'HOLIDAY HOURS:'
00013A 0003001'HOURLY RATE:'
00014A 0003020'BASE PAY:'
00015A 0003041'OVERTIME PAY:'
00016A 0004001'SICK PAY:'
00017A 0004020'HOLIDAY PAY:'
00018A TOT 3 1I001014DSPATR(UL) CHECK(ME RZ)
00019A COMP(EQ 0 11)
00020A 11 ERROR(30 'YOU MUST ENTER THE HOURS')
00021A OVER 3 1I 1 36DSPATR(UL)
00022A CHECK(FE RZ) 1
00023A 2 COMP(EQ 0 12)
00024A SICK 3 1I001052DSPATR(UL)
00025A CHECK(FE RZ) 8
00026A 3 COMP(EQ 0 13)
00027A HOL 3 1I001074DSPATR(UL) CHECK(FE RZ)
00028A 4 COMP(EQ 0 14)
00029A TEMP1 5 2I 5 1PMT(ENTER OVERTIME RATE)
00030A 12 CHECK(BY)
00031A TEMP2 5 2I 5 10PMT(ENTER SICK PAY RATE)
00032A 13 CHECK(BY)
00033A TEMP3 5 2I 5 20PMT(ENTER HOLIDAY RATE)
00034A 14 CHECK(BY)
00035A HOUR 3 2I003016CHECK(DR) DSPATR(UL)
00036A BASE 5 2I003070INSERT(HOUR*TOT) DSPATR(UL)
00037A OVERTI 4 2I003 5 INSERT(TEMP1*OVER) DSPATR(UL)
00038A SICKTI 5 2I004011INSERT(TEMP2*SICK) DSPATR(UL) 6
00039A HOLTI 5 2I004 7 INSERT(TEMP3*HOL) DSPATR(UL)
00040A GROSS 5 2I004060INSERT(BASE+OVERTI+SICKTI+HOLTI)
00041A 9 SETOF(11) SETOF(12)
00042A SETOF(13) SETOF(14)
00043A F PAYEX 60 DEVICE(DISK D1)

```

Figure 7-3. A Sample Program Showing the Complex Use of Indicators on the A-Specification

This program is prompting the operator for data about the hours an employee works. If the operator enters any nonzero value **1** in the overtime hours **2**, sick hours **3**, or holiday hours fields **4**, the program automatically provides a corresponding calculation **5**, **6**, and **7** that determines the correct amount of pay.



If the operator does not enter anything in the field, it is filled with zeros. The CHECK(RZ) **3** operation ensures this so that an accurate test can be made against the fields. Notice that a SETOF **4** operation is specified for each indicator. If the indicators are not reset, they remain on for the next record. When SETOF is used, the operator must be careful about using the backspace function. The condition of the indicators might may not be accurate. Using the rerun mode at the conclusion of the entry operation is one way to ensure accurate calculations and totals.

In conclusion, there are two things to remember when using indicators on the A-specification:

1. Successful tests turn indicators on.
2. When an indicator is on, either the field is bypassed or an error message is displayed.

## Chapter 8. Using Indicators for Application Programs

Indicators can be set in a variety of ways on the C-specification. The resulting indicator columns (54 through 59) determine the status of the operation that sets the indicator on. For example, these columns set indicators to signify that an arithmetic calculation has resulted in a positive, negative or zero result or that the end of the file has been reached on a READ operation. The same columns are also used by the compare operands to indicate that the result of a compare operation is high, low, or equal to the field being compared. Some I/O operations (such as READ) require a resulting indicator. The *DE/RPG Reference Manual* provides detailed information about conditioning indicators and about operations conditioned by indicators.

## **OPERATIONS THAT SET INDICATORS AND OPERATIONS CONDITIONED BY INDICATORS**

The following chart lists the operations that set indicators and the operations that are conditioned by indicators:

<b>Operations that Set Indicators</b>	<b>Operations Conditioned by Indicators</b>
ADD	All except
Z-ADD	TAG
SUB	ENDSR
MULT	BEGSR
DIV	
MVR	
COMP	
CAB	
CABEQ	
CABNE	
CABLE	
CABLT	
CABGE	
CABGT	
TESTB	
LOKUP	
READ	
READP	
CHAIN	
WRITE	
DELET	
SETLL	
OPEN	
CLOSE	
SETON	
UPDAT	
SETOF	
Z-SUB	
OPEN	

The operations that set indicators on the C-specifications fall into three general categories: I/O operations, arithmetic operations, and branching operations. Examples of conditioning each of these types of operations are included in this chapter.

## **A SAMPLE OF USING INDICATORS WITH I/O OPERATIONS**

Some of the I/O operations that set indicators are:

- Reading the last record in a data set
- Performing a chain operation when no record with the specified key exists in the data set

The sample in Figure 8-1 illustrates a simple program that uses indicators for I/O operations.

```

00001 Z*****
00002 Z* PROGRAM 50. FIGURE 8-1 FOR THE DE/RPG USER'S GUIDE *
00003 Z*****
00004 ZJ EXPLIND
00005 Z A1BEGIN          1E                      EOJ
00006 A              F ITMAST          25          DEVICE(DISK D1)
00007 A              R ITEM
00008 A              K CUSTN           4
00009 A              ITEM#            6
00010 A              QUANT             4
00011 A              PERPRI           5
00012 A              COST             6
00013 A              F DISP           25          DEVICE(CRT) DSPSIZ(6 80)
00014 A              R LOOK
00015 A              NUMBER           4   I   1PMT(ENTER THE NUMBER)
00016 A              R OK
00017 A              CUST#            4   B   INSERT(CUSTN)
00018 A              ITEM1            6   B   INSERT(ITEM#)
00019 A              QUANT1           4   B   INSERT(QUANT)
00020 A              PERPR1           5   B   INSERT(PERPRI)
00021 A              COST1            6   B   INSERT(COST)
00022 C              BEGIN            BEGSR
00023 C              BRANCH            TAG
00024 C
00025 C              2NUMBER            EXFMTLOOK          0105
00026 C              3N01              CHAINITEM
00027 C              N05                4GOTO BRANCH
00028 C              ENDSR

```

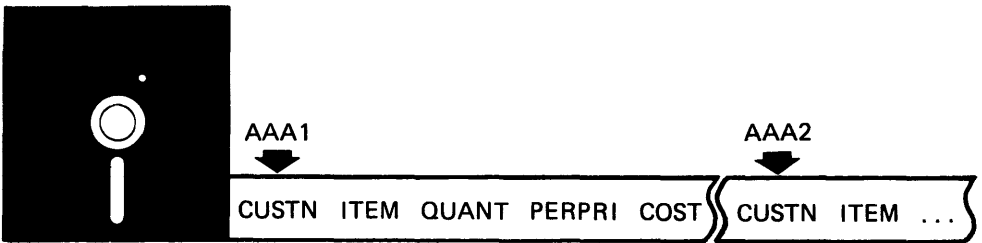
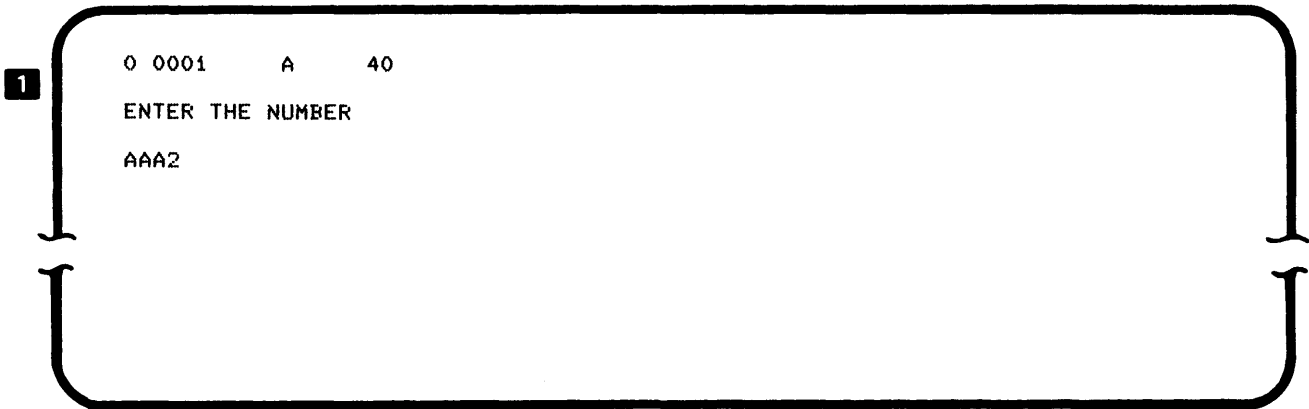


Figure 8-1 (Part 1 of 2). A Sample Program that Uses Indicators with I/O Operations

```
O 0001      A 40
AAA2584200002000010000200
```

Figure 8-1 (Part 2 of 2). A Sample Program that Uses Indicators with I/O Operations

This program prompts the operator for an identification number **1**. When the operator enters the number, the program exits to the C-specification and uses the number to locate the corresponding record on the diskette data set **2**. When a record with a matching identification value is found, indicator O1 is off and a display appears containing the value of the fields in that record **3**. When no matching record is found, indicator O1 is on **4**, and a display appears containing the prompt for another identification number **1**.

This process continues until the operator uses the End of Job key or until the last record is reached.

The sample in Figure 8-2 expands the program from Figure 8-1.

```

00001Z*****
00002Z* PROGRAM 51. FIGURE 8-2 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ EXPLIND
00005Z A1BEGIN 1E EOJ
00006A F DISP 7 DEVICE(CRT) DSPSIZ(6 80)
00007A R ONE
00008A NUMBER 6 I PMT(ENTER THE NUMBER ID)
00009A 1 R LOOK 3
00010A ITEMNU 6 I PMT(ENTER THE ITEM NU. IF CHGD.)
00011A FLDX 1 I 4 PMT(ENTER X FOR LAST ONE) DSPATR(RI)
00012A F ITMAST1 27 DEVICE(DISK D1)
00013A R ITEM
00014A K CUSTN 6
00015A ITEMNU 6
00016A QUANT 4
00017A PERPRI 5
00018A COST 6
00019C BEGIN BEGSR
00020C BRANCH TAG
00021C EXFMTONE
00022C 2 NUMBER CHAINITEM 01
00023C N01 EXFMTLOOK
00024C N01 3 UPDATITEM
00025C FLDX COMP 'X' 09
00026C 5 09 GOTO END
00027C GOTO BRANCH
00028C END ENDSR

```

Figure 8-2. A Sample Program Showing Multiple Use of Indicators for I/O Operations

This program does not display fields **1** from the record that matches the identification provided by the operator **2**, but it does allow the operator to alter the value of a field in the matching record **3**. Only the fields for records with matching identifiers are displayed. The operator terminates the program with an entry in a field on the display **4**. An indicator controls this function **5**.

## A SAMPLE OF USING INDICATORS WITH ARITHMETIC OPERATIONS

Some of the arithmetic operations that condition indicators are:

- A positive result for an arithmetic operation
- A negative result for an arithmetic operation
- A zero result for an arithmetic operation

The samples in Figures 8-3 and 8-4 illustrate using indicators with arithmetic operations to set and use indicators.

```

00001Z*****
00002Z* PROGRAM 52. FIGURE 8-3 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ EXPLIND
00005Z A1BEGIN          1E                      EOJ 7
00006A          F DISP          6          DEVICE(CRT) DSPSIZ(6 80)
00007A          R LOOK
00008A          NUMBER          6 I 2 PMT(ENTER THE BEGINNING NUMBER)
00009A          R MSG
00010A          1 I          PMT(WARNING. THE VALUE IS 0)
00011A          F ITMAST7        27          DEVICE(DISK D1)
00012A          R ITEM
00013A          K CUSTN          6
00014A          ITEM            6
00015A          QUANT           4
00016A          PERPRI          5
00017A          COST            6 2
00018A          F OTTER          15          DEVICE(DISK D1)
00019A          B EXMPB
00020A          1 INTER          15 2
00021C          BEGIN          BEGSR
00022C          EXFMTLOOK
00023C          N05NUMBER 3 CHAINITEM          01
00024C          BRANCH          TAG
00025 8 N01          5 Z-ADDCOST 4 INTER 152 02
00026 02          EXFMTMSG
00027 9 N01          6 WRITEEXMPB
00028C N01          READ ITEM          05
00029 10 N05N01          GOTO BRANCH
00030C          ENDSR

```

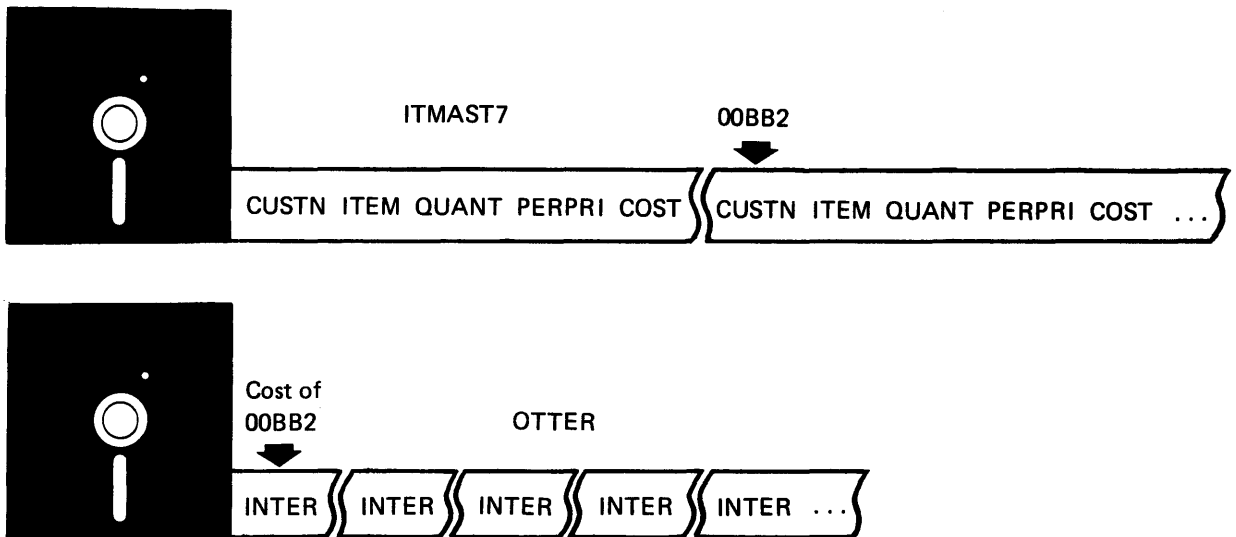


Figure 8-3. A Sample Program Showing the Use of Indicators with Arithmetic Operations

This program is creating a new data set that contains a field 1 from each record in the original data set beginning with the record that matches the identifier entered by the operator and ending with the last record. The program prompts the operator for an identification value 2. It uses this value to locate the beginning record in the data set 3. When a matching record is found, the program takes a field from the record 4, adds it to zero 5, and writes it out to the new data set 6. When the last record is reached, the program automatically terminates 7. Indicators control when fields are added to zero 8, when they are written in the new data set 9, and when the program terminates 10.



The program in Figure 8-4 is another version of the program in Figure 8-3.

```

00001 Z*****
00002 Z* PROGRAM 53. FIGURE 8-4 IN THE DE/RPG USER'S GUIDE *
00003 Z*****
00004 ZJ EXPLIND
00005 Z A1BEGIN 1E EOJ
00006 A F DISP 7 DEVICE(CRT) DSPSIZ(6 80)
00007 A R LOOK
00008 A NUMBER 6 I PMT(ENTER THE NUMBER ID)
00009 A F ITMASTF 27 DEVICE(DISK D1)
00010 A R ITEM
00011 A K CUSTN 6 1
00012 A ITEM# 6
00013 A QUANT 4
00014 A PERPRI 5
00015 A COST 6 2
00016 A F OUTGO 15 DEVICE(DISK D1)
00017 A R LAST
00018 A INTER 15 2
00019 C BEGIN BEGSR
00020 C AGAIN TAG
00021 C EXFMTLOOK
00022 C NUMBER CHAINITEM 02
00023 C 02 GOTO AGAIN
00024 C Z-ADDCOST INTER 152 03
00025 C BRANCH TAG
00026 C READ ITEM 05 4
00027 C 1 N05N03COST ADD INTER INTER
00028 C 2 N05 GOTO BRANCH
00029 C 3 WRITELAST
00030 C END ENDSR

```

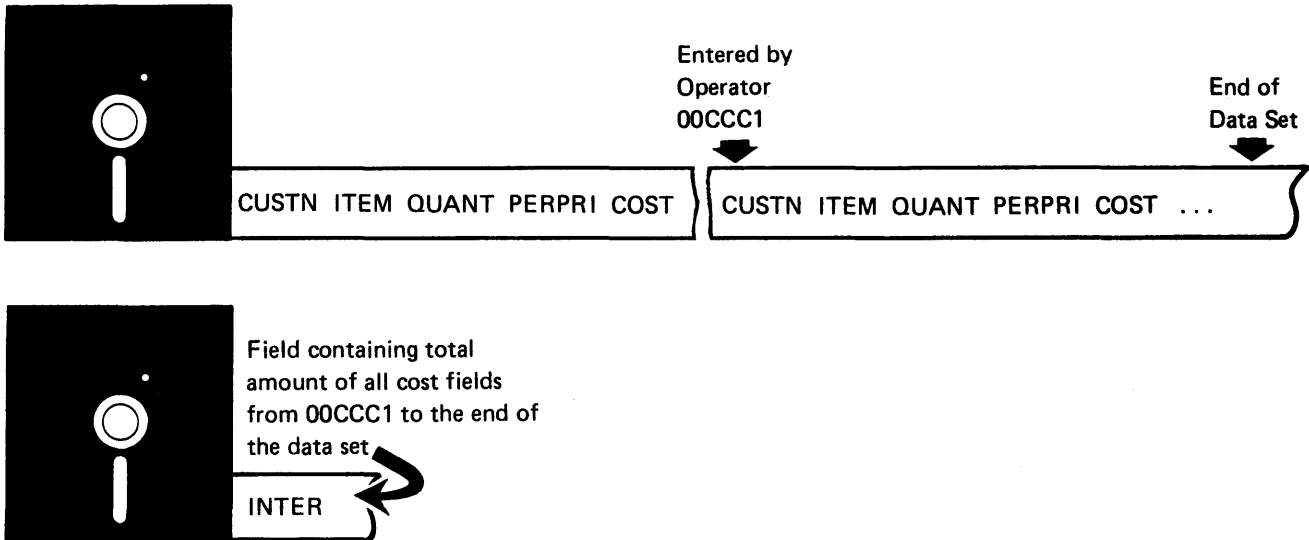


Figure 8-4. A Sample Program Showing the Use of Indicators to Accumulate a Batch Total

In Figure 8-4, the program is accumulating an online total of fields in the original data set beginning with the record specified by the operator's identifier and ending with the last record in the data set. The new data set that is written, contains one field which represents the total value of the fields that have been read from the original records.

In this sample, indicators are being used to condition arithmetic operations **1**, branching **2**, writing to the new data set **3**, and program termination **4**.

## A SAMPLE OF USING INDICATORS WITH BRANCHING OPERATIONS

Some of the branching operations that condition indicators are:

- The value in Factor 1 is greater than the value in Factor 2.
- The value in Factor 1 is less than the value in Factor 2.
- The value in Factor 1 is equal to the value in Factor 2.

The sample in Figure 8-5 illustrates using compare operations with indicators.

```

00001Z*****
00002Z*PROGRAM 54. FIGURE 8-5 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ EXPLIND
00005Z A1BEGIN 1E EOJ
00006A F DISP 16 DEVICE(CRT) DSPSIZ(6 80)
00007A R LOOK
00008A NUMBER 6 I PMT(ENTER THE FIRST NUMBER)
00009A NUMB2 6 I PMT(ENTER THE LAST NUMBER)
00010A R DONE
00011A 15 2I INSERT(INTER)
00012A 1 I PMT(THIS IS THE INTER RESULT. +
00012A PRESS ENTER TO CONTINUE)
00013A R TEST1
00014A 6 I INSERT(CUSTN)
00015A 6 I INSERT(NUMBER)
00016A 1 I PMT(FIRST IS CUSTN THEN NUMBER)
00017A R TEST2
00018A 15 2I INSERT(INTER)
00019A 1 I PMT(THIS IS INTER WHEN ADDED TO +
00019A COST)
00020A F ITMAST 27 DEVICE(DISK D1)
00021A R ITEM
00022A K CUSTN 6
00023A ITEM 6
00024A QUANT 4
00025A PERPRI 5
00026A COST 6 2
00027C BEGIN BEGSR
00028C HERE TAG
00029C EXFMTLOOK
00030C NOSNUMBER CHAINITEM 01
00031C NO1 INTER 152 02
00032C EXFMTDONE
00033C NEXT TAG
00034C NO1 READ ITEM 05
00035C 1 05 GOTO END
00036C NO1NUMB2 COMP CUSTN 0303
00037C EXFMTTEST1
00038C NO1NO2COST ADD INTER INTER
00039C EXFMTTEST2
00040C 2 03 GOTO NEXT
00041C 3 NO3 GOTO HERE
00042C END ENDSR
00043

```

Figure 8-5. A Sample Program Showing the Use of Indicators with Compare Operations

This program illustrates the use of indicators to control multiple branching operations within a single program. Based upon the condition of the various indicators, the program returns to one of three points in the program **1**, **2**, **3**. The formats for the display (reached when the EXFMT occurs), provide a way of tracking the program to determine which branch has been taken.

## COMPLEX USE OF INDICATORS ON THE C-SPECIFICATIONS

The sample in Figure 8-6 illustrates a complex application for using indicators on the C-specification.

```

00001Z*****
00002Z* PROGRAM 55. FIGURE 8-6 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ INDEMP
00005Z A1SUBEX      1E                      EOJ
00006A      F INPUTER      22          DEVICE(CRT) DSPSIZ(6 80)
00007A      R EXDFREC
00008A      NUMBER          6  I      1 PMT(ENTER THE CUSTOMER NUMBER)
00009A      TOTAL          15  2I      2 PMT(ENTER THE MATCHING TOTAL FROM T+
00009A                                     HE ACCOUNTING BOOKS)
00010A      R OUT
00011A                                     0      'THERE IS NO RECORD FOR THIS CUSTOM+
00011A                                     ER'
00012A                                     1  I      PMT(USE THE FLD EXIT KEY)
00013A                                     CHECK(FE)
00014A      R TELL
00015A                                     0      'THE TOTALS DO NOT TALLY'
00016A                                     1  I      PMT(USE THE FLD EXIT KEY) CHECK(FE)
00017A      F INVMAS
00018A      R BILMAS
00019A      K CUSTNO          6
00020A      ITEM            6
00021A      QUANT           4
00022A      PERPRI          5
00023A      COST           15  2
00024C      SUBEX          BEGSR
00025C      START          TAG
00026C      3 NUMBER      EXFMTXOFREC
00027C      4 01          CHAINBILMAS
00028C      01          EXFMTOUT          01
00029C      5 GOTO START
00030C      6 TOTAL      SUB COST      7 INTER  152  0203 8
00031C      NEXT          TAG
00032C      9 READ BILMAS          05 10
00033C      CUSTNO      COMP '999999'          04 12
00034C      04 05      11 GOTO END
00035C      02          13 GOTO ERROR
00036C      N03N04      GOTO ERROR
00037C      03N04      GOTO START
00038C      14 INTER      SUB COST      15 INTER          0203
00039C      N02          16 GOTO NEXT
00040C      ERROR      TAG
00041C      EXFMTTELL
00042C      GOTO START
00043C      END          TAG
00044C      ENDSR

```

Figure 8-6. A Sample Program Showing the Complex Use of Indicators

The object of this program is to determine whether the monthly billing data set maintained for the customers equals the monthly total in the accounting books. The following list shows the elements involved in using the program in Figure 8-6.

The flow of the program is as follows:

1. The operator is prompted to enter the customer ID **1** and the total **2** from the accounting books.
2. The program finds the first record in the data set that matches the customer ID **3**.  
  
**Note:** If there is no matching record, the program sets an indicator that conditions the operation that displays an error message **4**, and prompts the operator for the next customer information **5**.
3. The program then subtracts the total in the record from the total entered by the operator **6**. It places the results in a newly created field **7**. If the results of the calculation are zero or negative, it sets the appropriate indicator **8**.
4. The program then reads the next record in the data set **9**. (If it is the last record in the data set, it sets an indicator **10** and conditions an operation that ends the subroutine **11**.)
5. The program compares the identification numbers of the last record and the ID entered by the operator. If the numbers match, the program sets an indicator **12**.
6. The program checks for invalid combinations and if it finds one, branches to an error routine within the subroutine **13**.
7. If the result of the previous calculation was not zero and this record matches the entered ID **14**, the program subtracts the total in this record from the field created in the previous calculation and then puts the new results in the temporary field **15**.
8. The program branches back to the read operation **16** and begins the process again. When the results are zero and the last record with a matching ID has been found, the program returns to the beginning of the routine and prompts the operator for another number.

This process of reading a record and performing a calculation against the record continues until the operator uses the End of Job key, or the end of the data set is reached. As you can see in this one sample, a variety of operations can be used to set indicators and a variety of operations can be conditioned by indicators.



This part describes the various ways that you can retrieve data from diskette data sets.

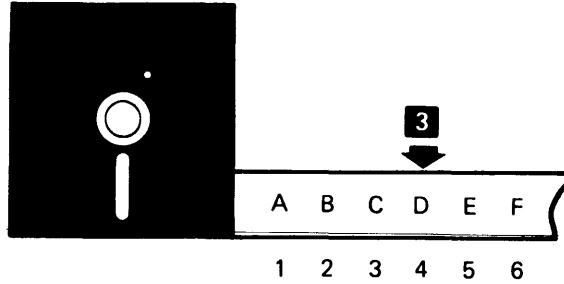
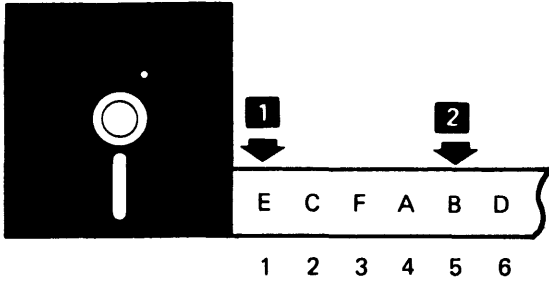
It contains two chapters:

- Chapter 9. Sequential Access Methods
- Chapter 10. Direct Access Methods

The organization of the data set determines the access method that can be used. The sample in Figure 9-1 illustrates the types of organization and access methods available with DE/RPG.

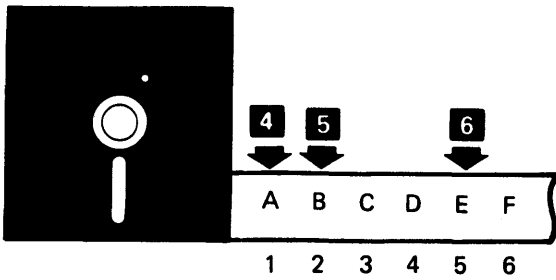
The DE/RPG compiler uses the C-specification operations (such as READ, WRITE, CHAIN, and UPDAT) to determine if the diskette data set is to be created, added to, updated, or read. For example, if only WRITE statements appear for a data set, the data set is recreated if it already exists; in other words, it is rewritten beginning with the first record. Therefore, if you want to add records to an existing data set without completely rewriting it, you must include a READ operation for the data set in the subroutine. The READ operation does not have to be executed; it simply indicates that the data set already exists.

1. The data set is sequential.



- 1 Can be accessed sequentially.
- 2 Can be accessed directly by relative record number.
- 3 Can be accessed directly by key if the field specified as the key occurs in ascending sequence in the data set when it is accessed.

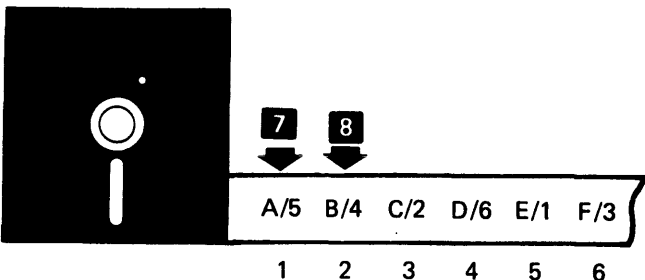
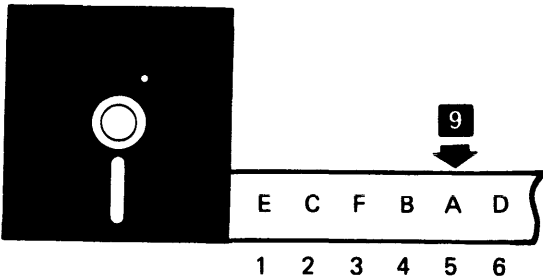
2. The data set organization is by key sequence.



- 4 Can be accessed sequentially by key.
- 5 Can be accessed directly by key.
- 6 Can be accessed directly by relative record number if a key field is not specified for the data set when it is accessed.

	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	
				SUB1										BEGSR											
												1		READ	REC										
2			5											CHAIN	REC										
3			KEY											CHAIN	REC										
														ENDSR											
				SUB2										BEGSR											
												4		READ	REC										
5			KEY											CHAIN	REC										
6			5											CHAIN	REC										
														ENDSR											
				SUB3										BEGSR											
													7	READ	REC										
8			KEY											CHAIN	REC										
9			5											CHAIN	REC										
														ENDSR											

3. The data set organization is sequential by key and with an index.



- 7 Can be accessed sequentially by key.
- 8 Can be accessed directly by key.
- 9 Can be accessed directly by relative record number if a key field is not specified for the data set when it is accessed.

Figure 9-1. Types of Access Methods Available with DE/RPG

Before beginning the topics of direct and sequential access methods, there is one additional consideration concerning the general topic of access methods—that is, the accessing of multivolume data sets. A multivolume data set is one that is resident on more than one diskette. An example of this might be a data set that contains information about the items your business sells. A single diskette might not be large enough to hold all of this information, so you must continue the data set onto another diskette.



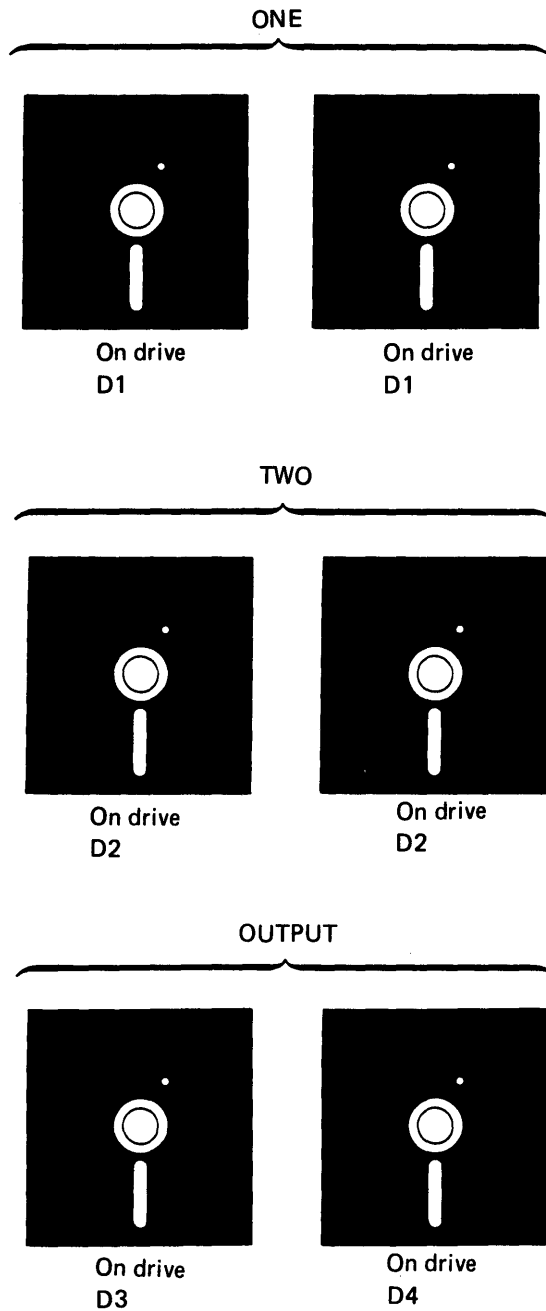
The sample in Figure 9-2 illustrates a program that reads from multivolume data sets and writes a single multivolume data set on the diskette.

```

00001Z*****
00002Z* PROGRAM 56. FIGURE 9-2 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ MULTVOL
00005Z S1ROUT 1E EOJ
00006A 1 F ONE 50 DEVICE(DISK D1 D1) 2
00007A R EX1
00008A FLD1 10
00009A FLD2 10
00010A FLD3 10
00011A FLD4 10
00012A FLD5 10
00013A 3 F TWO 50 DEVICE(DISK D2 D2)
00014A R EX2
00015A FLD6 20
00016A FLD7 20
00017A FLD8 10
00018A 4 F OUTPUT 100 DEVICE(DISK D3 D4)
00019A INDEX(CHECK)
00020A R RE1X
00021A K FLD1
00022A FLD2
00023A FLD3
00024A FLD4
00025A FLD5
00026A FLD6
00027A FLD7
00028A FLD8
00029C ROUT BEGSR
00030C LOOP 1 TAG
00031C NO1 NO2 3 READ EX1 01
00032C NO1 NO2 3 READ EX2 02
00033C WRITERE1X 4
00034C GOTO LOOP
00035C ENDSR

```

Figure 9-2 (Part 1 of 2). A Sample Program Showing the Use of Multivolume Data Sets



**Figure 9-2 (Part 2 of 2). A Sample Program Showing the Use of Multivolume Data Sets**

In this sample, the data set named ONE is read first **1**. The D1 D1 parameters of the DISK keyword **2** declare this data set as multivolume. By specifying D1 D1 (the same drive), the programmer is indicating that the operator removes the first diskette from the drive as needed and replaces it with the second diskette. In this case, the operator sees only one open prompt for the data set named ONE at the beginning of the program. As the need for the second volume arises, the open prompt reappears, alerting the operator to change diskettes in the drive.

The second data set that is read **3** also contains a data set that resides on more than one diskette. As before, these diskettes will be used in a single diskette drive. Finally, the data set that is written is also a multivolume data set **4**. The open prompts for both diskettes for the data set that is written appear at the beginning of the program. These data sets will use two different diskette drives. Both data sets, therefore, are opened at the beginning of the program.

If you are not using logical devices (D1, D2, etc.), then the specifications for the drives would appear as: X'4000', X'4400', etc.

Data sets created by transaction files are always sequentially written and nonkeyed or nonindexed; therefore, they can only be accessed sequentially or directly by a relative record number.

**Note:** The one exception to this is a data set that has been entered in a transaction data set in key order by the operator. In this case, the data set can be accessed as a keyed data set.

Data sets created by subroutines on the C-specifications can be sequentially written, or written in key sequence, or indexed by key sequence; therefore, they can be accessed using any of these methods.

Data sets created by using the Sort/Merge Program Product can be organized sequentially, or in key sequence, or by an indexed (ADDROUT) data set; therefore, they can be accessed by any of these methods.

In addition, the Sort/Merge Program Product can be used to create individual indexed data sets, and those can be merged into a master indexed data set.

Review Part 1 for a description of how data sets are created by data-entry and application programs. See the *IBM 5280 Sort/Merge Reference/Operation Manual*, SC21-7789 for details about creating data sets using the Sort/Merge Program Product.

User-programmed access to already existing data sets is through the READ or CHAIN operations on the C-specifications.

## Chapter 9. Sequential Access Methods

The read (READ) operation on the C-specification provides the sequential access method for DE/RPG. The read operation can be used either for sequential or indexed data sets. Sequential (nonindexed) data sets are read in an ascending relative record number sequence. That is, the first record in the data set is read and then the next one is read and so forth. Indexed data sets are read sequentially by key sequence. Data sets created in key sequence are read sequentially by key sequence.

## SEQUENTIAL ACCESS OF NONKEYED AND NONINDEXED DATA SETS

The sample in Figure 9-3 illustrates the use of the read operation for accessing a sequentially written (nonindexed) data set.

```

00001Z*****
00002Z* PROGRAM 57. FIGURE 9-3 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ ACCESS
00005Z F4SUB1          1E                      EOJ
00006A          F MASTHEAD          67          DEVICE(DISK D1)
00007A          R HEADER
00008A          CUSNA              30
00009A          ADDR              30
00010A          ID                 6
00011A          H                  1
00012A          F MIX              67          DEVICE(DISK D1)
00013A          R FIRST
00014A          ID
00015A          CUSNA
00016A          ADDR
00017A          H
00018C          SUB1          BEGSR
00019C          GO           TAG
00020C          1 READ HEADER          05
00021C          N05          WRITEFIRST
00022C          N05          GOTO GO
00023C          ENDSR
00024

```

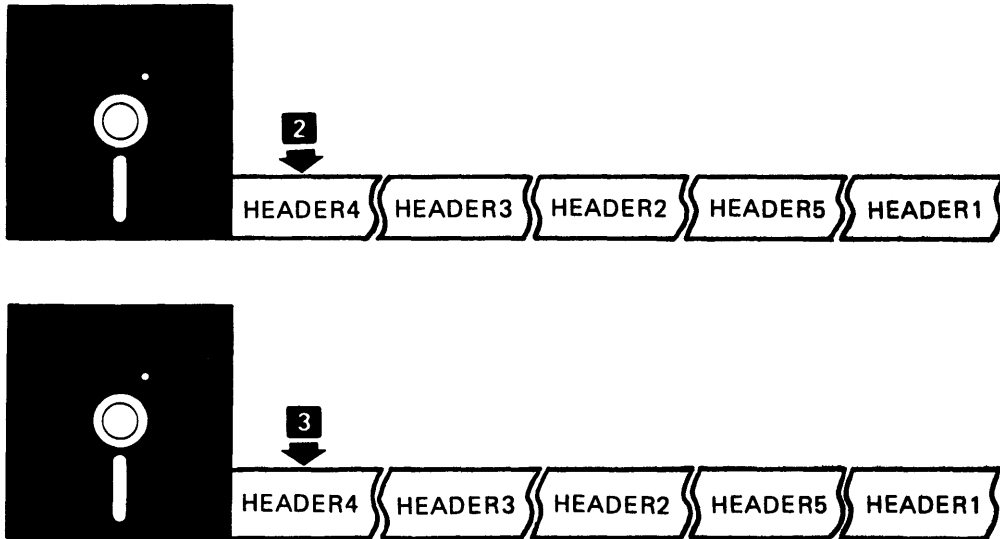


Figure 9-3. A Sample Program Showing the Sequential Access of a Nonkeyed, Nonindexed Data Set

In this sample, the records in the data set that is being read **1** are in a random sequence **2**. The resulting data set is also written in the same random sequence **3** because no key or index has been specified for it.

If the data set that is being read contains more than one record type, you should include the RECID operation (along with its associated indicators) to identify the correct record type to be processed by the program. The sample in Figure 9-4 illustrates this process.

```

00001Z*****
00002Z* PROGRAM 58. FIGURE 9-4 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ ACCESS
00005Z F4SUB1 1E EOJ
00006A F MASTBIL 67 DEVICE(DISK D1)
00007A 1 R HEADER SETOF(05) SETOF(06) RECID(*POS1 'H')
00008A SETON(05)
00009A H 1
00010A ID 6
00011A CUSNA 30
00012A ADDR 30
00013A 2 R DETAIL RECID(*POS1 'D') SETON(06)
00014A D 1
00015A ID 6
00016A ITEM 10
00017A QUANT 5
00018A COST 6
00019A F IMML 67 DEVICE(DISK D1)
00020A R HED
00021A H
00022A ID
00023A CUSNA
00024A ADDR
00025A R DET
00026A D
00027A ID
00028A ITEM
00029A QUANT
00030A COST
00031C SUB1 BEGSR
00032C BRANCH TAG
00033C 5 05 N01 READ MASTBIL 01
00034C 6 06 N01 WRITHEHED
00035C N01 WRITEDET
00036C N01 GOTO BRANCH
00037C ENDSR

```

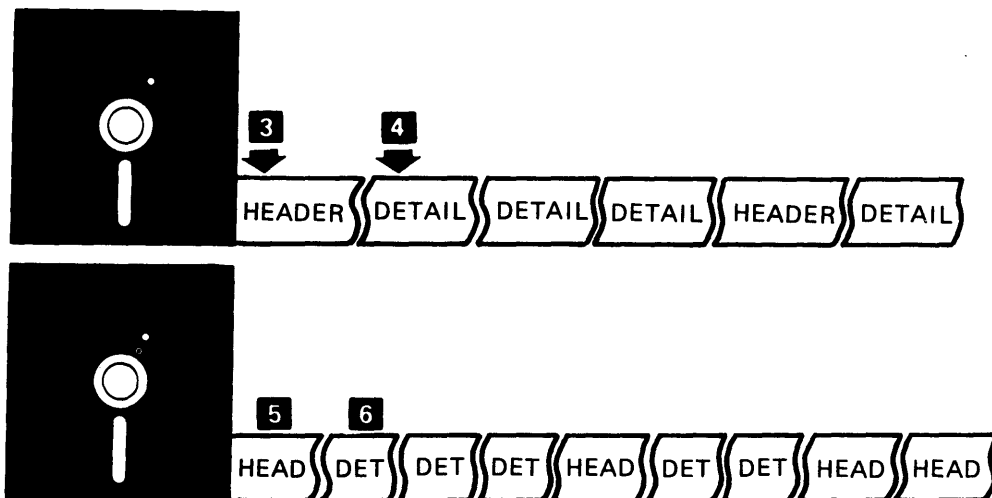


Figure 9-4. A Sample Program Showing Sequential Access of a Data Set with Multiple Record Types

Notice that the data set being read contains two types of records (header 1 and detail 2). All header records are identified by an H in position 1 3 and all detail records are identified by a D in position 1 4. The sample program is specifying that every time a header record is read, a new header record is written 5 and every time a detail record is read, a new detail record is written 6.

## SEQUENTIAL ACCESS OF KEYED AND NONINDEXED DATA SETS

Suppose you want to sequentially read records in a data set that is written in keyed sequence. The sample in Figure 9-5 illustrates this process.

```

00001Z*****
00002Z* PROGRAM 59. FIGURE 9-5 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ ACCESS
00005Z F4SUB1          1E                      F4
00006A                F MASTERAD           67          DEVICE(DISK)
00007A                R HEADER
00008A                CUSNA                 30
00009A                ADDR                 30
00010A                1 K ID                 6
00011A                H                     1
00012A                F FEL                 90          DEVICE(DISK D1)
00013A                R NEW
00015A                CUSNA
00016A                ID
00017A                ADDR
00018A                H
00019C                SUB1          BEGSR
00020C                BRANCH        TAG
00021C                READ HEADER          01
00022C                N01          WRITENW
00023C                N01          GOTO BRANCH
00024C                ENDSR

```

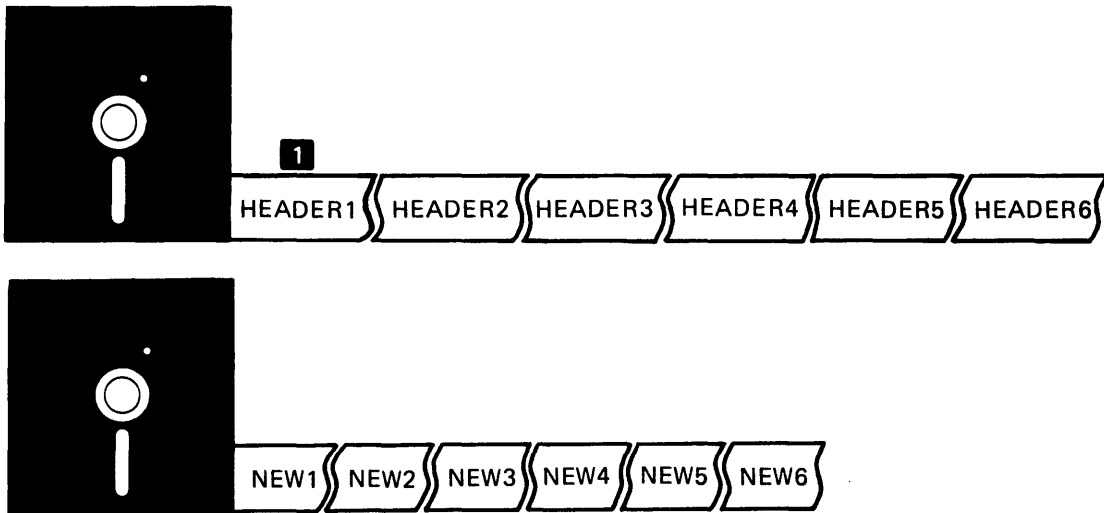


Figure 9-5. A Sample Program Showing Sequential Access of a Keyed Data Set

The data set is the same one used in Figure 9-2 but this time, it is organized by key sequence **1**. This time, the records are read in the sequence of their keys and written out in key sequence (which in this case corresponds to the sequence of the relative record numbers because the original data set is organized in key sequence). Even though a key field is not specified for the NEW record, the data set can be used as a keyed data set if the ID field is referenced as the key field. This is true because the data set from which this one was created was in ascending key sequence and therefore the NEW data set is also in ascending key sequence (via the ID field).

## SEQUENTIAL ACCESS OF INDEXED DATA SETS

Suppose that the data set being accessed is indexed. It is randomly organized, but it uses an indexed data set that is organized by key sequence. The sample in Figure 9-6 illustrates this process.

```

00001Z*****
00002Z* PROGRAM 60. FIGURE 9-6 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ ACCESS3
00005Z D1SUB13      1E      D1      2
00006A      F MASTHEA4      66      DEVICE(DISK D1) INDEX(LOOK)
00007A      R HEADER
00008A      K ID      6
00009A      CUSNA      30
00010A      ADDR      30
00011A      F TEMP8      66      DEVICE(DISK D1)
00012A      R NEW
00013A      ID
00014A      CUSNA
00015A      ADDR
00016C      SUB13      BEGSR
00017C      AGAIN      1 TAG
00018C      3 READ HEADER      05
00019C      N05
00020C      N05
00021C      ENDSR
    
```

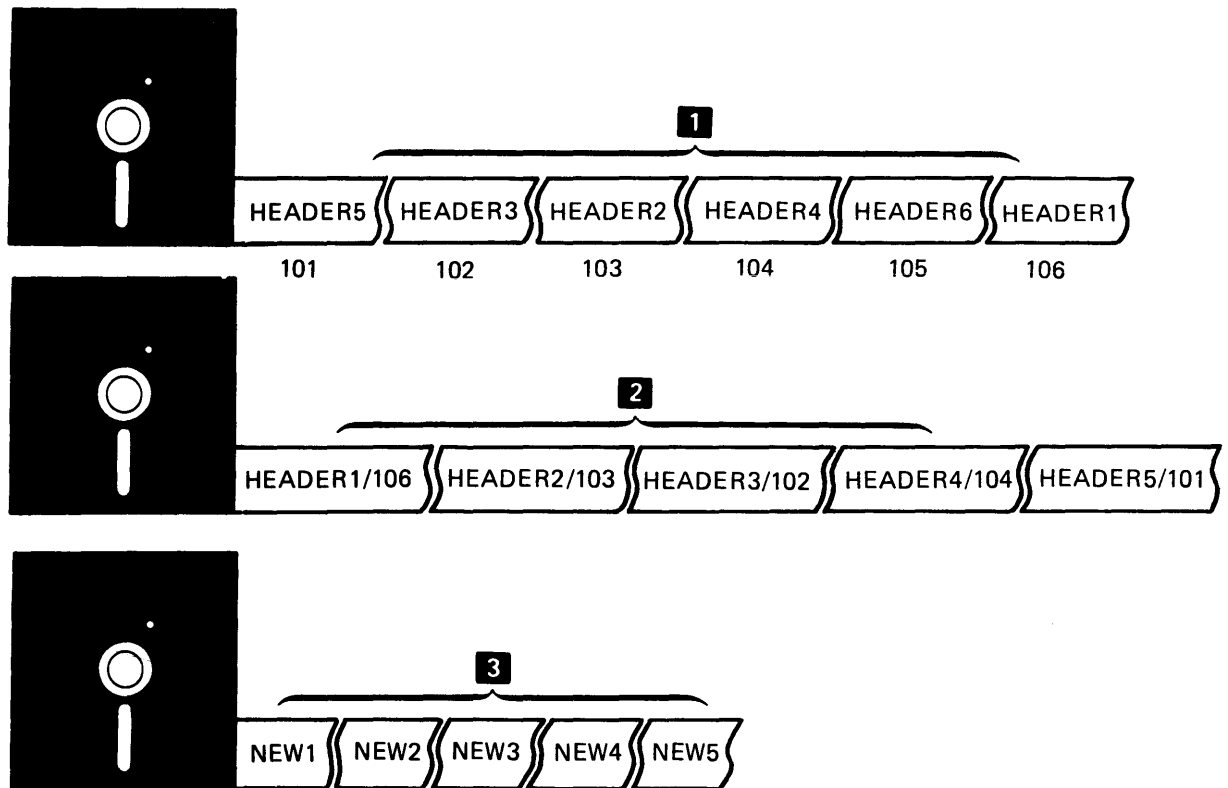


Figure 9-6. A Sample Program Showing Sequential Access of an Indexed Data Set

The read operation is performed sequentially, but this time it is performed against the indexed data set which in turn references the correct record to be read in the data set. With the use of the indexed data set, a data set that is randomly written can be accessed as if it were organized sequentially by key field. This improves the efficiency of first writing the records and later reading them.



(

(

## Chapter 10. Direct Access Methods

The chain operation (CHAIN) on the C-specification provides the direct access methods for DE/RPG. Data sets can be directly accessed in one of two ways: (1) by a reference to a specific relative record number or (2) by a reference to a specific key field value.

### DIRECT ACCESS FOR NONKEYED AND NONINDEXED DATA SETS

Nonkeyed and nonindexed data sets can only be directly accessed by relative record number. The sample in Figure 10-1 illustrates this process.

```

00001Z*****
00002Z* PROGRAM 61. FIGURE 10-1 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ ACCESS
00005Z C5SUB2          1E          C5
00006A          F MASTHEAD          66          DEVICE(DISK D1)
00007A          R HEADER
00008A          ID          6
00009A          CUSNA          30
00010A          ADDR          30
00011A          F OUTPUT          30          DEVICE(DISK D1)
00012A          R NEW
00013A          CUSNA          30
00014C          SUB2          BEGSR
00015C          1 5          CHAINHEADER          09
00016C          BRANCH          TAG
00017C          N09          WRITENew
00018C          3 READ HEADER          05
00019C          N05          GOTO BRANCH
00020C          ENDSR

```

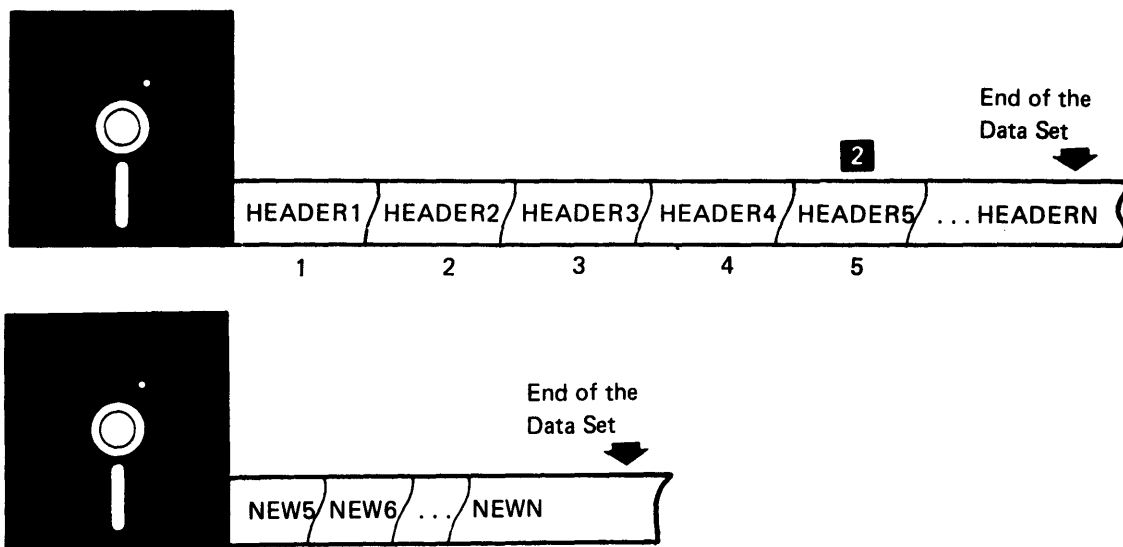


Figure 10-1. A Sample Program Showing Direct Access of a Nonkeyed Data Set

In this sample, factor 1 **1** contains the record number that you want the program to read first. DE/RPG will find and read **2** the record with this relative record number. If you want to read the records sequentially following this record, you must specify a READ operation **3**.

## DIRECT ACCESS FOR KEYED AND INDEXED DATA SETS

The CHAIN operation provides direct access to keyed data sets (both indexed and nonindexed). The sample in Figure 10-2 illustrates the chain process.

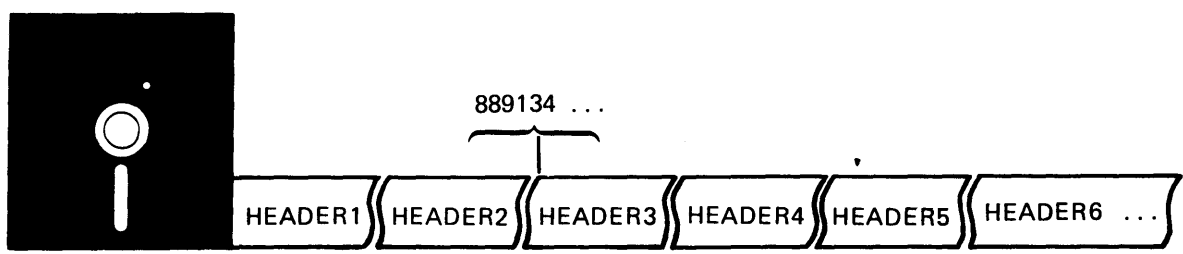
```
00001Z*****
00002Z* PROGRAM 62. FIGURE 10-2 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ ACCESS2
00005Z N1SUB2      1E                      EOJ
00006A          F DISP              67          DEVICE(CRT) DSPSIZ(6 80)
00007A          R GET
00008A          ID              6   I          PMT(ENTER THE RECORD ID YOU WANT)
00009A          R LOOK
00010A          NAME1            30   B          INSERT(CUSNA)
00011A          NAME2            30   B          INSERT(ADDR)
00012A          ID              1   I          PMT(PRESS ENTER TO CONTINUE)
00013A          F MASTHEAD        66          DEVICE(DISK D1)
00014A          R HEADER
00015A          K IDENT              6
00016A          CUSNA             30
00017A          ADDR              30
00018C          SUB2              BEGSR
00019C          EXFMTGET
00020C          ID              CHAINHEADER          01
00021C          EXFMTLOOK
00022C          ENDSR
```

Figure 10-2 (Part 1 of 2). A Sample Program Showing Direct Access of a Keyed Data Set

```

0                               X
ENTER THE RECORD YOU WANT
889134

```



```

0                               X
PRESS ENTER TO CONTINUE
DAN NELSEN                       99 EAST CENTER ST., ROCHESTER, MN.

```

**Figure 10-2 (Part 2 of 2). A Sample Program Showing Direct Access of a Keyed Data Set**

As the sample illustrates, the value of the key is contained in the field named ID **1**. This value is used by the program to determine the exact record in the data set to be read. If the data set is keyed and nonindexed, the program directly finds the correct record.

If the data set is keyed and indexed (or the ADDROUT data set is being used), the program uses the index data set to correctly locate the appropriate record in the data set. The sample in Figure 10-2 illustrates using a direct access method with a keyed and indexed data set.

```

00001Z*****
00002Z* PROGRAM 63. FIGURE 10-3 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ ACCESS3
00005Z D1SUB13      1E                      D1
00006A              F MASTHEA4      66      DEVICE(DISK D1) INDEX(LOOK)
00007A              R HEADER
00008A              K ID              6
00009A              CUSNA            30
00010A              ADDR             30
00011A              F MASDETAL      27      DEVICE(DISK D1) INDEX(FIND)
00012A              R DETAIL
00013A              K IDENT           6
00014A              ITEM             10
00015A              QUANT             5
00016A              COST              6
00017A              F TEMP8          87      DEVICE(DISK D1)
00018A              R NEW
00019A              IDENT             6
00020A              CUSNA            30
00021A              ADDR             30
00022A              ITEM             10
00023A              QUANT             5
00024A              COST              6
00025C              SUB13            BEGSR
00026C              AGAIN            TAG
00027C N05              READ HEADER      05
00028C              ID              CHAINDETAIL      02
00029C N02 N05          WRITENEW
00030C N05              GOTD AGAIN
00031C              ENDSR

```

**Figure 10-3 (Part 1 of 2). A Sample Program Showing Direct Access of a Keyed and Indexed Data Set**

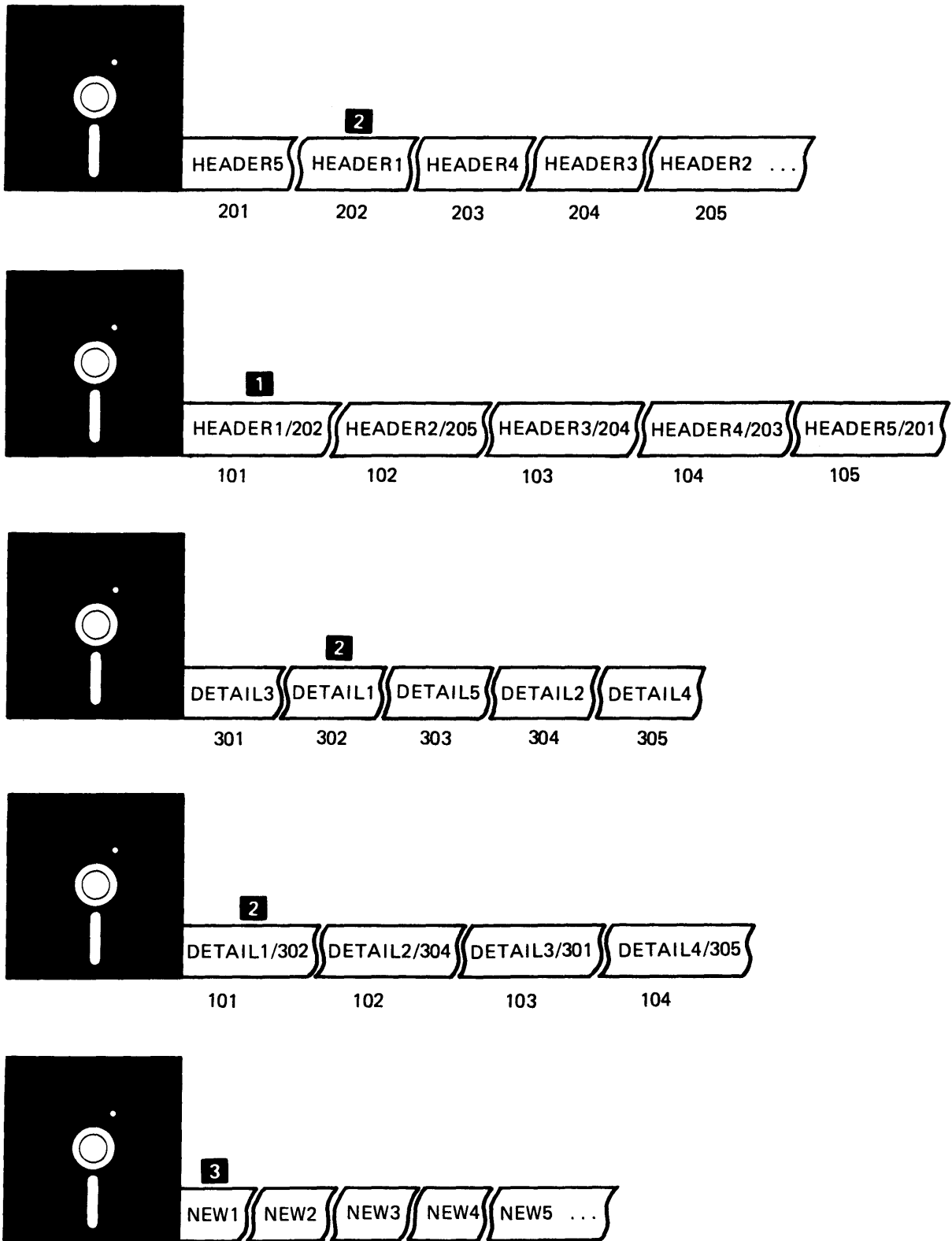


Figure 10-3 (Part 2 of 2). A Sample Program Showing Direct Access of a Keyed and Indexed Data Set.

The sample in Figure 10-3 illustrates a direct access method using a field from a previously read record as the key value. First, a record from the LOOK data set (which is the index data set for the MASTHEAD data set) is sequentially accessed. Since this data set is written in key sequence **1**, the first logical record (by key) is the first relative record. The key field from this record is extracted and used to identify the correct record to read from the MASTDETAL data set **2**. This record is directly accessed. After the read and chain operations have been concluded, a new record, which contains fields from the previous two records that were read, is written into a new data set **3**.

Remember that the way the data set is created directly determines how it can be accessed. Part 1 contains related information about creating data sets using data-entry and application programs.

A data set created with the ADDRROUT file in the Sort/Merge Program Product is similar to an indexed data set with the following exceptions. When accessed:

- The data set must not specify a key field.
- The name provided as a parameter for the INDEX keyword is the name of the ADDRROUT file.

If you are using a large indexed data set, you may want to include a storage parameter for the INDEX keyword. The *DE/RPG Reference Manual* contains additional details about a storage parameter.

This part describes considerations involved in using data tables and in creating tables. It contains two chapters:

- Chapter 11. Using Data Tables in Data-Entry Programs
- Chapter 12. Using Arrays in Application Programs

There are two basic ways of thinking of data tables in DE/RPG: (1) as lists and (2) as one-dimensional RPG arrays. The table functions performed on the A-specification are primarily for data-entry purposes. They provide a variety of ways to ensure correct entry or to automatically supply values for entries. The array operation provided on the C-specification compares a value (field or constant) with entries in a table and allows you to manipulate a table and index as a field.

The table functions that can be provided on the A-specification are:

LOOK	Determines if the field matches a table entry.
RANGET	Determines if the field is within an acceptable range of values.
SUBST	Replaces the value of the field with an entry from the table.
XCHK	Cross-checks the values of two fields against a table to ensure that the combination of their values matches.

The array operations that can be performed on the C-specifications are:

LOKUP	Compares the value named in Factor 1 with the entries in a table for high, low, and equal.
<i>Arrayname,index</i>	Allows the programmer to use the <i>arrayname,index</i> combination as any field is used on the C-specifications.

The *Introduction to DE/RPG* manual describes the process of using tables and of creating them (both in separate data sets and in the using program). This chapter describes considerations for using tables; it does not repeat the material provided in the *Introduction to DE/RPG* manual. The *DE/RPG Reference Manual* provides additional information about coding programs to use tables.





## Chapter 11. Using Data Tables in Data-Entry Programs

The specific topics described in Chapter 11 are:

- Considerations about declaring indexes
- Arranging data in tables
- Arranging tables in data sets

### CONSIDERATIONS ABOUT DECLARING INDEXES

Indexes are numeric variables which contain the number of a table entry. The number that the index contains depends on the table operation with which it is used. For example, when an index is used with a LOOK operation, the index contains the number of the table entry that matches the contents of the current field.

You have the choice of either declaring the indexes to be used in your program or of accepting the default characteristics for the indexes. The default characteristics for each index are five positions wide and numeric. All indexes must be numeric, so there is no difficulty in accepting the default from this viewpoint. However, all indexes are not necessarily five positions wide. If you accept this default and you use indexes in a XCHK operation, you must create the XCHK table so that its entries are five positions wide. If you do not accept the default, DE/RPG is not able to provide the correct value.

Of course, you have the most control when you create the indexes. Creating indexes as work space fields prevents their being displayed or written to diskette. The sample in Figure 11-1 illustrates the declaration of an index named PEEK.

```

00001Z*****
00002Z* PROGRAM 64. FIGURE 11-1 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ SAMP TFILE(ONLY)
00005Z B1TEMP 1E B1
00006A F DISFL 4 3 DEVICE(CRT)
00007A R TEMP 2 1 0W 1
00008A PEEK 4 2I
00009A AMT 4 2I PMT(ENTER AMOUNT)
00010A LOOK(AMTAB PEEK)
00011A F TABLE1 4 NUMENT(9)
00012A T AMTAB 4 2
00013A F ONLY 4 DEVICE(DISK D1)
**CTDATA TABLE1
1000
2000
3000
4000
5000
6000
7000
8000
9000

```

**Figure 11-1. A Sample Program Showing the Use of an Index with a Table**

This index is described as a work space **1**. It is one position wide **2**. Because it is only one position wide, it can be used effectively only with a table that contains fewer than 10 entries. The 0 in the decimal usage column **3** describes the field as being numeric.

## ARRANGING DATA IN TABLES

The type of table operation you want to use determines the arrangement of the data in the table. For example, two tables that are to be used in LOOK operations in the same program, can be created simultaneously by alternating entries. The sample in Figure 11-2 illustrates this table entry process.

```

00001Z*****
00002Z* PROGRAM 65. FIGURE 11-2 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ TABEXMP TFIL(TEMPORAR)
00005Z Y1EXMPTB 1E Y1
00006A F INPUT 11 DEVICE(CRT) DSPSIZ(6 80)
00007A R EXMPTB
00008A 0001001'ITEM NUMBER'
00009A 0002001'PRICE'
00010A 0001014LOOK(ITEM) 2
00011A 3 5 2I002008LOOK(PRICTA) CHECK(RZ)
00012A F LOOKTBS 11 NUMENT(4) 4
00013A T ITEM 6
00014A T PRICTA 5 2
00015A F TEMPORAR 11 DEVICE(DISK D1)
5 **CTDATA LOOKTBS 7
100-A200895
350-A100150
580-A208969 6
678-A115555

```

Figure 11-2. A Sample Program Showing a Compile-Time Table

In this program, the operator is entering an item number **1**. This item number is being validated by entries in a table **2**. If a matching entry is found in the table, the entered number is considered valid. In the same manner, the price **3** that is entered by the operator is being validated by entries in another table **4**. These two tables can be created at compile-time **5**. Because they are both being used in LOOK operations, they should be created in alternating sequence **6** in the same data set **7**. No indexes are used in this program.

If two tables are being used for a substitute operation, they should also be created in alternating sequence in the same data set. The sample in Figure 11-3 illustrates a program using the SUBST operation.

```

00001Z*****
00002Z* PROGRAM 66. FIGURE 11-3 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ  SAMPLET                                TFILE(SHIPTEM)
00005Z  A1HEADER          NE
00006A          F DISFL          73          DEVICE(CRT) DSPSIZ(6 80)
00007A          R HEADER
00008A                                0001001 'CUSTOMER NAME'
00009A                                0002001 'ADDRESS'
00010A                                0003001 'SHIP CODE'
00011A                                I001015
00012A                                I002015
00013A                                I003015CHECK(RB) SUBST(SHIPCD SHIPTO)
00014A          F SHIPTEM          3 13          DEVICE(DISK D1)
00015A          F SHIPTAB          15          NUMENT(4)
00016A          T SHIFCD          2
00017A          T SHIPTO          13
**CTI 2 A SHIPTAB
1 01RAILFREIGHT
  02INSTIMAIL
  03EDDY'S TRUCKS
  04MASSIVE TRANS

```

Figure 11-3. A Sample Program Showing the Use of the SUBST Operation

In this program, the operator is entering a two-digit ship code **1**. The program is exchanging this two-digit code for a descriptive title taken from the substitution table **2**. As you can see, the organization of the table data set consists of alternating entries for the abbreviated and extended entries.

Remember, when you are using a SUBST operation, that the field receiving the substituted data must be long enough to accept it. For example, in Figure 11-3 if the receiving field **3** had been only two positions to accept the initial operator entry, only the first two characters of the extended description could have been substituted. If the code is 03 and the field length is 2, the field would contain ED (rather than EDDY'S TRUCKS) after the substitution. Another consideration when using the SUBST operation is to keep the entries of the two tables aligned. For example if the tables were not carefully created, the incorrect description might be substituted for the coded entry.

Entries for tables being used by the RANGET operation should be paired. The sample in Figure 11-4 illustrates this operation.

```

00001Z*****
00002Z* PROGRAM 67. FIGURE 11-4 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ PARTSAM TFILE(DISRAT)
00005Z C1REC 1E
00006A F DISPC 15 DEVICE(CRT) DSPSIZ(6 80)
00007A R REC
00008A 5 2I 1 PMT(ENTER AMT SOLD TO DETERMINE THE--
00008A DISCOUNT) RANGET(DISCT FIND)
00009A CHECK(RZ)
00010A 5 2I INSERT(FIND)
00011A FIND 1 OW
00012A F DISCOUNT 5 NUMENT(6)
00013A T DISCT 5 2
00014A F DISRAT 15 DEVICE(DISK D1)
**CTDATA DISCOUNT 2
01000
02000
02010
05000
05010
19990

```

Figure 11-4. A Sample Program Showing the Use of the RANGET Operation

In this sample, the program is testing the entry **1** to determine if it is in an acceptable range of values. The table being used to determine the range is created at compile-time within the program **2**. If you look at the entries for the table, you see that the first and second entries form a pair of ranges, the third and fourth form another pair and so on. For example, if the entry is 03000, it is in the range of 02010 to 05000; therefore, the index (FIND) contains the number 3. Remember that indexes that are not declared (set up as separate fields) default to numeric fields with a length of 5.

The coding required to determine the amount to be discounted for this value is not included in the program.

## ARRANGING TABLES IN DATA SETS

Multiple tables can be contained in a single data set. Before you group multiple tables in the same data set, however, consider that all tables used by a program must be resident in storage for the entire use of the program. Only those tables being used specifically for the program should be in the data set. If you are using the same tables in a variety of programs, try to group them in data sets separate from the program so unused tables are not loaded into storage when the programs are operating.

The sample in Figure 11-5 illustrates the creation of a data set that uses multiple tables.

```

00001Z*****
00002Z* PROGRAM 68. FIGURE 11-5 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ TODAY                                TFILE(BILLING)
00005Z                                        CFILE(MASTHEAD)
00006Z A0SCRATCH 1E                          X1          WRITE(*NO)
00007Z X1DET     NE                          X2          WRITE(DETAIL)
00008Z X2TRAIL  1E                          A0          WRITE(TRAILER)
00009Z          R *POS150 'H'                0
00010Z          R *POS150 'D'                X1
00011Z          R *POS150 'T'                X2
00012Z          F DISP 150                   DEVICE(CRT)
00013Z          DSPSIZ(6 80)
00014Z          R SCRATCH
00015Z          6 I001060PMT(MAKE AUTODUP ACTIVE AND ENTER T+
00015A          HE DATE IF THERE IS ONE)
00016A          AUXST(DATE)
00017A          0002001'USE THE SEARCH KEYS TO FIND THE CO+
00017A          RRECT HEADER RECORD IN THE MASTHEAD-
00017A          DATA SET.'
00018A          0003001'SEARCH FOR A MATCH TO THE CUSTOMER-
00018A          NUMBER ON THE ORDER FORM.'
00019A          0004001'NEXT USE THE COPY KEYS TO COPY DAT+
00019A          A FROM THE HEADER INTO THIS DATA SE+
00019A          T.'
00020A          1 I I PMT(USE THE FIELD EXIT KEY TO LEAVE-
00020A          THE DISPLAY)
00021A          CHECK(FE)
00022A          R DET
00023A          DAT 6 I002001AUXDUP(DATE)
00024A          SALS 3X I002021PMT(ENTER THE SALESMAN'S INITIALS)
00025A          CHECK(DR BC)
00026A          ITEM 6C I002031PMT(ENTER THE ITEM CODE) +
00026A          SHIFT(DDXXXD)
00027A          LOOK(ITEM A)
00028A          CHECK(BC)
00029A          DESC 30 I003001PMT(ENTER THE DESCRIPTION)
00030A          QUANT 4 0I003041PMT(ENTER THE QUANTITY)
00031A          CHECK(RZ DR)
00032A          PRICE 5 2I004001PMT(ENTER THE PRICE) CHECK(RZ)
00033A          LOOK(PRICET B)
00034A          XCHK(INVENT A B)
00035A          0004060'TOTAL:'
00036A          COST 9 2I004070INSERT(PRICE*QUANT)
00037A          TADD(*TOT1) DSPATR(HI)
00038A          CUSN 5 I005001PMT(ENTER THE CUSTOMER NUMBER)
00039A          MARK2 1 I005079INSERT('D')
00040A          R TRAIL
00041A          DAY 6D I PMT(ENTER THE CURRENT DATE)
00042A          CUST 5 I INSERT(CUSN)
00043A          TOT 15 2I INSERT(*TOT1)
00044A          RESET(*TOT1)
00045A          MARK3 1 I INSERT('T')
00046A          F BILLING 150 DEVICE(DISK X'4000')
00047A          R DETAIL
00048A          DAT 001
00049A          CUSN 007
00050A          SALS 012
00051A          ITEM 015
00052A          DESC 021
00053A          QUANT 051
00054A          PRICE 055
00055A          COST 060
00056A          MARK2 150
00057A          R TRAILER
00058A          DAY 001
00059A          CUST 007
00060A          TOT 012
00061A          MARK3 150
00062A          F MASTHEAD 150 DEVICE(DISK X'4000')
00063A          F TABLE1 11 NUMENT(6)
00064A          T ITEM 6
00065A          T PRICET 5 2
00066A          F TABLE2 1 NUMENT(12)
00067A          T INVENT 1
**CTDATA TABLE2
1
1
2
2
3
3
4
4
5
5
6
6
**CTDATA TABLE1
00AAA109950
00AAB200230
00ABB318970
00BBB400013
00BBB500405
00BBB609875

```

Figure 11-5. A Sample Program that Illustrates the Use of Multiple Tables

## Chapter 12. Using Arrays in Application Programs

This chapter describes how to use the LOKUP and MOVEA operands on the C-specification and how to use arrayname, index as a field.

### USING LOKUP ON THE C-SPECIFICATIONS

The array operation that is available on the C-specification is LOKUP. This operation is similar to the LOOK operation on the A-specification. In addition to the equal-to check provided by the LOOK operation, the LOKUP operation also provides higher-than and lower-than checks. When a table is referred to by a LOKUP operation on the C-specification, it must be described on the A-specification in the program. The sample in Figure 12-1 illustrates a program that uses the LOKUP operation.

```

00001Z*****
00002Z*PROGRAM 69. FIGURE 12-1 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ LOKEXM
00005Z X1SUB          1E                                EOJ
00006A                F INPUT                6          DEVICE(CRT) DSPSIZ(6 80)
00007A                R CHECK
00008A                5 0I                      INSERT(A)
00009A                1 I
00010A                F PREP                 10          DEVICE(DISK D1)
00011A                R ONE
00012A                FLD1                   5 0
00013A                FLD2                   3
00014A                FLD3                   2
00015A                F RANTAB               5          DEVICE(DISK D1) NUMENT(3)
00016A                T EXTAB               5 0
00017C                SUB                   BEGSR
00018C                READ ONE                99
00019C                Z-ADD1                 A          0050
00020C                2 FLD1                 1          LOKUPEXTAB,A          01
00021C                01                    EXFMTCHECK
00022C                ENDSR

```

Figure 12-1. A Sample Program Showing the Use of the LOKUP Operation

Notice that the index is described on the C-specification **1** before it is used with the table. Also, notice how the index is used with the table to locate the position of the successful match in the table **2**.



## USING ARRAYNAME, INDEX AS FIELDS

On the C-specification, arrayname, index combinations can be used wherever a field can be used. The sample in Figure 12-2 illustrates one such use.

```

00001Z*****
00002Z* PROGRAM 70. FIGURE 12-2 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ TFIELD
00005Z A1TRANS      1E              A1      WRITE(*ND)
00006A              F INPUT        100      DEVICE(CRT) DSPSIZ(6 80)
00007A              R TRANS
00008A              INTER1         15 0W     INSERT(00000000000000)
00009A              INTER2         15 0W     1  ERT(00000000000000)
00010A              FLD1           5  I  2  2PMT(ENTER THE ITEM NUMBER)
00011A              FLD2           5  I  4  LOOK(ITEMT A) 3
00012A              FLD2           5  I  4  2PMT(ENTER THE PRICE) LOOK(PRICET B) 6
00013A              EXSR(CALCS)
00014A              15 0I           INSERT(INTER1)
00015A              15 0I           INSERT(INTER2)
00016A              1  I
00017A              F TABLE1      10              NUMENT(3)
00018A              T ITEM         5
00019A              T PRICET       5
00020C              CALCS          BEGSR
00021C              MOVE ITEMT,A 7 INTER1
00022C              MOVE PRICET,B 8 NTER2
00023C              ENDSR
2  CTDA 5  TABLE1
AAAA100001
AAAA200002
AAAA300003

```

Figure 12-2. A Sample Program Showing the Use of the Table, Index Operation

In this sample, the operator enters the item number as prompted 1. The program finds the corresponding entry in the ITEMT table 2 and places the position number in the A index 3. Next, the program prompts the operator for the price 4. It searches the PRICET table 5 and places the position at which the match is found in the B index 6. At this point, the program exits to the C-specifications where it places the value of the ITEMT found in the position designated by index A in the field named INTER1 7. Correspondingly, the program places the matching value from the PRICET table in INTER2 8.

The values of INTER1 and INTER2 are displayed. The displays indicate the process involved in using the program.

## USING THE MOVEA OPERATION

On the C-specification, the MOVEA operation transfers characters to a field.

```

Z*****
Z* PROGRAM 83. FIGURE 12-3 IN THE DE/RPG USER'S GUIDE *
Z*****
ZJ  PROG83
Z 1  SCAN          1E                      EOJ
A      F CRT              80                DEVICE(CRT)
A      R MSG1
A      MSG80           80  B                FMT(ENTER 80 BYTE MESSAGE FOR SCAN)
A      R MSG2
A      LETTER         160  O                'THE MESSAGE CONTAINED LETTERS '
A      BLANKS         160  W
A      F TBLFILE1      1
A      T SCTBL         1                    NUMENT(80)
A      F TBLFILE2      1                    NUMENT(160)
A      T LTBL         1
C* SUBROUTINE SCAN: THIS ROUTINE INPUTS A LINE OF DATA, SCANS AND TABULATES
C* WHICH LETTERS WERE USED, AND DISPLAYS THOSE LETTERS.
C
C      SCAN          BEGSR                START SUBR. SCAN
C      LOOP1         TAG
C                      EXFMTMSG1          GET 80 BYTE MESSAGE
C                      MOVEABLANKS        L TBL          BLANK OUT CHAR. TABLE
C                      MOVEAMSG80        SCTBL           MOVE MSG. INTO TABLE
C                      Z-ADD0             L              30          ZERO LETTER COUNT
C                      Z-ADD1             I              20          INITIALIZE LOOP CTR.
C
C      LOOP2         TAG
C                      MOVE SCTBL,I       CHAR          1          GET NEXT CHARACTER
C                      CABGTCHAR         ENDLP2          IF < 'A', BRANCH
C                      CABLTCHAR         ENDLP2          IF > 'Z', BRANCH
C                      LOKUPLTBL         03CHECK IF CHAR. IN TBL
C                      GOTO ENDLP2        IF 30, BRANCH
C
C      L             ADD 1                L              INCREMENT TABLE INDEX
C      I             CABEQL               SKIPC          IF FIRST, BRANCH
C                      MOVE ',,'         LTBL,L         ELSE INSERT COMMA
C                      L                 L              INCR. INDEX PAST ',,'
C      SKIPC         TAG
C                      MOVE CHAR         LTBL,L         MOVE CHAR INTO TABLE
C
C      ENDLP2        TAG
C                      ADD 1             I              INCREMENT LOOP CTR.
C                      I                 LOOP2          BRANCH IF MORE CHARS.
C                      I                 LETTER         MOVE TABLE INTO FIELD
C                      MOVEALTBL         AND DISPLAY RESULTS
C                      EXFMTMSG2        GO DO ANOTHER
C                      GOTO LOOP1
C      ENDSR

```

Figure 12-3. A Sample Program Showing the Use of the MOVEA Operation

This program prompts the operator to enter a message to be scanned for the letters in it. To facilitate scanning, the program moves the message into a table using the MOVEA operation. Each letter is looked-up in a table used to collect the letters. If the letter is not in the table, the letter is added to the table. A separating comma is added for punctuation. After the entire message is processed, the letters used in the message are moved into a field and displayed.

(

This section contains information about printing data using the IBM 5280. It consists of two chapters:

- **Chapter 13. Unformatted Printing for Data-Entry Programs**
- **Chapter 14. Formatted Printing for Application Programs**

(

## Chapter 13. Unformatted Printing for Data-Entry Programs

Unformatted printing is only available for programs using transaction files (TFILE specified in the job statement). Chapter 1 describes this type of program in detail. Unformatted printing consists of printing the contents of the current buffer (current record) without providing format control such as line skipping. The result is that the default records are printed one at a time with fields in the record strung together. To print an unformatted record, three conditions must exist: (1) PRTFILE(datasetname) must appear in the job statement, (2) a printer data set must be specified on the A-specification, and (3) the operator must use the PRINT key.

The sample in Figure 13-1 illustrates a program that provides unformatted printing.

```
00001Z*****
00002Z* PROGRAM 71. FIGURE 13- 1 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ INTERX TFILE(TEMJOB)
00005Z 1 PRTFILE(OUTPUT)
00006Z B1PURCH NE
00007Z R R1
00008A F EXMP 15 DEVICE(CRT) DSPSIZ(6 80)
00009A R PURCH
00010A 0001001'ITEM'
00011A 0002001'PRICE'
00012A 0003001'QUANT'
00013A FLD1 6 I001010
00014A FLD2 5 2I002010CHECK(RZ)
00015A FLD3 4 0I003010CHECK(RZ)
00016A 4 F TEMJOB 15 DEVICE(DISK D1)
00017A 3 F OUTPUT 2 15 DEVICE(PRINTER X'8000')
00018
```

Figure 13-1. A Sample Program Showing the Use of the PRTFILE Operation

Note that only the input fields 1 in the record are printed. The reason is that only input fields are in the default diskette record. To print the three records as shown in the sample, the operator has to press the Print key three times or use the Auto Rec Adv function. Look at the length field in the file statement for the printer 2. The length of the printed record is determined by the length provided in the file statement. For example, if the length in the print file statement for the sample were 10, and the length of the default record were 15, the last five characters in the record would be lost and the resulting printed record for the first line is:

00AAA10099

Also notice that no redefinition of fields follows the file statement for the printer 3 or for the diskette 4.

If a record has been reformatted, and unformatted printing is used, the results will match the format of the diskette record.

Because unformatted printing is available for programs using transaction files, it is available in the enter, update, verify, and copy operating modes. The mode in which the operator uses the Print key determines what is printed. If the operator uses the Print key for an unformatted printing operation when the enter mode is in effect, only the last record in the data set is printed. The reason for this is that the enter mode goes to the end of the data set in anticipation of adding new records. If you are using the unformatted printing operation, be sure you know which mode is in effect when the operation is used.

## Chapter 14. Formatted Printing for Application Programs

Formatted printing differs from unformatted printing in two ways:

- The program determines the arrangement of data as it is printed.
- Printing is controlled by operations in subroutines on the C-specifications rather than by the operator using the Print key.

Formatted printing can be used with all programs except those exclusively using transaction files (no subroutines). Chapter 1 provides information about the various types of programs available for use in DE/RPG.

To use formatted printing, a print data set must be described on the A-specification and a write operation for the print data set must be specified on the C-specification.



The sample in Figure 14-1 illustrates a very simple formatted printing operation.

```

00001Z*****
00002Z* PROGRAM 72. FIGURE 14-1 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ PRTEX
00005Z R1HEADER      E                                WRITE(*NO)
00006A      F VIEW          90                        DEVICE(CRT) DSPSIZ(6 80)
00007A      R HEADER
00008A      CUSNA          20      I      1 PMT(ENTER CUSTOMER NAME)
00009A      STREET        20      I      1 PMT(ENTER STREET ADDRESS) 2
00010A      CISTA         20      I      PMT(ENTER CITY AND STATE) EXSR(PRNT)
00011A      F PRINT       132
00012A
00013A      R NAME
00014A      CUSNA          20
00015A      R ADDR
00016A      STREET        20
00017A      R CITY
00018A      CISTA         20
00019A      F CUSTOMER    90
00020A      R HEAD
00021A      K CUSNA        20
00022A      STREET        20
00023A      CISTA         20
00024C      PRNT          BEGSR
00025C      WRITEADDR
00026C      5 WRITEADDR
00027C      WRITECITY
00028C      WRITEHEAD    7
00029C      ENDSR
005
010
040
3 SPACEB(2) SPACEA(2) 4
6 SPACEA(0)
SPACEA(3)
DEVICE(PRINTER P1)
FORM(55)

```

●		●
●	MR. N.E. EDDY	●
●	233 OSBORNE DRIVE	●
●		●
●		●
●	FRAMPTON, NEW MEXICO	●
●		●
●	MRS. JANE RANSOM	●
●	16 FORWIRTH ST.	●
●		●
●		●
●	NEW HAMPTON, CONN.	●
●		●
●		●

Figure 14-1. A Sample Program Showing a Simple Formatted Print Operation

This program is prompting the operator for customer identification information **1**. When the entry for a customer is complete, the program exits to a subroutine on the C-specification **2**. The subroutine writes three records: NAME, ADDR, and CITY via the printer.

Notice that the printing does not begin on the first line of the paper. The SPACEB(2) **3** operation is telling the IBM 5280 to skip two lines before printing the record. The SPACEA(2) **4** operation is telling the IBM 5280 to print the first record and then skip two lines before printing the next record.

The second record **5** is printed and no lines are skipped before the last record **6** is printed. After the last record is printed, four lines are skipped. The last line of the subroutine **7** writes a single record (containing three fields) in key sequence into the diskette data set. The entire process, beginning with the prompt to the operator, begins again.

This was a simple sample of formatted printing. More complex samples are provided later in this chapter. The topics that are presented in this chapter are:

- Designing a print data set to be used with an existing form
- Designing a print data set to create a form

**DESIGNING A PRINT DATA SET TO BE USED WITH AN EXISTING FORM**

The sample in Figure 14-2 illustrates the form you use to design a print data set. This form represents a typical preprinted form used by many businesses. The form is marked to indicate the amount of space between entry areas.

↙ Top of Page

Align the Printer at this Mark 2 Inches from Alignment Mark to First Print Line

**RAYMOND B. ATHERTON LAW FIRM**  
*Bismarck, North Dakota*

Date of Service	Type of Service	Associate Who Helped You
25.6 mm (1 Inch)	112. mm (4.4 Inches)	46 mm (1.8 Inches)
12.8 mm (0.5 Inch)		

Figure 14-2. The Form to be Used for the Print Operation in Figure 14-4

The first rule to remember when working with preprinted forms is that DE/RPG prints the data exactly where you specify.

This means that you must tell DE/RPG exactly where to place each piece of data. The placement of data not only includes the line location, it also includes the space location. To understand how to make these measurements, you must know something about the printer.

For example, the IBM 5256 printer prints 132 characters per line and six or eight lines per inch. The maximum line width is 13.5 inches. This means that each character occupies approximately 0.1 inch (13.5 divided by 132) and that each printed line occurs approximately every 0.17 inches (1 divided by 6) for six lines per inch and approximately every 0.13 inches for eight lines per inch. If the form you are using is 10 inches long, the number of lines that can be printed on the form at six lines per inch is 60 lines (10 inches multiplied by 6). If your form is 8.5 inches wide, the maximum number of characters that can be printed on the form at 10 characters per inch is 85 characters (8.5 divided by 0.1).

You must measure your preprinted form to find the width and length of the form. Then determine where you want to print each line of data. Review the illustration of the form in the sample in Figure 14-3.

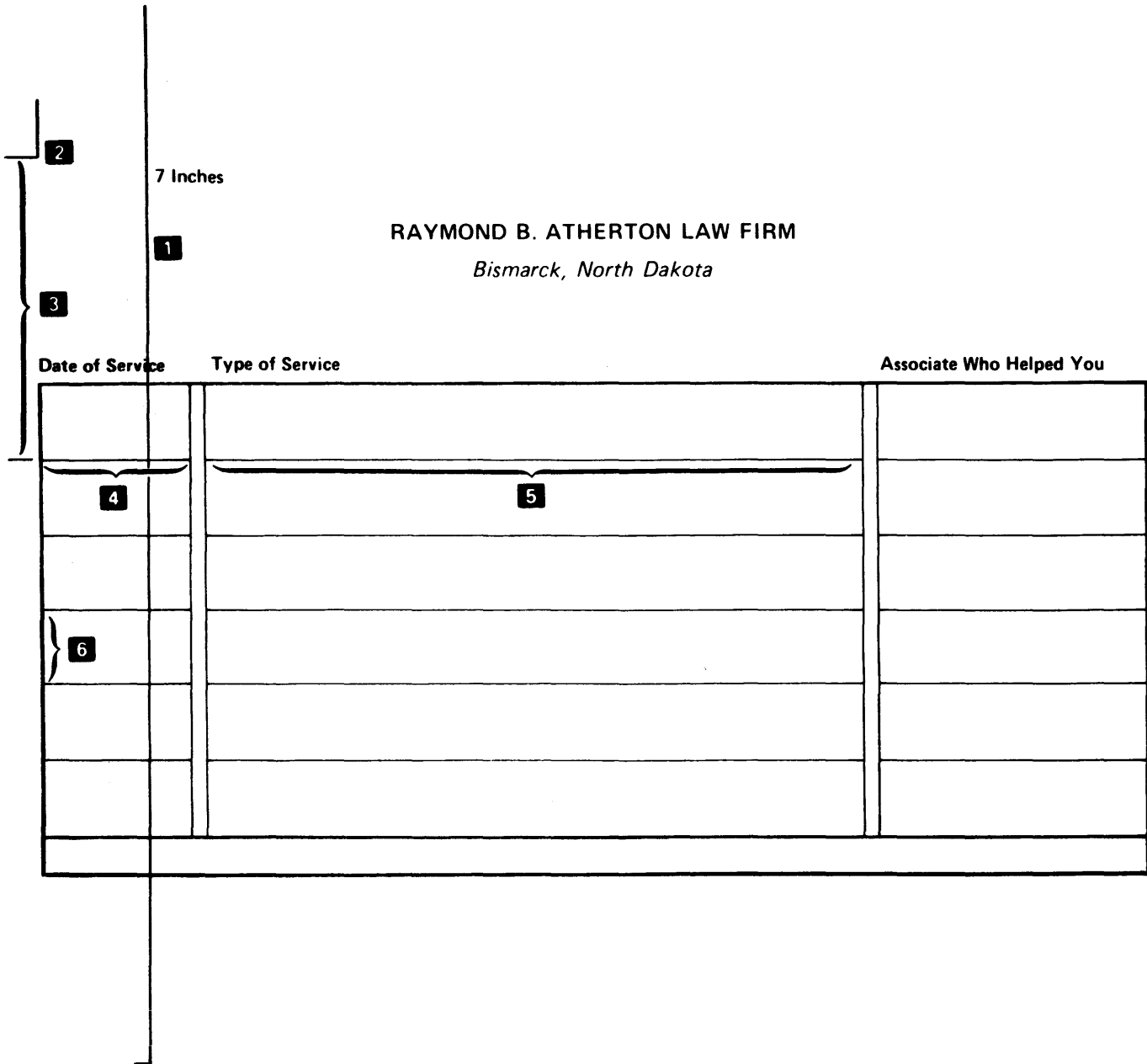


Figure 14-3. The Measurements that Determine the Spacing Required for the Program in Figure 14-4

Assume that this form is 7 inches long **1**; that is, 42 lines long (six lines per inch multiplied by 7 inches). The alignment mark **2** tells the operator to load the form with space at the top. Counting from the alignment mark, there are 2 inches **3** to the first print line. Two inches is 12 lines (2 multiplied by 6). Therefore, you know that you want to skip 12 lines before printing the first line. Next, look at the marks on the first print line. The first field to be printed is 1 inch long **4**. This is a date field. The date is always six characters long (mmyydd). If each character takes 0.1 inch, then 6 characters take 0.6 inch. This does not fill the field. You will have to do something to start the printing for the next field in the proper column. If you measure across the vertical separators, you find that you need an extra six characters to ensure that the next field **5** is printed in the proper beginning position.

You can determine the maximum length of the second field by dividing the length of the field by the number of inches per character. For example, the field is 4.4 inches long and each character takes 0.1 inch; therefore, the maximum number of characters that can be printed in this field is 44. The length of the last field must be determined next. This field is 1.8 inches, so divide 1.8 by 0.1. The field can be a maximum of 18 characters.

Next, you must determine how many spaces to skip between lines of print. You know from the marks that there is 0.5 inches between lines **6**. If a line is printed every 0.17 inches (6 lines per inch) and there is 0.5 inch between the lines you want printed, then you should skip 3 lines after printing a line (0.5 divided by 0.17).

You have enough information to begin writing the program that prints the data from a diskette data set onto this form in the locations you specify. The sample in Figure 14-4 illustrates this program.

```

00001Z*****
00002Z* PROGRAM 73. FIGURE 14-4 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004Z I PRNTJOB
0000 1 Q1START 1E EOJ
00006A F BILMAST 164 DEVICE(DISK D1)
00007A R BILLING
00008A ID 6 0
00009
00010A CUSNA 90
00011A DATE 6
00012A DESC 44
00013A 5 ASSOC 15 18 6
00014A 4 F PRINOUT 132 7 DEVICE(PRINTER P1) FORM(42)
00015A R FIRSTA 8 SKIPB(12)
00016A DATE 6 1 9
00017A 10 DESC 44 11 13
00018A R FIRST SPACEB(1)
00019A DATE 6 001
00020A DESC 44 13
00021A R SECOND SPACEA(2) 13
00022A ASSOC 14 18 058
00023A R NEXT SKIPB(40) SKIPA(1) 22
00024A CUSNA 90 1 21
00025C START BEGSR
00026C 2 READ BILLING 05
00027C NEXTON TAG
00028C 3 WRITEFIRSTA
00029C NOW TAG
00030C Z-ADDID 12 TEMP 60
00031C WRITESECOND
00032C ID READ BILLING 16 05
00033C 17 COMP TEMP 18 01
00034C 01 WRITEFIRST
00035C 01 19 GOTO NOW
00036C READPBILLING 05
00037C 20 WRITENEXT
00038C READ BILLING 05
00039C N05 GOTO NEXTON
00040C 23 ENDSR
00041
00042

```

Figure 14-4. A Sample Program that Prints a Form

The program begins with an exit to a subroutine **1**. The subroutine is reading a record **2** from the diskette data set. The subroutine then tells the IBM 5280 to print the record named FIRSTA **3**. The contents of this record are described on the A-specification **4**. Notice that a file description for the printer precedes the record description **5**. This file description contains an operation that tells the IBM 5280 the length of the form **6**. You had determined this length by multiplying the form length in inches times the number of lines per inch that could be printed. Next, the first record that is printed is described. The record description contains the operation that skips the first 12 lines of the form **7** to properly position the printing at the first vertical print position.

The field descriptions determine what is printed on the line. The first field is printing the date **8** from the data set in print positions 1 through 6 **9**. The second **10** is inserting the contents of the description in the data set into print positions 13 through 57 **11**.

When this record is completed, the subroutine adds a field in the record to zero and places it in a newly created field **12**. This new field will be used later in the program to determine if the next record that is read is for the same customer and, therefore, if it should be printed on the same form. The SECOND record is written.

The SPACEA(2) **13** operation in the record advances the printer 2 lines when this record is completed. That locates the next print line in the proper location for the next record. The field currently being printed is 18 characters long **14**. The field is taken from the diskette data set **15**. At the completion of this record, the subroutine reads the next record **16**. The subroutine compares the identification number in the record to the identification number of the previous record **17**. If these numbers match, an indicator **18** is set. This indicator conditions an operation **19** that directs the subroutine back to the set of operations that printed the first line. In this way, multiple lines of printing are performed for a customer who has more than one transaction. If the indicator is off (there are no other transactions), the subroutine prints the customer's name at the bottom of the form **20**. The SKIPB(40) **21** places the print position at line 40. The SKIPA(1) **22** advances the print position past the end of the form and forces a new page. The next record (for a new customer) is read and printed. When all records have been read from the data set (indicator 05 is on), the subroutine ends **23**.

The sample in Figure 14-5 illustrates the result of using this program.



RAYMOND B. ATHERTON LAW FIRM  
Bismarck, North Dakota

Date of Service	Type of Service	Associate Who Helped You
061781	WILL \$150.00	JUSTIN R. GROSSMAN
063081	ADOPTION PAPERS \$55.00	JUSTIN R. GROSSMAN

SILAS MERINA, APT 88, RIDEPORT ROAD, BISMARCK, N.D.

Figure 14-5 (Part 1 of 2). An Example of the Printed Form Resulting from Using the Program in Figure 14-4

RAYMOND B. ATHERTON LAW FIRM

*Bismarck, North Dakota*

Date of Service	Type of Service	Associate Who Helped You
06038i	LIEN SETTLEMENT \$83.95	MARVIN F. MAYNARD

MARJORIE DAVIS, 1615 DARTMOUTH AVE., BISMARCK, N.D.

Figure 14-5 (Part 2 of 2). An Example of the Printed Form Resulting from Using the Program in Figure 14-4

## DESIGNING A PRINT DATA SET THAT DESIGNS THE FORM TO BE USED

Before you can begin writing the program that creates the form, you must design the form on paper. Part of this design involves the appearance of the form, and the other part involves the measurements to be used in describing the form in the program.

The sample in Figure 14-6 illustrates the form design that is used in this topic.

The programmer has decided to use 8.5- by 11-inch continuous forms rather than having special forms made. This means that each form is 11 inches long (66 lines long-11 inches by 6 lines per inch).

The form begins 2 inches from the top perforations **1**. The programmer wants the headings centered, so he must first calculate how many characters there are in each heading. JERRY'S SALVAGE YARD contains 20 characters. Because each character occupies 0.1 inch, 2 inches are required for this heading. There are 8.5 inches across the page. The center of the page is 4.2 inches **2** from each margin. If the heading is 1 inch on either side of this center line, the heading is centered. Mark the heading to begin 3.2 inches **3** from the left margin.

Leave a space between this first heading and the address to make both more readable. The second heading has 14 characters (1.4 inches). This means that 0.7 inches should be on either side of the center mark to center the heading. Mark the heading to begin 3.4 inches **4** from the left margin.

The third heading is 15 characters. Mark the heading to begin in the same position as the second heading **5**.

The heading is complete. Next space down 3 lines (which is 0.5 inches). Begin the customer address 1 inch (10 characters) from the left margin **6**. Leave enough space for the entire customer address as shown in the illustration **7**.

Start the column headings for the bill. Beginning at the left margin, indent 1 inch and write Date **8**. The heading Date should occupy 0.5 inches. You know that the dates require 0.6 inches (6 digits at 0.1 inch each), so leave about 4 characters (0.4 inches) space after the heading. This space provides separation between this field and the next field. Write the heading Parts **9**. Include enough space after this heading so the associated field has enough space to provide a description of the parts that are being billed. Four inches following the heading Parts should be sufficient **10**.

Write the heading for the last column (Cost) **11**. In this sample, the maximum cost of a part sold by this company is 5 digits long (2 decimal positions). Therefore, the maximum width the field requires is 0.6 inches (1 character for the decimal point). Make the column width 1 inch to provide margin space.

Next, you need to provide a line beneath the column headings to separate the headings from the entries. This line should begin 1 inch from the left margin and should be 6.5 inches long.

Finally, you should have an area that gives the accumulated balance of the bill. Leave 4 inches for individual entries **12**. At the bottom of the form, skip three lines after the last entry. Now go to the right 6 inches and write the heading Balance in the lower right corner of the bill **13**. Leave enough space for a 9-character field.

You now have enough information to begin writing the program that will print this form.

Sample 14-6

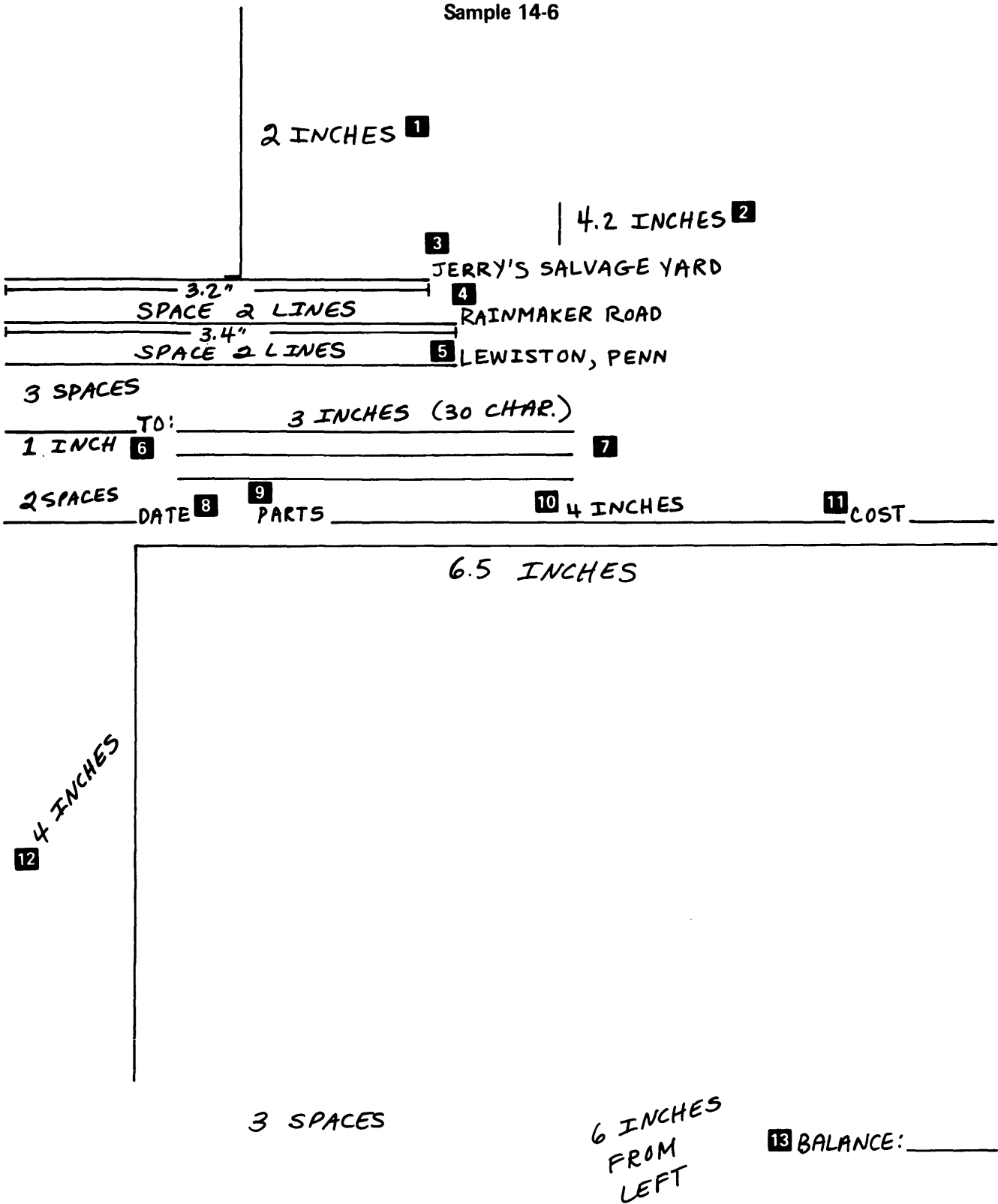


Figure 14-6. The Design for the Form to Be Created by the Program in Figure 14-7

The sample in Figure 14-7 illustrates the program that creates this form. An illustration is included to show you a printed copy of the form. To include entries on the form as it is created, simply use the principles you learned in the first part of this chapter.

```

00001 Z*****
00002 Z* PROGRAM 74. FIGURE 14-7 IN THE DE/RPG USER'S GUIDE *
00003 Z*****
00004 ZJ PRTEX
00005 Z P1FORM 1E EOJ
00006 A F FORMPRT 132 DEVICE(PRINTER P1) FORM(66)
00007 A R NAMEHEAD SKIPB(12) SPACEA(2)
00008 A 033'JERRY'S SALVAGE YARD'
00009 A R ADDHEAD SPACEA(2)
00010 A 035'RAINMAKER ROAD'
00011 A R CITHEAD SPACEA(3)
00012 A 035'LEWISTON, PENN'
00013 A R CUSNAHD SPACEA(1)
00014 A 011'TO:'
00015 A R CUSADHD SPACEA(1)
00016 A 011' '
00017 A R CUSCITHD SPACEA(2)
00018 A 011' '
00019 A R ENTRYHD SPACEA(1)
00020 A 011'DATE'
00021 A 020'PARTS'
00022 A 065'COST'
00023 A R LINE
00024 A 011'-----+
00026 A R LINEND SKIPB(45) SPACEA(2)
00029 A R BAL SKIPA(55)
00030 A 060'BALANCE:'
00031 C FORM BEGSR
00032 C NEXT TAG
00033 C WRITENAMEHEAD
00034 C WRITEADDHEAD
00035 C WRITECITHEAD
00036 C WRITECUSNAHD
00037 C WRITECUSADHD
00038 C WRITECUSCITHD
00039 C WRITEENTRYHD
00040 C WRITELINE
00041 C WRITELINEND
00042 C WRITEBAL
00043 C GOTO NEXT
00044 C ENDSR

```

Figure 14-7. A Sample Program that Creates a Form

JERRY'S SALVAGE YARD  
RAINMAKER ROAD  
LEWISTON, PENN

TO:

DATE	PARTS	COST
-----		

BALANCE :

Figure 14-8. The Form that Is Created When the Program in Figure 14-6 Is Used

## Part 7. Using Calculations

This section provides information about various ways to perform calculations using DE/RPG.

The types of calculations that are available on the A-specification:

- INSERT(constantoperatorconstant)
- INSERT(constantoperatornamed field)
- INSERT(named fieldoperatornamed field)
- ADD(\*TOTn or named field)
- TADD(\*TOTn or named field)
- TSUB(\*TOTn or named field)
- SUB(\*TOTn or named field)

operator can be + (addition), - (subtraction), \* (multiplication) or / (division)

\*TOTn stands for counters 1 through 9

The topics presented in this part are:

- Considerations about using calculations
- Using named fields in calculations
- Using counters in calculations
- Keeping online totals using the A-specification



## USING NAMED FIELDS IN CALCULATIONS

The sample in Figure 15-1 illustrates a simple data-entry program that uses calculations. If named fields are used in a calculation, they must have been created as numeric fields; that is, they must have had a decimal position entry **1** when they were created. Numeric fields are a maximum of 15 positions long.

```
00001Z*****
00002Z* PROGRAM 75. FIGURE 15-1 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ INDICEX TFILE(PUTOUT)
00005Z G1SHOW 1E
00006Z R G1
00007A F EXMW 13 1 DEVICE(CRT) DSFSIZ(6 80)
00008A R SHOW
00009A QUANT 3 01 PMT(ENTER THE QUANTITY)
00010A COST 4 2I PMT(ENTER THE COST)
00011A PAY 6 2I INSERT(QUANT*COST)
00012A F PUTOUT 13 DEVICE(DISK D1)
```

Figure 15-1. A Sample Program Showing the Use of Named Fields in Calculations

If a field is not named, it cannot be referred to. If a field is not created as a numeric field, it cannot be used in the calculation. A field with no decimal position entry is considered a character field. Character data cannot be used in a calculation.

When you include an arithmetic expression in an insert operation, be aware that DE/RPG does not include embedded parentheses to denote that an expression should be handled as a unit; all expressions are evaluated from left to right with multiplication and division being performed first and then subtraction and addition being performed. For example, the expression `INSERT(1+3*14)` results in 43 (1+42) rather than 56 (4\*14).

When you use constants in a calculation for an insert operation, decimal positions are permitted. For example,

```
INSERT(.15*PAY)
```

could be used to determine 15% of the field named PAY.

The lengths of the source field and the receiving field must be considered. For example, if you are adding two fields, each with a length of 5, the receiving field should have at least a length of 6. See the sample in Figure 15-2.

```
00001Z*****
00002Z* PROGRAM 76. FIGURE 15-2 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ  CALCEXM                                TFILE(KEEP)
00005Z  G2FIGURE      1E
00006Z                R                        G2
00007A                F DISP                    15      DEVICE(CRT) DSPSIZ(6 80)
00008A                P FIGURE
00009A                1 COST1                    5  2I    PMT(ENTER THE PRICE OF ITEM1)
00010A                2 COST2                    5  2I    PMT(ENTER THE PRICE OF ITEM2)
00011A                PAY                        3  5  2I    INSERT(COST1+COST2)
00012A                F KEEP                    15      DEVICE(DISK D1)
```

Figure 15-2. A Sample Program Showing the Effect of the Length for Fields Involved in a Calculation

If the receiving field has a length less than the source fields, significant digits can be lost from the calculation.

If, `COST1` **1** equals 500.00, `COST2` **2** equals 600.00 and `PAY` **3** has a length of 5 **4** with two decimal positions, `PAY` will be 100.00 rather than the 1100.00 it should be.

## USING COUNTERS IN CALCULATIONS

The counters (\*TOT1 through \*TOT9) are 15 positions wide. All entry of data into a counter is accomplished by the ADD, TADD, TSUB, or SUB operations on arithmetic calculation operations. All entry of data in a counter is right-aligned.

Transfer of data from a counter is accomplished with the insert operation. The insertion of data from a counter into a receiving field is right-aligned. See the sample in Figure 15-3.

```

00001Z*****
00002Z* PROGRAM 77. FIGURE 15-3 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ  COUNTEX                                TFILE(DISKET)
00005Z  H2EXMPLE      1E
00006Z                R                        H2
00007A                F CALCL                60      DEVICE(CRT) DSPSIZ(6 80)
00008A                R EXMPLE
00009A                NAME                    6 0I    1    PMT(ENTER THE FIRST PRICE) ADD(*TOT+
00009A                PRICE                  6 2I    3    1)
00010A                LOOK                   6 2I    3    PMT(ENTER THE SECOND PRICE)
00011A                OKAY                   6 2I    3    ADD(*TOT2)
00012A                F DISKET               60      4    INSERT(*TOT1)
00013A                F DISKET               60      5    INSERT(*TOT2)
00014A                F DISKET               60      DEVICE(DISK D1)

```

**Figure 15-3. A Sample Program Showing the Use of Counters**

If you assume that the entries are both 004488 **1**, then the contents of \*TOT1 is 0000000000004488 **2** and the contents of \*TOT2 **3** is 0000000000000044.

When \*TOT1 is inserted into the field **4**, the field contains 4488.00. When \*TOT2 is inserted into the field **5**, the field contains 0044.00.

The counters use two functions that provide decimal alignment (ADD and SUB) and two that do not provide decimal alignment (TADD and TSUB).

Consider an example contrasting the ADD and TADD functions. The sample in Figure 15-4 illustrates the differences.

```

00001Z*****
00002Z* PROGRAM 78. FIGURE 15-4 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ DECOUNT                                TFILE(WRITE)
00005Z I1DUMMY          1E
00006Z                      R                      I1
00007A          F XAMP          50          DEVICE(CRT) DSPSIZ(6 80)
00008A          R DUMMY
00009A          QUANT 1          2 0I          PMT(ENTER THE QUANTITY)
00010A          2 PRICE          3 2I          PMT(ENTER THE PRICE)
00011A          TOTAL 3          5 2I          INSERT(PRICE*QUANT) ADD(*TOT1) 4
00012A          5 TADD(*TOT2)
00013A          15 2I          INSERT(*TOT1)
00014A          15 2I          INSERT(*TOT2)
00015A          F WRITE          50          DEVICE(DISK D1)

```

Figure 15-4. A Sample Program Showing the Difference between the ADD and TADD Operations

If QUANT **1** equals 2 and PRICE **2** equals 2.44, the product of multiplying QUANT and PRICE is 4.88. TOTAL **3** contains 0044.88. \*TOT1 **4** contains 0000000000000004. \*TOT2 **5** contains 0000000000000488.

The fields in which \*TOT1 and \*TOT2 are inserted respectively are:

00000000000000400 and 0000000000048800.

If you intend to use the counters in online totals, be careful of the way you use decimal positions.

Two more consideration are important in using counters for online totals: (1) resetting the counters and (2) controlling the counters as data is loaded into them.

The sample in Figure 15-5 illustrates a program that has three types of records in the data set: header **1**, detail **2**, and trailer **3**. There is always one **4** header record, multiple **5** detail records (this number varies), and one **6** trailer record. The counter is being used to keep an online total of the number of purchases made on the detail records for a single customer.

```

00001Z*****
00002Z* PROGRAM 79. FIGURE 15-5 IN THE DE/RPG USER'S GUIDE *
00003Z*****
00004ZJ RESEXMF TFILE(BILLEX)
00005Z K1HEADER 1E 4 K2
00006Z K2DETAIL 5 NE K3
00007Z K3TRAILER 1E 6 K1
00008A F INPUT 75 DEVICE(CRT) DSFSIZ(6 80)
00009A 1 R HEADER
00010A CUSNA 30 I PMT(CUSTOMER NAME)
00011A ADDR 30 I PMT(ADDRESS)
00012A COUNT 15 2W
00013A 2 R DETAIL
00014A ITEM# 6 I PMT(ENTER THE ITEM#)
00015A NUMSLD 3 0I PMT(QUANTITY SOLD)
00016A PERPRI 4 2I PMT(PRICE PER ITEM)
00017A TOTAL 6 2I INSERT(NUMSLD*PERPRI)
00018A 7 ADD(COUNT)
00019A 3 R TRAILER
00020A 9 2I INSERT(COUNT) 8
00021A 9 RESET(COUNT)
00022A F BILLEX 75 DEVICE(DISK D1)

```

**Figure 15-5. A Sample Program Showing the Use of a Named Field as an Online Counter**

The sample in Figure 15-5 illustrates how this online total is being kept by adding the results from each detail record into the counter **7** at the completion of the record. At the completion of all detail records for a single customer, the contents of the counter is inserted **8** into the trailer record. The counter is then reset **9** so that it does not carry a balance from one customer to another.

This chapter describes how to use the EOJ keyword on the Z-specification to chain DE/RPG jobs. Chaining links one DE/RPG job to the next job. The next job in the chain can be another DE/RPG job or any other program.

### USING EOJ ON THE Z-SPECIFICATION

The name parameter in the EOJ keyword provides the name of the object program data set (as it appears on the diskette) that you want to go to upon completion of the current program. The name parameter can be either a constant name or a variable name.

A constant name cannot be changed by an entry from the keyboard. The name is coded on the Z-specification enclosed in apostrophes. For example, if you always follow a payroll program with the check-writing program, you name the check-writing program in the EOJ keyword parameter within apostrophes.

A variable (dummy) name is one that is set by the program to select the next program to be loaded. The variable name is not enclosed in apostrophes.

This chapter illustrates both techniques that can be used to chain programs. A DE/RPG program that loads a utility program illustrates the use of a constant name. Another program illustrates how you can use a menu to allow the operator to select the next program.

## USING A CONSTANT NAME TO SELECT THE NEXT PROGRAM

The sample program in Figure 16-1 shows the constant name **SYSPRINT** on the EOJ keyword. **SYSPRINT** is loaded when the current program is completed. Everytime the current program is run, **SYSPRINT** is the next program run.

```

00001 Z*****
00002 Z* PROGRAM 80. FIGURE 16-1 IN THE DE/RPG USER'S GUIDE *
00003 Z*****
00004 ZJ LOKEXM
00005 Z X1SUB          1E          EOJ('SYSPRINT' X'4800')
00006 A             F INPUT          6          DEVICE(CRT) DSPSIZ(6 80)
00007 A             R CHECK
00008 A             5 0I          INSERT(A)
00010 A             1 I          PMT(PRESS EOJ TO SKIP SYSPRINT)
00011 A             F PREP          10         DEVICE(DISK D1)
00012 A             R ONE
00013 A             FLD1          5 0
00014 A             FLD2          3
00016 A             FLD3          2
00017 A             F RANTAB        5          DEVICE(DISK D1) NUMENT(3)
00018 A             T EXTAB        5 0
00019 C             SUB          BEGSR
00020 C             READ ONE          99
00022 C             Z-ADD1          A          50
00023 C N99          FLD1          LOKUPEXTAB,A          01
00024 C 01
00025 C             ENDSR

* ADDR CONSTANT
* 02F0 'SYSPRINT'
* 02F8 'PRESS EOJ TO SKIP SYSPRINT'
* 0312 1
*
* ADDR NAME
* 0313 FLD1
* 0318 FLD2
* 031B FLD3
* 031D A
*
* OBJECT PROGRAM MAP
*ROUTINE ENTRY POINTS
*EP RTN DESCRIPTION
*05CC RG99 - End of job processor
*063C RG80 - Verify mode error display
*0700 RG86 - Physical buffer allocation
*07CC RG01 - Keyboard external status routine
*0A1C RG03 - KB/CRT I/O management routine
*0BC4 RG30 - Diskette external status routine
*0D28 RG32 - Diskette I/O management routine
*1110 RG51 - I/O driver - full function
*
*1744 Z-spec driver entry point
*18FC Program entry point
* 7,936 Is the program length.

```

Figure 16-1. A Sample Program Showing the Use of a Constant Program Name

## USING A VARIABLE NAME TO SELECT THE NEXT PROGRAM

The sample program in Figure 16-2 shows how to chain one program to another by using the variable name parameter in the EOJ keyword.

The variable name PROG **1** is coded in the name parameter of the EOJ keyword and loads the next program when the format OPT **2** is executed. The operator is prompted for the program name of the program to run next **3**.

The prompt that appears on the display screen for this sample program looks like this:

```

0 0001      A 08 40          1
SELECT YOUR PROGRAM NAME: | | | | | | | |

```

```

00001      Z*****
00002      Z* PROGRAM B1.  FIGURE 16-2  IN THE DE/RPG USER'S GUIDE  *
00003      Z*****
00004      ZJ  SHOWEOJ
00005      Z 1 OPT          1E          EOJ(PROG X'4000') 1
00006      Z 2              WRITE(*NO)
00007      A          F DSPLY          480          DEVICE(CRT)
00009      A          R OPT
00010      A          PROG          8  I  2  2PMT(SELECT YOUR PROGRAM NAME:) 3
00011      A          CHECK(DR) DSPATR(CS)

* ADDR  CONSTANT
* 02F0  'SELECT YOUR PROGRAM NAME:'
*
* ADDR  NAME
* 0309  PROG
*
*      OBJECT PROGRAM MAP
*ROUTINE ENTRY POINTS
*EP     RTN     DESCRIPTION
*0AFC   RG99   - End of job processor
*0B6C   RG86   - Physical buffer allocation
*0C38   RG01   - Keyboard external status routine
*0E88   RG03   - KB/CRT I/O management routine
*1030   RG50   - I/O driver routine
*
*12F4   Z-spec driver entry point
*14CC   Program entry point
* 6,400  Is the program length.

```

Figure 16-2. A Sample Program Showing the Use of a Variable Program Name





**access methods:** A technique for moving data between main storage and I/O devices.

**attribute:** A characteristic. For example, attributes of a displayed field could include high intensity, reverse image, and column separators.

**background program:** An application program that can be executed in a background partition.

**complex reformatting:** The changing of the arrangement of data from its entry sequence to its diskette format. This might include the omission of fields or the merging of fields from multiple record formats.

**counters:** A register or storage location used to accumulate a user-defined numeric value identified by \*TOTn.

**data-entry program:** A DE/RPG program that is used primarily for putting volumes of data on diskette. Normally it makes use of the edit and check functions provided by DE/RPG.

**data set:** An organized collection of related data records treated as a unit and existing on diskette.

**End of Job key:** For the data entry keyboard, the Cmd numeric shift Dup key sequence that an operator uses to terminate a job. For the typewriter keyboard, the CMD, 7 key sequence that an operator uses to terminate a job.

**format:** A specific arrangement of information in a record or on a display screen.

**formatted printing:** A print operation in which each field is specified by the program.

**indexed data set/file:** A data set in which the position of each record is recorded in a separate file called an index. The index contains an index key and disk address for each record in the file.

**indexes:** A field that is used with a table function to contain the table position at which a match occurs.

**indicator:** A switch identified by two digits, which are either OFF or ON and that can be used to condition actions on the A- and C-specifications.

**interactive application program:** A DE/RPG program that is used primarily for processing data. This type of program requires some data input from the operator in order to perform the processing. Some edit and check functions may be used.

**I/O (input/output):** Components that receive or return data, such as the keyboard, display, printer, magnetic stripe reader, and diskette.

**keys:** One or more characters included in a data record that are used to identify or control the use of that data.

**keyword:** For the A- and Z-specifications the control characters that define the action to be taken, such as CHECK, or specify characteristics of the program, such as TFILE.

**logical device:** A two-character identifier that can be used to specify the device to be used, for example D1.

**mode:** The operational category of a data station, such as enter, verify, update, rerun, and execute.

**noninteractive application program:** A DE/RPG program that is used primarily for processing data. This type of program requires little if any operator interaction.

**operation:** For the C-specifications the control characters that define the action to be taken, such as EXFMT for execute the format named in factor 2.

**partition:** An area of the IBM 5280 storage in which programs can execute.

**physical device:** A four-character address for the device, such as 4000 for a disk or 8000 for a printer.

**status line:** For the IBM 5280, the first line on a display screen. This line provides operational information.

**subroutine:** A group of instructions that always returns the control to the calling routine. Subroutines occur on C-specifications and always begin with a BEGSR and end with an ENDSR operation.

**unformatted printing:** A print operation that is initiated by the Print key.



1-10 11-20 21-30 31-40 41-50 51-60 61-70 71-80 81-90 91-100 101-110

1.5 Identifies the source statement order.

6 Identifies the type of source statement.

7 An \* indicates a user comment.

8 Reserved.

9-10 Specifies the indicator that is used to control field bypassing or displaying user error codes.

11-16 Reserved.

17 Defines the type of statement: F = data set, R = record, K = key field, T = table, blank = field.

18 Reserved.

19-26 Specifies the name for: data set (max 8 characters), record (max 8 characters), field (max 6 characters) or table (max 6 characters).

27-29 Reserved.

30-34 Specifies the length:  
data set = maximum record length is required.  
record = number of characters (1-8192)  
field = number of characters (1-256 for alphanumeric or 1-15 for numeric).

35 Defines the data type for the field:

A = alpha	S = signed numeric
B = binary	V = right half only
C = use SHIFT keyword	W = right half shift
D = digits only	X = alpha only
H = hexadecimal	Y = numeric only
N = numeric	b = alpha or numeric
P = packed	depending on the field type.

36 Reserved.

37 Specifies the number of decimal positions (0-9).

38 Specifies how the data in a field on the display screen is processed. I = input, O = output, B = both, W = workspace.

39-44 Specifies the location of the field within a record or on the display screen.

45-80 Specifies parameters for data sets, files, records, tables, and fields:

*Data sets and files (F in column 17)*

BLKING ([\*DBL] [\*FMTU or \*FMST])—specifies blocking characteristics for data sets:  
\*DBL specifies to use two physical buffers  
\*FMTU specifies that the records are unblocked (Basic or H data exchange)  
\*FMST specifies that the records are blocked and spanned (1 data exchange)

DEVICE (dev-type address)—physical device type for the data set:  
dev-type is COMM or COMM 3270 (communications), CRT (keyboard/display), DISK (diskette), MREAD (magnetic stripe reader), PRINTER (printer).  
address is the 2-character logical ID or the 4-character device address (X'xxxx' where xxxx is the physical address).

DSPSIZ (lines 80)—specifies display size: lines = 6, 12, or 24.

FORM (length [overflow-line overflow-ind])—Specifies the printer page size; length specifies the lines available on the page, overflow-line specifies the line that sets the overflow indicator on, and overflow-ind specifies the indicator that is set on.

INDEX ([storage] [data set])—At least one parameter must be specified. Specifies the storage reserved for the sparse index and the index data set name: storage specifies the space required for the index, data set specifies the name of the index data set.

LABEL (name of data set)—diskette data set name.

LOGON ('message' or name)—Specifies the log on information when required for communications. The parameter can be either a message enclosed in single quotes or a variable name.

NUMENT (number)—number of records in a data set when used for dynamic allocation of the data set or the number of entries in a table.

*Records (R in column 17)*

DSPATR (attr...)—Specifies the display attributes that apply to all the fields in the record.

MARK (\*POSnnnn)—Specifies the position in a data record where an E is placed if the Field Mark key is pressed.

VMARK (\*POSnnnn)—Specifies the position in a data record where a V is placed after the record is verified.

RECID (\*POSnnnn 'c')—Specifies the position that identifies the single character record type 'c' from a data set with more than one record type (nnnn is 1 to 1892).

SPACEA (n)—Causes the printer to space n lines after the record is printed.

SPACEB (n)—Causes the printer to space n lines before the record is printed.

SKIPA (n)—Causes the printer to skip to line n after the record is printed.

SKIPB (n)—Causes the printer to skip to line n before the record is printed.

*Field (Blank in column 17)*

ADD (name)—Adds the data in the current field to the named field with decimal alignment.

AUXDUP (name)—Duplicates data from the named field if the Dup key is pressed or the Auto Dup/Skip switch is on.

AUXST (name)—Stores the current field in the named field if the Auto/Dup switch is on.

CHECK (parameter)—Specifies the keyboard edits to be applied to the field.

COMP (test fld 1 @...fldn 'literal' [indicator])—Compares the current field with a named field, the specified expression, or a literal and optionally turns on an indicator if the compare is true.

DSPATR (attr...)—Controls the display attributes for each field.

EDTCDE (code 'float')—Specifies the editing that is to be applied to data in numeric fields, where:  
• code is a single character that controls the use of editing characters specified by the EDITC keyword.  
• float can be either:  
\*, which places asterisks in the character positions to the left of the first digit  
cu, which floats the two-character currency symbol used on EDITC.

ERROR (code ['message'])—Locks the keyboard, displays an error code, and optionally displays an error message (when the Help key is pressed) if the specified indicator is turned on.

EXSR (subroutine)—Branches to the named calculation subroutine.

INSERT (fld 1 @...fldn 'literal')—Inserts the named field, expression or literal into the current field.

LOOK (table [index])—Compares the current field for a match in a table, and optionally places the index value of the table entry in index.

PMT (prompt)—Displays the prompt message when the current field is entered.

RANGE (low high)—Specifies the low and high limits for data that can be entered into the current field.

RANGET (table[index])—Compares the current field for a match in a table of low and high limits, and optionally places the index value of the table entry in index.

RESET ([\*TOTn] [name])—Only one parameter is allowed. Sets the named counter to 0.

SEQ (test)—Sequence checks the data in the current field against the data from the previous sequence check using the specified test.

SETOF (ind)—Turns the specified indicator off.

SETON (ind)—Turns the specified indicator on.

SHIFT (shift)—Specifies the shift and character set for each character in a field when C is specified for data type.

SUB (name)—Subtracts the data in the current field from the named field with decimal alignment.

SUBST (table 1 table 2 [index])—Compares the current field for a match in table 1. If there is a match, replaces the current field with data from the corresponding entry in table 2. Optionally places the index value of the table entry in index.

TADD ([\*TOTn] [name])—Only one parameter is allowed. Adds the current field to the named counter.

TSUB ([\*TOTn] [name])—Only one parameter is allowed. Subtracts the current field from the named counter.

XCHK (table index1 index2)—Compares the indexes to see if they match an entry in a named table of index pairs.

Continuation—Specifies to continue on the next line  
+ specifies to continue with the first nonblank character in position 45-80 on the next line (ignore leading blanks).  
- specifies to continue from position 45 on the next line (leading blanks are included).

1-15 16-30 31-45 46-60 61-75 76-90 91-105 106-120 121-135 136-150 151-165



1-10	11-20	21-30	31-40	41-50	51-60	61-70	71-80	81-90	91-100	101-110
1-5	Identifies the source statement order.		31-32	Reserved.						JOB OPT(*NOPMT [*NOOPEN])—At least one of the parameters must be specified. Where:
6	Identifies the type of source statement.		33-34	The characters EQ or blank when a character to test for is specified in position 35-37.						— *NOPMT specifies to bypass the prompts for data set information at the beginning of the job.
7	Names the type of source statement: *—User comment J—Job specification blank—Format specification		35-37	Specifies the character that controls format selection if it matches the character in the data record.						— *NOOPEN specifies to bypass the automatic opening of all files except the transaction file specified by the TFIL keyword.
8-9	The identification associated with this format: 1 through 9—A single numeric character ID. A0 through Z9—A two-character ID consisting of an alphabetic character followed by a numeric character.		38-44	Reserved.						PRTFILE (data set)—Includes the PRINT function in the job. The parameter data set is the data set name to be assigned to the printer.
10-17	The name used to: — identify the job (J in column 7).  — identify the format or subroutine (blank in column 7).  These columns are not used if column 21 contains an R.		45-46	Specifies the identification of the format used for the entry or display of the next record. If columns 22-37 are specified, the format is selected when a match occurs. If columns 22-37 are not specified in enter mode (E in column 21), the format is selected when the repeat count (column 21) is met or the NEXT FMT key is pressed. If columns 22-37 are not specified in review mode (R in column 21), the format is selected if no previous match occurs.						SHARE(names)—Allows other programs to read or write records in the data set specified by the names parameter while this program is executing.
18-19	Reserved.		47-54	Reserved.						SHARER(names)—Allows other programs to read records in the data set specified by the names parameter while this program is executing.
Note:	Columns 20-54 are not used if column 7 contains a J.		55-80	Keywords that specify information used for jobs or formats: <i>JOB specifications (J in column 7):</i> CFILE (data set)—Includes the COPY function in the job. The parameter data set is the data set name from which records will be copied. DATE(*DMY/*YMD)—The format of the date available in UDATE. The default is *MDY, where M = month, D = day, and Y = year. EDITC(cuptd)—Five characters that define the editing control for output fields, where: — cu is a two-character currency symbol (default = b\$). — p is the decimal point character (default = .). — t is the thousand separator character (default = ,). — d is the date separator character (default = /). The system default for this option is b\$.,/ if EDITC is not specified.						STATUS(name)—Establishes a variable that can be used to check the status of an I/O device after an I/O operation. The parameter name is the name assigned to the variable.
20	Specifies the number of times the format is repeated before the next format is used: 1 through 9—Repeat the format for the specified number of times unless the SEL FMT or NEXT FMT key is pressed. blank or N—Repeat the format until the SEL FMT or NEXT FMT key is pressed.									TFILE(data set [delfreq])—Specifies the data set where records will be written after a format is completed, where: — data set is the name of the data set that receives the transaction records. — delfreq specifies how often deleted records are automatically inserted in the transaction data set.
21	Specifies how the format is used: E—(Entry) used to enter and display data. R—(Review) used to select a format for scan, update, or verify of existing records.									<i>Format Specifications (blank in column 7):</i> CLRL (number)—Specifies the number of display lines cleared, starting from the first line of the display, when a new record is to be entered. If *NO is specified, none of the display lines are cleared.
22-37	Used for logical selection of a format. Multiple tests are allowed. In enter mode, the format selected is used to format the <i>next</i> record entered. In review mode, the format selected is used to display the <i>current</i> record.									EOJ ([('job' dev] [*PASS]))—Causes the end of the job upon completion of the format. The optional parameters are: job—name of the next job to execute. dev—the device address where the next job is located. *PASS—suppress job production statistics.
22	In review mode (column 21 contains an R), an A specifies the <i>anding</i> of two characters in the data record to create a unique record identifier.									SLNO (line)—Specifies the uppermost display line that can be used. All display line references are based on the specified line as line one.
23-30	*POSnnnn identifies the position in the data record to be tested, where nnnn is a numeric value from 1 to 1024.									WRITE (name)—Specifies that the current data is written to the data set in the record format specified by <i>name</i> . If *NO is specified, the current data is not written to the data set.
										Continuation can be specified by a + or - as the last character on the line, where: + specifies to continue with the first nonblank character in positions 55-80 on the next line (ignore leading blanks). - specifies to continue from position 55 on the next line (including leading blanks).

1-15

16-30

31-45

46-60

61-75

76-90

91-105

106-120

121-135

136-150

151-165





**This page is intentionally left blank**

\*POS  
 entry formats 53, 55, 57, 58, 61, 62, 64, 96, 123  
 review formats 64, 96  
 \*TOT 166

access methods  
 as affected by data set organization 117  
 as performed against multivolume data sets 120  
 direct access 129  
 of indexed data sets 130  
 of keyed and nonindexed data sets 131  
 of nonkeyed and nonindexed data sets 129  
 sequential access 123-127  
 of nonkeyed and nonindexed data sets 124  
 of keyed and nonindexed data sets 126  
 of indexed data sets 127

ADD 167, 170-172  
 ADDRROUT files 131, 134  
 alternating entry formats 53  
 ending tests on formats 53, 55, 58  
 apostrophes in literals 88, 165  
 application programs  
 controlling the I/O devices 37  
 data set organization 32  
 data set update 35  
 keys that are active 31  
 arithmetic operations 167  
 (see also *calculations*)  
 arrayname,index combination 144  
 AUXDUP 9, 99

BEGSR 67  
 BL 88  
 blink attribute 88

calculations  
 on the A-specification 168, 169  
 on the C-specification 93, 111-114  
 with counters 171  
 with named fields and INSERT 169

calling subroutines  
 through EXSR on A-specification 66, 71  
 through Z-specification 65, 67

CFILE 22, 10  
 CHAIN 40, 47  
 (see also *direct access methods*)  
 chaining jobs 173  
 changing the entry and review formats 78  
 CHECK(BY) 96, 57, 61, 97, 103  
 CHECK(DR) 43, 51, 103  
 CHECK(FE RZ) 80, 103  
 CHECK(FE) 22  
 clearing lines on the display 86, 88  
 CLOSE 42  
 CLRL 10, 86, 88  
 column separators  
 with a formatted display 76, 88  
 with an unformatted display 74, 82  
 combination programs 43  
 communications 42  
 COMP  
 on A-specification 102, 103  
 on C-specification 65, 69  
 compile-time tables 139  
 complex format selection 61, 78, 88  
 complex reformatting for the diskette 92  
 constant name 174  
 copying 22  
 creating multiple records from a single record input 67  
 current field attributes 84  
  
 data set open prompt 17, 122  
 data sets  
 access methods for 117  
 created using application programs 9  
 created using data-entry programs 9  
 multivolume 120  
 reformatting 9, 92  
 data-entry programs  
 characteristics of 15  
 differences with application programs 9  
 keys that are active 21  
 default diskette records for transaction files 9  
 default display format 49  
 (see also *format 0*)  
 DEVICE 2, 42  
 differences between data-entry and application programs 9  
 direct access methods 129-134  
 displaying multiple records at the same time 86, 88  
 DSPATR(BL HI) 88  
 DSPATR(CS) (see *column separators*)  
 DSPATR(ND) 83  
 DSPATR(RI) 83  
 DSPATR(UL) 103  
 DSPSIZ 2  
 dynamically created fields 92, 111, 114

EDTCDE 10, 86  
ENDSR 2, 3  
enter mode 15, 16  
ENTRATR 84, 85  
EOJ 10, 173  
ERROR 102, 103  
execute mode 31  
EXFMT 31, 37, 47  
EXITATR 85  
EXSR 70

fill-in-the-blank displays 83  
FORM 152  
format 0 58, 75  
formatted displays 76  
formatting 73, 91  
  definition of 47  
  diskette formats 67, 91  
  display formats 65, 73  
  entry formats 50  
  format selection by testing 55  
  review formats 54

GOTO 3, 110, 112, 114

highlight attribute 88

INDEX 33, 120, 127, 130  
indexed data sets 32, 127, 130  
indicators  
  for conditioning errors 101, 102  
  for conditioning fields 103  
  on A-specification 99, 103  
  operations conditioned by 106  
  operations that condition 106  
  resetting 104  
  using to branch 110, 113  
  using to condition arithmetic operations 110, 111, 112  
  using to condition end of job 110, 113  
  using to condition EXFMT 10, 113  
  using with I/O operations 108  
INSERT  
  with arithmetic expressions 99, 168, 169, 170  
  with named fields for complex reformatting 43, 94  
  with record identifiers 53

key fields 32, 126, 130  
key-initiated modes 20  
keyed data sets 126, 130  
keys 21, 31

literals 78  
loading an object program 16  
logical device ID 122  
LOKUP 135, 143  
LOOK 135, 137, 139

menus 53, 80  
merging multiple fields from multiple records 94, 112  
mode select prompt 17  
modes  
  automatically selecting 18, 50  
  designations 20  
MOVE 144  
MOVEA 145  
MULT 93  
multiple entry formats 71, 88, 94  
multiple tables 142  
multivolume data sets 120-121

ND (nondisplay attribute) 83  
nondisplay attribute 83  
NUMENT 139, 140

online totals 112, 170-171  
OPEN 42

padding records 57  
partitions 7  
physical device ID 122  
PMT 2  
positioning fields in a display 75, 79  
positioning fields in a data set 95  
positioning fields on a printout 157, 162  
printing a form 162  
printing on a form 154-161

## READER'S COMMENT FORM

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

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

*Page Number    Error*

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

*Page Number    Comment*

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

Name \_\_\_\_\_

Address \_\_\_\_\_

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

Cut Along Line

Fold and tape

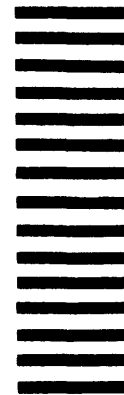
Please do not staple

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:

IBM CORPORATION  
General Systems Division  
Information Design and Development  
Publications, Dept 997  
11400 Burnet Rd  
Austin, Texas 78758

Fold and tape

Please do not staple

Fold and tape



International Business Machines Corporation

General Systems Division  
4111 Northside Parkway N.W.  
P.O. Box 2150  
Atlanta, Georgia 30055  
(U.S.A. only)

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

IBM 5280 DE/FPG User's Guide (File No. S5280-28) Printed in U.S.A. SC21-7804-1

1

2



**International Business Machines Corporation**

**General Systems Division  
4111 Northside Parkway N.W.  
P.O. Box 2150  
Atlanta, Georgia 30055  
(U.S.A. only)**

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

IBM 5280 DE/RPG User's Guide (File No. S5280-28) Printed in U.S.A. SC21-7804-1