IBM System/3
RPG II Disk File Processing
Programmer's Guide

GC21-7566-1
File No. S3-28

IBM System/3
RPG II Disk File Processing
Programmer's Guide

Second Edition (July 1974)

This is a major revision of, and obsoletes, GC21-7566-0 and Technical Newsletter GN21-7657. This manual has been extensively revised and should be reviewed in its entirety.

Information concerning IBM System/3 Model 15 has been added to the manual. The entire section entitled *Chapter 5. Using Console Devices* has been removed from this manual. The KEY (Model 6 only) and DSPLY operation codes, formerly described in Chapter 5, are described in your RPG II reference manual. For information concerning the RPG II Inquiry facility (Rollout/Rollin), also formerly described in Chapter 5, see *Related Publications*, listed in the *Preface*.

Changes are periodically made to the information herein; before using this publication in connection with the operation of IBM systems, refer to the *IBM System/3 Bibliography*, GC20-8080, for the editions that are applicable and current.

Use this publication only for the purposes stated in the *Preface*.

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 IBM branch office serving your locality.

This publication could contain technical inaccuracies or typographical errors. Use the Reader's Comment Form at the back of this publication to make comments about this publication. If the form has been removed, address your comments to IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901. Comments become the property of IBM.

Preface

This manual explains the RPG II specifications necessary to process disk files using the IBM System/3 Model 6, Model 10 Disk System, and Model 15. The following types of files are discussed:

- Sequential
- Indexed
- Direct
- Record Address

Included are examples of creating and maintaining each of these file types. For information on how this publication should be read, see *How to Use This Publication,* following the table of contents.

This manual is intended for programmers who have a basic knowledge of the RPG II language, including the ability to describe sequential and indexed files as input files. The reader must be familiar with the information presented in the *IBM System/3 Disk Concepts and Planning Guide,* GC21-7571.

Note: For detailed information concerning multivolume disk files – concepts; Operation Control Language considerations; RPG II processing – see the publications listed under *Related Publications* which are appropriate for your System/3 model.

Related Publications

These publications are recommended for additional reference:

General System/3

- *IBM System/3 Disk Concepts and Planning Guide,* GC21-7571

RPG II References

- *IBM System/3 Model 6 RPG II Reference Manual,* SC21-7517
- *IBM System/3 RPG II Reference Manual,* SC21-7504

OCL References

- *IBM System/3 Models 8 and 10 System Control Programming Reference Manual,* GC21-7512
- *IBM System/3 Model 15 System Control Programming Reference Manual,* GC21-5077
- *IBM System/3 Models 4 and 6 Operation Control Language and Disk Utility Programs Reference Manual,* GC21-7516
- *IBM System/3 Model 15 System Control Programming Concepts and Reference Manual,* GC21-5162

Contents

CHAPTER 1. SEQUENTIAL FILES 1
Creating a Sequential File 1
Input Records 3
Output Record 3
Maintaining a Sequential File 6
Adding Records 7
Tagging Records for Deletion 16
Updating Records 16
Reorganizing a File 22

CHAPTER 2. INDEXED FILES 26
Creating an Indexed File 26
Creating an Ordered Indexed File 27
Creating an Unordered Indexed File 28
Input Record 29
Output Record 29
Maintaining an Indexed File 31
Updating Records 32
Adding Records 37
Tagging Records for Deletion 49
Reorganizing a File 49
Other Ways to Process Indexed Files 50
Processing an Indexed File Consecutively . . . 50
Processing an Indexed File Randomly by Relative Record Number 50

CHAPTER 3. DIRECT FILES 51
Creating a Direct File 51
Creating a Direct File without Synonyms . . . 53
Input Record 54
Output Record 54
Creating a Direct File with Synonyms 58
Retrieval of Records in a Direct File 60
Consecutive Retrieval 61
Random Retrieval 66
Maintaining a Direct File 70
Adding Records 70
Tagging Records for Deletion 70
Updating Records 71

CHAPTER 4. RECORD ADDRESS FILES 79
Files Containing Relative Record Numbers (ADDROUT Files) 79
Creating an ADDROUT File 79
Processing by an ADDROUT File 80
Files Containing Record Key Limits 83
Creating a File with Record Key Limits 83
Processing Sequentially within Limits 84

INDEX 89

How To Use This Publication

This publication is divided into four chapters:

1. Sequential Files
2. Indexed Files
3. Direct Files
4. Record Address Files

Each of the chapters discusses the RPG II specifications needed to create and maintain a certain type of file. Since the discussions in each chapter apply to a particular file organization, you need read only the chapter for the file organization you will use.

This manual is designed to show you the entries that *must* be made in order for the system to identify the file and determine what functions are to be done. Other entries will be needed, but they depend on the job you are doing. Refer to the reference manual for your system if you need information on these additional entries.

Note Concerning Examples

Most of the examples in this manual use MFCU input files and use 96-column card images to represent input records. This does not imply that either the MFCU or 96-column cards must be used for input; any of several input devices can be used, depending on which System/3 model and configuration you have.

If you are a Model 6 user, you would probably not use a card file for input, although you may use an online data recorder for card input. Instead, you would create sequential disk transaction files in one of the following ways:

- Using the Keyboard Data Entry conversational utility program (see *IBM System/3 Model 6 Conversational Utility Programs Reference Manual,* SC21-7528)
- Using an interactive RPG II, FORTRAN, or BASIC program to build the disk file from keyboard input

The disk transaction file would then be used to update a disk master file in the same way MFCU files are used in the examples in this manual. Thus, the MFCU files in the examples could be replaced by sequential disk files. For example:

File Description Specification

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/L/K	I/D/T or 2	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Name of Label Exit	Core Index	Option	Entry	R/U/N
02	F																				
03	F																				
04	F	TRANS	I	P		A	F	96	96						MFCU1						
05	F																				
06	F	TRANS	I	P		A	F	96	96						DISK						
07	F																				
08	F																				
09	F																				
10	F																				
	F																				
	F																				

MFCU File

Sequential Disk File

Depending on the System/3 model and the I/O devices available, MFCU input files shown in most of the examples can be replaced by 80-column card files, console files, tape files, 5445 disk files, or other types of input files. Models 10, 12, and 15 users can substitute DISK45 (5445 disk) or DISK40 (3340 disk) for the device entry DISK (5444 disk) in the examples, depending on the equipment they have. Model 6 users should replace the device entry PRINTER in the examples with the device entry TRACTR1, and replace Block Length and Record Length entries with the beginning and ending print positions.

DIRECT ACCESS STORAGE FOR MODELS 12 AND 15

The IBM 3340 Direct Access Storage Facility attaches to System/3 Model 12, to System/3 Model 15 (5704-RG1), and to System/3 Model 15 (5704-RG2).

The IBM 3344 Direct Access Storage Facility also attaches to System/3 Model 15 (5704-RG2).

Certain areas on the 3340 and 3344 disks are treated as 5444 disks. These areas, known as 5444 simulation areas, are used for the program libraries and can be also used for certain data files. The remainder of the disk space, known as main data area, can only be used for data files.

References in this manual to DISK, DISK40, and DISK45 are to be interpreted according to which disk storage device(s) is attached to the system. The following table should be used to determine the meaning of the reference:

Device Type	Model 15 (5704-RG1)	Model 12 or 15 (5704-RG1)	Model 15 (5704-RG2)
DISK	5444 Disk Storage Drive	5444 simulation area on 3340	5444 simulation area on 3340 or 3344
DISK40	Not applicable	Main data area on 3340	Main data area on 3340 or 3344
DISK45	5445 Disk Storage	Main data area on 3340	Main data area on 3340 or 3344

IBM System/3 5448 Disk Storage Drive

The IBM System/3 5448 Disk Storage Drive on System/3 Models 8 and 10 uses the same program product support as the IBM 5445 Disk Storage. However, a separate system control program feature is required for the 5448. In general, references to 5445 in this manual also apply to 5448. For specific information about 5448 operating characteristics and programming support, see *IBM System/3 5448 Disk Storage Drive Program Reference Manual,* GC21-5168.

Chapter 1. Sequential Files

A sequential disk file is a group of records arranged in a particular sequence and processed consecutively, one after another in the order they occur.

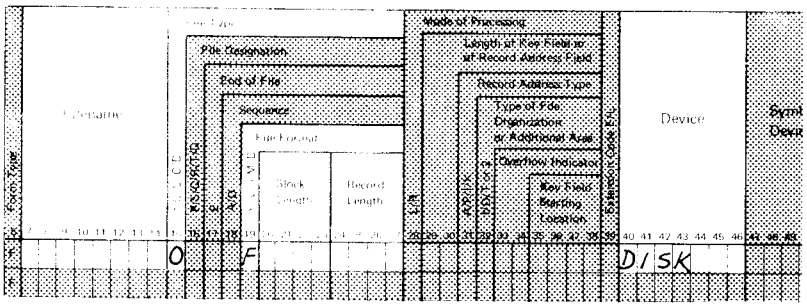
This chapter explains how to use the RPG II language to create and maintain a sequential file. Sample jobs are used to illustrate these functions.

To understand the sample jobs, a basic knowledge of RPG II is necessary. If you do not fully recall some of the coding used in the sample jobs, you should refer to the *IBM System/3 RPG II Reference Manual,* SC21-7504 (for Model 10 Disk System or Model 15), or the *IBM System/3 Model 6 RPG II Reference Manual,* SC21-7517, depending on the system you have.

CREATING A SEQUENTIAL FILE

To create a sequential file, you make certain entries on the File Description sheet. The following entries are required to describe various characteristics of the disk file:

File Description Specifications



The disk filename must be entered in columns 7 through 14. Column 15 must contain an O to indicate the file is an output file.

All records in a file must be the same length. Thus, column 19 must contain an F to specify that the record length is fixed.

A number that is equal to or a multiple of the disk record length must be entered in columns 20 through 23. This entry determines the size of the input/output area allocated by RPG II. For an explanation on block length calculation, see the *IBM System/3 Disk Concepts and Planning Guide,* GC21-7571. If you want block length calculated for you by RPG II, assign a block length equal to the record length. By blocking disk records you can increase the input/output efficiency of your program by reducing the number of accesses. You must be sure, however, that enough main storage is available for your input/output area.

Columns 24 through 27 must contain the length of the disk record. Whenever a disk file is being described, DISK (Model 6, 10, 15), DISK40 (Model 12 or 15), or DISK45 (Model 10, 12, or 15) is required in the Device columns.

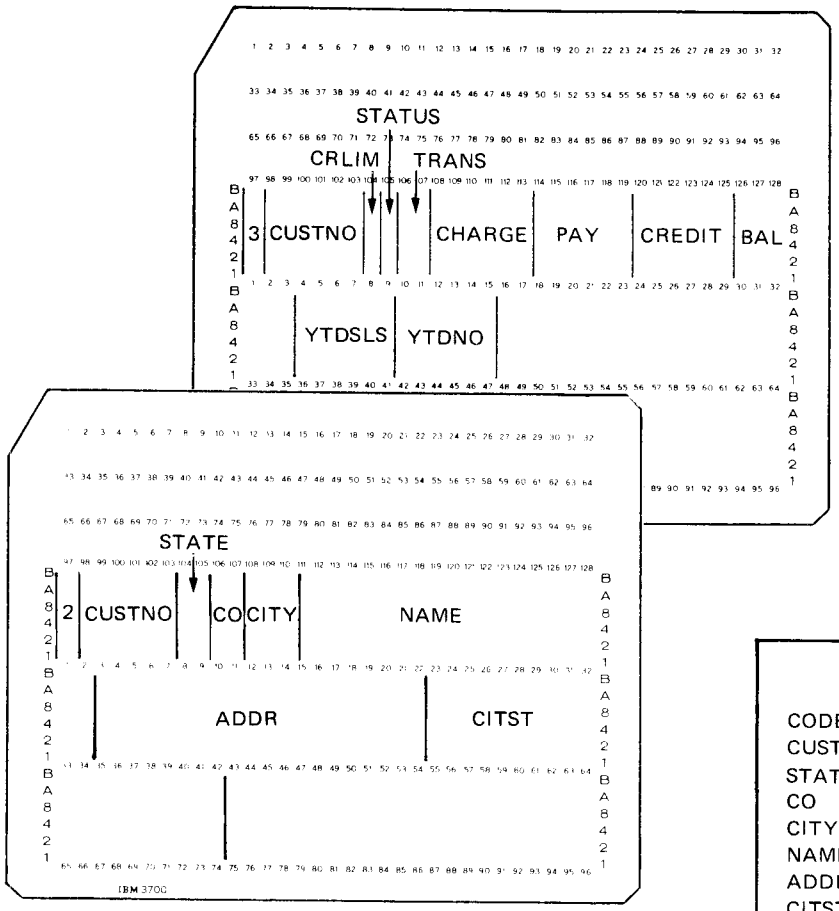
Example of Creating a Sequential File

Suppose you want to create a customer file on disk. Customer numbers are to be assigned on a sequential basis; new customers are assigned the next higher number. The file will be used to produce monthly reports of each customer's status. Thus, a sequential file will serve your needs.

To create the sequential file, you must first determine the input record format and the output record format (Figure 1). The file is created by writing the customer data from the input records onto disk. Notice that space is provided on the output record so additional information can be added to the record later if necessary. Basic information about each customer in the file is also printed in the report shown in Figure 2.

Figure 3 shows the RPG II coding necessary to create the sequential customer file and print the report.

Input Records



Note: The input file need not be a card file. See *How to Use This Manual, Note Concerning Examples* at the beginning of this manual.

Key

CODE	= Output code (CM)
CUSTNO	= Customer number
STATE	= State
CO	= County
CITY	= City
NAME	= Customer name
ADDR	= Customer street address
CITST	= City and state
CRLIM	= Credit limit
STATUS	= Status
TRANS	= Number of transactions this month
CHARGE	= Current month charges
PAY	= Current month payments
CREDIT	= Current month credits
BAL	= Balance
YTDSLS	= Year-to-date sales amount
YTDNO	= Year-to-date number of sales
DELETE	= Output record code

Output Record

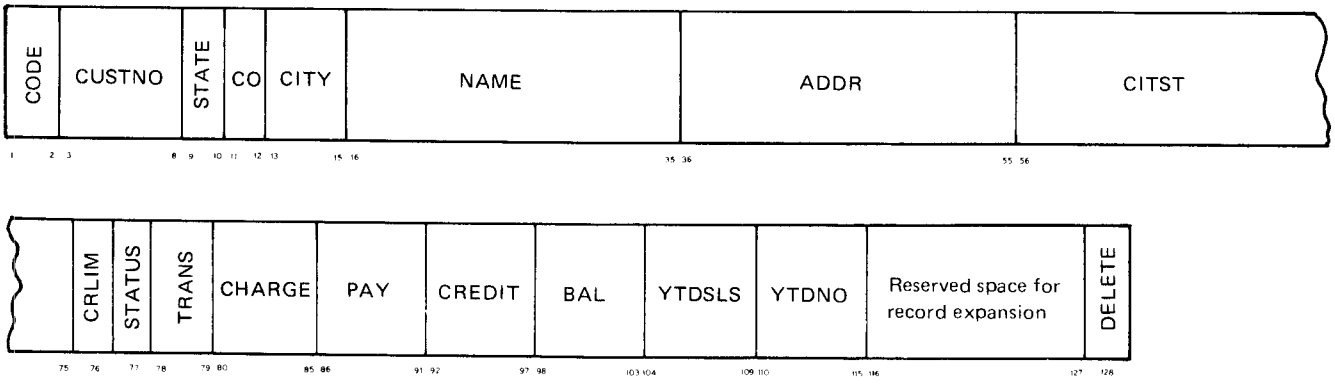
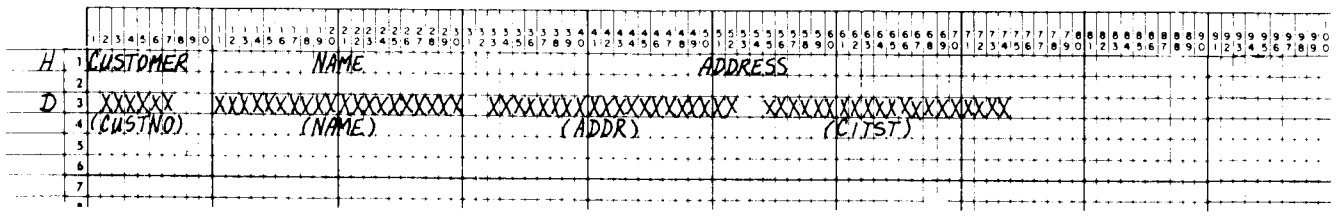


Figure 1. Input Record and Output Record Formats



CUSTOMER	NAME	ADDRESS	
136728	JONES VARIETY	14 S MAIN	BEDROCK, TEX
301628	JIM'S 5 AND 10	1103 FRANKLIN ST	GLENCOE, MINN
795246	SCHMIDT HARDWARE	600 1ST ST NW	HILL CITY, MD

Figure 2. Report of Customer Records

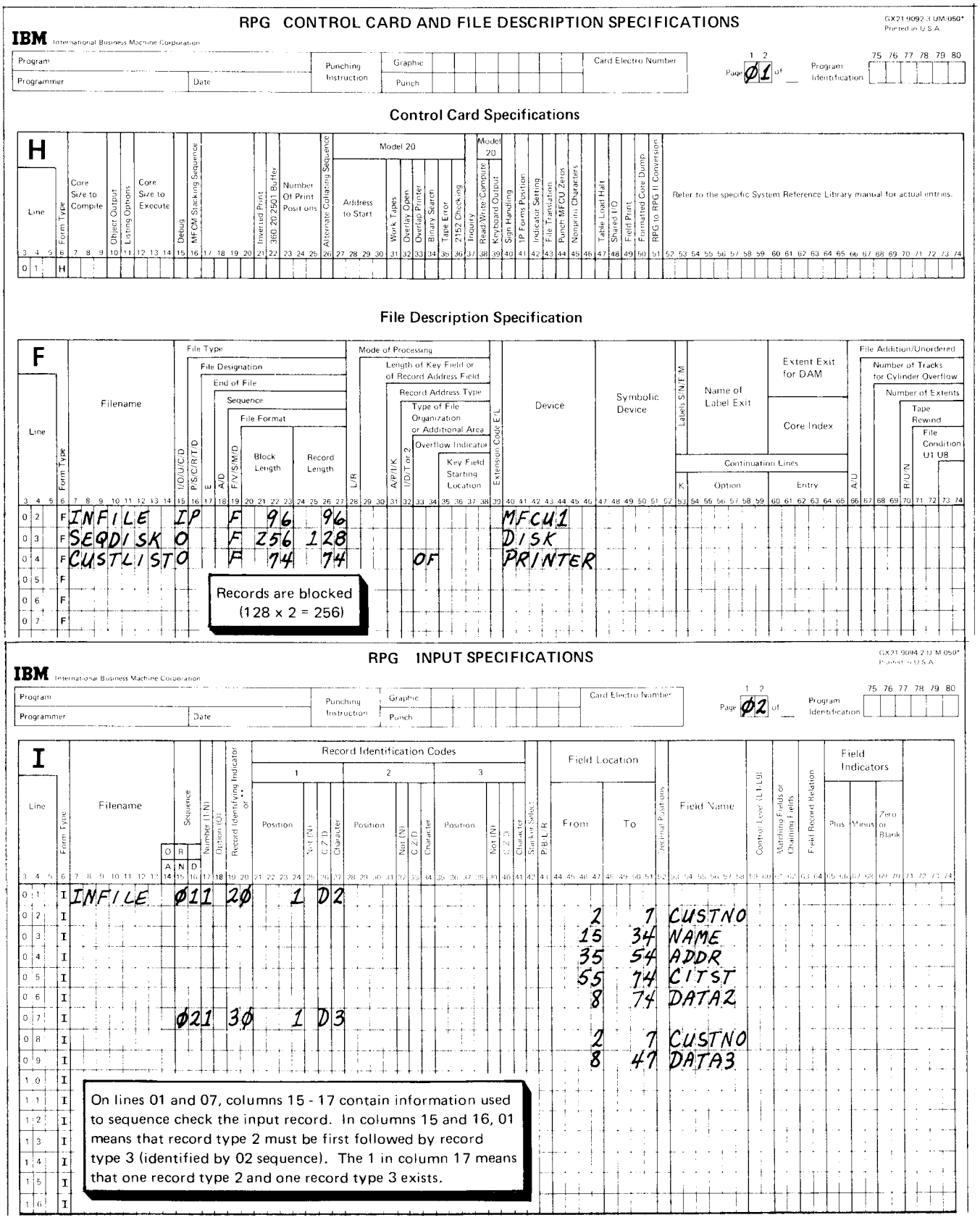


Figure 3 (Part 1 of 2). Creating a Sequential Customer File

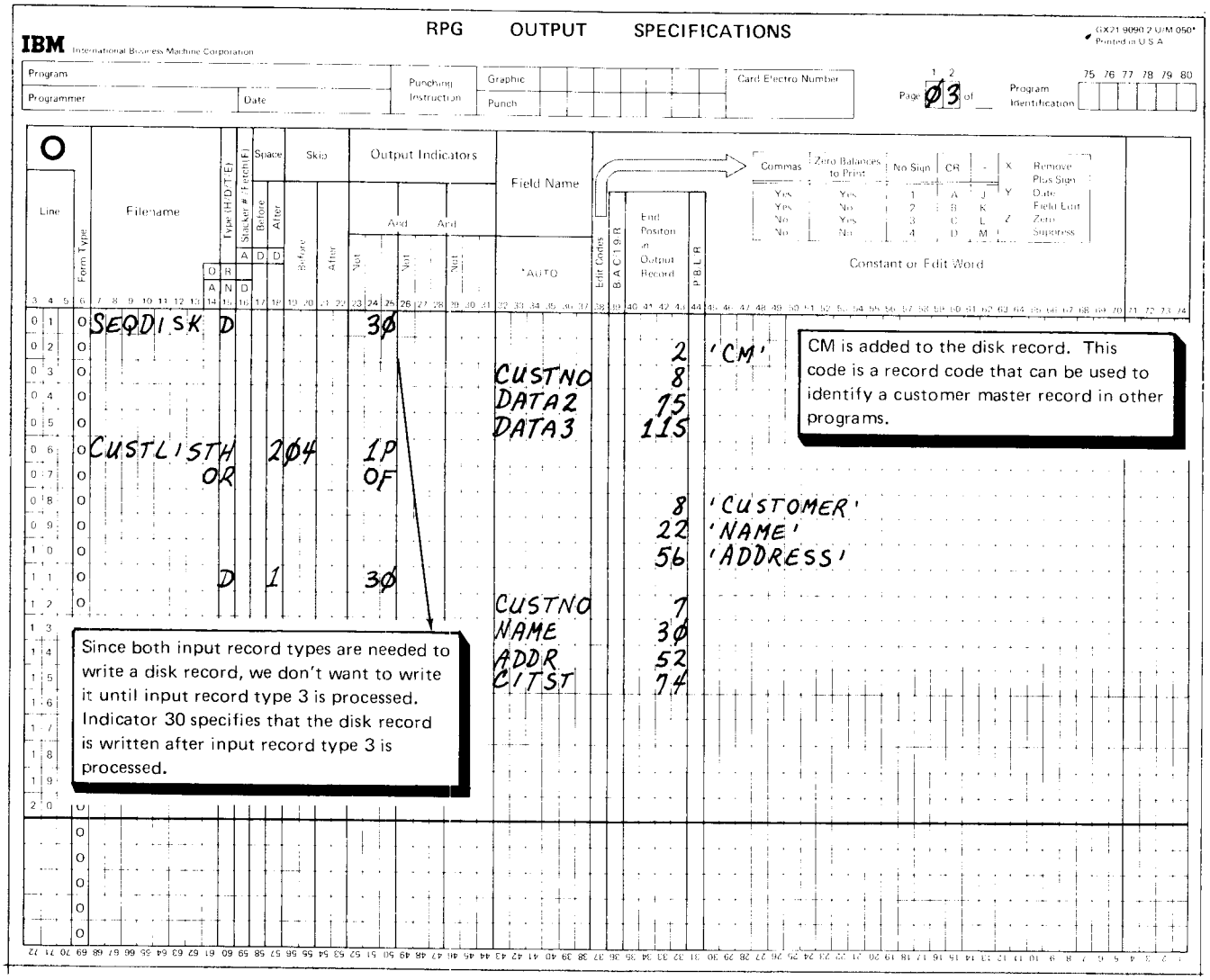


Figure 3 (Part 2 of 2). Creating a Sequential Customer File

MAINTAINING A SEQUENTIAL FILE

Once a file has been created, it usually needs to be maintained. File maintenance means performing those functions that keep a file current for daily processing needs. Four common file maintenance functions apply to sequential files:

1. Adding records.
2. Tagging records for deletion.
3. Updating records.
4. Reorganizing a file.

Adding Records

After a file is created, you can add records to it. Records can be added to a sequential file in either of two ways:

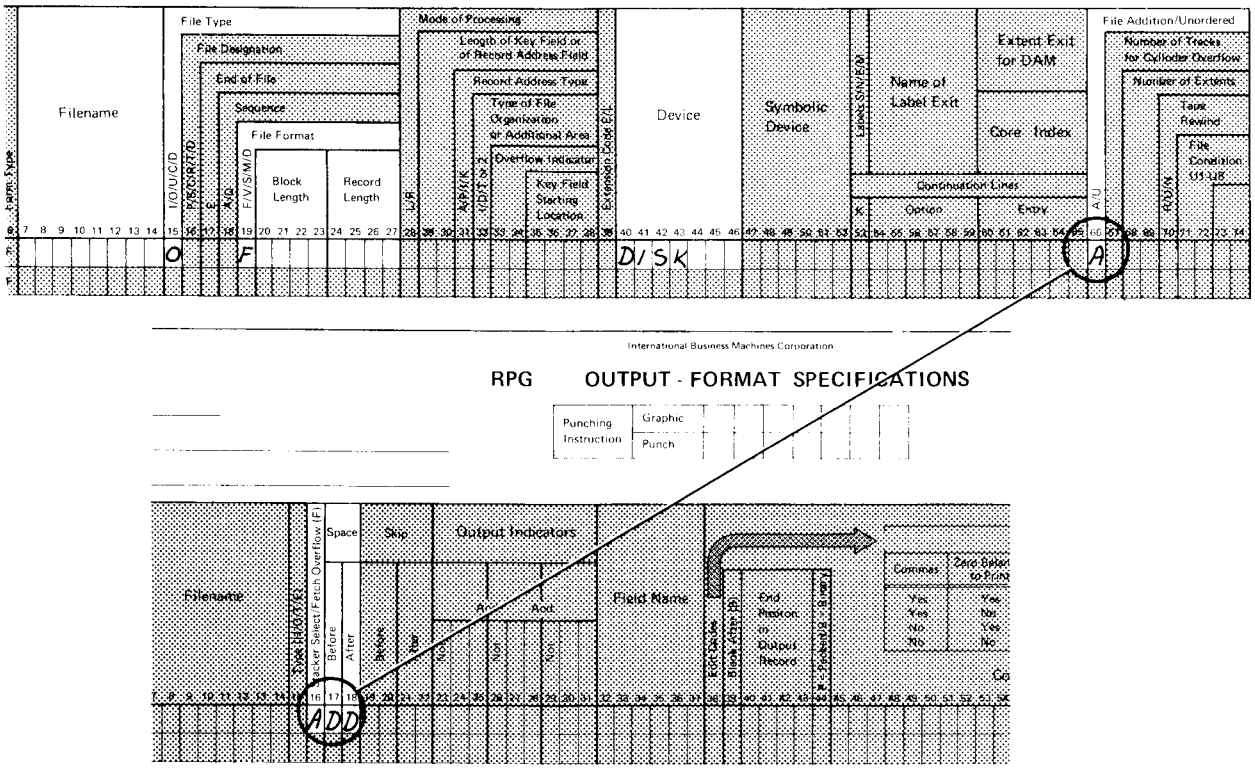
1. At the end of records in the file.

2. Between records in the file (creating a new file).

Adding Records at the End of Records in the File

To add records at the end of the file, entries are required on the File Description and Output-Format sheets:

File Description Specifications



Some of the entries on the File Description sheet are the same ones needed to create a sequential file (see *Creating a Sequential File* in this chapter).

Two new disk entries are also needed, one on the File Description sheet and one on the Output-Format sheet. These entries are circled.

An A in column 66 on the File Description sheet tells the system that records will be added to the file described on this line.

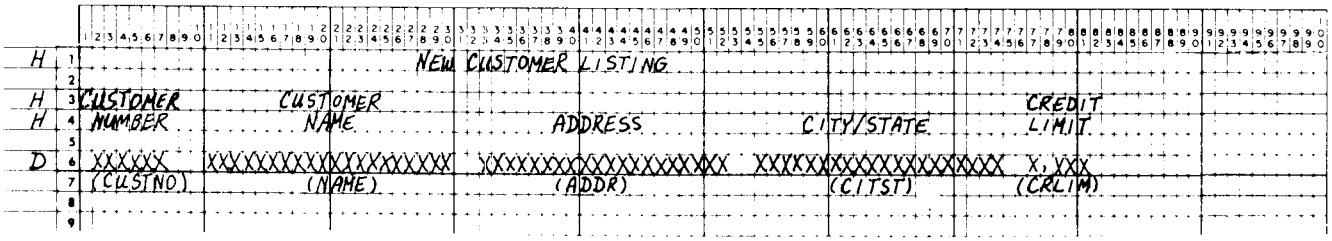
The entry on the Output-Format sheet is an ADD in columns 16 through 18. This entry tells the system that the fields defined on the following lines constitute the record to be added to the file specified in columns 7 through 14.

Example of Adding Records to the End of the File

As you get new customers, you will want to add them to the sequential customer file you created in the *Example of Creating a Sequential File.* Since customer numbers are assigned sequentially, each new customer record can be added following the records now in the file.

The input records and output records will be in the same format as the records used to create the file. A report can be printed like the one shown in Figure 4 to ensure that the new records have been added to the file.

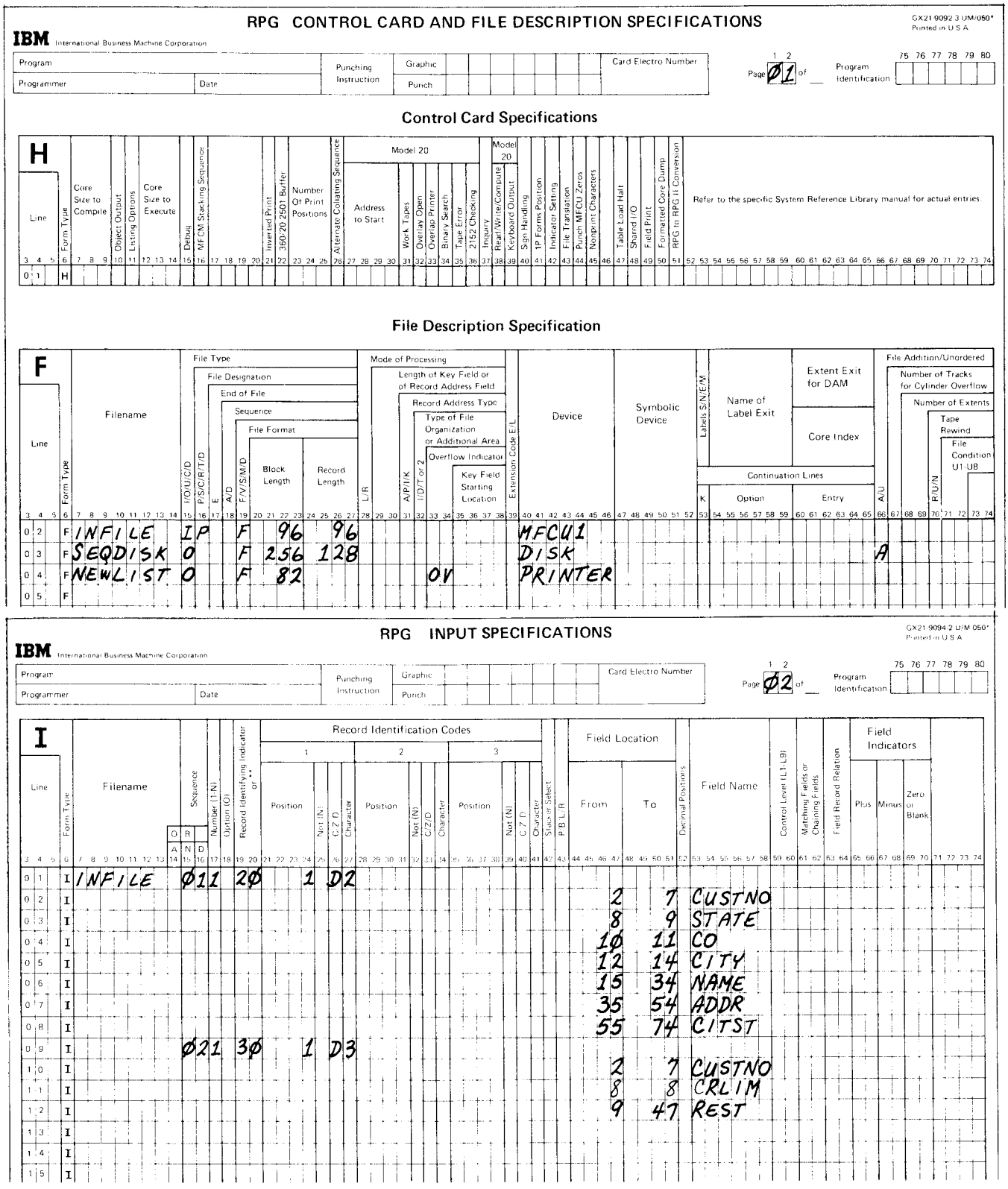
Figure 5 shows the RPG II coding necessary to add records to the customer file and print the report.



NEW CUSTOMER LISTING

CUSTOMER NUMBER	CUSTOMER NAME	ADDRESS	CITY/STATE	CREDIT LIMIT
136728	JONES VARIETY	14 S MAIN	BEDROCK, TEX	1,000
301628	JIM'S 5 AND 10	1103 FRANKLIN ST	GLENCOE, MINN	250
795246	SCHMIDT HARDWARE	600 1ST ST NW	HILL CITY, MD	2,000

Figure 4. Report of New Customers Added to the Customer File



RPG CONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS

GX21-9092-3 U/M 050* Printed in U.S.A.

IBM International Business Machine Corporation

Program | Punching Instruction | Graphic | Card Electro Number

Programmer | Date | Punch

Page 01 of __ | Program Identification

Control Card Specifications

H

Line | Form Type | Core Size to Compile | Object Output | Listing Options | Core Size to Execute | Debug | MFCM Stacking Sequence | Inverted Print | 360/20 2501 Buffer | Number Of Print Positions | Alternate Collating Sequence | Model 20: Address to Start | Work Tapes | Overlay Open | Overlap Printer | Binary Search | Tape Error | 2152 Checking | Inquiry | Model 20: Real-Time Compute | Keyboard Output | Sign Handling | 1P Forms Position | Indicator Setting | File Translation | Punch MFCU Zeros | Nonprint Characters | Table Load Halt | Shared I/O | Field Print | Formatted Core Dump | RPG to RPG II Conversion | Refer to the specific System Reference Library manual for actual entries.

File Description Specification

F

Line | Form Type | Filename | File Type | File Designation | End of File | Sequence | File Format | Block Length | Record Length | Mode of Processing | Length of Key Field or of Record Address Field | Record Address Type | Type of File Organization or Additional Area | Overflow Indicator | Key Field Starting Location | Extension Code E/L | Device | Symbolic Device | Labels S/N/E/M | Name of Label Exit | Extent Exit for DAM | Core Index | Continuation Lines: K, Option, Entry | File Addition/Unordered A/U | Number of Tracks for Cylinder Overflow | Number of Extents | Tape Rewind R/U/N | File Condition U1-U8

Line	Form Type	Filename	File Type	File Designation	End of File	Sequence	File Format	Block Length	Record Length	Mode of Processing	Length of Key Field or of Record Address Field	Record Address Type	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Core Index	Continuation Lines: K, Option, Entry	File Addition/Unordered A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind R/U/N	File Condition U1-U8
02	F	INFILE	I	P			F	96	96								MFCU1											
03	F	SEQDISK	O				F	256	128								DISK							A				
04	F	NEWLIST	O				F	82						OV			PRINTER											
05	F																											

RPG INPUT SPECIFICATIONS

GX21-9094-2 U/M 050* Printed in U.S.A.

IBM International Business Machine Corporation

Program | Punching Instruction | Graphic | Card Electro Number

Programmer | Date | Punch

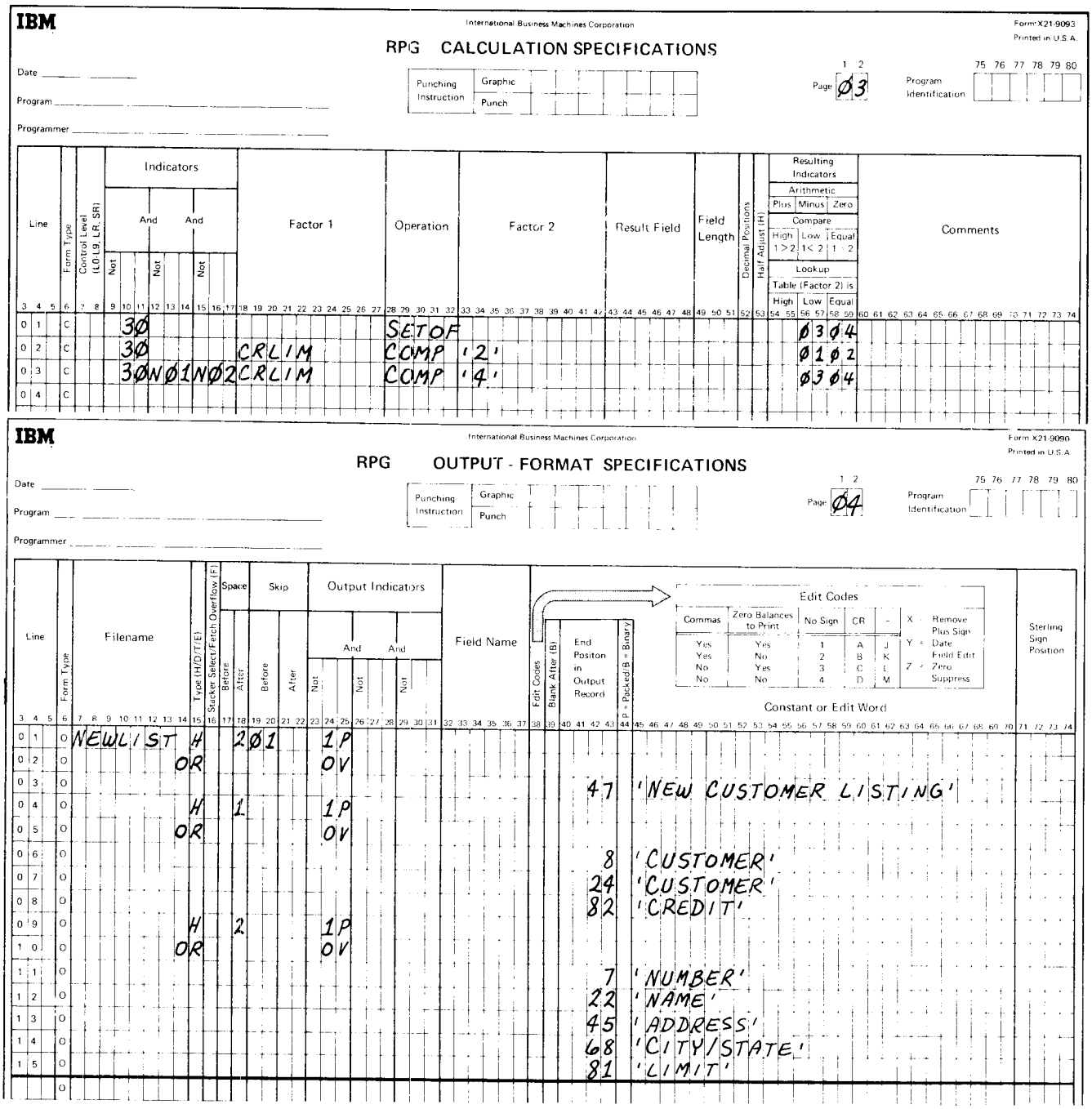
Page 02 of __ | Program Identification

I

Line | Form Type | Filename | Sequence | Number (1-N) | Option (O) | Record Identifying Indicator or: | Record Identification Codes 1: Position, Not (N), C/Z/D, Character | 2: Position, Not (N), C/Z/D, Character | 3: Position, Not (N), C/Z/D, Character | Stacker Select | P/B/L/R | Field Location: From, To | Decimal Positions | Field Name | Control Level (L1-L9) | Matching Fields or Chaining Fields | Field Record Relation | Field Indicators: Plus, Minus, Zero or Blank

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or:	Record Identification Codes 1: Position, Not (N), C/Z/D, Character	2: Position, Not (N), C/Z/D, Character	3: Position, Not (N), C/Z/D, Character	Stacker Select	P/B/L/R	Field Location: From, To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus, Minus, Zero or Blank
01	I	INFILE		01		20	1 D2											
02	I											2 7		CUSTNO				
03	I											8 9		STATE				
04	I											10 11		CO				
05	I											12 14		CITY				
06	I											15 34		NAME				
07	I											35 54		ADDR				
08	I											55 74		CITST				
09	I		021			30	1 D3											
10	I											2 7		CUSTNO				
11	I											8 8		CRLIM				
12	I											9 47		REST				
13	I																	
14	I																	
15	I																	

Figure 5 (Part 1 of 3). Adding Records to the End of the Sequential Customer File



IBM — International Business Machines Corporation — Form X21-9093 Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____ Program _____ Programmer _____

Punching Instruction: Graphic / Punch

Page 03 Program Identification

Line	Form Type	Control Level (L0,L9, LR, SR)	Indicators: And / And — Not	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus / Minus / Zero); Compare (High 1>2 / Low 1<2 / Equal 1=2); Lookup Table (Factor 2) is (High / Low / Equal)	Comments
01	C		30		SETOF						034	
02	C		30	CRLIM	COMP '2'						012	
03	C		30N01N02	CRLIM	COMP '4'						034	
04	C											

IBM — International Business Machines Corporation — Form X21-9090 Printed in U.S.A.

RPG OUTPUT - FORMAT SPECIFICATIONS

Date _____ Program _____ Programmer _____

Punching Instruction: Graphic / Punch

Page 04 Program Identification

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select/Fetch Overflow (F)	Space: Before	Space: After	Skip: Before	Skip: After	Output Indicators: And / And — Not	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	P = Packed/B = Binary	Constant or Edit Word	Sterling Sign Position
01	O	NEWLIST	H		2	01			1P							
02	O		OR						OV							
03	O														47 'NEW CUSTOMER LISTING'	
04	O		H		1				1P							
05	O		OR						OV							
06	O														8 'CUSTOMER'	
07	O														24 'CUSTOMER'	
08	O														82 'CREDIT'	
09	O		H		2				1P							
10	O		OR						OV							
11	O														7 'NUMBER'	
12	O														22 'NAME'	
13	O														45 'ADDRESS'	
14	O														68 'CITY/STATE'	
15	O														81 'LIMIT'	
	O															

Edit Codes

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

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

Figure 5 (Part 2 of 3). Adding Records to the End of the Sequential Customer File

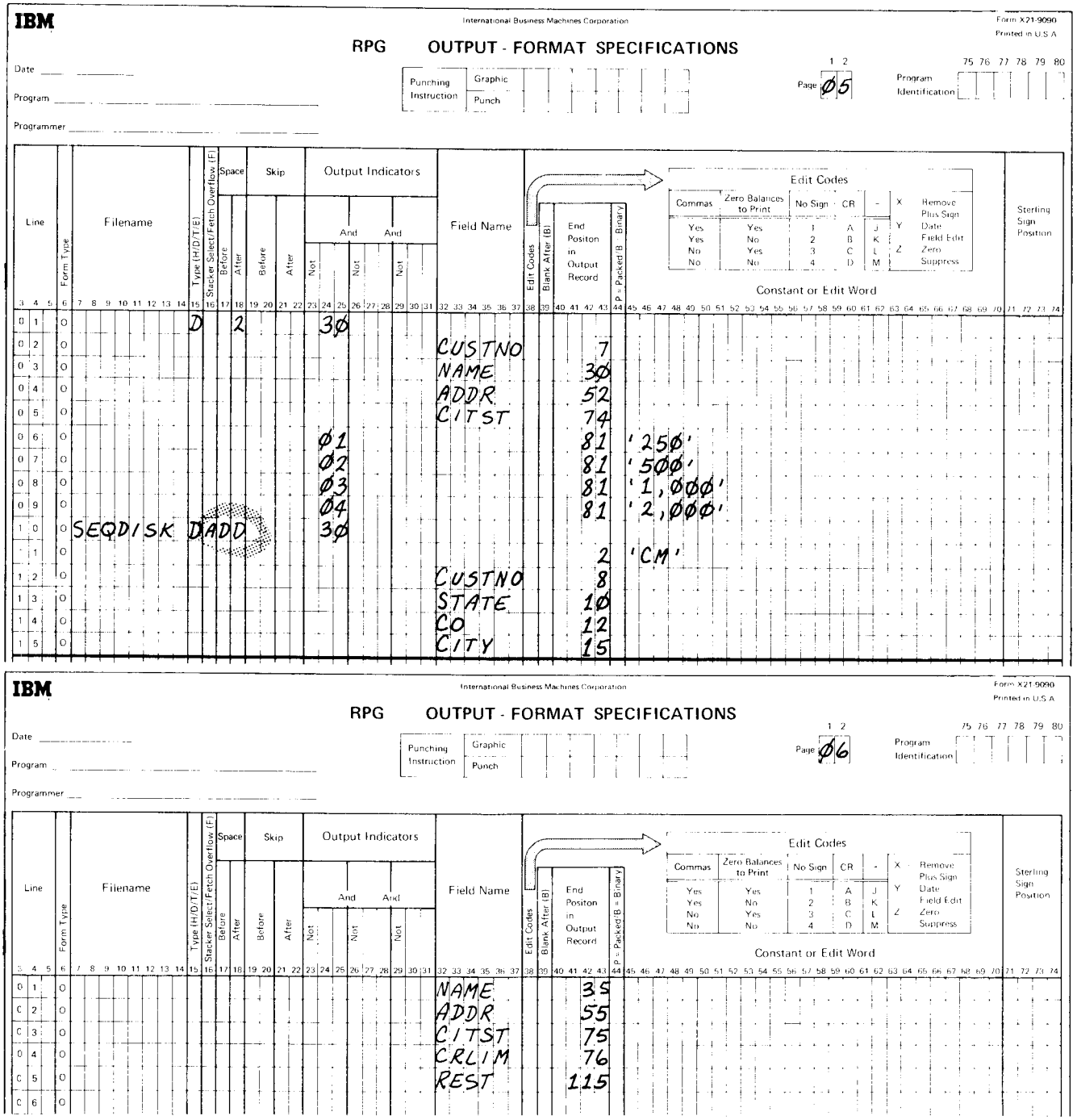


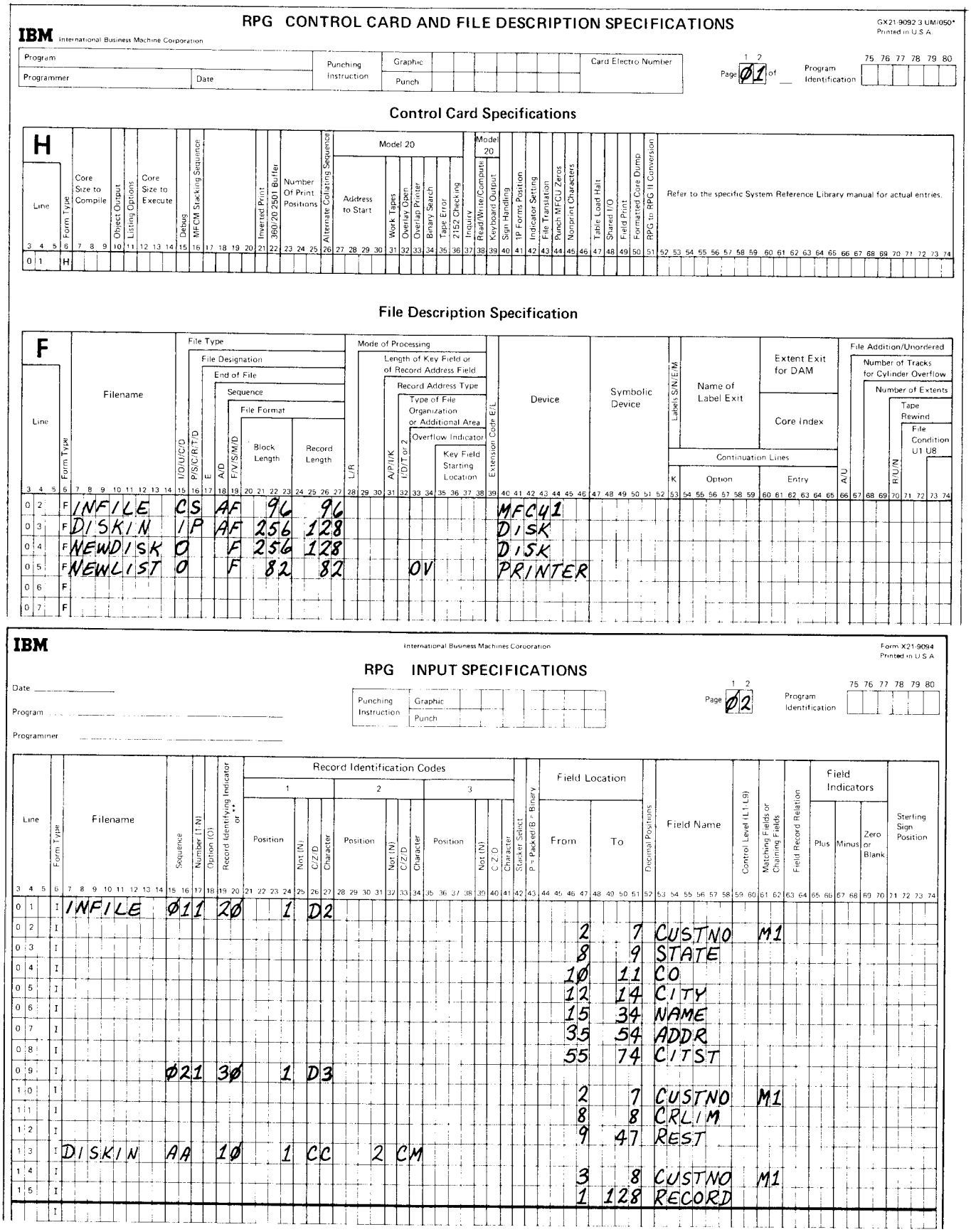
Figure 5 (Part 3 of 3). Adding Records to the End of the Sequential Customer File

Merging Records Between Records in the File

Often records must be added between existing records in a sequential disk file. When records must be added in this manner, you must sort the new records and re-create the file. This new file contains the added records merged in correct order with the records from the original file.

Example of Merging Records Between Records in the File

Figure 6 shows the RPG II coding necessary to merge records in the file. This example is similar to the *Example of Adding Records to the End of the File* in that the input and output records are the same format and the same report is printed.



RPG CONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS

IBM International Business Machine Corporation

GX21-9092-3 UM/050* Printed in U.S.A.

Program | Programmer | Date | Punching Instruction | Graphic | Punch | Card Electro Number | Page 01 of __ | Program Identification

Control Card Specifications

H

Line	Form Type	Core Size to Compile	Object Output	Listing Options	Core Size to Execute	Debug	MFCM Stacking Sequence	Inverted Print	360/20 2501 Buffer	Number Of Print Positions	Alternate Collating Sequence	Model 20: Address to Start	Model 20: Work Tapes	Model 20: Overlay Open	Model 20: Overlap Printer	Model 20: Binary Search	Model 20: Tape Error	Model 20: 2152 Checking	Model 20: Inquiry	Model 20: Read/Write/Compute	Model 20: Keyboard Output	Sign Handling	1P Forms Position	Indicator Setting	File Translation	Punch MFCU Zeros	Nonprint Characters	Table Load Halt	Shared I/O	Field Print	Formatted Core Dump	RPG to RPG II Conversion	Refer to the specific System Reference Library manual for actual entries.
01	H																															

File Description Specification

F

Line	Form Type	Filename	File Type: I/O/U/C/D	File Designation: P/S/C/R/T/D	End of File: E	Sequence: A/D	File Format: F/V/S/M/D	Block Length	Record Length	Mode of Processing	Length of Key Field or of Record Address Field	Record Address Type: A/P/N/K	Type of File Organization or Additional Area: I/O/T or 2	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Core Index	Continuation Lines: K	Continuation Lines: Option	Continuation Lines: Entry	File Addition/Unordered: A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind: R/U/N	File Condition U1 U8
02	F	INFILE	C	S		A	F	96	96								MFC41													
03	F	DISKIN	I	P		A	F	256	128								DISK													
04	F	NEWDISK	O				F	256	128								DISK													
05	F	NEWLIST	O				F	82	82					OV			PRINTER													
06	F																													
07	F																													

IBM International Business Machines Corporation

Form X21-9094 Printed in U.S.A.

RPG INPUT SPECIFICATIONS

Date _____ Program _____ Programmer _____

Punching Instruction | Graphic | Punch | Page 02 | Program Identification

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P = Packed/B = Binary	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank	Sterling Sign Position
01	I	INFILE	01	1		20		1	D	2																				
02	I																				2	7		CUSTNO	M1					
03	I																				8	9		STATE						
04	I																				10	11		CO						
05	I																				12	14		CITY						
06	I																				15	34		NAME						
07	I																				35	54		ADDR						
08	I																				55	74		CITST						
09	I		021			30		1	D	3																				
10	I																				2	7		CUSTNO	M1					
11	I																				8	8		CRLIM						
12	I																				9	47		REST						
13	I	DISKIN	AA			10		1	C	C		2	C	M																
14	I																				3	8		CUSTNO	M1					
15	I																				1	128		RECORD						
	I																													

Figure 6 (Part 1 of 3). Merging Records Between Records in the Sequential Customer File

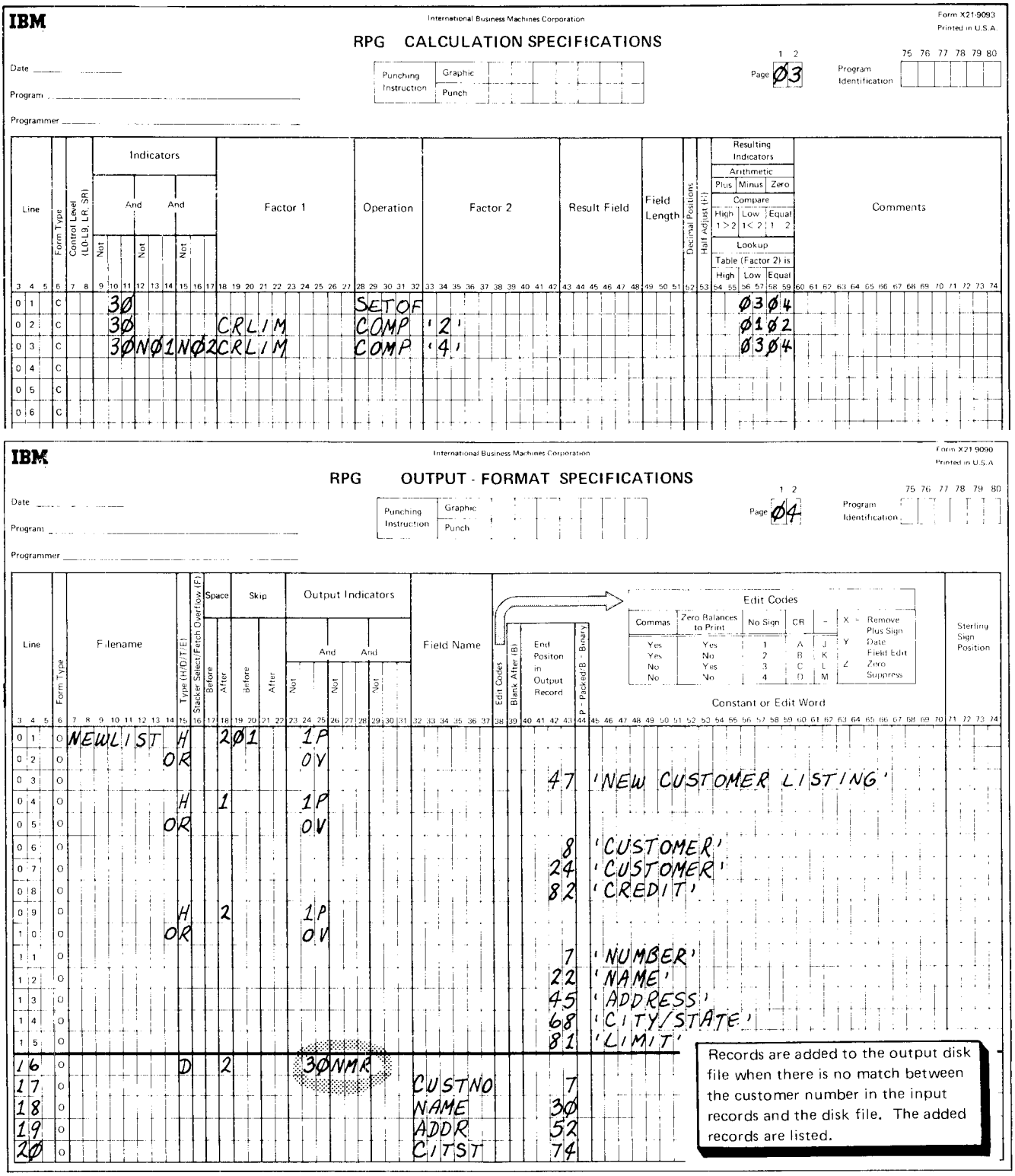


Figure 6 (Part 2 of 3). Merging Records Between Records in the Sequential Customer File

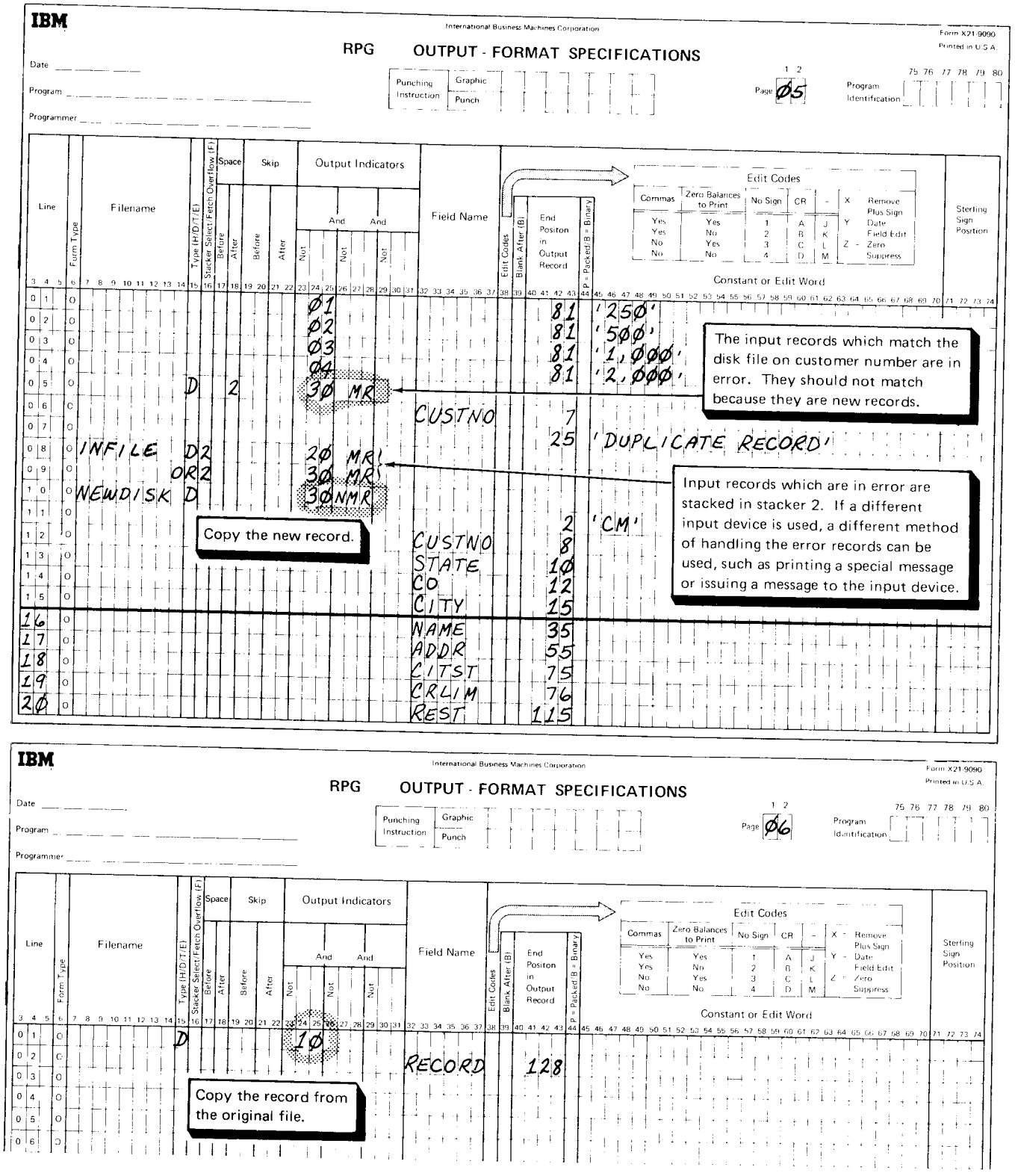


Figure 6 (Part 3 of 3). Merging Records Between Records in the Sequential Customer File

Tagging Records for Deletion

When a record becomes inactive, you may no longer want to process it with the other records. Since the record is not physically removed from the file, you must identify the record so it can be bypassed. One way to do this is to put a code, called a delete code, in a particular location in the record. This code can be any character you want. Any program that should not process deleted records can check for this code. If the code is present, the program can bypass the record. The deleted records can be physically removed when the file is reorganized (see *Reorganizing a File* in this section).

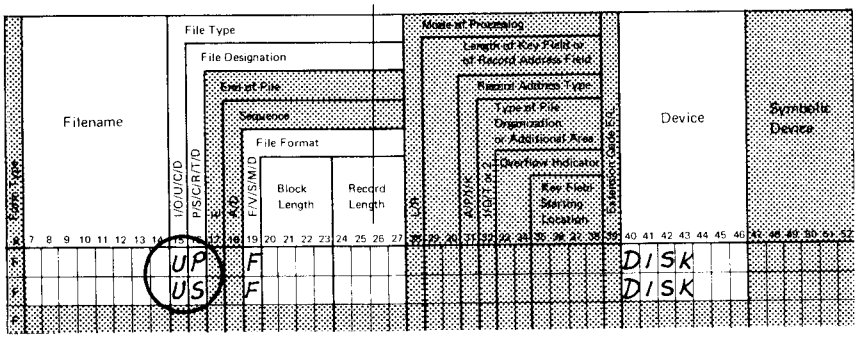
Example of Tagging Records for Deletion

An example of tagging records for deletion is shown in this section under *Updating Records, Example of Updating Records.*

Updating Records

Many jobs require changes to certain data in a record. This function is called updating. To update disk files, the following entries are required on the File Description sheet:

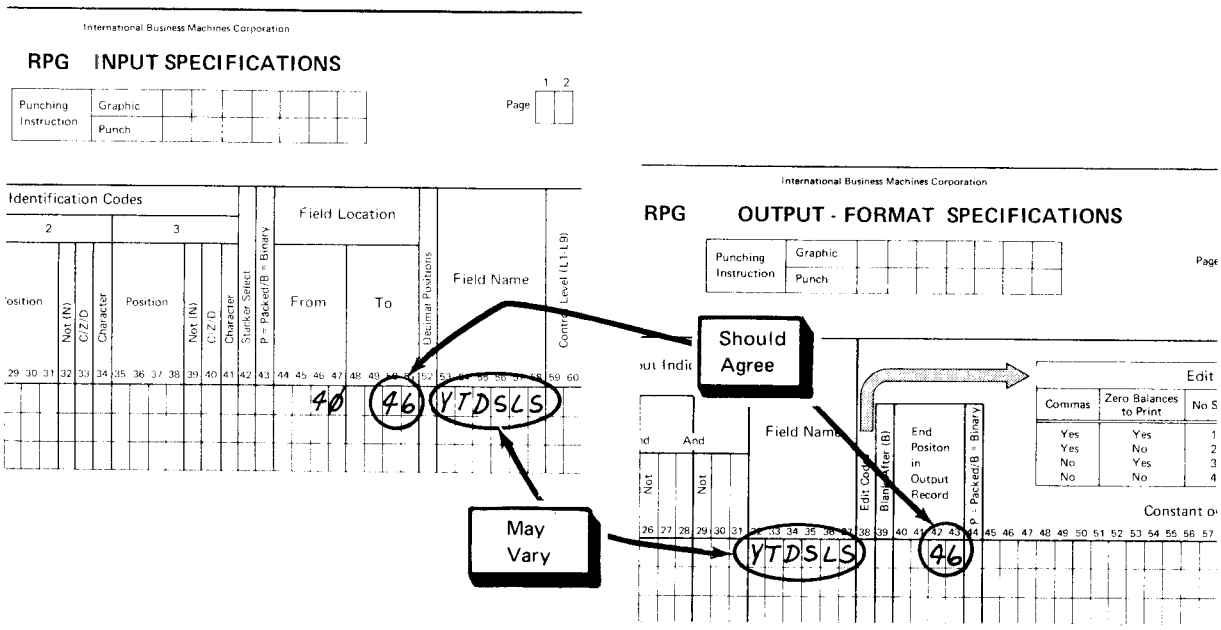
File Description Specifications



Some of the entries are the same entries needed to create a sequential file and were discussed in this chapter under *Creating a Sequential File.*

The two new entries are circled. Column 15 must contain a U to indicate that the file is an update file. Column 16 can contain either a P or an S depending on whether the file is a primary file or a secondary file.

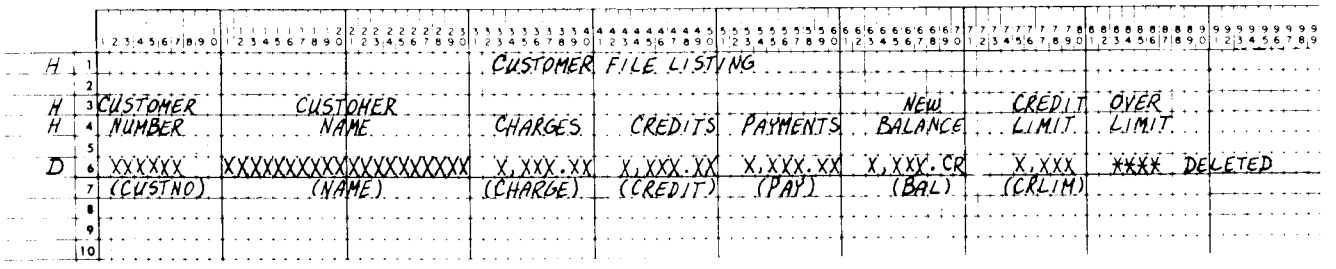
Since updating means getting a record from a disk file, changing some data, and putting the record back in its original location, an update file is like a combination input/output file. For this reason, the file to be updated must be specified on both the Input and Output-Format sheets. Field locations should agree between the two sheets. Field names may vary depending on the updating being done. Field lengths must agree.



Example of Updating Records

Periodically, you might want to update the accounting information for each customer in your customer file. You might also want to tag some customer records for deletion. A printed report, like the one shown in Figure 7, lists the updated information and the records tagged for deletion.

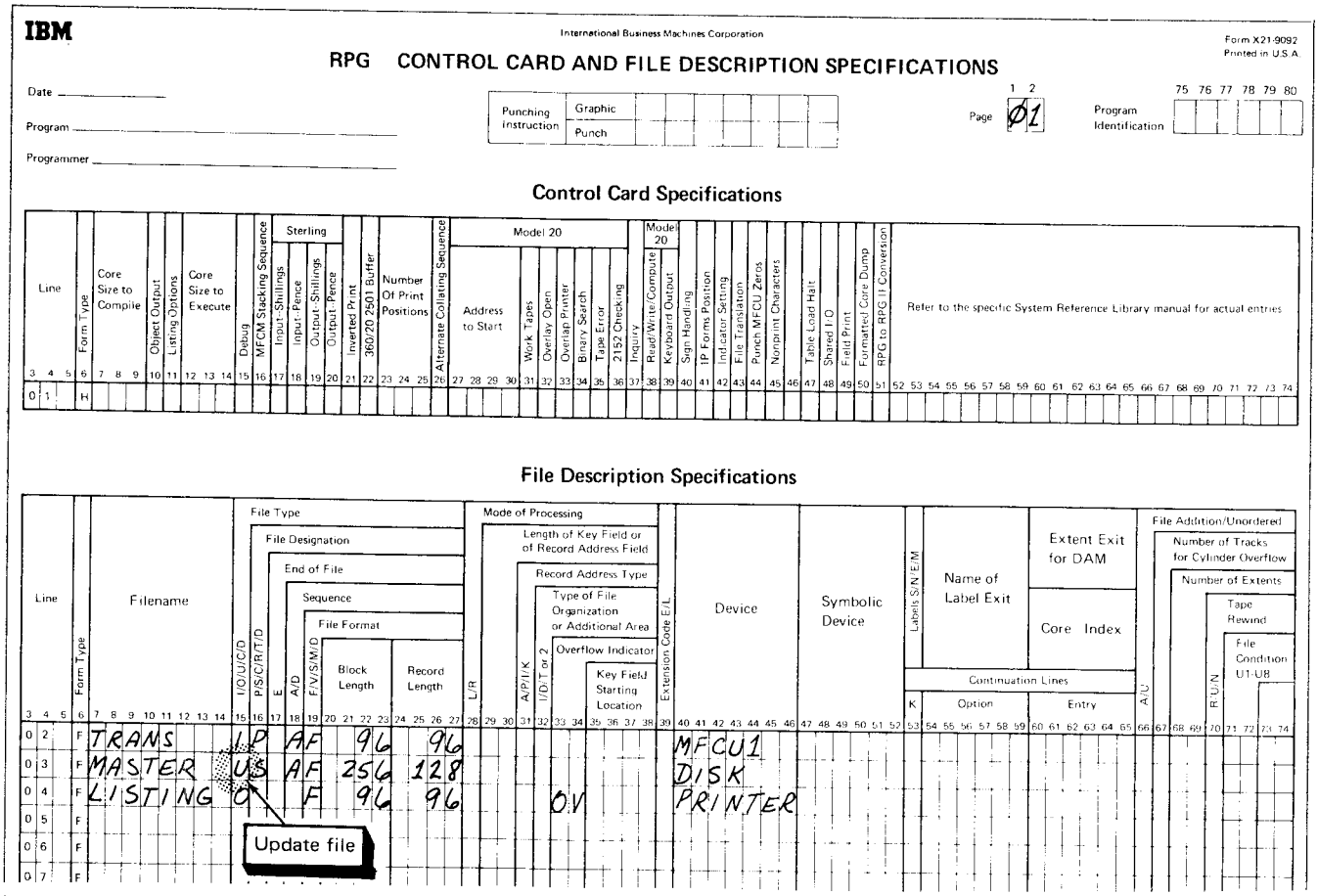
The TRANS file contains two input record types. One type identifies disk records to be deleted (D in column 1); the other type contains information needed to update the MASTER file (3 in column 1). Figure 8 shows the RPG II coding necessary to update records and to tag records for deletion.



CUSTOMER FILE LISTING

CUSTOMER NUMBER	CUSTOMER NAME	CHARGES	CREDITS	PAYMENTS	NEW BALANCE	CREDIT LIMIT	OVER LIMIT
136728	JONES VARIETY	5,000.00	3,000.00	1,500.00	500.00	1,000	
301628	JIM'S 5 AND 10	600.00	200.00	100.00	300.00	250	****
795246	SCHMIDT HARDWARE	2,500.00	1,000.00	500.00	1,000.00	2,000	

Figure 7. Report of Updated Customer Records and Deleted Customer Records



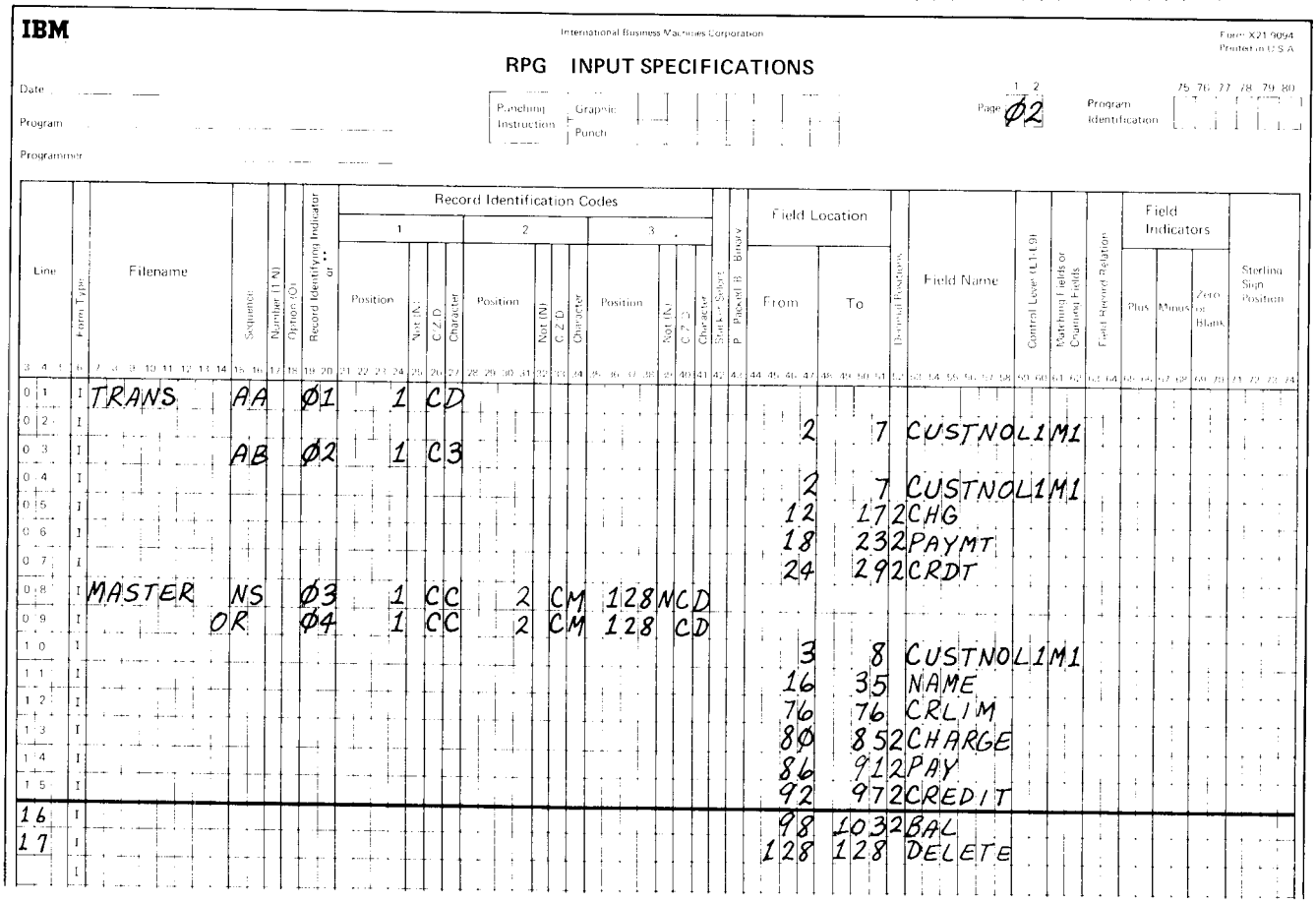
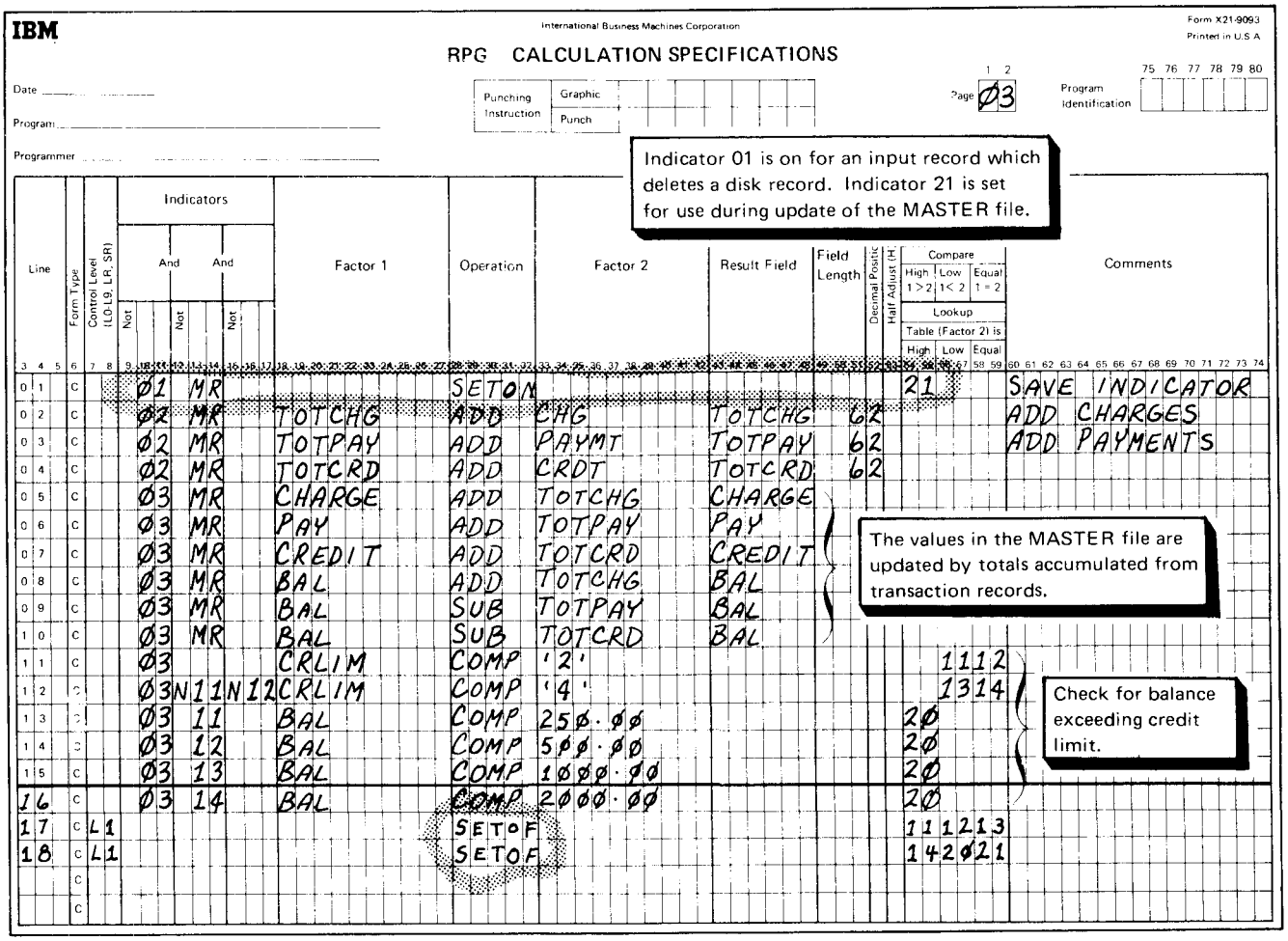


Figure 8 (Part 1 of 4). Updating and Deleting Records in the Sequential Customer File



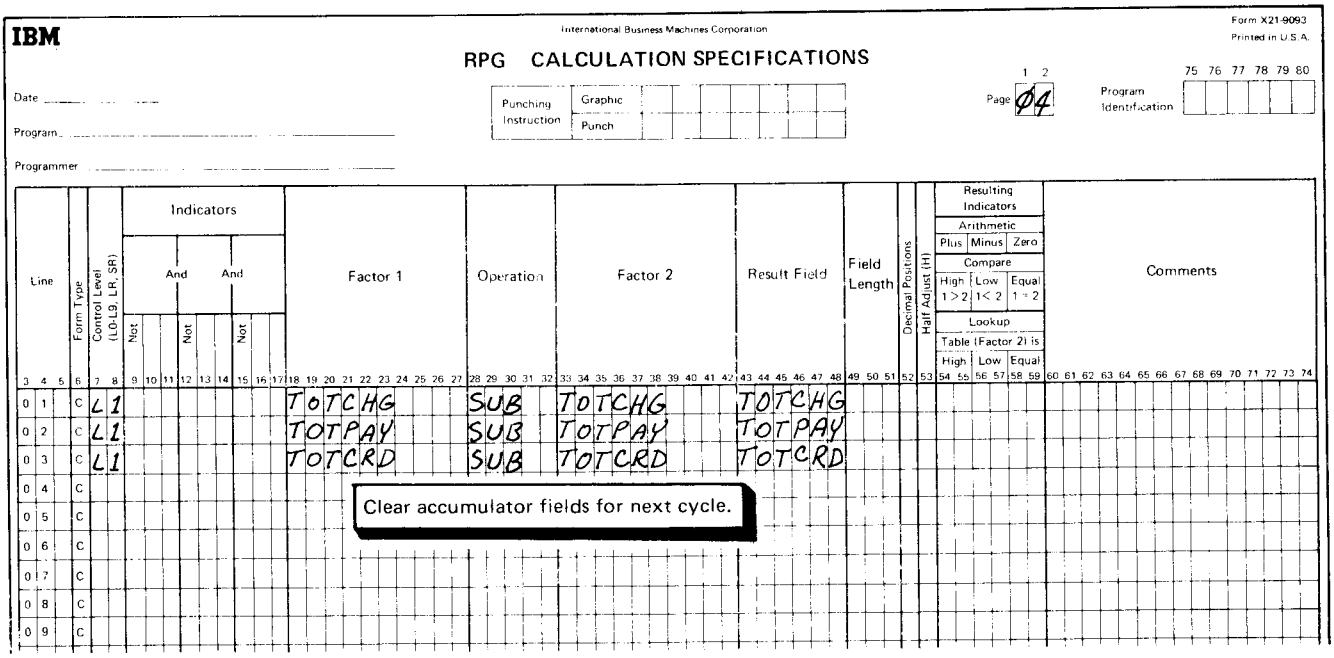


Figure 8 (Part 2 of 4). Updating and Deleting Records in the Sequential Customer File

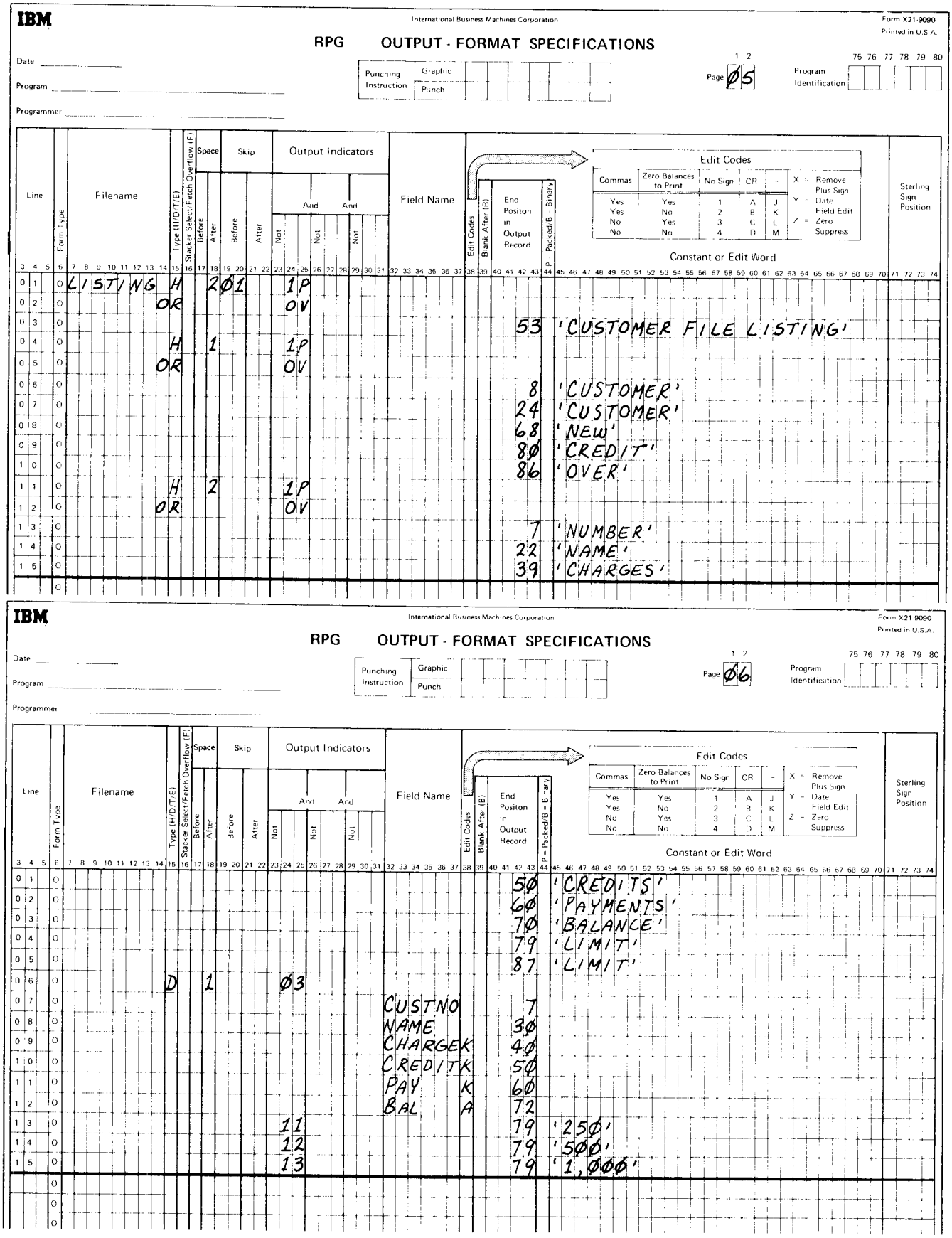


Figure 8 (Part 3 of 4). Updating and Deleting Records in the Sequential Customer File

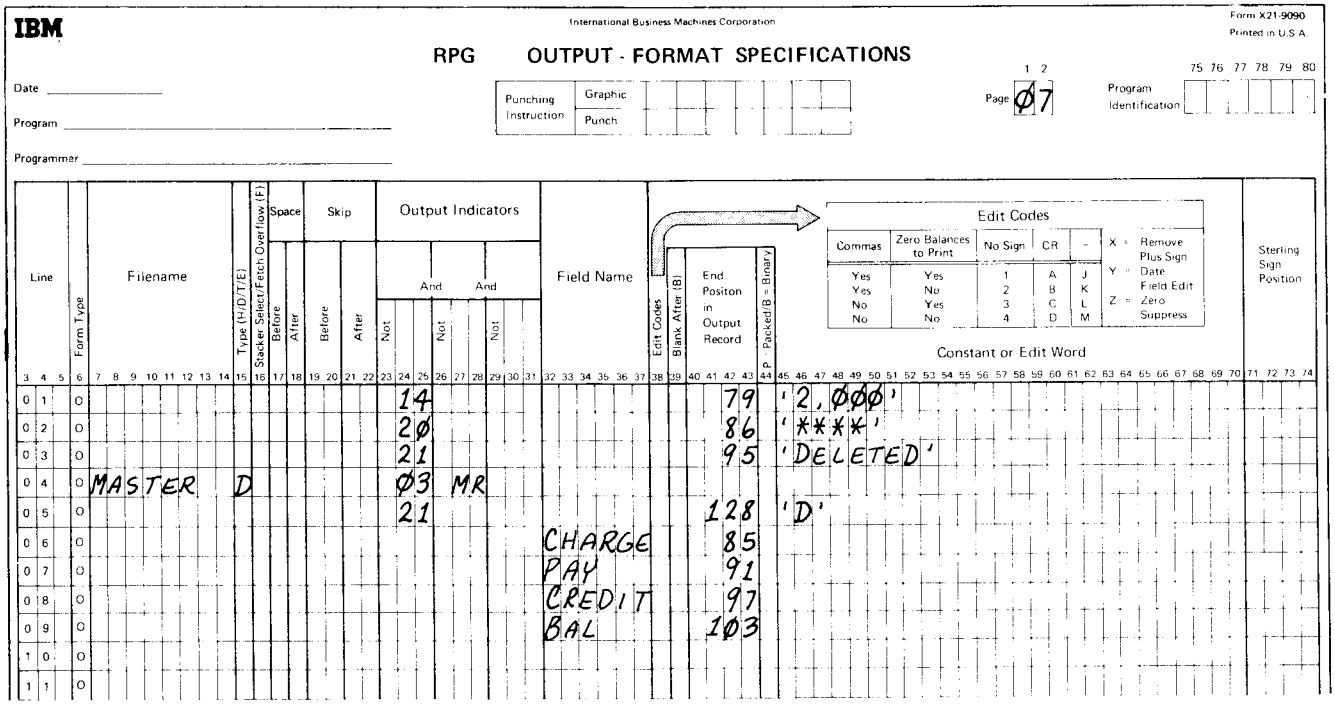


Figure 8 (Part 4 of 4). Updating and Deleting Records in the Sequential Customer File

Reorganizing a File

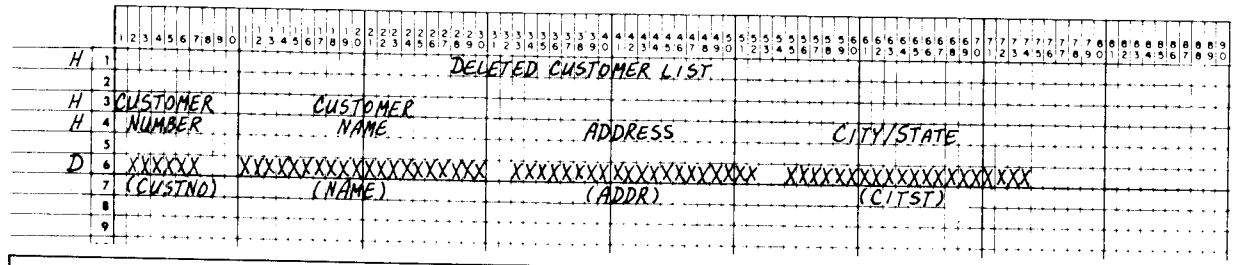
The space reserved for a file often becomes filled as additions are made. Reorganizing is a means of freeing space by physically removing inactive records (those with a delete code) from a file.

You can reorganize your file using the Disk Copy/Dump program. The DELETE or OMIT parameter causes records of a specified type to be deleted from a file as it is copied. You can also use the Disk Sort program to remove the records tagged for deletion. For information on this program, see the *IBM System/3 Disk Sort Reference Manual,* SC21-7522.

Example of Reorganizing a File

Records were tagged for deletion in the *Example of Updating Records* by placing a D in position 128. Now suppose you want to reorganize your customer file to physically remove these records from the file. You also want to print a report listing the customer records deleted (Figure 9).

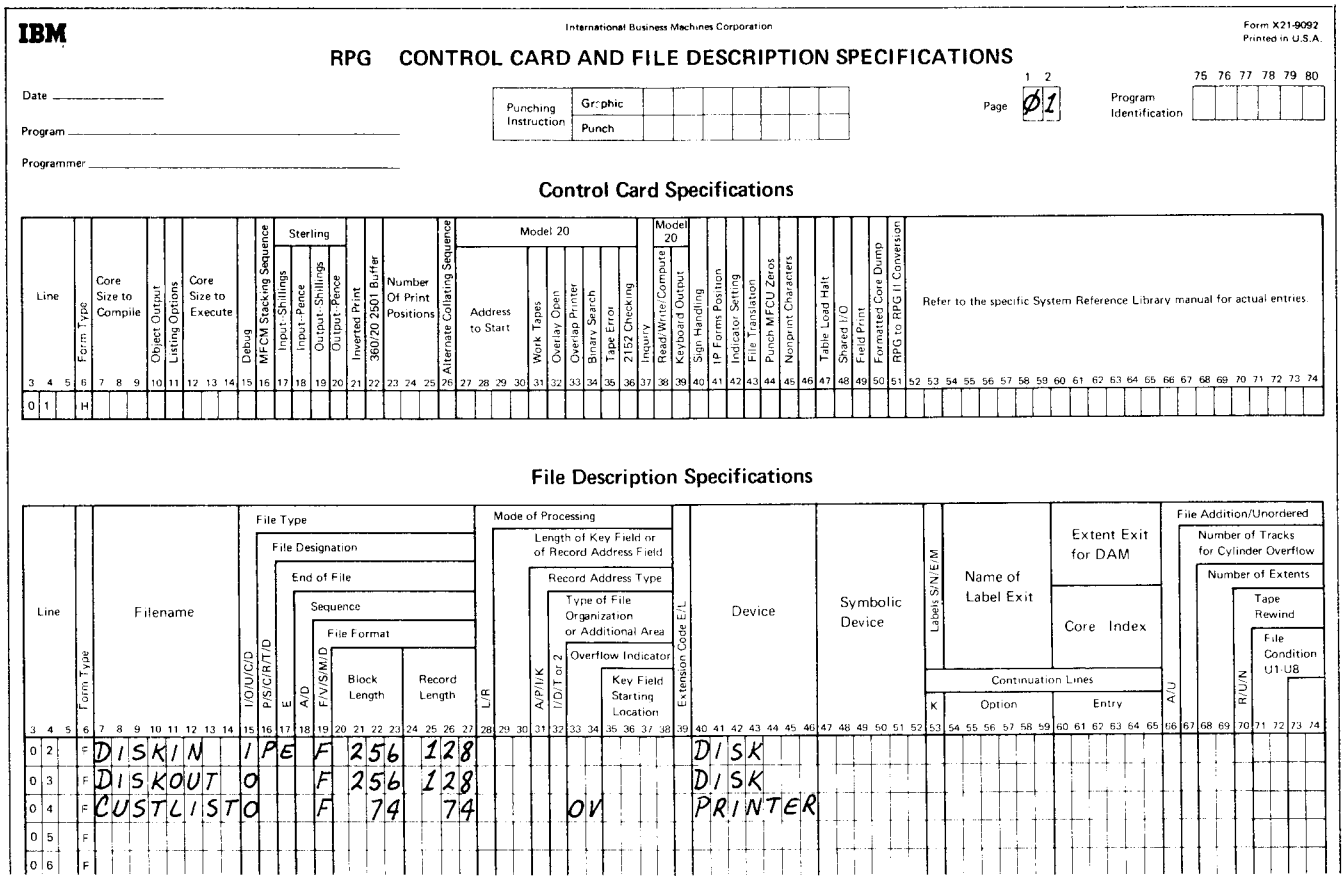
To delete these records you want to copy all records that do not have a D in position 128. Figure 10 shows the RPG II specifications necessary to reorganize the file.



DELETED CUSTOMER LIST

CUSTOMER NUMBER	CUSTOMER NAME	ADDRESS	CITY/STATE
652791	GREEN GROCERY, INC	1739-6TH ST SW	BIG CITY, CALIF
576290	STAR MARKET	3278 ADAMS ST	GOODTOWN, GA

Figure 9. Report of Deleted Customer Records



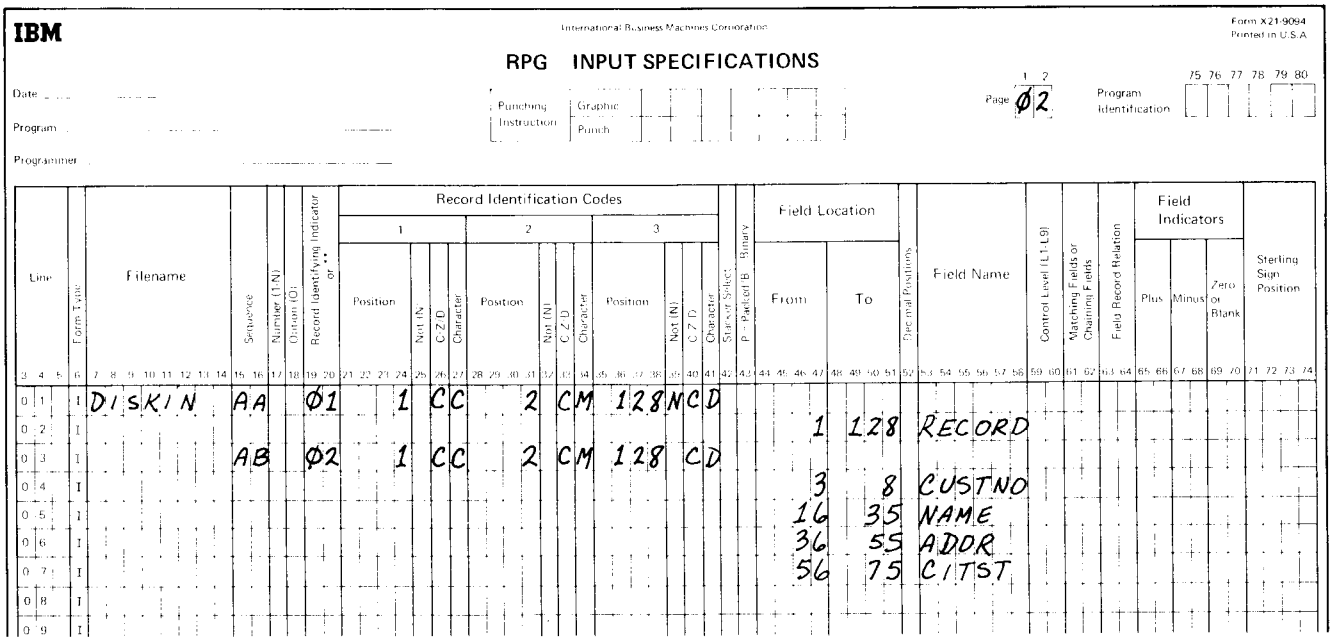


Figure 10 (Part 1 of 2). Reorganizing the Sequential Customer File to Remove Deleted Records

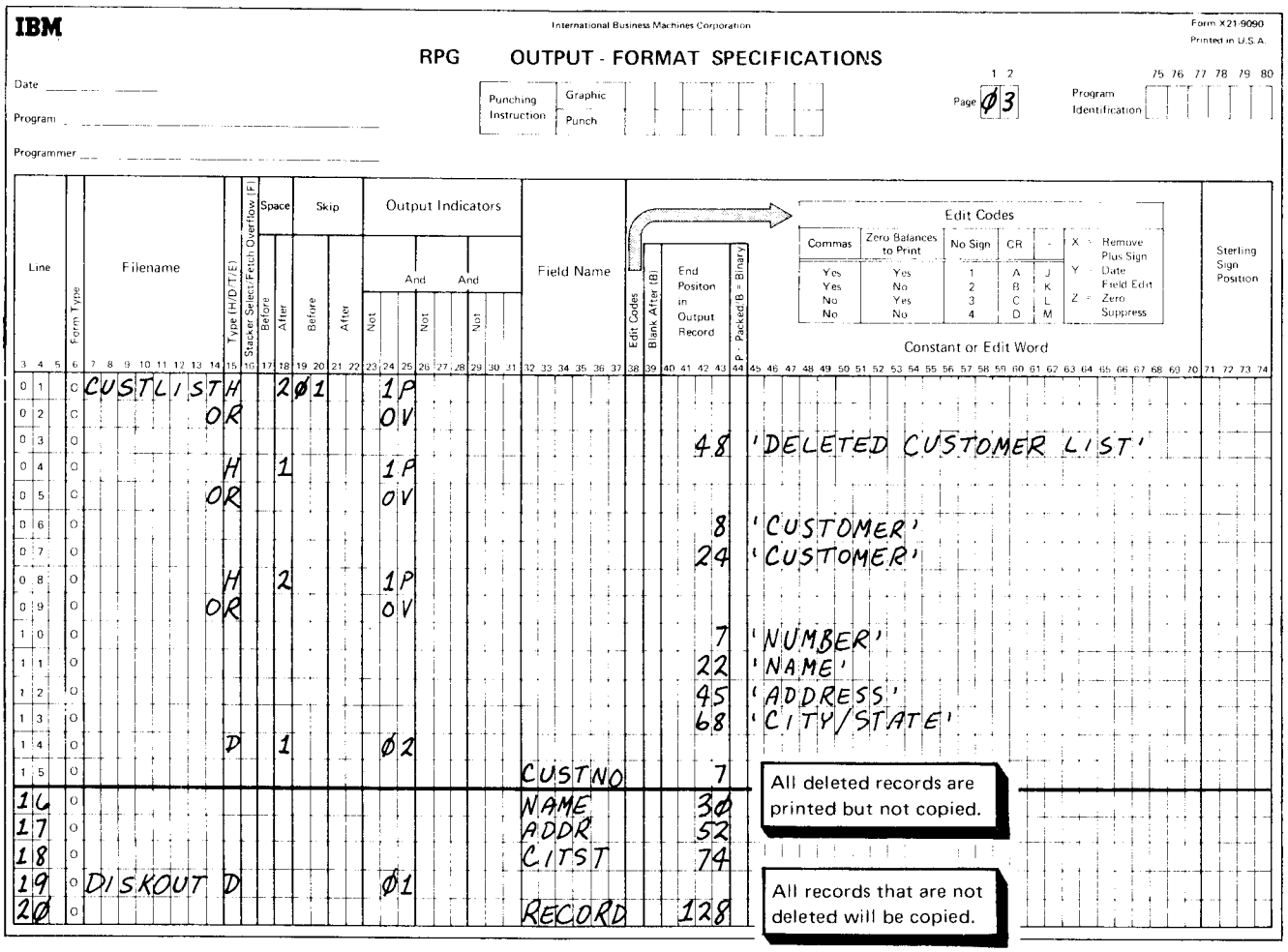


Figure 10 (Part 2 of 2). Reorganizing the Sequential Customer File to Remove Deleted Records

Chapter 2. Indexed Files

An indexed disk file has two parts: an index and the records. The index contains the key field and disk address of each record in the file; it is organized sequentially by key field. Records in the file are organized in the order in which they are loaded. Records in an indexed file can be processed in the following ways:

1. Sequentially by key.
2. Sequentially within limits.
3. Randomly by key or relative record number.
4. Consecutively (without keys).
5. By ADDROUT file.

This chapter explains how to use the RPG II language to create and maintain a single volume indexed file. Sample jobs are used to illustrate these functions.

To understand the sample jobs, a basic knowledge of RPG II is necessary. If you do not fully recall some of the coding used in the sample jobs, you should refer to the *IBM System/3 RPG II Reference Manual,* SC21-7504 (for the Model 10 Disk System or the Model 15), or the *IBM System/3 Model 6 RPG II Reference Manual,* SC21-7517, depending on the system you have.

CREATING AN INDEXED FILE

You can create a single volume indexed file with records that are in ordered or unordered sequence. An ordered sequence means the records are arranged in order according to the major control field that will be used as the key of the file. If an ordered sequence is specified, records are sequence-checked by System/3 data management.

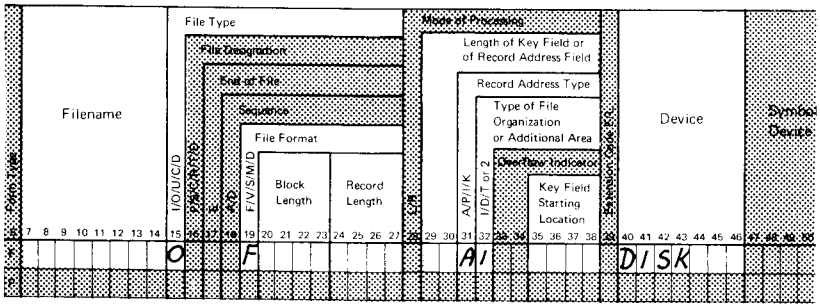
An unordered sequence means the records are in no particular order. For example, if records in an inventory item file were organized by frequency of use, they would be in an unordered sequence with the most active items at the beginning of the file. System/3 data management will not sequence check the records or check for duplicate records while an indexed file is being loaded in unordered sequence. The index of an unordered file is sorted into ascending sequence after all the records have been loaded. At this time, System/3 data management will detect duplicate records.

Note: Multivolume indexed files must be loaded in key field sequence, that is, in ordered sequence. See *Related Publications* in the *Preface* for sources of additional information about multivolume files.

Creating an Ordered Indexed File

To create an indexed file in an ordered sequence, you must make the following entries on the File Description sheet:

File Description Specifications



The disk filename must be entered in columns 7 through 14. Column 15 must contain an O to indicate that the file is an output file.

All records in a file must be the same length, so column 19 must contain an F to specify that the record length is fixed.

A number equal to, or a multiple of, the disk record length must be entered in columns 20 through 23. This entry determines the size of the input/output area allocated by RPG II. If you want block length calculated for you by RPG II, assign a block length equal to the record length. By blocking disk records, you can increase the input/output efficiency of your program by reducing the number of accesses. You must be sure, however, that enough main storage is available for your input/output area.

Note: Block length calculation and input/output area allocation are described in *IBM System/3 Disk Concepts and Planning Guide,* GC21-7571. On Model 15, RPG II uses double buffering for indexed file output; therefore, main storage must be available for at least twice the block length plus twice the index buffer length.

Columns 24 through 27 must contain the length of the disk record. Whenever a disk file is being described, DISK (Model 6, 10, 15), DISK40 (Model 12, 15), or DISK45 (Model 10, 12, 15) is required in the Device columns.

Indexed files are allowed only on the main data area, and are not allowed on the simulation area. Models with 5445 or 3340 drives must use DISK45 and DISK40 respectively.

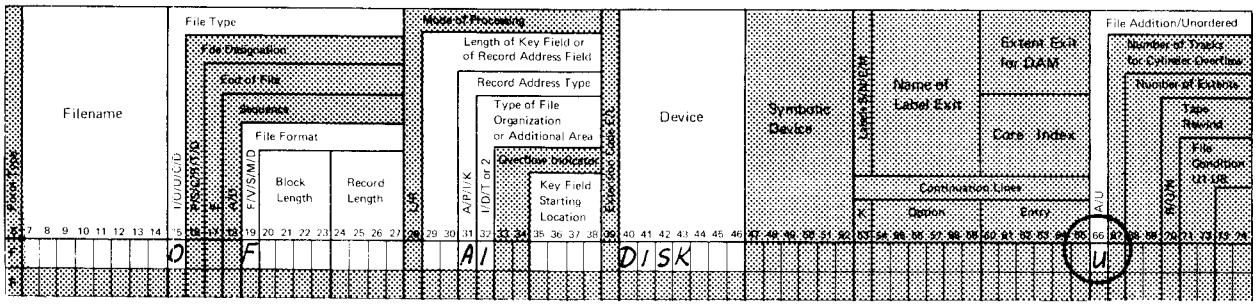
Entries in columns 31 and 32 indicate that an indexed file is to be created. An I in column 32 specifies an indexed file; an A in column 31 specifies that a key field exists.

The other two entries describe the length and location of the key. Key length, entered in columns 29 through 30, is the same for all records in a file. The maximum key length is 29 characters. The location of the first character of the key is specified in columns 35 through 38. Thus, if a 6-character key field were located in positions 73 through 78 of a disk record, you would enter a 73 in columns 37 through 38 of the File Description sheet and a 6 in column 30.

Creating an Unordered Indexed File

To create an indexed file in an unordered sequence, one entry is needed on the File Description sheet in addition to the entries needed to create an ordered indexed file. This entry is a U in column 66, indicating that the records are to be loaded in an unordered sequence:

File Description Specifications



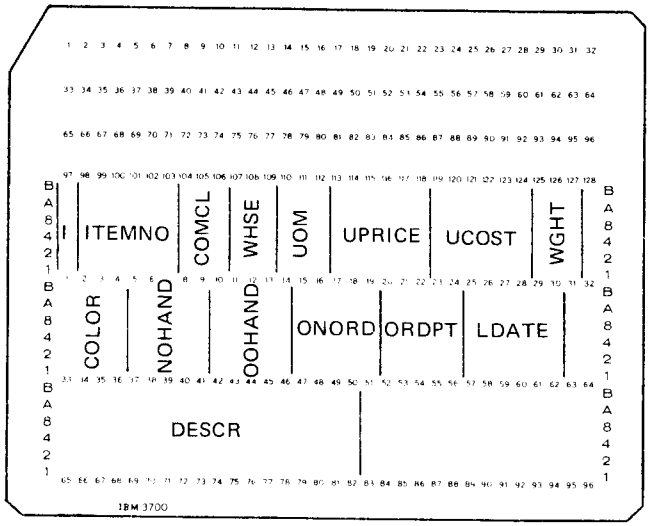
Example of Creating an Indexed File

Suppose you want to create an inventory file on disk and want to process the records in various ways (sequentially, randomly, within limits). An indexed file provides this processing flexibility. Whether you load your records in an ordered or an unordered sequence depends on which sequence gives you the most processing efficiency.

To create an indexed file, you must first determine the input record format and the output record format (Figure 11). The file is created by writing the inventory item data from the input records onto disk. Notice that the output record format provides space so additional information can be added to the record later if necessary.

Figure 12 shows the RPG II coding necessary to create the indexed inventory file. The program will count the number of records loaded and print the total after the file has been created.

Input Record



Note: The input file need not be a card file. See *How to Use This Manual, Note Concerning Examples* at the beginning of this manual.

Key

CODE	=	Record identifying code (I)
ITEMNO	=	Item number
COMCL	=	Commodity class
WHSE	=	Warehouse location
UOM	=	Unit of measure
UPRICE	=	Unit price
UCOST	=	Unit cost
WGHT	=	Weight
COLOR	=	Color
NOHAND	=	New on hand quantity
OOHAND	=	Old on hand quantity
ONORD	=	Quantity on order
ORDPT	=	Order point
LDATE	=	Latest transaction date
DESCR	=	Description
DELETE	=	Output record delete code

Output Record

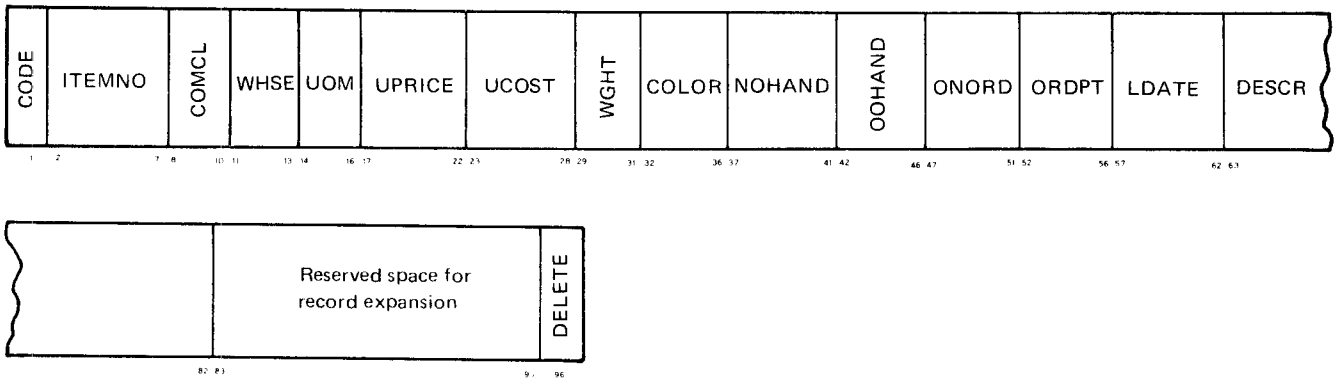


Figure 11. Input Record and Output Record Formats

RPG CONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS

IBM International Business Machine Corporation

GX21-9092-3 UM:050* Printed in U.S.A.

Program | Punching Instruction: Graphic / Punch | Card Electro Number

Programmer | Date

Page 01 of __ | Program Identification

Control Card Specifications

H

Line	Form Type	Core Size to Compile	Object Output	Listing Options	Core Size to Execute	Debug	MP-UM Stacking Sequence	Inverted Print	360/20 2501 Buffer	Number Of Print Positions	Alternate Collating Sequence	Model 20: Address to Start	Work Tapes	Overlay Open	Overlap Printer	Binary Search	Tape Error	2152 Checking	Model 20: Inquiry	Read/Write/Compute	Keyboard Output	Sign Handling	1P Forms Position	Indicator Setting	File Translation	Punch MFCU Zeros	Nonprint Characters	Table Load Halt	Shared I/O	Field Print	Formatted Core Dump	RPG to RPG II Conversion	Refer to the specific System Reference Library manual for actual entries.
01	H																															

File Description Specification

F

Line	Form Type	Filename	File Type	File Designation	End of File	Sequence	File Format	Block Length	Record Length	Mode of Processing	Length of Key Field or of Record Address Field	Record Address Type	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Core Index	Continuation Lines: K	Option	Entry	File Addition/Unordered	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	File Condition U1 U8
02	F	INFILE	I	P	E		F	96	96								MFCU1													
03	F	OUTDISK	O				F	192	96	R	Ø6	A	I			2	DISK									U				
04	F	REPORT	O				F	2Ø	2Ø								PRINTER													
05	F																													
06	F																													
07	F																													
08	F																													
09	F																													

Make this specification only if you want the records to be loaded in an unordered sequence.

RPG INPUT SPECIFICATIONS

IBM International Business Machine Corporation

GX21-9094-2 U M 050* Printed in U.S.A.

Program | Punching Instruction: Graphic / Punch | Card Electro Number

Programmer | Date

Page 02 of __ | Program Identification

I

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank
			O R A N D																										
01	I	INFILE	AA				Ø1		1	C1																			
02	I																			1	96		RECORD						
03	I																												
04	I																												
05	I																												
06	I																												

Figure 12 (Part 1 of 2). Creating an Indexed Inventory File

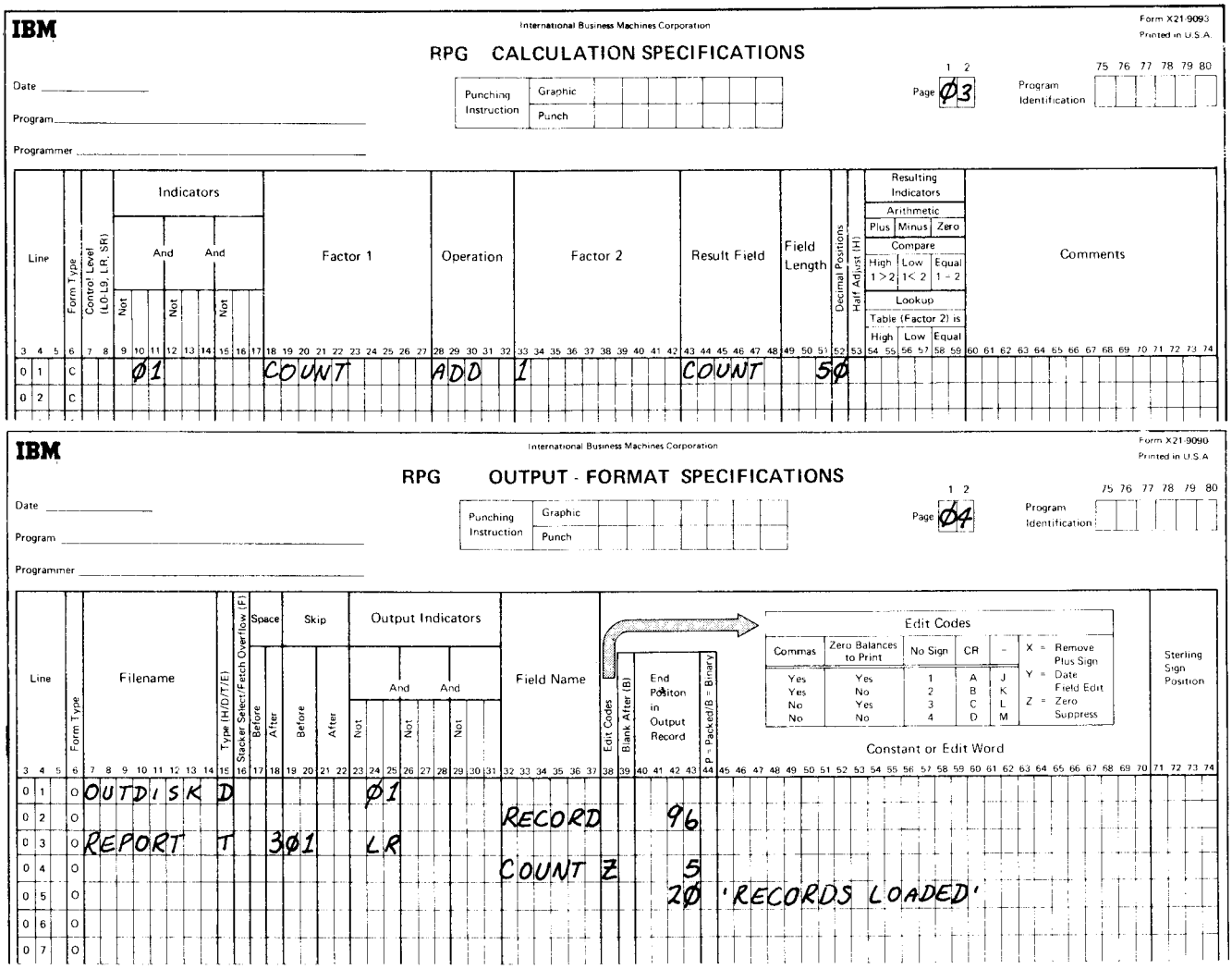


Figure 12 (Part 2 of 2). Creating an Indexed Inventory File

MAINTAINING AN INDEXED FILE

Once a file is created, it usually needs to be maintained. File maintenance means performing those functions that keep a file current for daily processing needs. Four file maintenance functions apply to indexed files:

1. Updating records.
2. Adding records.
3. Tagging records for deletion.
4. Reorganizing a file.

Updating Records

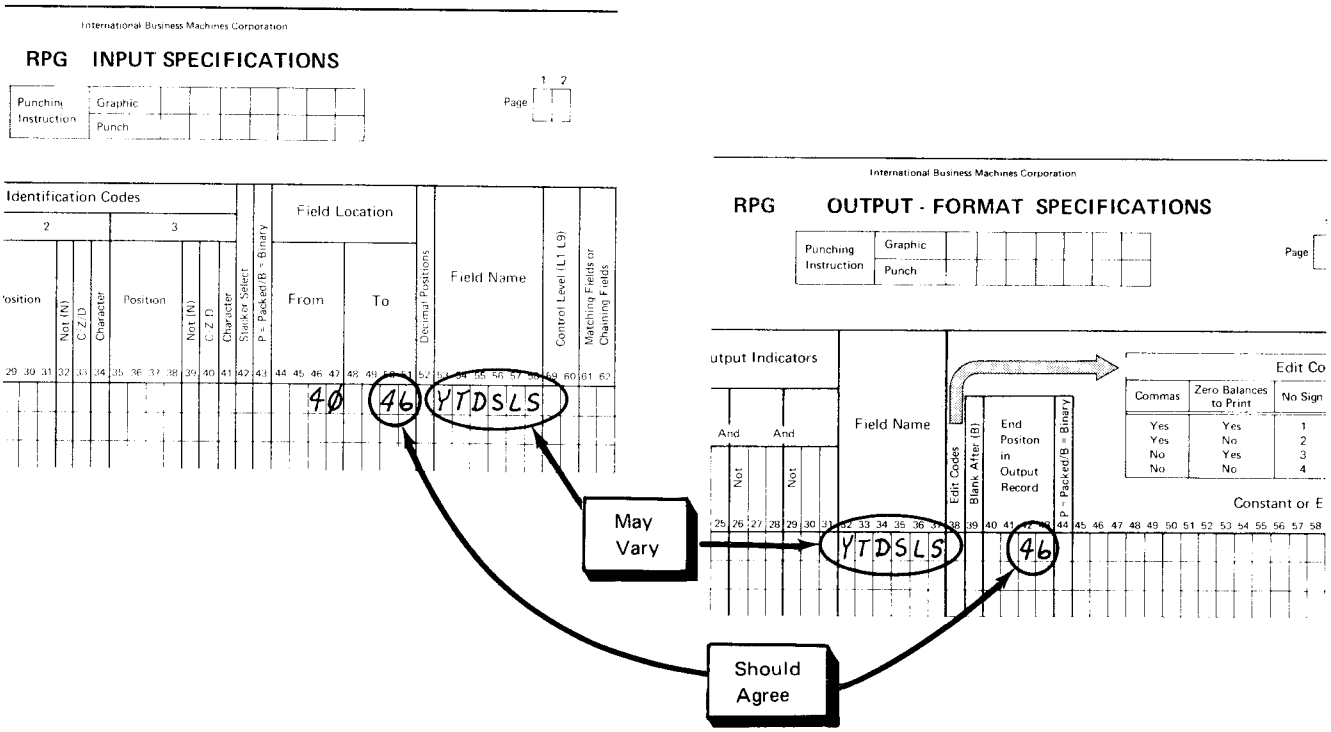
Many jobs require you to change certain data in a record. This function is called updating. You can update the records in one of three ways:

1. Sequentially by key.

2. Randomly by key.

3. Sequentially within limits.

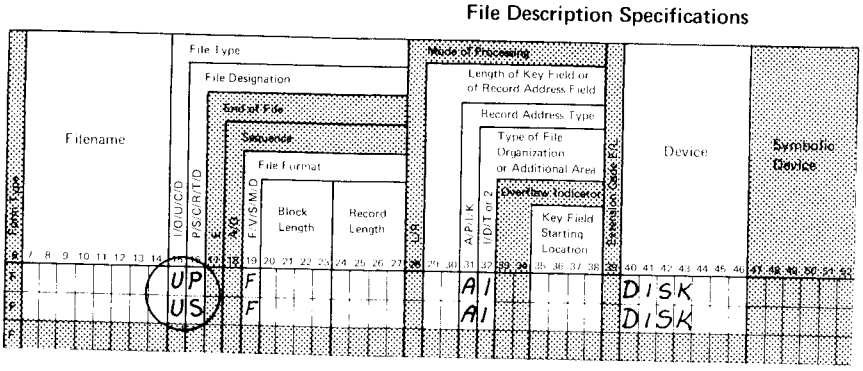
Since updating means getting a record from a disk file, changing some data, and putting the record back in its original location, an update file is like a combination input/output file. For this reason, the file to be updated must be specified on both the Input and Output-Format sheets. Field locations should agree between the two sheets; field names may vary depending on the updating being done. Field lengths must agree.



Updating Records Sequentially by Key

Records are usually updated sequentially by key when you want to update a large part of the file. To update records in this way, the following entries are required on the File Description sheet:

File Description Specifications



Some of the entries are the same entries needed to create an indexed file and were discussed in this chapter under *Creating an Indexed File.*

The two new entries are circled. Column 15 must contain a U to indicate that the file is an update file. Column 16 can contain either a P or an S depending on whether the file is a primary or a secondary file.

Example of Updating Records Sequentially by Key

For examples of this kind of updating, see:

- *Updating Records, Example of Updating Records* in Chapter 1. This coding applies if the necessary File Description sheet entries are made.
- *Adding Records, Example of Adding Records Sequentially by Key* later in this chapter.

Updating Records Randomly by Key

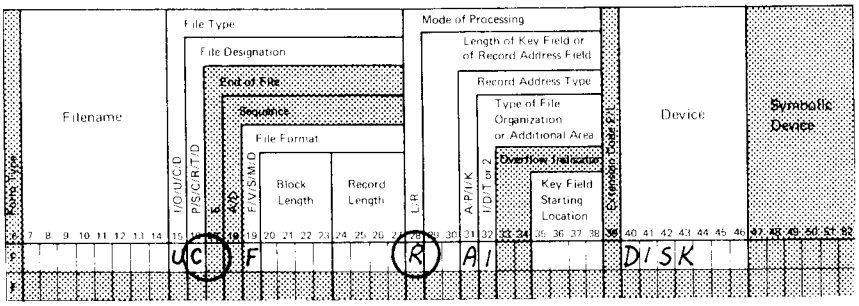
When an indexed file is updated randomly by key, input records are *chained* to the indexed file. Chaining means comparing the record key with the key in the index. If the keys are equal, the corresponding data record is made available. The data used as a record key in the chain operation can be a field in an input record or it can be created in your program.

Records are read from the indexed file during the calculation phase of the program. Fields of records read from chained update files can be read during total calculations and updated during total output, or read during detail calculations and updated during detail output. Records may also be read during detail calculations from chained update files and updated during total output.

When you update records randomly by key, entries must be made on the File Description and Calculation sheets.

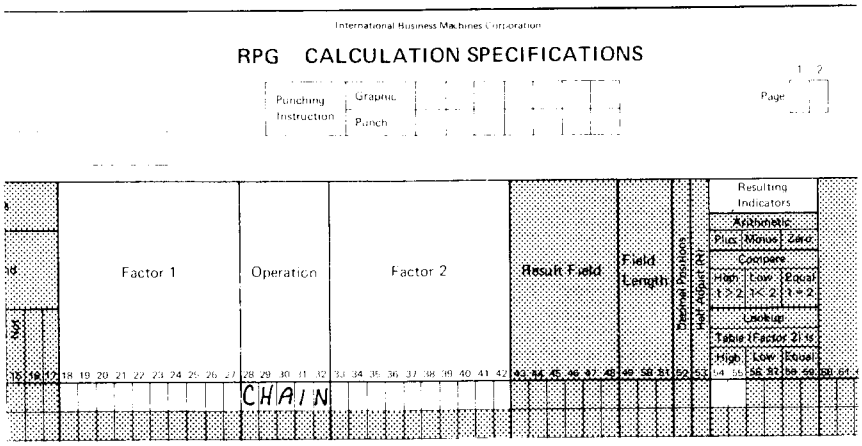
One new entry is needed on the File Description sheet in addition to the entries needed to update a file sequentially by key. One of the previous entries also has to be changed.

File Description Specifications



The changed entry is a C in column 16, which identifies the file as a chained file. The new entry is an R in column 28, indicating that the file is processed randomly by key.

The following entries are needed on the Calculation sheet to describe the chain operation:



Factor 1 must contain the name of the field to be used during the search for a match in the index, CHAIN must be entered as the operation, and Factor 2 must contain the name of the update file.

A resulting indicator should be specified in columns 54 and 55 to indicate whether the record to be updated is in the index. (If the indicator is omitted, you will get an error during compilation.) Refer to the *IBM System/3 RPG II Reference Manual*, SC21-7504 for more information. When a match is found in the index, the disk address of the corresponding record is available. The desired record can then be located and read into storage. At that time the record can be updated. Your program can check to see if the required record is in the file by checking the specified resulting indicator. When the indicator is off, the record was found and can be updated; when the indicator is on, the record was not found.

Example of Updating Records Randomly by Key

Suppose that whenever a transaction is made you want to update the item record in the inventory file. Since the transaction records are not in sequence, you will process the file randomly by key. You might also want to tag some records for deletion while the file is being processed.

Figure 13 shows the RPG II coding necessary to update the records and tag records for deletion. The TRANS file contains the update and deletion information.

Refer to Appendix H for more information on blocking records when processing an indexed file randomly.

This page intentionally left blank

IBM

International Business Machines Corporation

Form X21-9092
Printed in U.S.A.

RPG CONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page 01

Program Identification

Control Card Specifications

Line	Form Type	Core Size to Compile	Object Output	Listing Options	Core Size to Execute	Debug	MFCM Stacking Sequence	Sterling: Input-Shillings	Sterling: Input-Pence	Sterling: Output-Shillings	Sterling: Output-Pence	Inverted Print	360/20 2501 Buffer	Number Of Print Positions	Alternate Collating Sequence	Model 20: Address to Start	Model 20: Work Tapes	Model 20: Overlay Open	Model 20: Overlap Printer	Model 20: Binary Search	Model 20: Tape Error	Model 20: 2152 Checking Inquiry	Model 20: Read/Write/Compute	Model 20: Keyboard Output	Model 20: Sign Handling	1P Forms Position	Indicator Setting	File Translation	Punch MFCU Zeros	Nonprint Characters	Table Load Halt	Shared I/O	Field Print	Formatted Core Dump	RPG to RPG II Conversion	Refer to the specific System Reference Library manual for actual entries.
01	H																																		

File Description Specifications

Line	Form Type	Filename	File Type	File Designation	End of File	Sequence	File Format	Block Length	Record Length	Mode of Processing	Length of Key Field or of Record Address Field	Record Address Type	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Core Index	Continuation Lines: K	Continuation Lines: Option	Continuation Lines: Entry	File Addition/Unordered	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	File Condition U1-U8
02	F	TRANS	C	P	E	F		96	96								MFCU1													
03	F	MASTER	U	C		F		192	96	R	Ø6	A	I			2	DISK													
04	F	LIST	O			F		96	96								PRINTER													
05	F																													
06	F																													

IBM

International Business Machines Corporation

Form X21-9094
Printed in U.S.A.

RPG INPUT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page 02

Program Identification

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or *	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	Record Identification Codes 2: Position	2: Not (N)	2: C/Z/D	2: Character	Record Identification Codes 3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P = Packed / B = Binary	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank	Sterling Sign Position
01	I	TRANS	AA			Ø1		1	C	D																					
02	I																				2	7		ITEMNO							
03	I		AB			Ø2		1	C	R																					
04	I																				2	7		ITEMNO							
05	I																				8	12	Ø	QTYREC							
06	I	MASTER	BB			Ø3		1	C	I																					
07	I																				2	7		ITEMNO							
08	I																				37	41	Ø	NOHAND							
09	I																				42	46	Ø	OHAND							
10	I																				47	51	Ø	NORD							
11	I																														
12	I																														

Figure 13 (Part 1 of 2). Updating and Deleting Records in the Indexed Inventory File

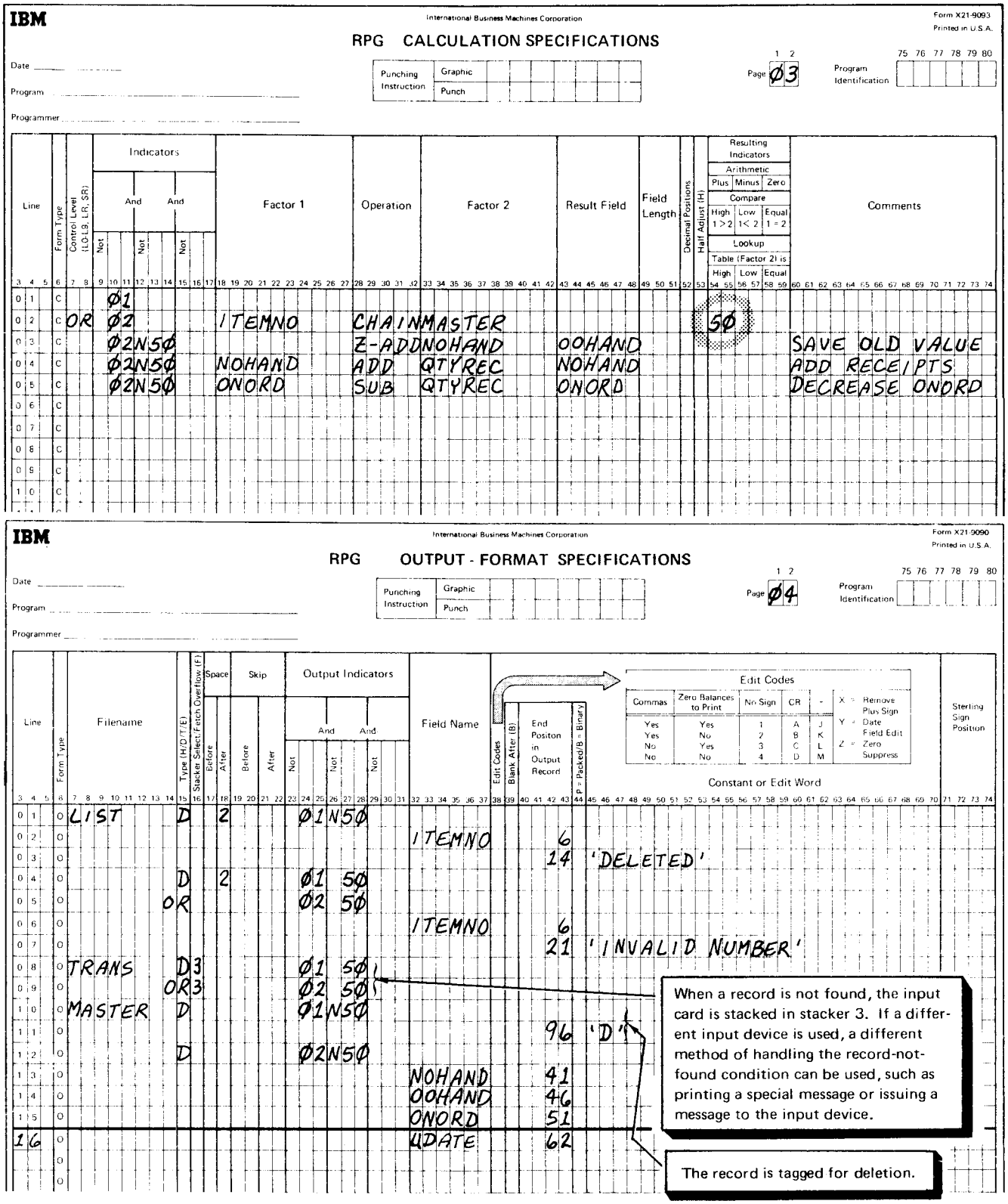


Figure 13 (Part 2 of 2). Updating and Deleting Records in the Indexed Inventory File

Updating Records Sequentially within Limits

See Chapter 4 in this manual for information on updating records sequentially within limits.

Adding Records

After a file is created, it is often necessary to add records to the file. For an indexed file, records can be added that contain keys which are: above the highest key presently in the file; below the lowest key presently in the file; or in between keys presently in the file. In any case, the added records are placed at the end of the records presently in the file. The index entry for each added record is written at the end of the current entries in the index area. After all records are added, the index is automatically sorted by the system.

If many records are to be added to the file, the time required for the index sort can be decreased by allocating a special work file This requires no special RPG II coding but does require a special OCL statement. See the section concerning use of OCL in the manuals listed in the *Preface* under *OCL References* for additional information and an example of this option. For a further discussion of indexed file performance considerations, see *IBM System/3 Disk Concepts and Planning Guide*, GC21-7571.

You can add records to a file in one of two ways:

1. Randomly by key.
2. Sequentially by key.

Adding Records Randomly by Key Using Chaining

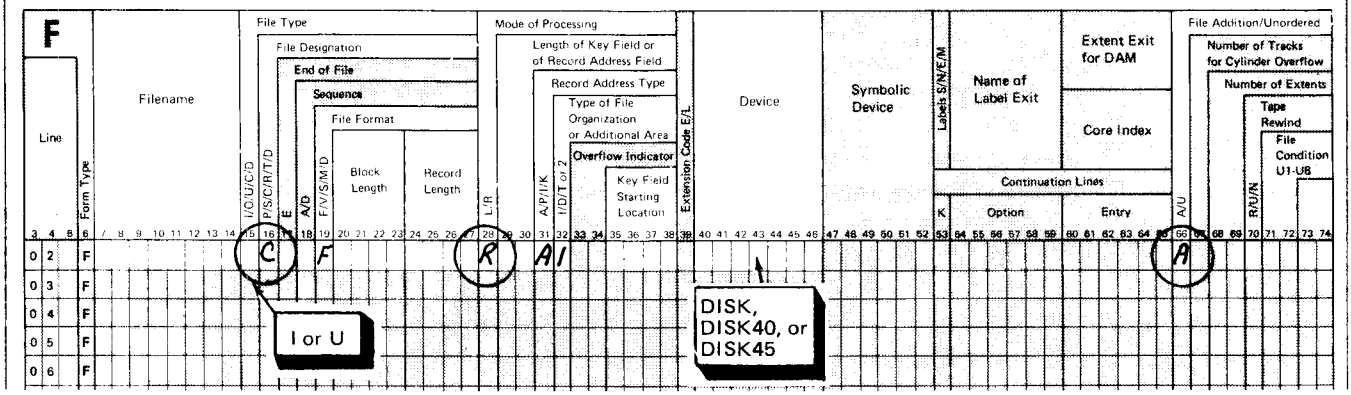
Records can be added randomly by key to an indexed file in two ways, either with or without chaining.

Chaining means matching the record key of the record to be added with the keys of the index. This matching is done as a check to ensure that the record to be added is not a duplicate of a record already in the file. The data used as a record key in the chain operation can be either a field in an input record or it can be created in your program.

When adding records randomly by key using the chaining method, entries must be made on the File Description, Calculation, and Output-Format sheets.

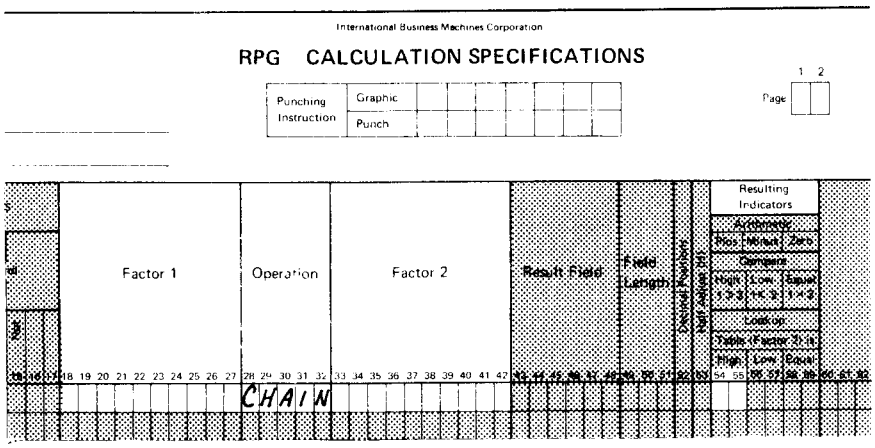
The special File Description entries are shown below:

File Description Specification



Column 16 must contain a C to indicate the file is a chained file. An R in column 28 indicates the file is processed randomly by key. Column 66 contains an A, indicating records are added to the file. The File Type entry (column 15) may be either I or U. The I entry indicates the file will be read, but the file will not be updated. The U entry indicates the file will be read and existing records will be updated.

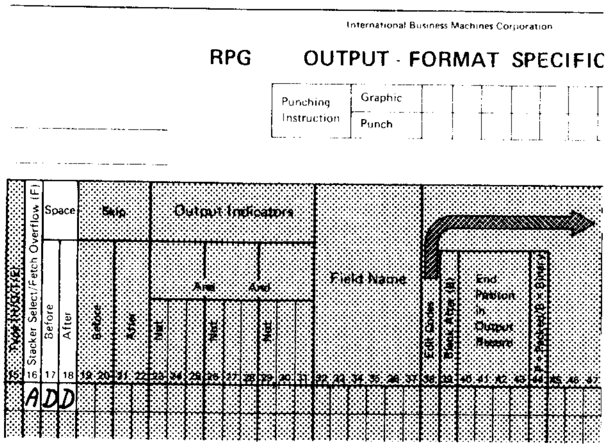
The following entries are needed on the Calculation sheet to describe the chain operation:



Factor 1 must be the name of the field to be used in the search for a match in the index, CHAIN must be entered as the operation, and Factor 2 must contain the name of the file.

A resulting indicator should be specified in columns 54 and 55 to indicate whether the record to be added is a duplicate of a record already in the file. The record key is used to find the index entry that contains the same key. If a duplicate is not found, the record can be added to the file. Your program can check for duplicates by checking the specified resulting indicator. If the indicator is off, do not add the record because it is a duplicate; if the indicator is on, the record can be added.

The entry on the Output-Format sheet is an ADD in columns 16 through 18:

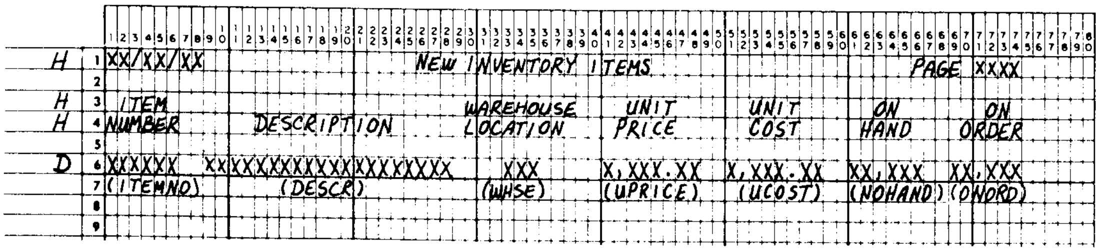


This entry tells the system that the fields defined on the following specification lines constitute the record to be added to the file specified in columns 7 through 14.

Example of Adding Records Randomly by Key Using Chaining

Suppose you want to add new inventory items to the indexed inventory file created in the *Example of Creating an Indexed File.* The new records are not in sequence. New record keys may be lower, between, or higher than keys presently in the file.

Input and output records will be in the same format as the records used to create the file. A printed report (Figure 14) will list all the new records added to the file.




```
06/18/70                  NEW INVENTORY ITEMS                      PAGE     1

 ITEM                            WAREHOUSE    UNIT       UNIT       ON       ON
NUMBER        DESCRIPTION        LOCATION     PRICE      COST      HAND     ORDER

413010  CH001 BOX 100A FLUSH       768         4.90       4.00       10        6
412146  CH143 BREAKER 15A          913          .89        .59      100       75
411126  1500 TWIN SOCKET B         493         1.12        .97      500      325
```

Figure 14. Report of New Items Added to the Inventory File

Figure 15 shows the RPG II coding necessary to add records to the inventory file and print the report.

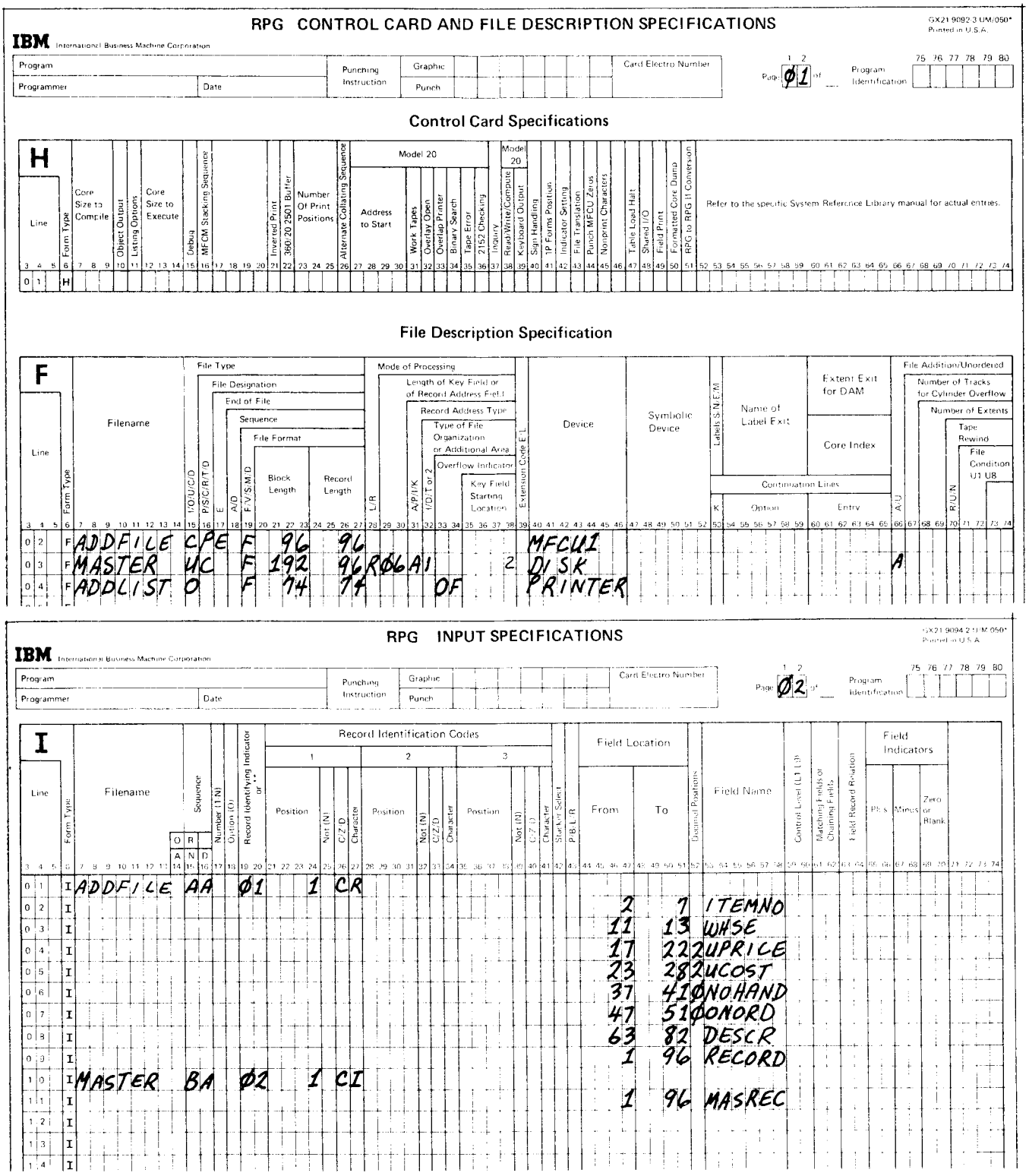
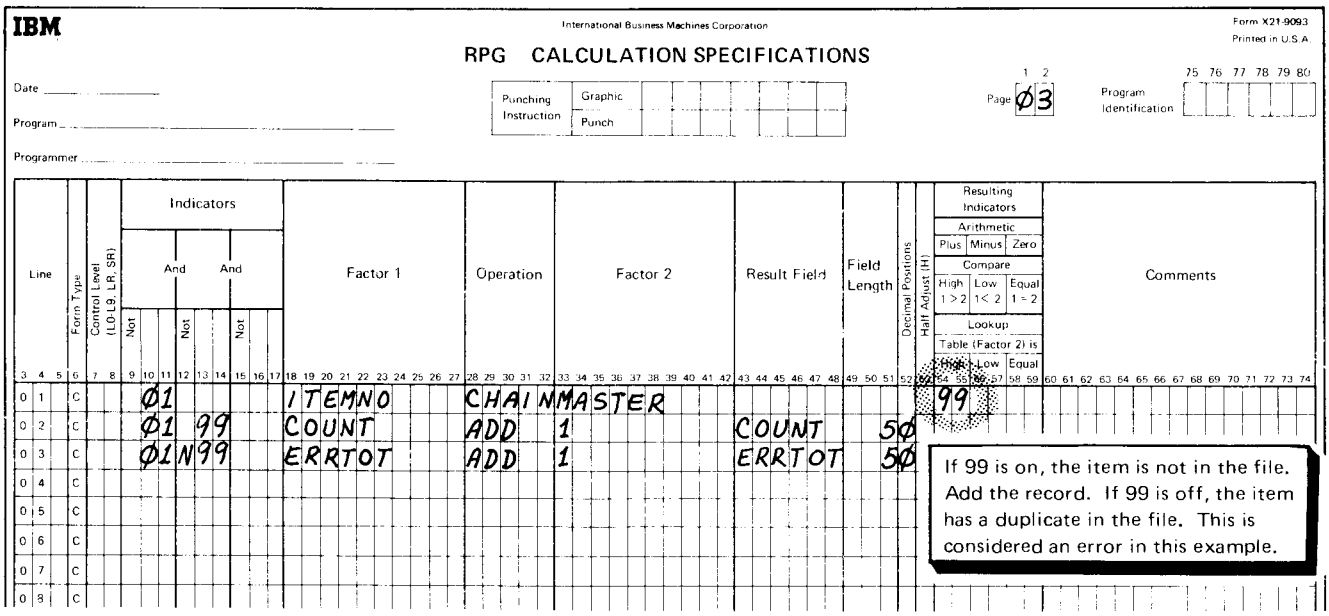


Figure 15 (Part 1 of 3). Adding Records to the Indexed Inventory File



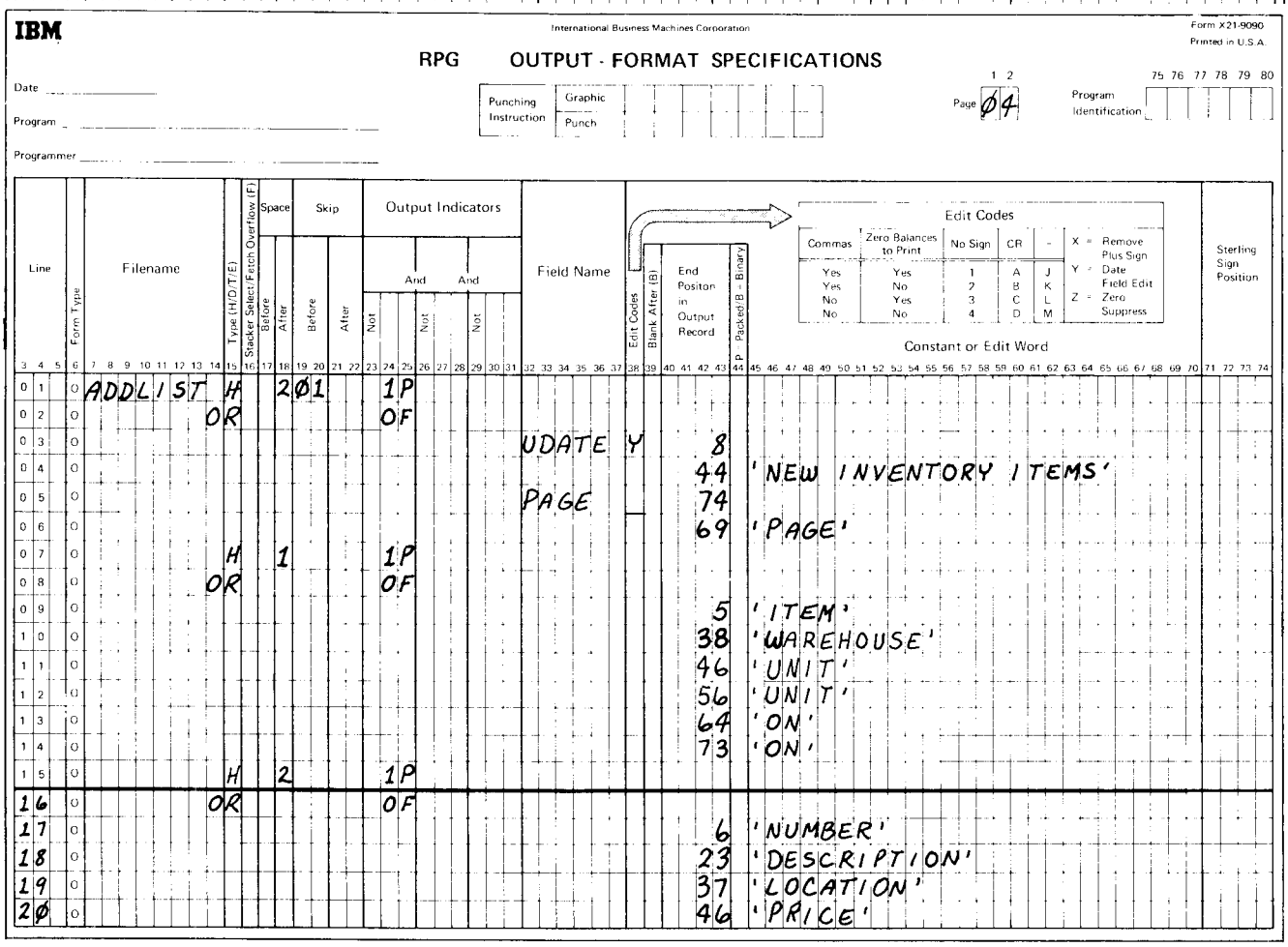


Figure 15 (Part 2 of 3). Adding Records to the Indexed Inventory File

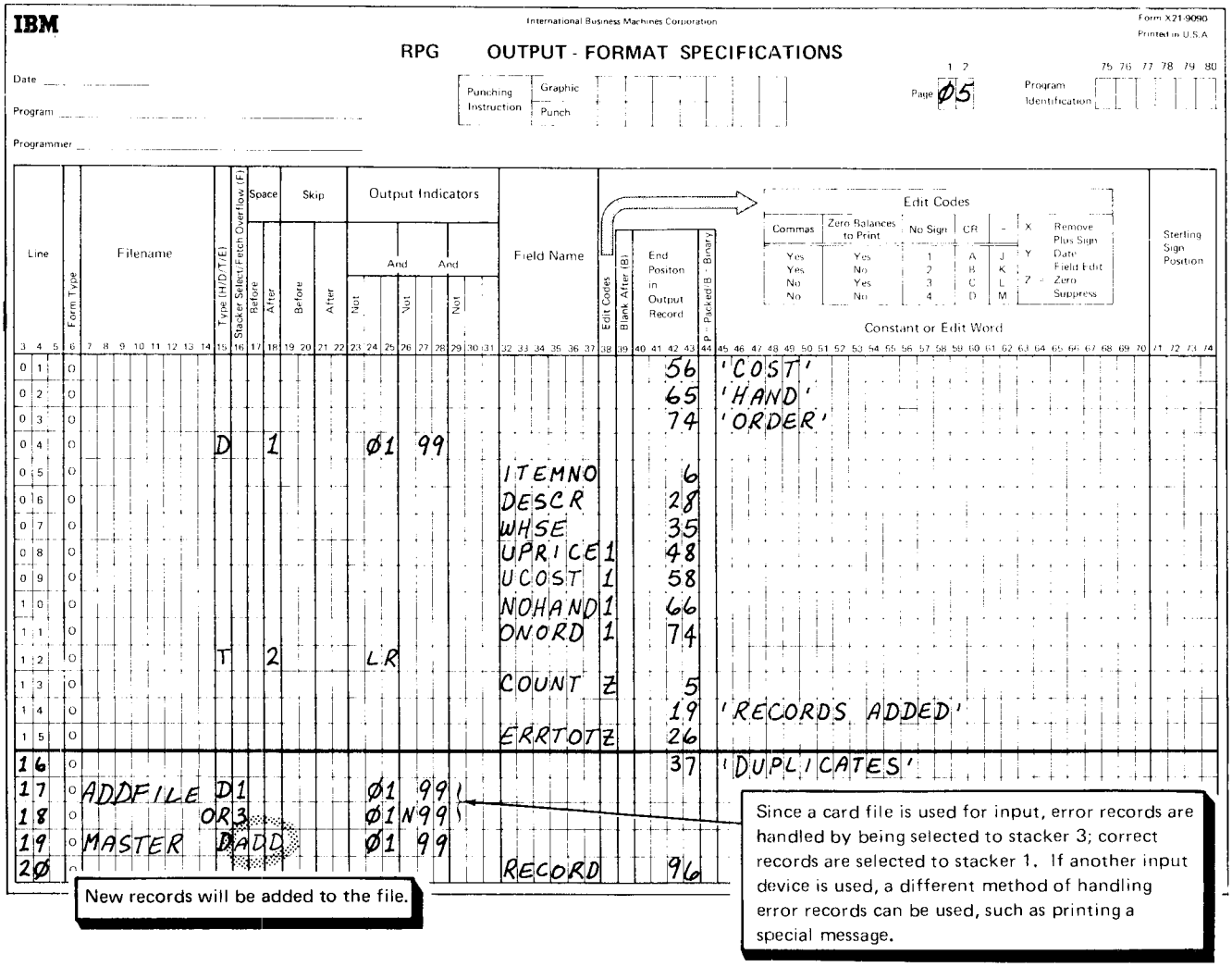


Figure 15 (Part 3 of 3). Adding Records to the Indexed Inventory File

Adding Records Randomly by Key Without Chaining

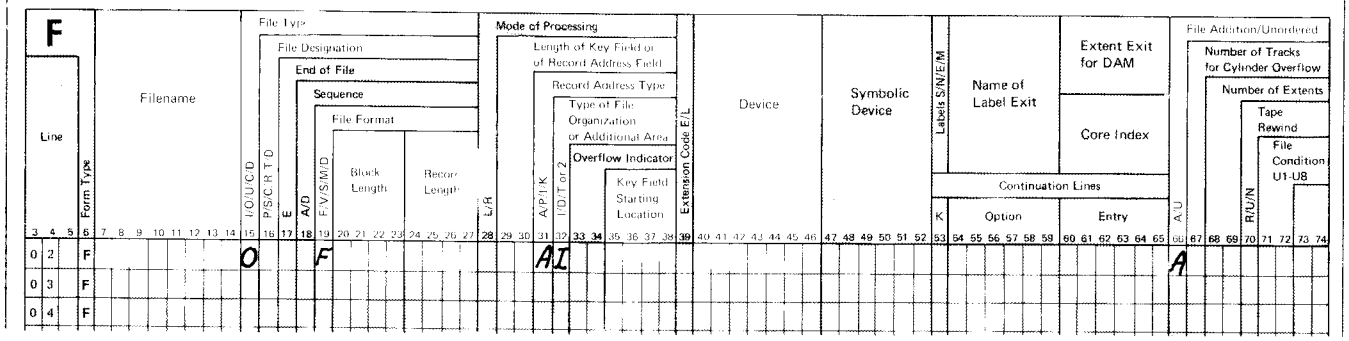
Chaining is not necessary if no checking is to be done by the program for duplicate records and the file is not being read or updated. (The system will check for duplicate records and return a halt if duplicates exist, even if chaining is not done.) In this method, the indexed file is defined as an output file and records to be added are read from a separate file (or generated in the program) and written out to the indexed file. The added records are placed at the end of the file. After the program has finished, the index is sorted into key field sequence. Records added in this manner may:

1. Contain keys that are above the highest presently in the file. In this case, the records constitute an extension of the file.

2. Contain keys that are either lower than the lowest presently in the file, or fall between those already in the file.

The File Description entries characteristic of this method of adding records to an indexed file are:

File Description Specification



For an example of this kind of add, see *Adding Records, Example of Adding Records to the End of the File* in Chapter 1. The coding used in that example applies if the preceding File Description entries are used for the disk file. Use 06 for length of key field (columns 29 and 30); enter 3 in column 38 for key field starting location.

Adding Records Sequentially by Key

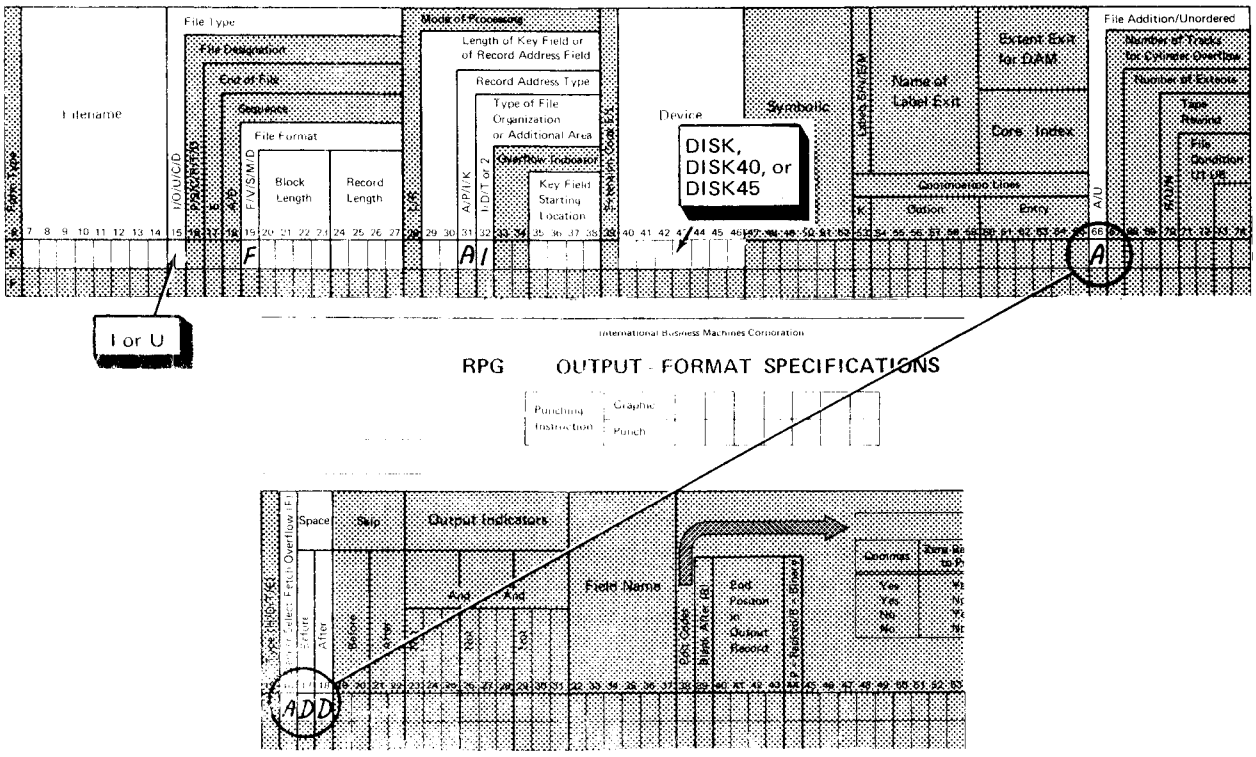
Records added to an indexed file sequentially by key (either single volume or multiple volumes) must either:

1. Be added between existing records (that is, the key of the record being added must be lower than the last key retrieved and higher than the preceding key) or

2. Have keys that are higher than existing keys in the file (that is, the indexed file is at end of file).

The unshaded columns on the File Description and Output-Format sheets below indicate entries that must be made to add records sequentially by key. Some of the File Description entries are also used to create an indexed file; these entries are discussed under *Creating an Indexed File,* previously in this chapter.

File Description Specifications



The File Type entry (column 15) can contain either an I or a U. The I entry indicates that the file will be read, records will be added, but no updating will be done. The U entry indicates that records will be read and updated and new records will be added.

Two special entries are needed to add records, one on the File Description sheet and one on the Output-Format sheet. These entries are circled.

The entry on the File Description sheet is an A in column 66. This entry tells the system that records will be added to the file described on this line.

The entry on the Output-Format sheet is an ADD in columns 16 through 18. This entry tells the system that the fields defined on the following lines constitute the records to be added to the file specified in columns 7 through 14.

Records might be added sequentially to an indexed file in situations where the activity of the file is high and the records to be added have been previously sorted into ascending sequence by key field. Adding sequentially in this kind of situation can result in better performance than adding records randomly with chaining by allowing large blocking factors.

Example of Adding Records Sequentially by Key

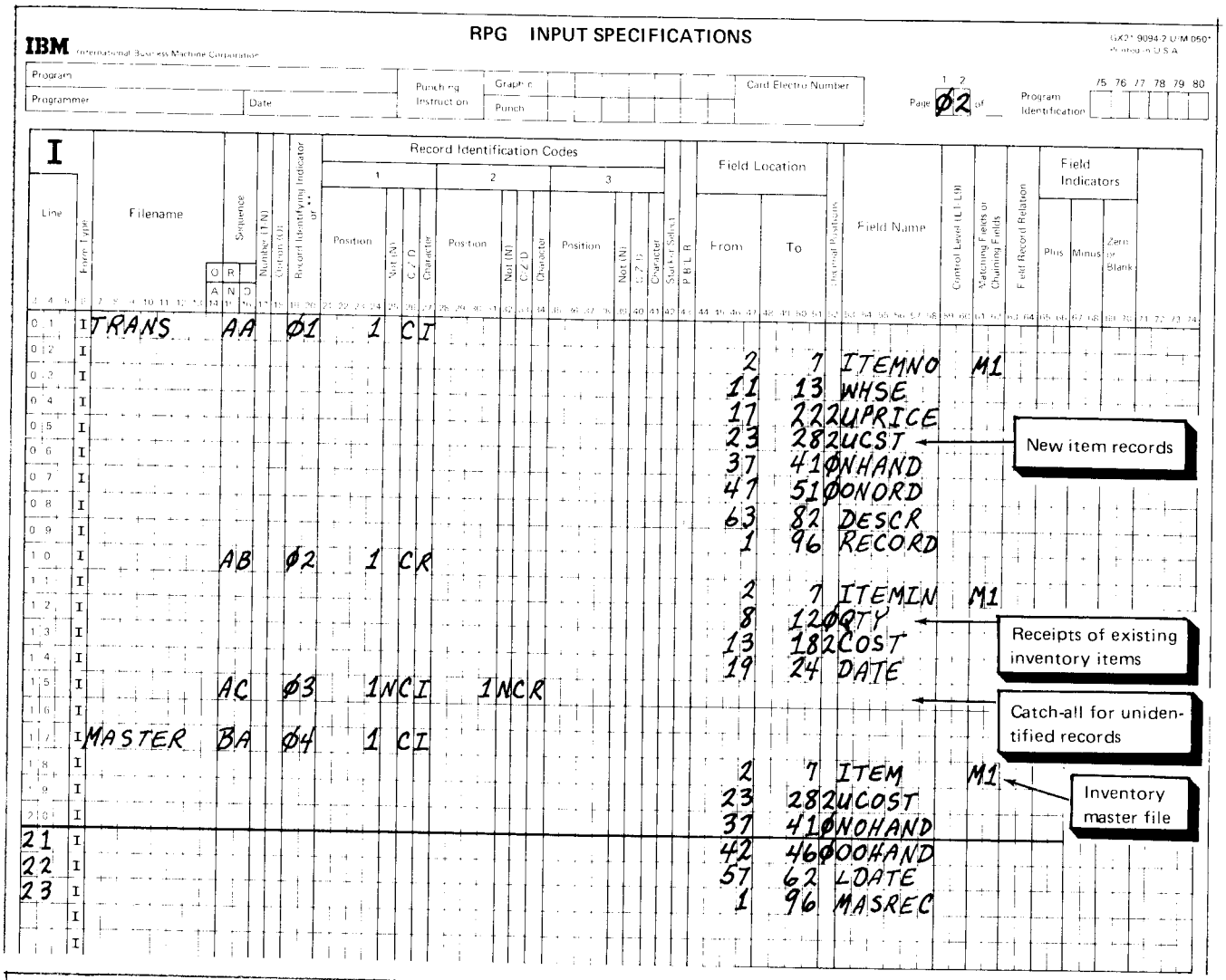
Suppose you want to add new inventory items to the indexed inventory file created in *Example of Creating an Indexed File,* previously in this chapter. The new item records are merged with records of receipts of existing items; thus, existing inventory records are updated and new records are added during the same program run. The activity of the inventory file (number of transactions against the file) is high, so the transaction file, containing new items and receipts, is organized into ascending sequence by item number (key field), enabling the file to be processed sequentially by key. Input records for new items are in the same format as the inventory records.

Figure 16 shows the coding to update the inventory file and to add new item records to the file. At the same time, a list of new inventory items is printed, similar to the list produced in the example of adding records randomly to the inventory file (Figure 13).

File Description Specification

Line	Form Type	Filename	File Type	File Designation	End of File	Sequence	File Format	Block Length	Record Length	Mode of Processing	Length of Key Field or of Record Address Field	Record Address Type	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code	Device	Symbolic Device	Labels S/N/F/M	Name of Label Exit	Extent Exit for DAM	Core Index	File Addition/Unordered	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	File Condition U1-U8
			I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D			L/R	A/P/I/K	I/D-Tol 2														R/U/N	
																			K	Continuation Lines: Option	Continuation Lines: Entry		A/U				
02	F	TRANS	C	P	E	A	F	96	96								MFCU1										
03	F	MASTER	U	S			F	192	96		06	AI			2		DISK						A				
04	F	REPORT	O				F	96	96				OF				PRINTER										
05	F																										
06	F																										
07	F																										
08	F																										
09	F																										
10	F																										
	F																										
	F																										

Figure 16 (Part 1 of 4) Adding Records Sequentially by Key



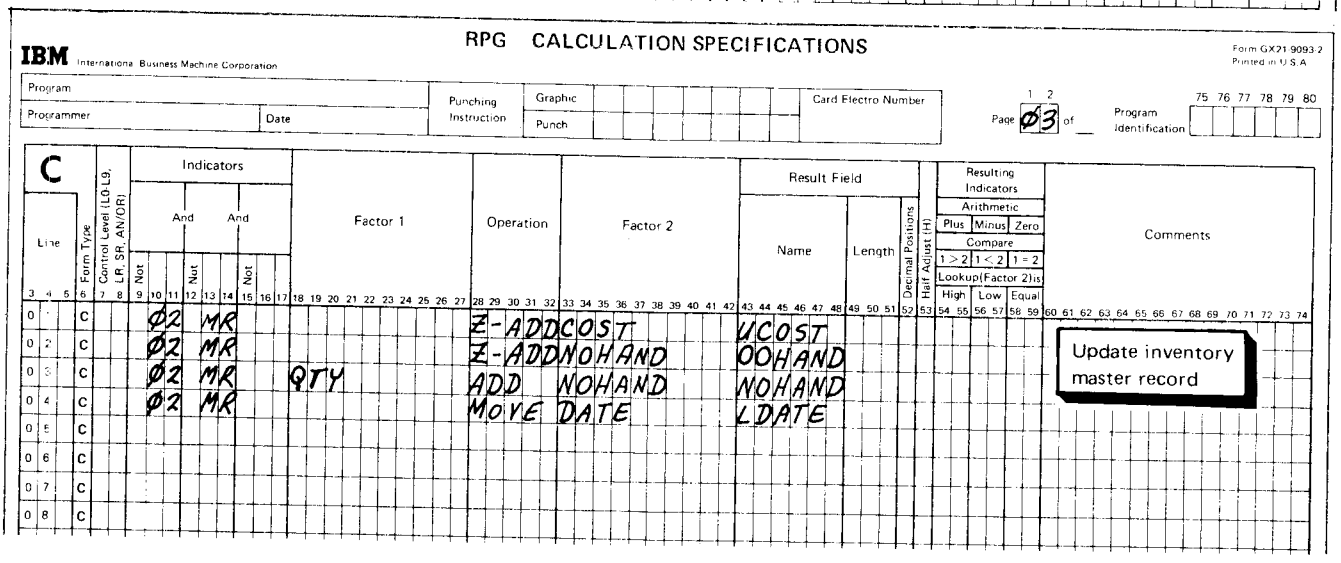


Figure 16 (Part 2 of 4) Adding Records Sequentially by Key

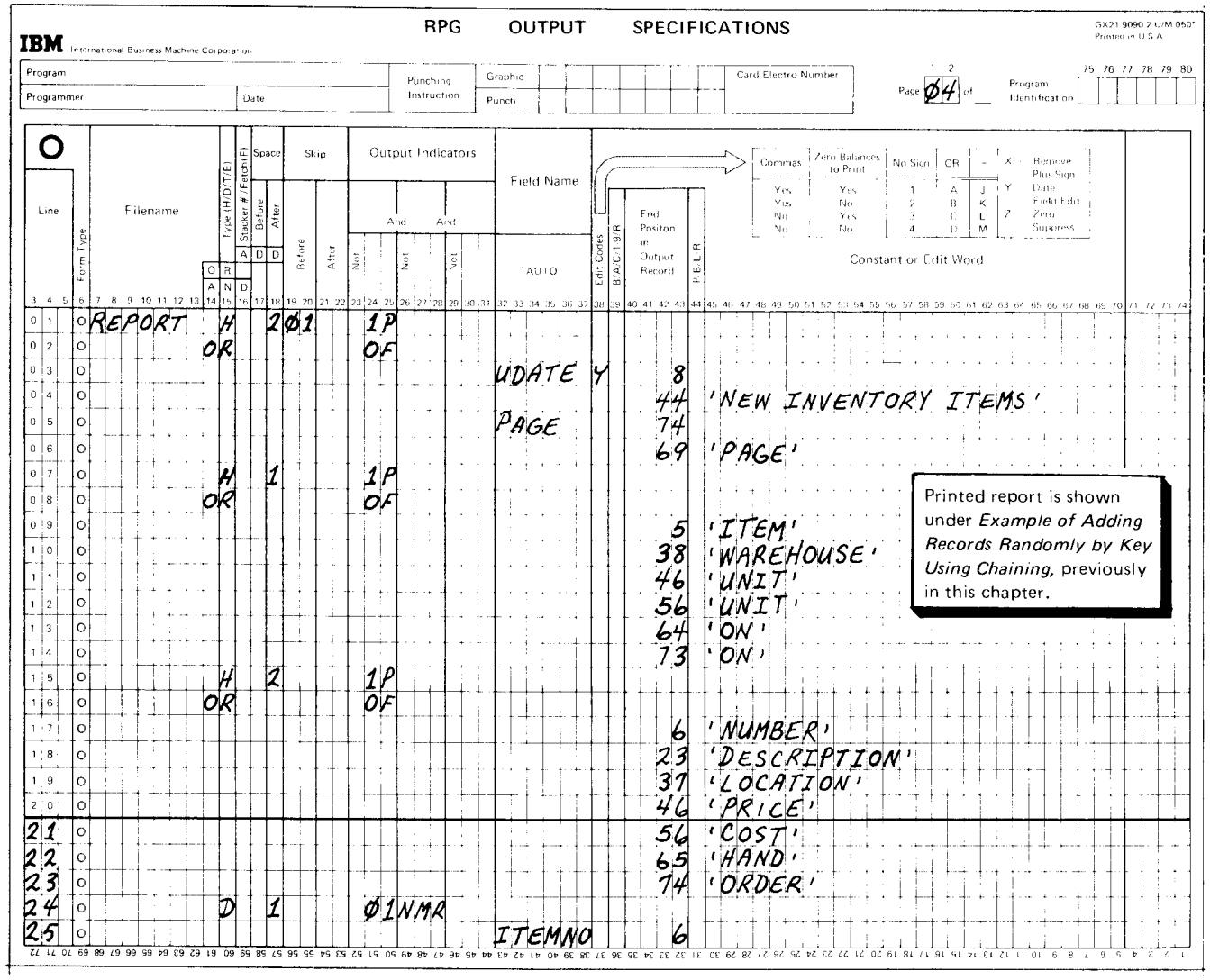


Figure 16 (Part 3 of 4) Adding Records Sequentially by Key

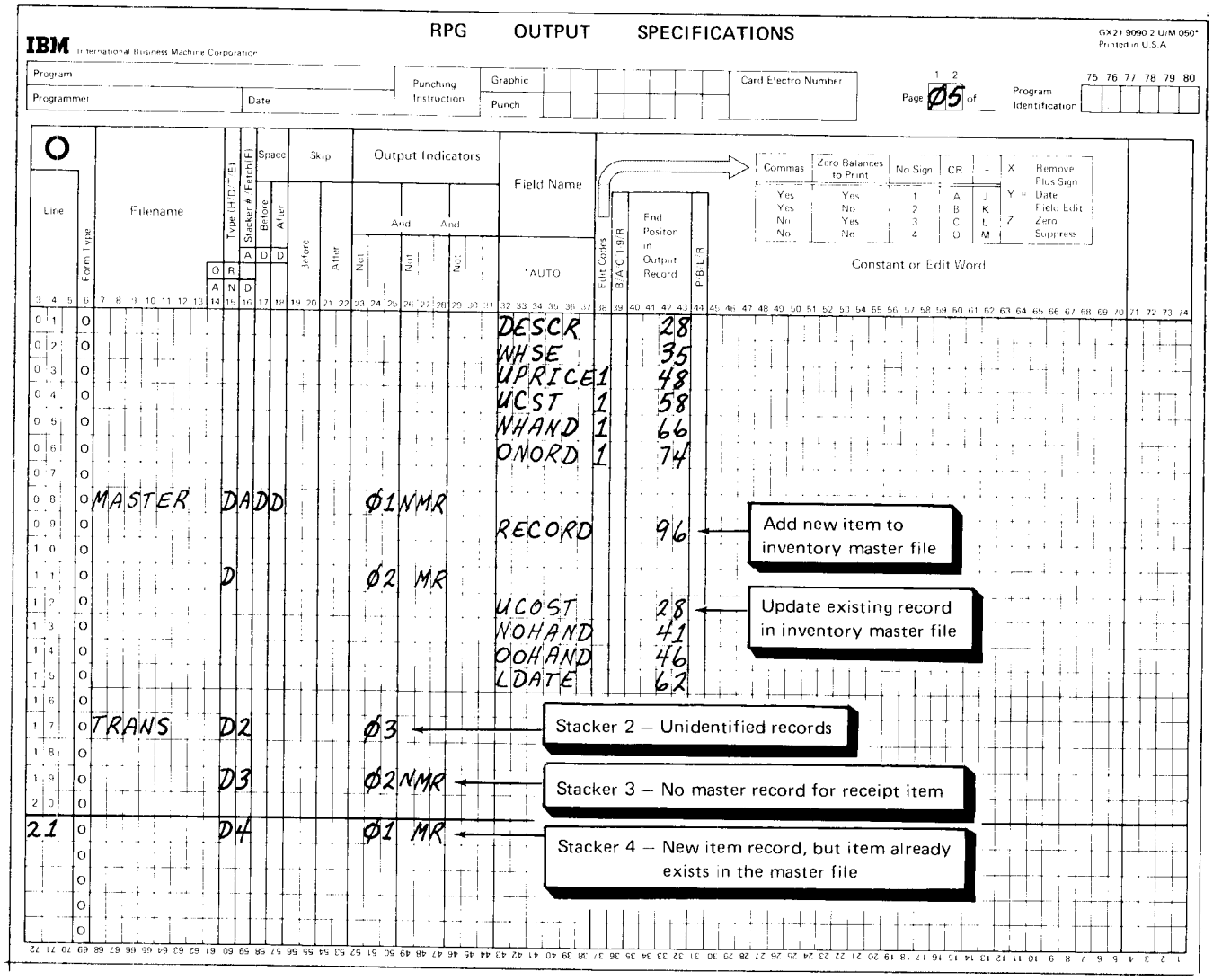


Figure 16 (Part 4 of 4) Adding Records Sequentially by Key

Tagging Records for Deletion

When a record becomes inactive, you probably will not want to process it with the other records. Since the record might not be physically removed from the file, however, you must identify it so it can be bypassed. One way to identify the record is to put a code, called a delete code, in a particular location in the record. This code can be any character you want. Any program which should not process deleted records can check for this code. If the code is present, the program bypasses the record. The deleted records can be physically removed from the file by reorganizing the file (see *Reorganizing a File* in this section).

Example of Tagging Records for Deletion

An example of tagging records for deletion is shown in this section under *Updating Records, Example of Updating Records Randomly by Key.*

Reorganizing a File

Reorganizing a file is similar to creating the file. Reorganization may be necessary for two reasons:

1. To increase processing efficiency.

2. To free disk space.

You can increase processing efficiency by restoring your file to its original sequence. When you add records to a file, these records are added at the end of the records already in the file; however, the keys are always in order in the index. When the file is processed sequentially by key, the disk access arm moves back and forth between the sequenced records (those originally created) and the added records. This increases processing time.

Disk space can be made available by removing inactive records during a file copy. All records with a delete code can be physically deleted from the file.

If you want your file in an ordered sequence, you can reorganize the file to place the added records in sequence with the records originally created and remove records tagged for deletion. To reorganize the file in this manner, you can use the IBM Disk Copy/Dump program. For an explanation on how to use this program, see the *IBM System/3 Model 10 Disk System Control Programming Reference Manual,* GC21-7512, the *IBM System/3 Model 15 System Control Programming Reference Manual,* GC21-5077, or the *IBM System/3 Model 6 Operation Control Language and Disk Utility Programs Reference Manual,* GC21-7516, depending on the system you have.

If you want your file in an unordered sequence and want only to delete records in the file, you can also use the IBM Disk Copy/Dump program.

When you want to change the order of the records and maintain an unordered sequence, you must determine what method can be used to produce the file in the required order. For example, if a count of activity was maintained in each master record, the file could be sorted in descending sequence on this activity field, then reloaded as an indexed file. This would place the most active records at the front of the file.

OTHER WAYS TO PROCESS INDEXED FILES

Processing an Indexed File Consecutively

An indexed file may be processed consecutively (read only) by defining the indexed file as a sequential input file in the File Description Specifications. When an indexed file is processed consecutively, the file index is bypassed and data records are read consecutively from the beginning of the file to the end, exactly as a sequential file. Indexed files may not be created, added to, or updated consecutively.

Consecutive processing of an indexed file is useful, for example, for reading records from an indexed file when the file index is unusable for some reason.

Processing an Indexed File Randomly by Relative Record Number

An indexed file may be processed randomly by relative record number if the file is an input file. The file must be described as a sequential file (that is, columns 31 and 32 must be blank) and must be described as a chained file (C in column 16 of the File Description sheet). The CHAIN operation code must be used in calculations to read records from the file. Records may not be updated, added, or written out to an indexed file using this method.

Random processing of an indexed file by relative record number can provide improved performance over random processing by key, because the file index need not be read. However, it is the user's responsibility to ensure that records in the indexed file are properly sequenced for this type of processing.

A direct file is one in which records are assigned specific record locations on disk. Direct file organization enables the program to access any record in the file without examining other records or searching an index.

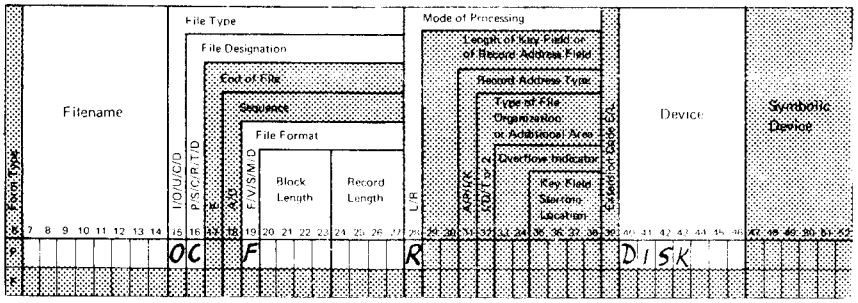
This chapter explains how to use the RPG II language to create and maintain a direct file and to retrieve records from the file. Sample jobs illustrate these functions.

To understand the sample jobs, a basic knowledge of RPG II is necessary. If you do not fully recall some of the coding used in the sample jobs, refer to the *IBM System/3 RPG II Reference Manual,* SC21-7504 (for the Model 10 Disk System or the Model 15), or the *IBM System/3 Model 6 RPG II Reference Manual,* SC21-7517, depending on the system you have.

CREATING A DIRECT FILE

To create a direct file, you must define a disk file as a chained output file. The following entries are needed on the File Description sheet to describe various characteristics of the disk file:

File Description Specifications



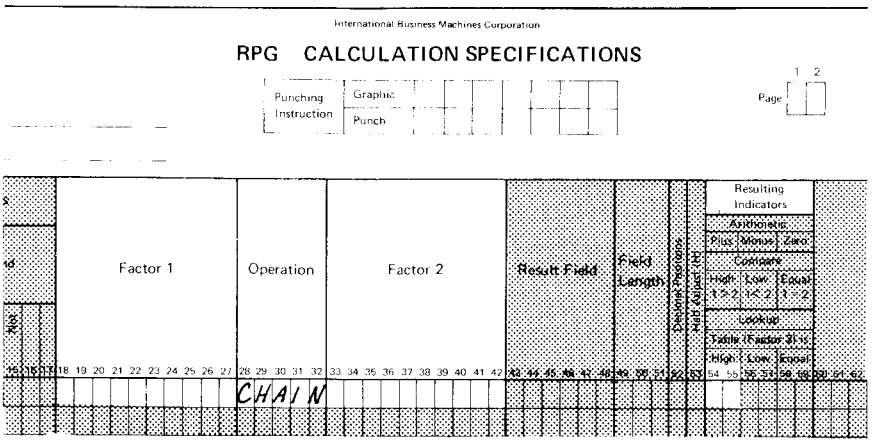
The disk filename must be entered in columns 7 through 14; column 15 must contain an O, and column 16 a C, indicating that the file is a chained output file.

All records in the file must be the same length. Thus, column 19 must contain an F to specify that the record length is fixed.

A number equal to or a multiple of the disk record length must be entered in columns 20 through 23. This entry determines the size of the input/output area allocated by RPG II. For an explanation on block length calculation, see the *IBM System/3 Disk Concepts and Planning Guide,* GC21-7571. If you want block length calculated for you by RPG II, assign a block length equal to the record length. By blocking disk records you can increase the input/output efficiency of your program by reducing the number of accesses. You must be sure, however, that enough main storage is available for your input/output area.

Columns 24 through 27 must contain the length of the disk record. Column 28 contains an R to indicate that random processing is to take place. Whenever a disk file is being described, DISK (Model 6, 10, 15), DISK40 (Model 12 or 15), or DISK45 (Model 10, 12, or 15) is required in columns 40 through 46.

Relative record numbers are always used with the CHAIN operation code in your program to make the corresponding record locations in a direct file available for loading. The data used as a relative record number in the chain operation can be either a field in an input record or it can be created in your program. To use the chain operation, you must make the following entries on the Calculation sheet:



Factor 1 must contain either the name of the field containing the relative record number or the relative record number itself. CHAIN must be entered as the operation, and Factor 2 must contain the name of the file to be loaded.

A resulting indicator should be specified in columns 54 and 55 with the CHAIN operation. If the record is not found (that is, the record location does not exist in the file) the indicator specified in columns 54 and 55 is turned on. This situation can occur either when the relative record number is higher than the highest record location in the file or when the relative record number is invalid for some other reason. If an indicator is not specified in columns 54 and 55 and the record is not found, the program halts.

When a direct file is loaded as a chained output file, disk system management clears the entire file area to blanks before records are loaded. Thus, if a record is not loaded, the space reserved for it remains blank. Programs written to access records in the direct file should check each record for blanks before attempting to process it, since the record may not have been previously loaded.

The method you use to write data records on the file depends on whether you must check for synonyms among those records. Synonyms are two or more records whose control fields yield the same relative record number. For more information on synonyms, see the *IBM System/3 Disk Concepts and Planning Guide,* GC21-7571.

Creating a Direct File without Synonyms

A direct file can be created without synonyms when the relative record number either corresponds to a field containing sequential values or is derived in such a way that no synonyms are produced.

When you do not have synonyms, you can load records into a direct file in a single pass. You do this by specifying a chained output file and writing records in the file by means of the chain operation. Record locations cannot be inspected before they are filled with data. If a synonym is encountered, it is written over the previous record. The previous record is lost.

Example of Creating a Direct File without Synonyms

Suppose you want to create a customer file on disk. The following are significant characteristics of the file:

- Customer numbers are assigned on a sequential basis; new customers are assigned the next higher number.
- Deletions from the file are few.
- The file will be used to process invoices, orders, and cash payments in a random manner.
- The file must allow direct inquiry to any customer's record.
- The file has low activity; for example, out of 5000 customer records, only 100 invoices are processed per day.

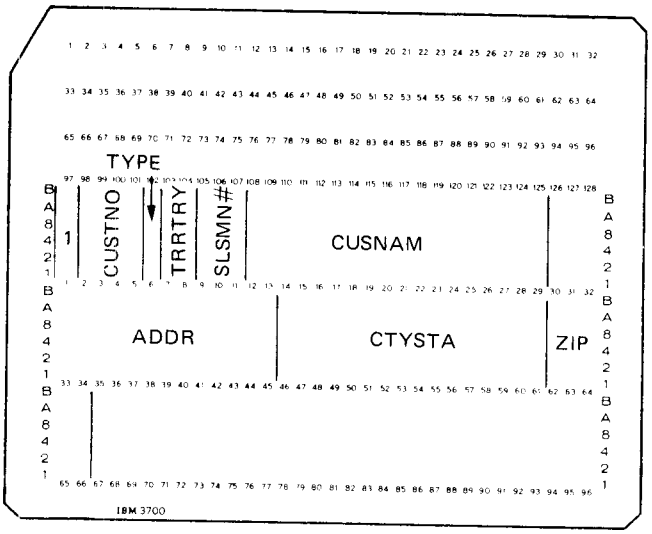
You need both the direct and consecutive processing capabilities offered by indexed and direct file organizations. Because the customer numbers are assigned consecutively, synonym records are not a consideration. For this reason, and because there will be few deletions from the file creating wasted space, direct file organization provides maximum flexibility and access speed.

Your first step, then, is to create the direct file. The record format shown in Figure 16 satisfies your information needs. Additional fields in the record will contain information to be used in specific jobs, such as customer payments, invoicing, and sales analysis. (Various applications using the customer file are described later in this chapter.)

The file is created from data on input records (Figure 17). The customer number (CUSTNO) is used as the relative record number to chain to the direct file. The customer data from the input records is then written on disk. Each record written on disk is also printed in the report shown in Figure 18.

Figure 19 shows the RPG II coding necessary to create the direct customer file.

Input Record



Note: The input file need not be a card file. See *How to Use This Manual, Note Concerning Examples* at the beginning of this manual.

Key

IDCODE	=	Identification code
CUSTNO	=	Customer number
TYPE	=	Type
TRRTRY	=	Territory
SLSMN#	=	Salesman number
CUSNAM	=	Customer name
ADDR	=	Customer street address
CTYSTA	=	City and state
ZIP	=	Zip code
DELETE	=	Output record delete code

Output Record

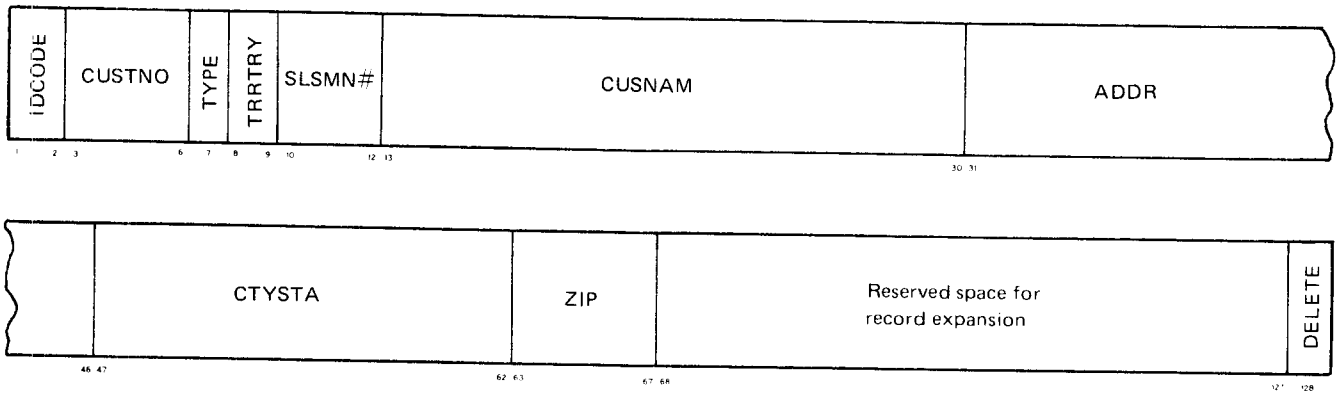
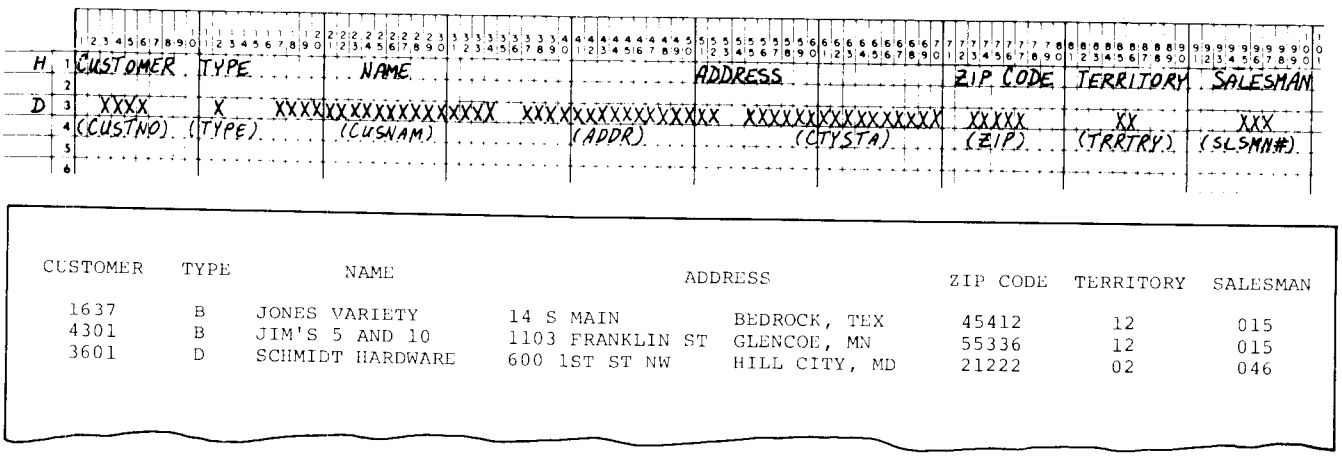
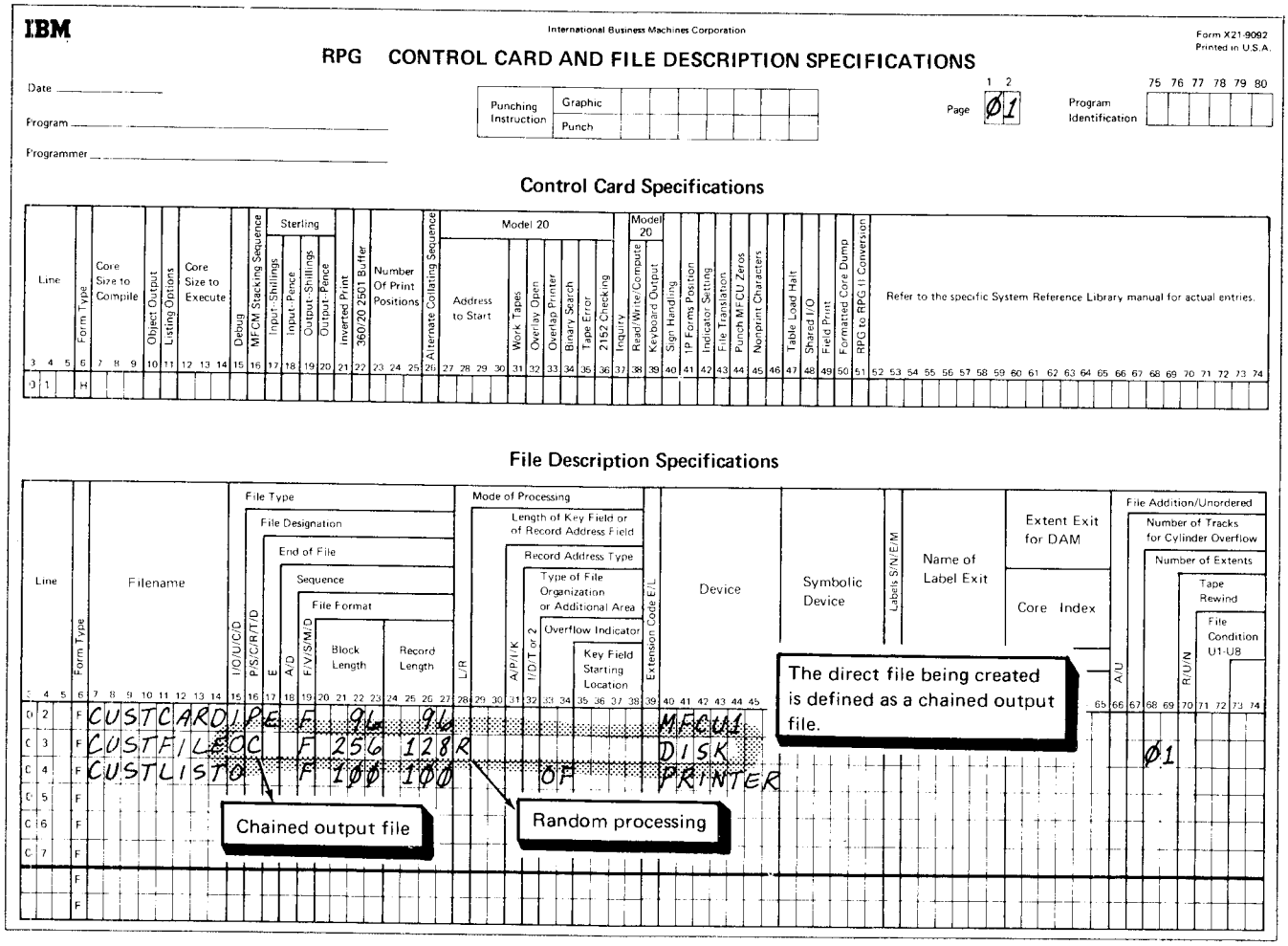


Figure 17. Input Record and Output Record Formats



CUSTOMER	TYPE	NAME	ADDRESS		ZIP CODE	TERRITORY	SALESMAN
1637	B	JONES VARIETY	14 S MAIN	BEDROCK, TEX	45412	12	015
4301	B	JIM'S 5 AND 10	1103 FRANKLIN ST	GLENCOE, MN	55336	12	015
3601	D	SCHMIDT HARDWARE	600 1ST ST NW	HILL CITY, MD	21222	02	046

Figure 18. Report of Customer Records



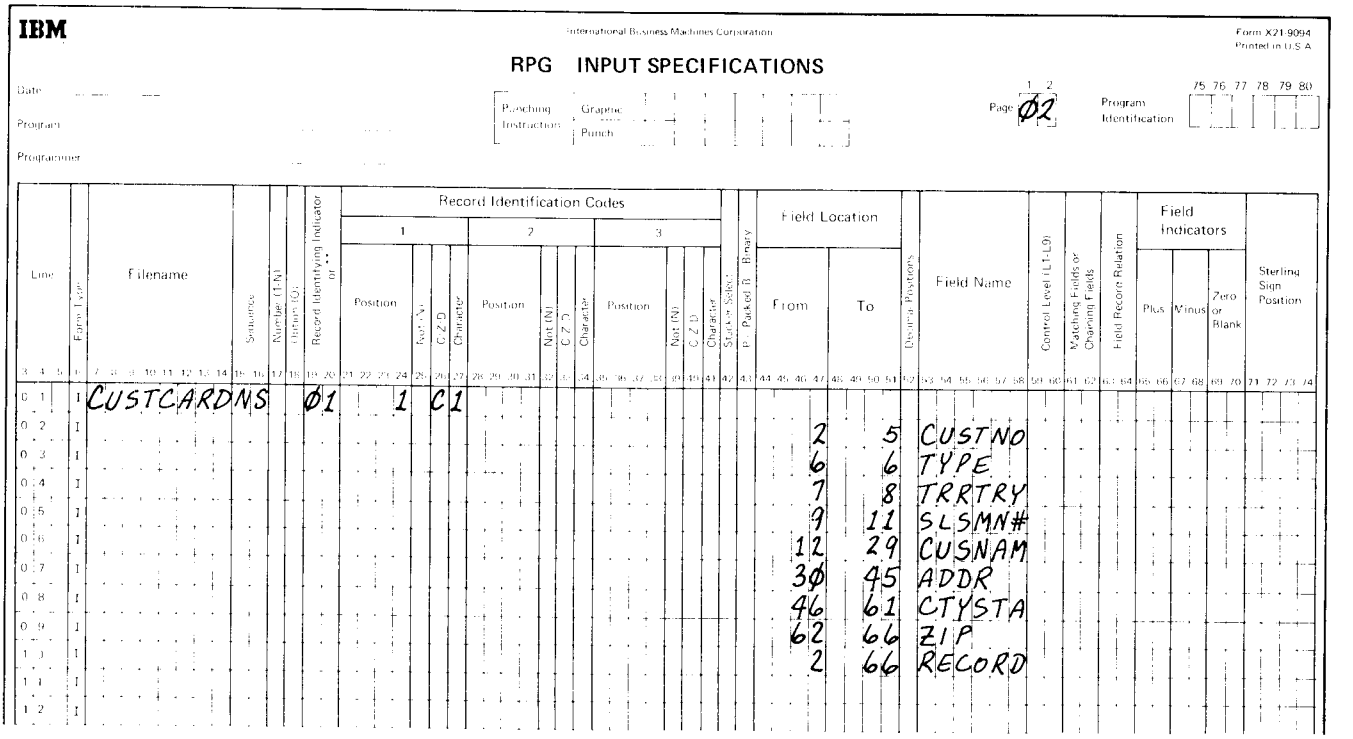
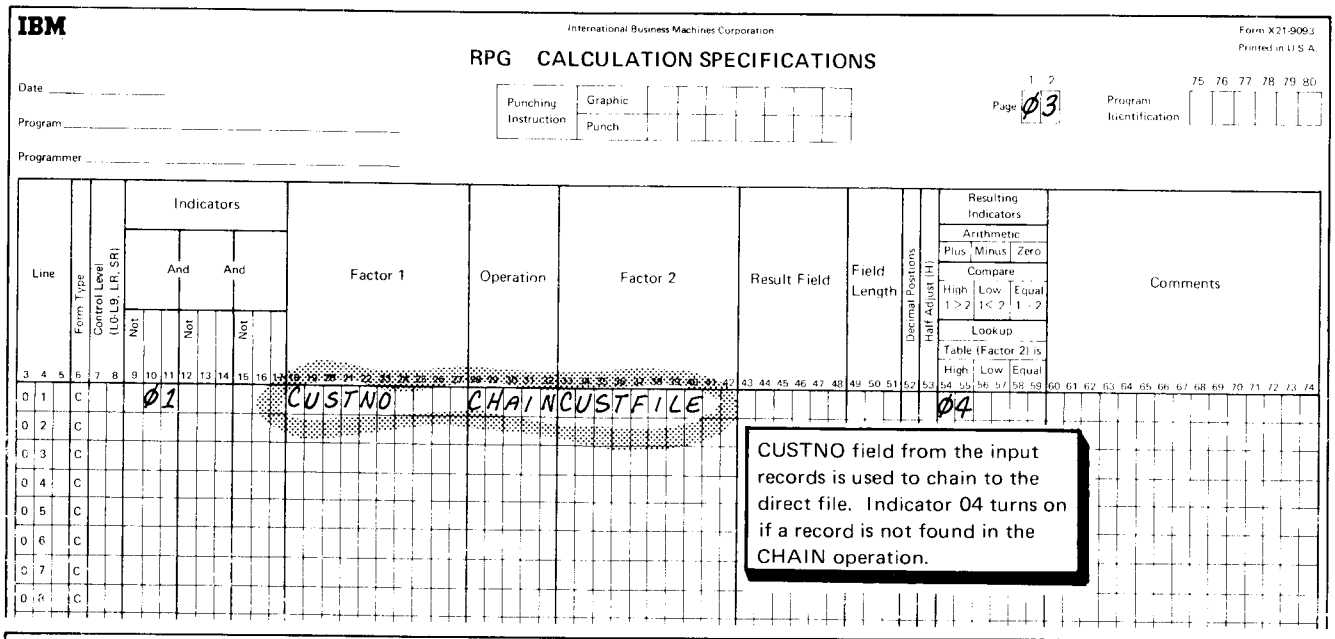


Figure 19 (Part 1 of 2). Creating a Direct Customer File



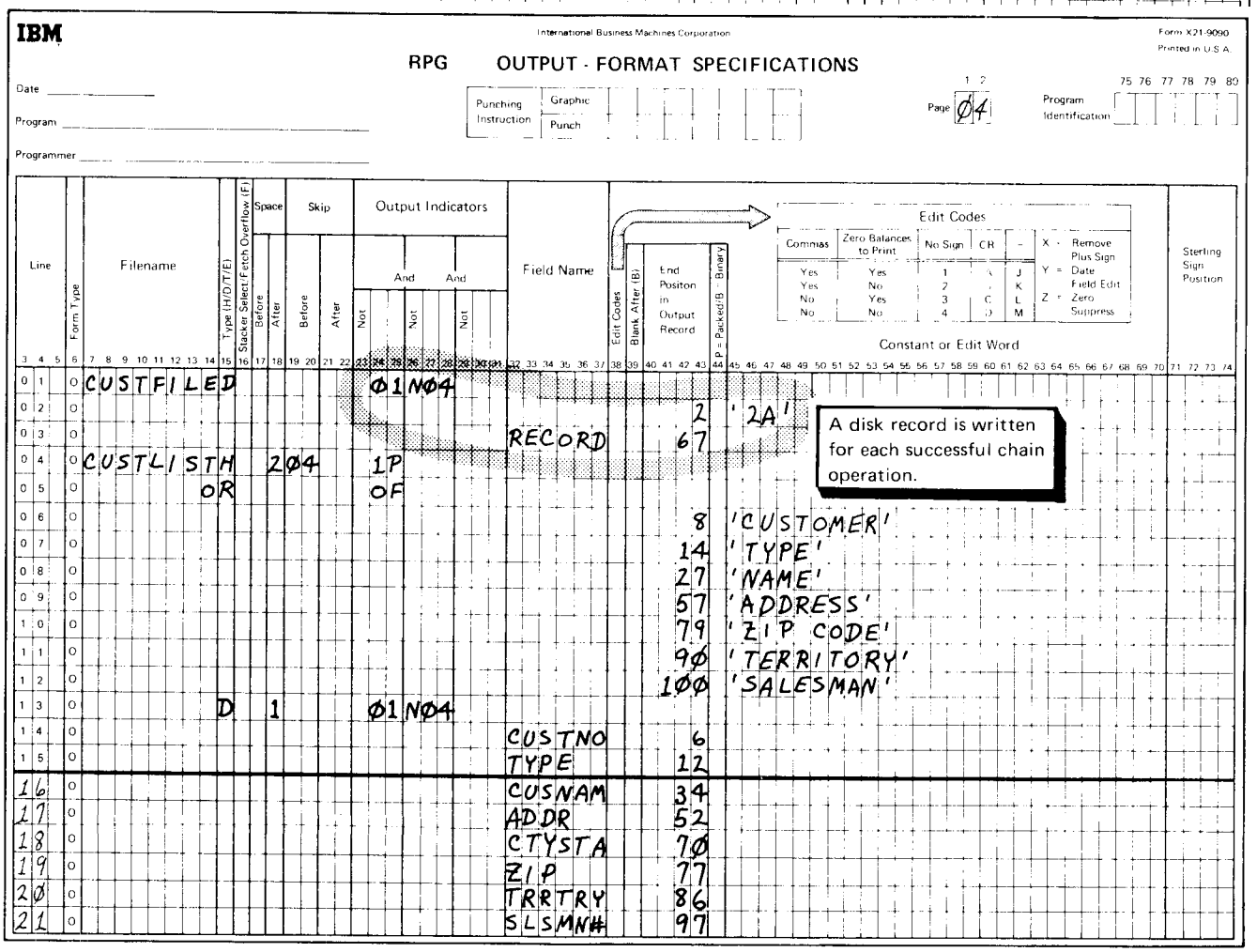


Figure 19 (Part 2 of 2). Creating a Direct Customer File

Creating a Direct File with Synonyms

If you have synonyms, you can create a direct file by using more than one job to load records into the file. The exact method you use depends on how you plan to handle synonym records. Your first job must define the disk file as a direct output file, causing disk system management to clear it to blanks. Since blank record locations indicate an unused location, clearing the file to blanks is necessary to ensure that no unwanted characters are left in the file area. Once the file has been cleared, one or more subsequent jobs can be run using the update function to read record locations and check for synonyms while loading the file.

Figure 20 shows a method of defining a direct file and clearing it to blanks. In this method, the input record file from which the direct file is created is placed in the MFCU; another type of input, such as a sequential disk file, could be used.

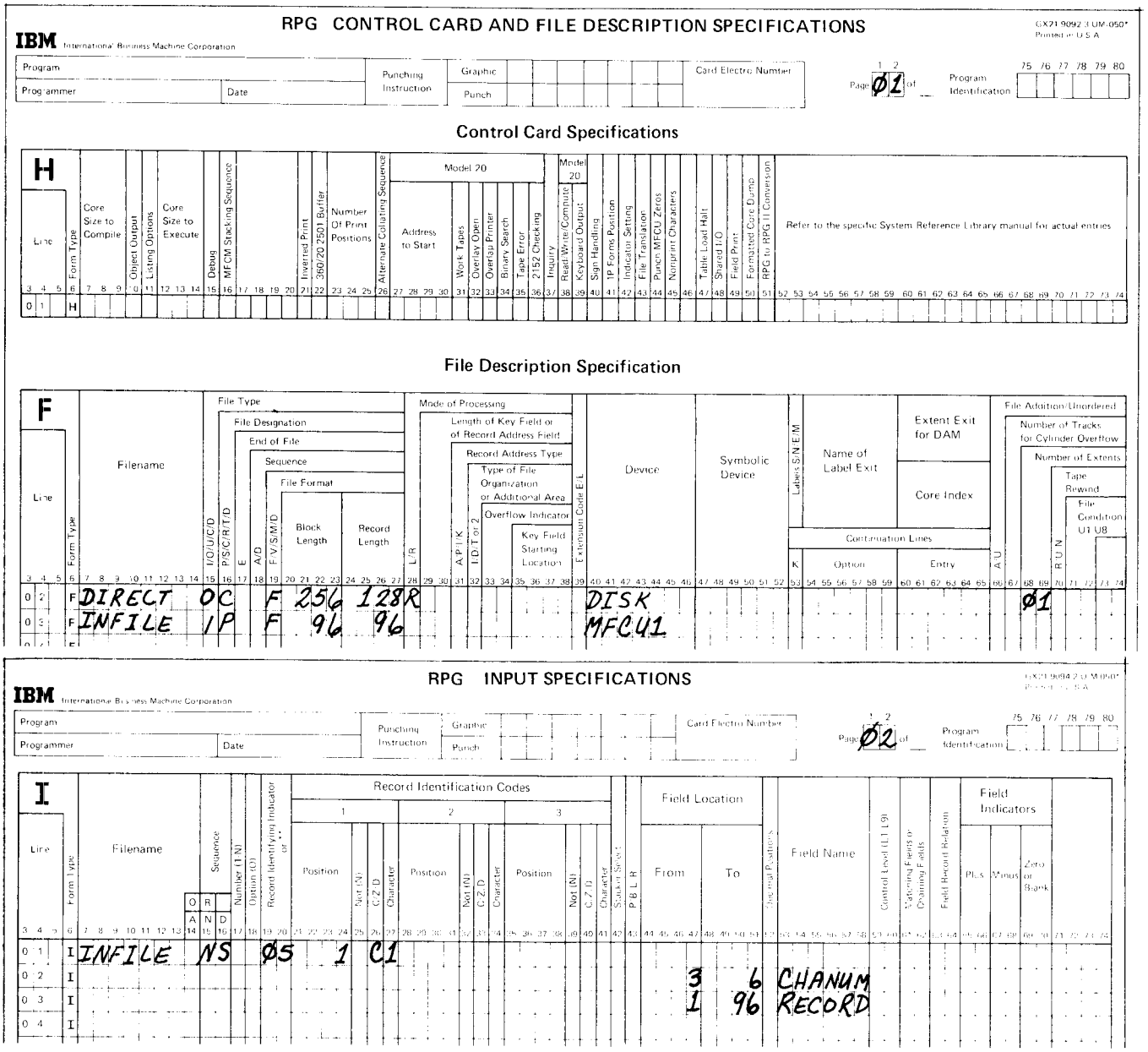


Figure 20 (Part 1 of 2). Defining a Direct File and Loading Only the First Record

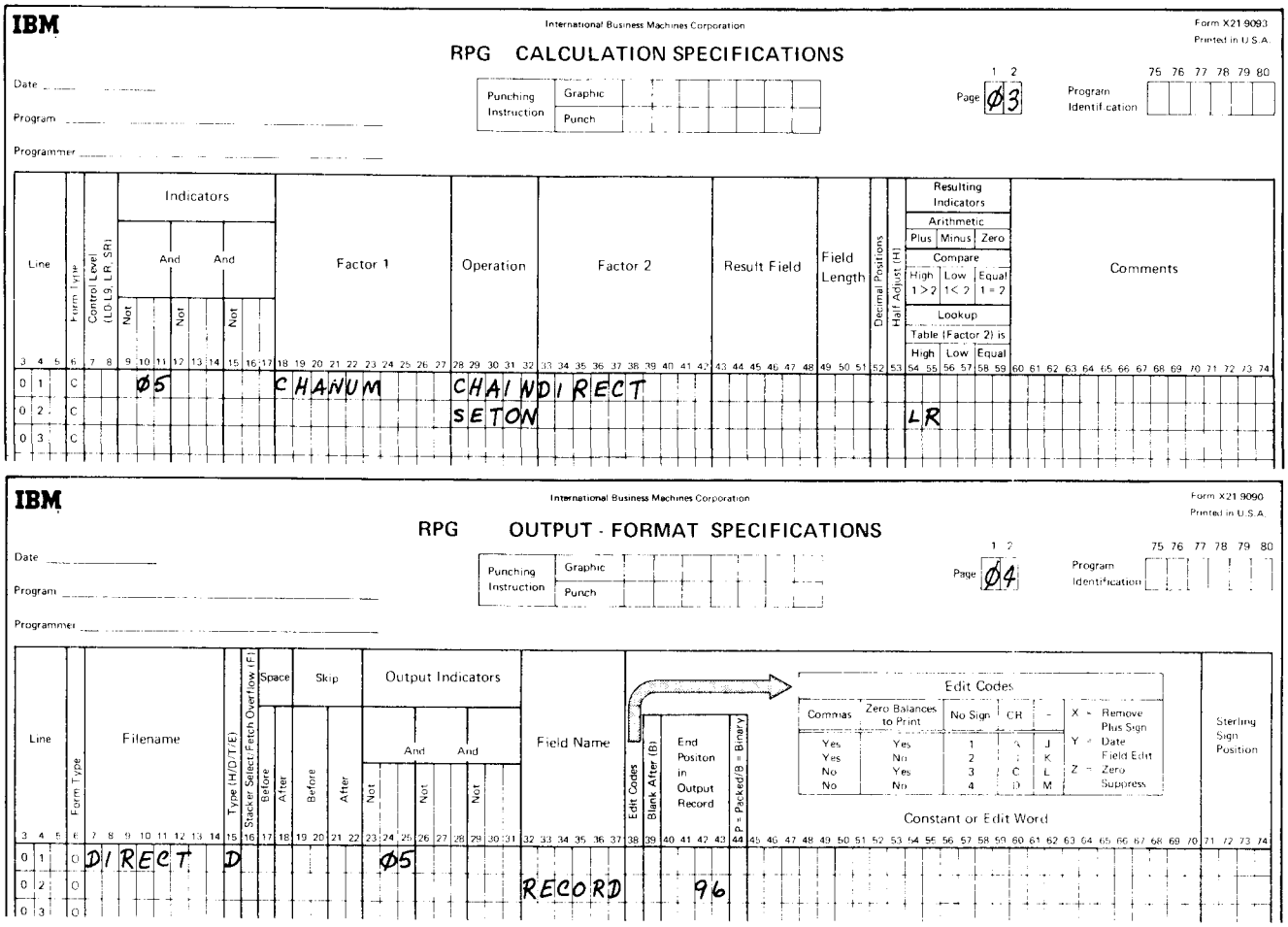


Figure 20 (Part 2 of 2). Defining a Direct File and Loading Only the First Record

The disk file, which is specified as a chained output file, is cleared to blanks by disk system management after the job begins. The CHANUM field from the input file is used to chain to the corresponding location in the direct file, and the first record is placed in the file. The last record (LR) indicator is then turned on by a SETON operation, forcing the end-of-job condition. The direct file now contains a single record. This job can be immediately followed by one or more jobs which read the remaining records from the MFCU and write out the disk records, using the update function.

After your direct file is defined and cleared to blanks, different steps are required to put records into the file, depending on the method you use to handle synonyms. There are several ways to handle synonyms. Two of the most common methods are:

1. Storing all synonyms in an area of the file set aside for them.

2. Storing synonyms in unused record locations between the records in the file.

If the first method is used, all records can be placed in the direct file in a single job. That job would retrieve and check each record location before it is filled. If the location already contains a record (that is, the record to be written is a synonym), the synonym is stored in the next available location in the part of the file set aside for synonyms. Thus, all home records and synonyms are placed in the file in a single job. A home record is the first synonym record; it is stored in the record location indicated by its relative record number. The rest of the synonyms are then linked together so they can all be found by locating the home record. For more information, see the *IBM System/3 Disk Concepts and Planning Guide,* GC21-7571.

If the second method is used, two jobs are required to place home records and synonyms in the direct file. The first job loads all home records; synonyms are bypassed. The second job loads synonyms in the record locations available between home records. Both jobs are done using the update function to check each record location.

Whatever method you use to handle synonym records, you will have to devise a sequence of jobs similar to those just described. Remember:

1. You load a disk file as a direct file by specifying a chained output file.

2. In order to check for synonyms, you must use the update function. Random update with a direct file is described later in this chapter.

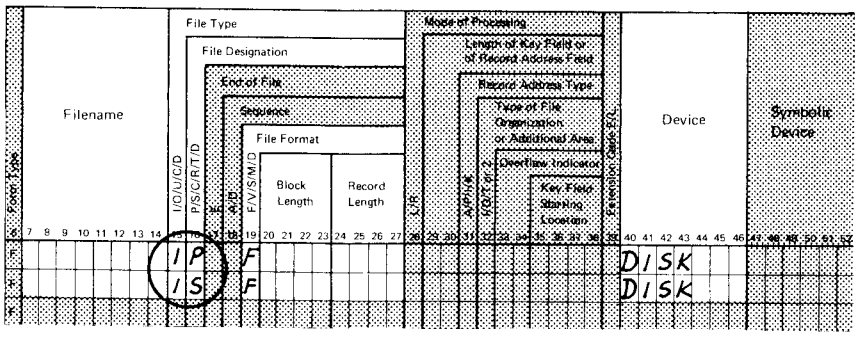
RETRIEVAL OF RECORDS IN A DIRECT FILE

Record retrieval is used when you want to get information from a record, but do not want to modify the record. You would normally use this method when you want to get data from a record to produce a report. Record retrieval can be done either consecutively, randomly by relative record number, or randomly by ADDROUT file. For a discussion of record retrieval done randomly by ADDROUT file, see Chapter 4 in this manual.

Consecutive Retrieval

Consecutive retrieval of records from a direct file requires the following entries on the File Description sheet:

File Description Specifications



Some of these entries are the same ones needed to create a direct file and were discussed in this chapter under *Creating a Direct File.*

The new entries are circled. An I is entered in column 15 to indicate that the file is an input file. Column 16 can contain either a P or an S depending on whether the file is a primary or secondary file. Because the file is an input file, it must also be defined on the Input sheet.

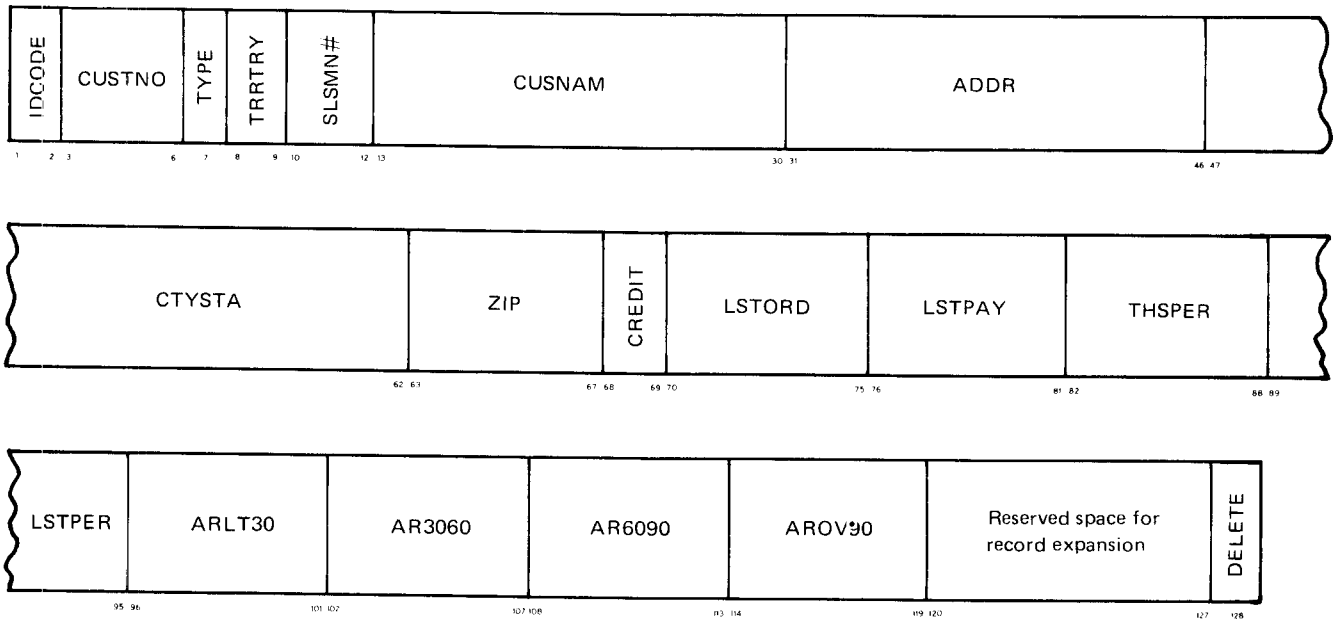
Example of Consecutive Retrieval

Suppose you want to process the direct customer file, CUSTFILE, created in the *Example of Creating a Direct File without Synonyms,* to produce a monthly report. This report lists all customers that have had no sales activity during the period. This report is analyzed by sales personnel, who then make follow-up calls. Since all the customer records will be checked and since the file is in sequence by customer number, the report is produced by consecutive processing of the direct file.

The format of the disk records in CUSTFILE is shown in Figure 21. Figure 22 shows a part of the report produced by the consecutive processing job. The report consists of fields selected from CUSTFILE and an accumulated total for accounts receivable (TOTAR).

Figure 23 shows the specification sheets necessary to consecutively retrieve records from CUSTFILE to produce REPORT1, which is a list of recently inactive customers.

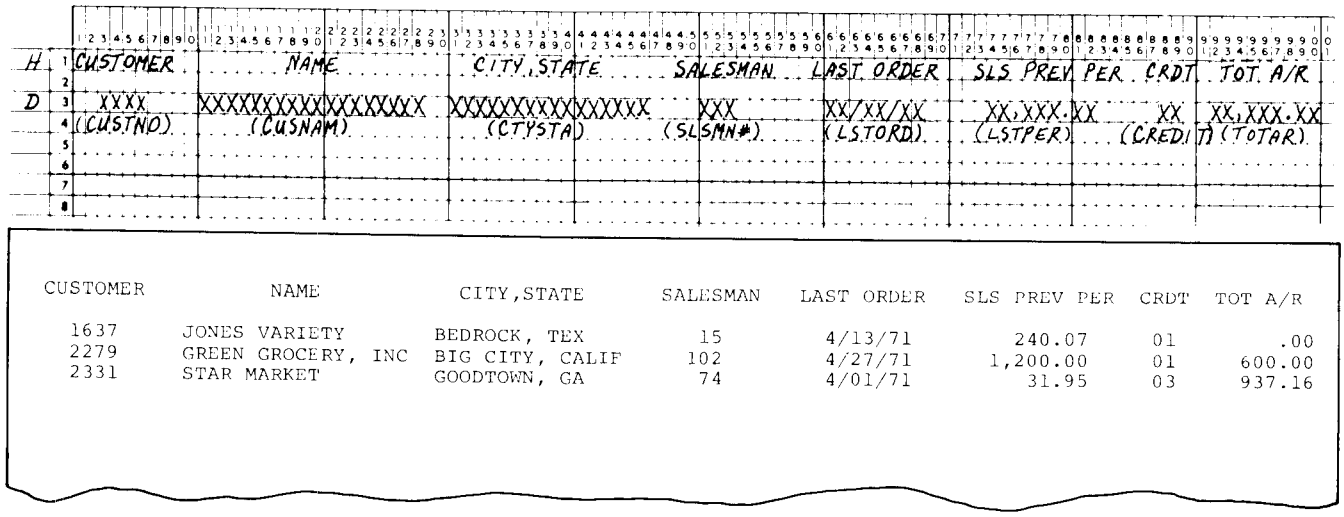
Since the direct file probably contains blank record locations and inactive records, a technique is employed on the Input sheet to bypass such records (Figure 23). If a method is not used to bypass unidentified records, the program halts when they are encountered.



Key

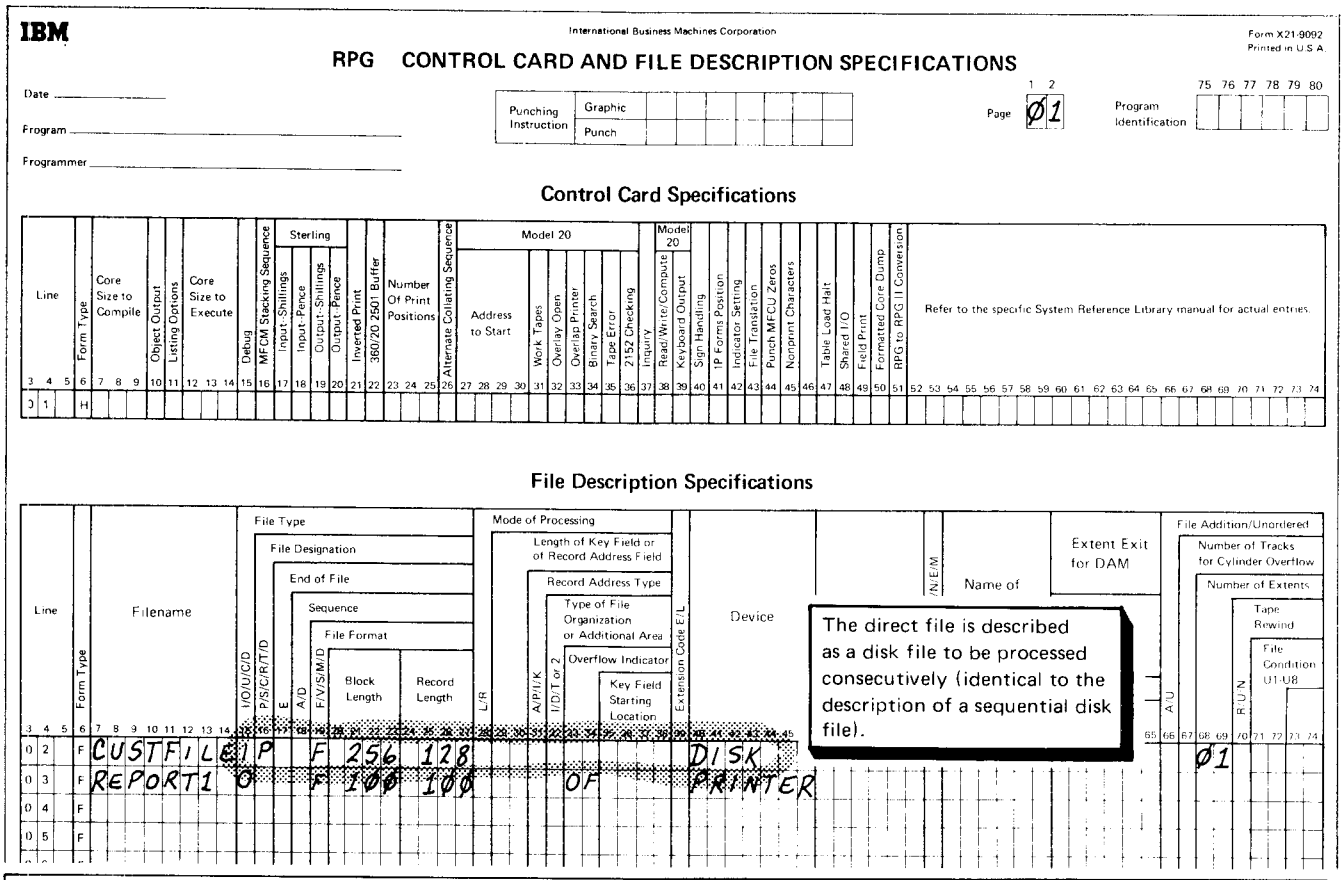
IDCODE	= Identification code	LSTORD	= Last order date
CUSTNO	= Customer number	LSTPAY	= Last pay date
TYPE	= Type	THSPER	= Charges for this period (month)
TRRTRY	= Territory	LSTPER	= Charges for last period (month)
SLSMN#	= Salesman number	ARLT30	= Accounts receivable for less than 30 days
CUSNAM	= Customer name	AR3060	= Accounts receivable for 30 to 60 days
ADDR	= Customer street address	AR6090	= Accounts receivable for 60 to 90 days
CTYSTA	= City and state	AROV90	= Accounts receivable for over 90 days
ZIP	= Zip code	DELETE	= Delete code
CREDIT	= Credit code		

Figure 21. Disk Record Format for the Direct Customer File



CUSTOMER	NAME	CITY,STATE	SALESMAN	LAST ORDER	SLS PREV PER	CRDT	TOT A/R
1637	JONES VARIETY	BEDROCK, TEX	15	4/13/71	240.07	01	.00
2279	GREEN GROCERY, INC	BIG CITY, CALIF	102	4/27/71	1,200.00	01	600.00
2331	STAR MARKET	GOODTOWN, GA	74	4/01/71	31.95	03	937.16

Figure 22. Report of Inactive Customers in the Direct Customer File



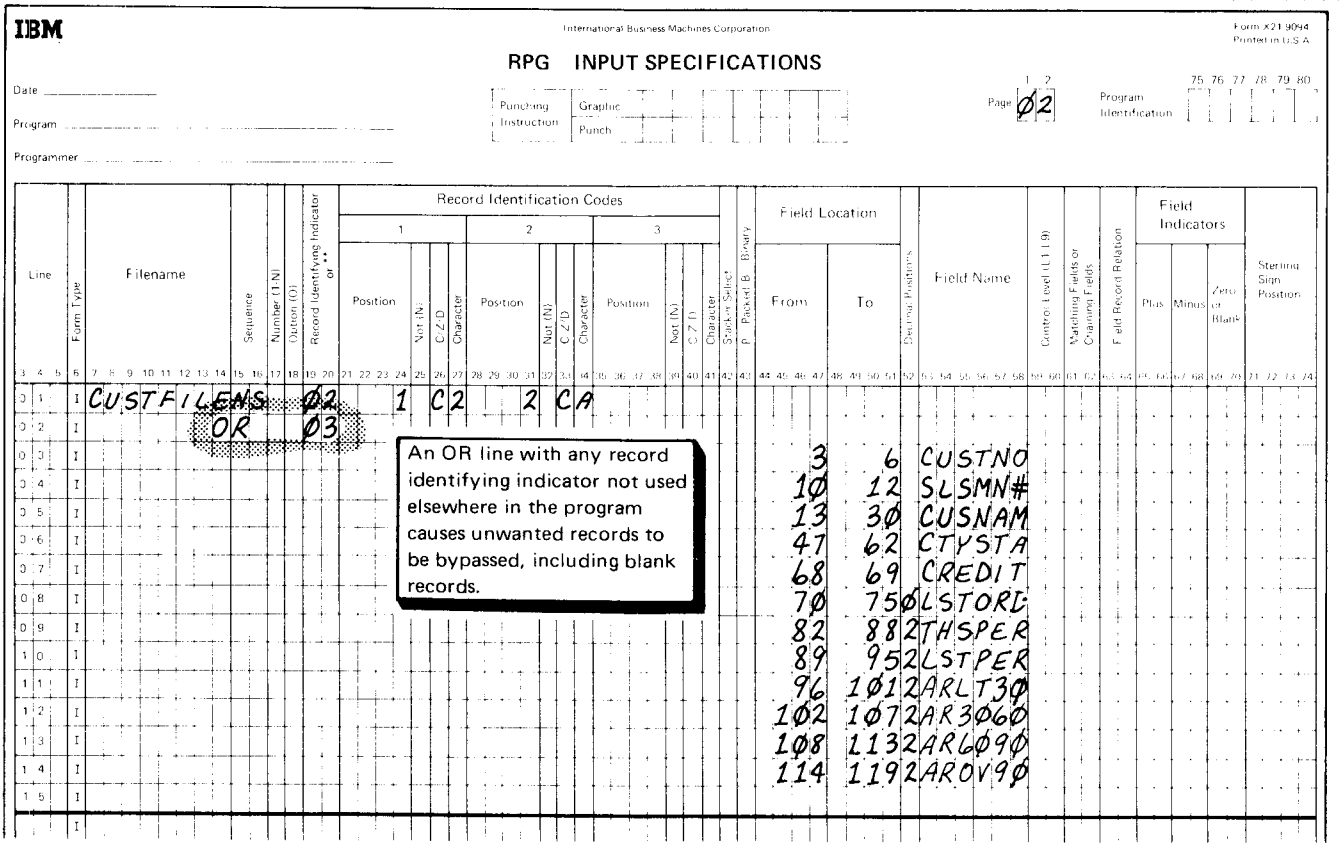
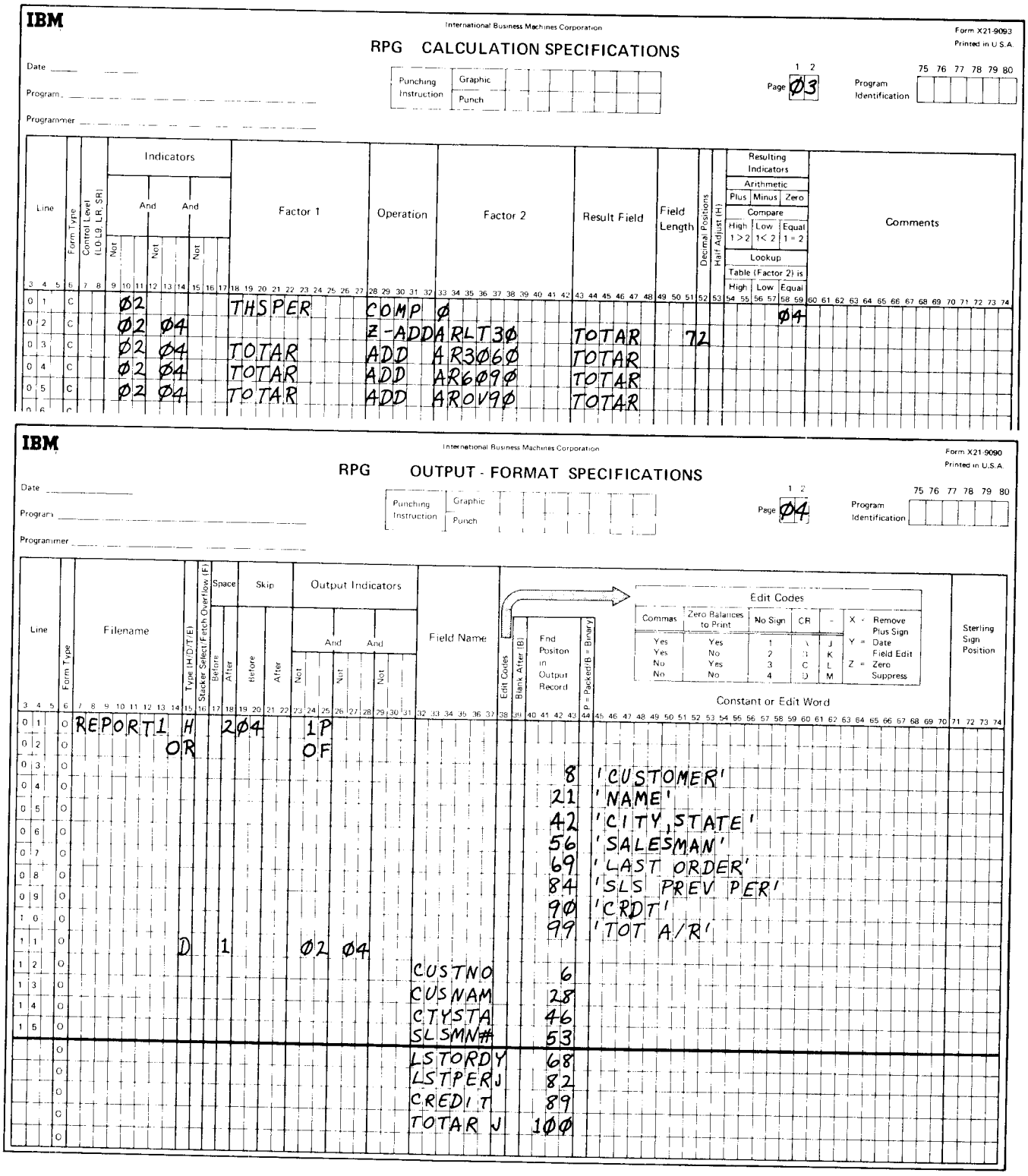


Figure 23 (Part 1 of 2). Consecutive Retrieval of Inactive Customers in the Direct Customer File



RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And/Not	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus	Arithmetic Minus	Arithmetic Zero	Compare High 1>2	Compare Low 1<2	Compare Equal 1=2	Lookup Table (Factor 2) is High	Lookup Table (Factor 2) is Low	Lookup Table (Factor 2) is Equal	Comments
01	C		Ø2	THSPER	COMP	Ø													Ø4	
02	C		Ø2 Ø4		Z-ADD	ARLT3Ø	TOTAR	72												
03	C		Ø2 Ø4	TOTAR	ADD	AR3Ø6Ø	TOTAR													
04	C		Ø2 Ø4	TOTAR	ADD	AR6Ø9Ø	TOTAR													
05	C		Ø2 Ø4	TOTAR	ADD	ARØV9Ø	TOTAR													

RPG OUTPUT - FORMAT SPECIFICATIONS

Edit Codes

Commas	Zero Balances to Print	No Sign	CR	-	
Yes	Yes	1	A	J	X = Remove Plus Sign
Yes	No	2	B	K	Y = Date Field Edit
No	Yes	3	C	L	Z = Zero Suppress
No	No	4	D	M	

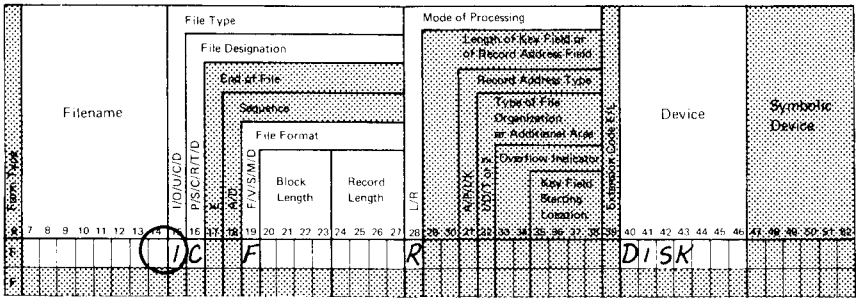
Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select/Fetch Overflow (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators: And/Not	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	P = Packed/B = Binary	Constant or Edit Word	Sterling Sign Position
01	O	REPORT1	H		2	Ø4			1P							
02	O	OR							OF							
03	O												8		'CUSTOMER'	
04	O												21		'NAME'	
05	O												42		'CITY,STATE'	
06	O												56		'SALESMAN'	
07	O												69		'LAST ORDER'	
08	O												84		'SLS PREV PER'	
09	O												9Ø		'CRDT'	
10	O												99		'TOT A/R'	
11	O		D		1				Ø2 Ø4							
12	O									CUSTNO			6			
13	O									CUSNAM			28			
14	O									CTYSTA			46			
15	O									SLSMN#			53			
	O									LSTORDY			68			
	O									LSTPERJ			82			
	O									CREDIT			89			
	O									TOTAR J			1ØØ			

Figure 23 (Part 2 of 2). Consecutive Retrieval of Inactive Customers in the Direct Customer File

Random Retrieval

Random retrieval of records from a direct file requires the following entries on the File Description sheet:

File Description Specifications

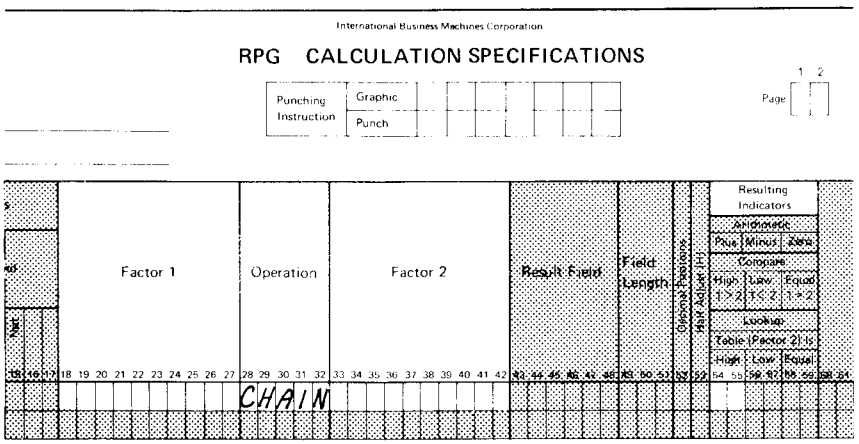


Some of these entries are the same ones needed to create a direct file and were discussed in this chapter under *Creating a Direct File.*

The circled entry, an I in column 15, is new. This entry indicates the file is an input file.

The records in the direct file to be retrieved must be further described on input specifications. On the Input sheet, a direct file being retrieved must have an alphabetic sequence entry (columns 15-16), because sequence checking cannot be done for chained files.

If the direct file being retrieved contains synonym records, calculations must be included in the program to test for synonyms and retrieve the desired record. The CHAIN operation code must be specified on the Calculation sheet to randomly retrieve records from a direct file. The entries on this sheet are the same ones discussed in this chapter under *Creating a Direct File:*



Example of Random Retrieval

Suppose the direct customer file, CUSTFILE, created in the *Example of Creating a Direct File without Synonyms* and processed consecutively in the *Example of Consecutive Retrieval,* is to be retrieved randomly. You want to make demand inquiries each day concerning customer sales and account information. Inquiries are received for records containing an I in column 1 followed by the customer number of the record to be retrieved (Figure 24). Inquiry records are read from the primary MFCU hopper in this example; substitute devices such as console/keyboard devices, can also be used. When an inquiry record is read, the customer number (CSTMER) is used as the relative record number to chain to CUSTFILE.

The format of the disk records in CUSTFILE is the same as the disk record format used in the consecutive retrieval example.

If a record that corresponds to the number on the inquiry record is found in CUSTFILE, a response is printed in the format shown in Figure 25. This response lists pertinent sales information and the total accounts receivable amount.

The RPG II coding for the random inquiry application is shown in Figure 26.

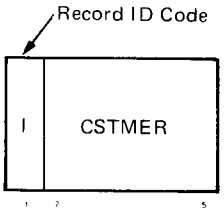
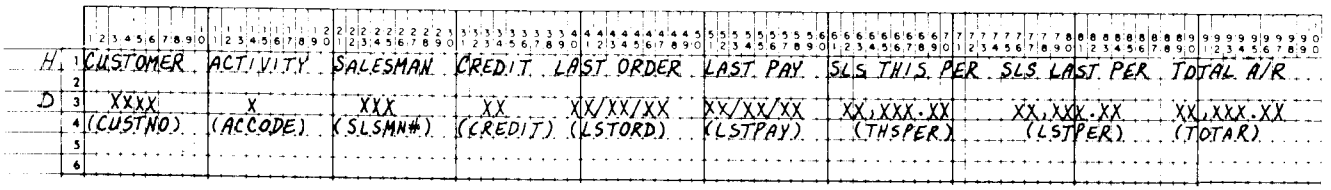


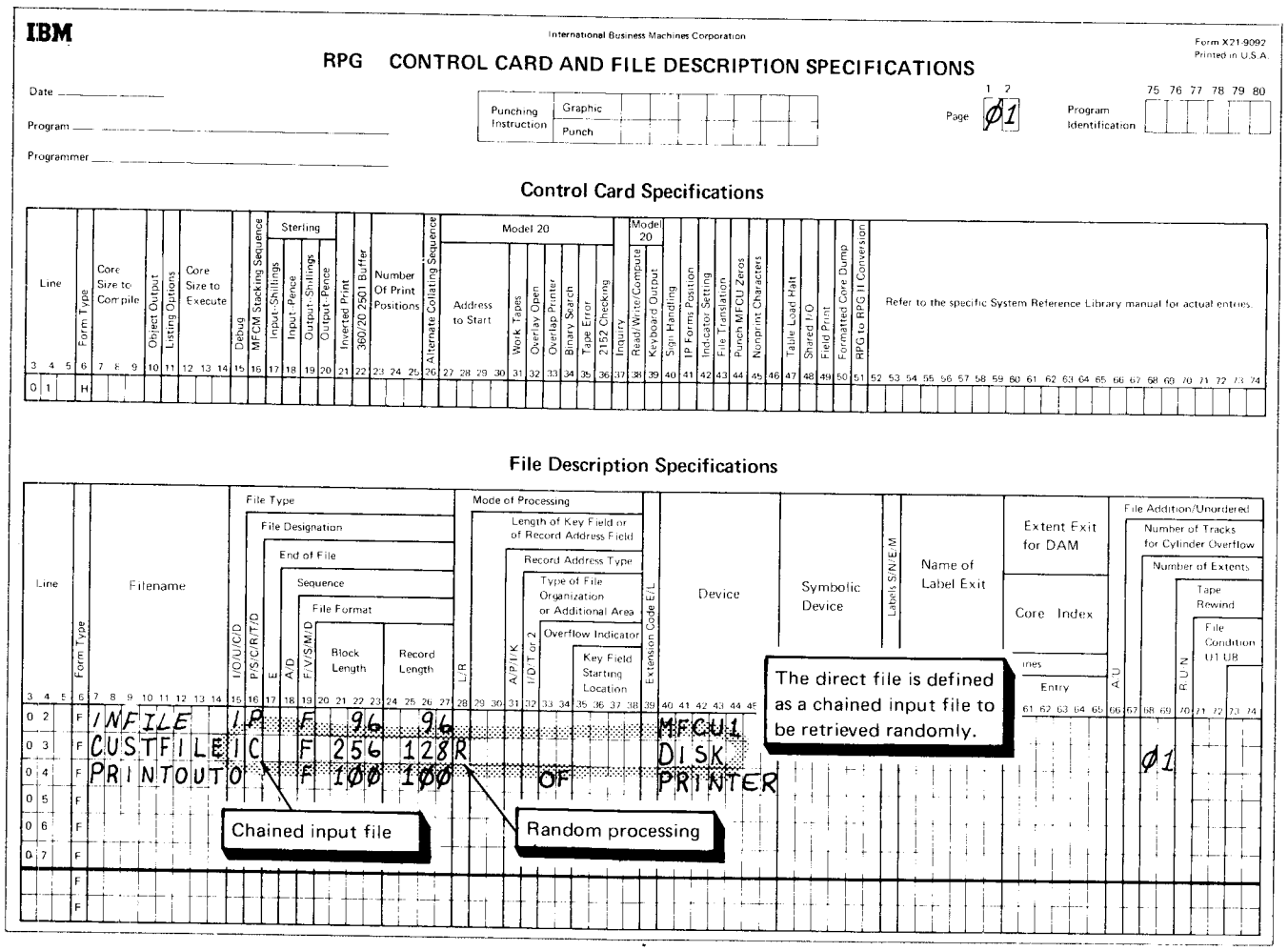
Figure 24. Inquiry Record Format




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

  3119        A        105      01      4/17/71    4/01/71        360.00        239.50     360.00
  6678 RECORD NOT FOUND--INVALID RECORD NUMBER
  1703        I         35      03     11/19/70   12/01/70           .00           .00        .00
```

Figure 25. Printer Output from Random Inquiry Requests



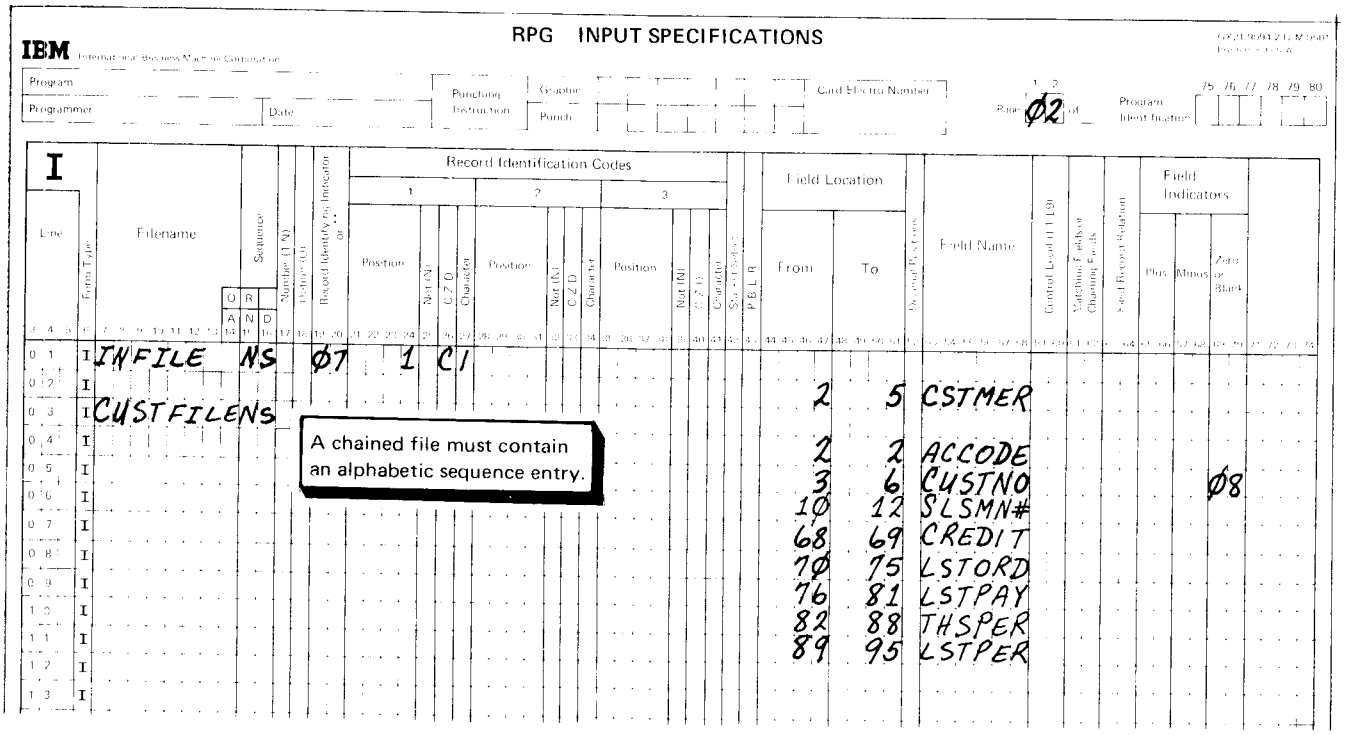
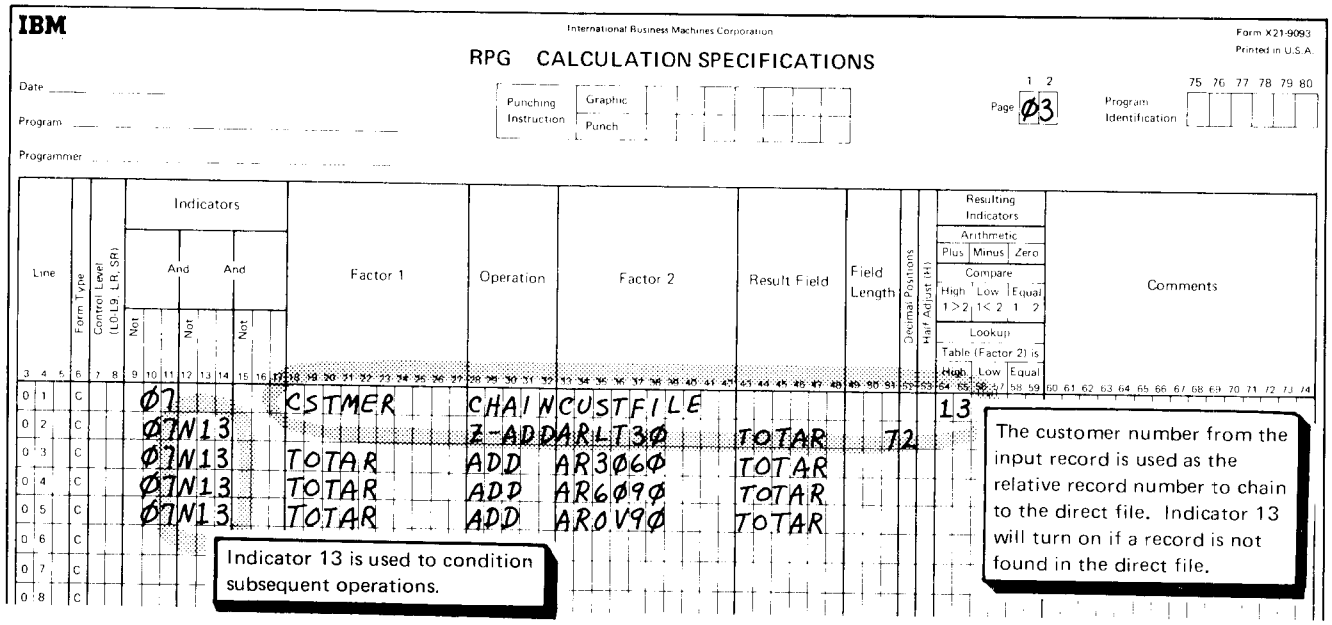


Figure 26 (Part 1 of 2). Random Retrieval of Inquiry Requests on the Direct Customer File



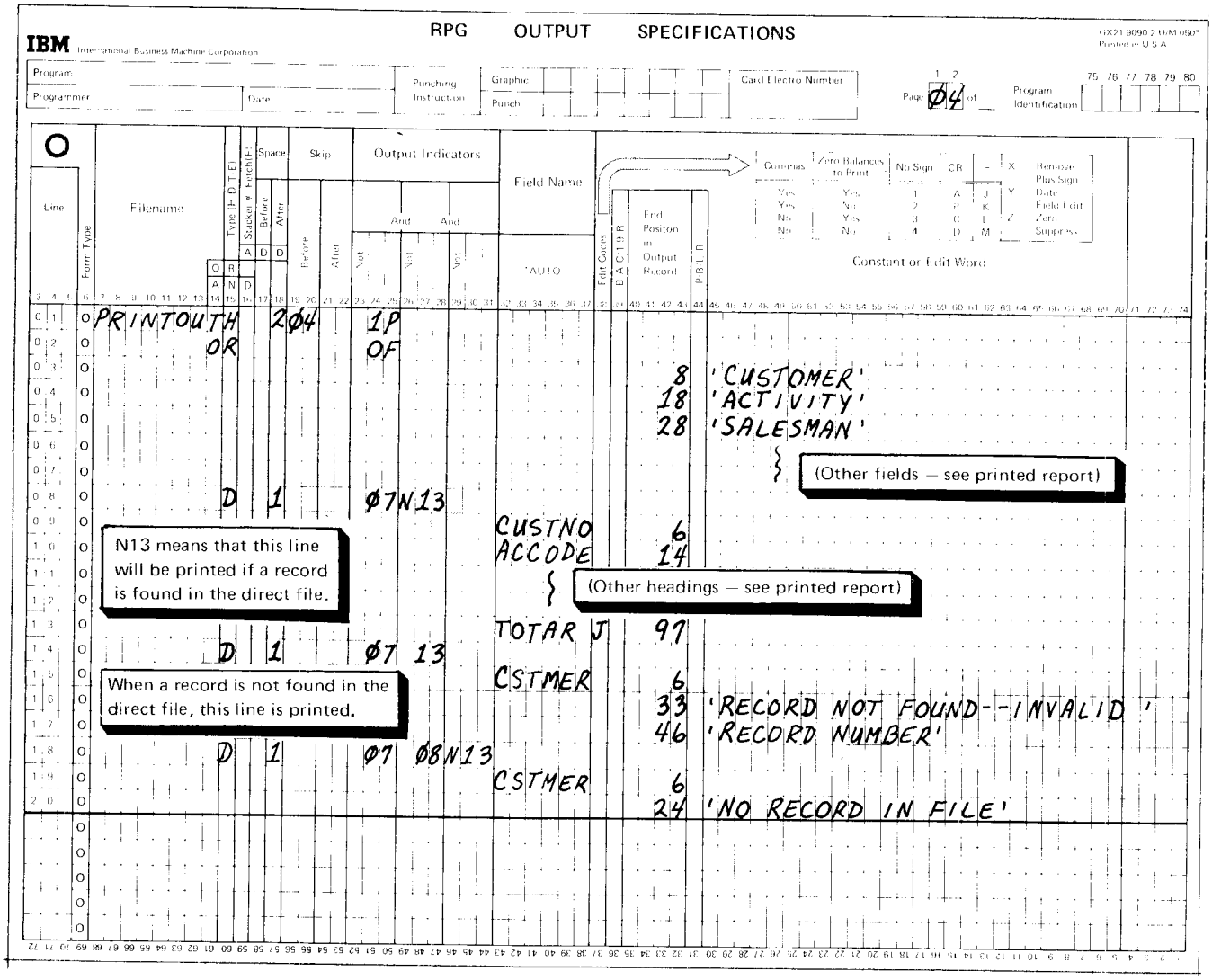


Figure 26 (Part 2 of 2). Random Retrieval of Inquiry Requests on the Direct Customer File

MAINTAINING A DIRECT FILE

After a file is created, file maintenance is usually necessary to keep the file current. Three file maintenance functions apply to direct files:

1. Adding records.
2. Tagging records for deletion.
3. Updating records.

Adding Records

Unlike sequential and indexed files, direct files can have space available between records for new records to be added. Records are added to a direct file by a normal update operation (either consecutive or random processing) as follows:

1. The relative record number is developed for the record to be added.
2. The location is read into main storage.
3. If the location is blank, the new record can be stored.
4. If the location is occupied, and the program can handle synonyms, the new record can be stored as a synonym.

For a discussion of the entries needed to add records consecutively, see *Updating Records, Consecutive Updating of Records* in this chapter. For a discussion of the entries needed to add records randomly, see *Updating Records, Random Updating of Records* in this chapter.

If records must be added but the allotted file space is full, you must increase the total space available for the file. The Disk Copy/Dump program can be used to copy the file into a larger area. For an explanation on how to use this program to copy the file, see the *IBM System/3 Model 10 Disk System Control Programming Reference Manual,* GC21-7512, the *IBM System/3 Model 15 System Control Programming Reference Manual,* GC21-5077, or the *IBM System/3 Model 6 Operation Control Language and Disk Utility Programs Reference Manual,* GC21-7516, depending on the system you have.

Tagging Records for Deletion

Like sequential and indexed file records, direct file records can be identified for deletion by a delete code. This code is usually a single character at a particular location in the record. When the file is processed, your program must check for the delete code; if the code is present, the record can be bypassed.

Since the record has a delete code, the record location is available for a new record. Either a synonym for a different location can be stored there, or the location can be reused by assigning the relative record number to a new record. If the file contains synonyms, be careful not to delete synonym linkage information when you delete a record and reuse the location.

Another method of deleting records, which may sometimes be preferable to using a delete code, is to restore the record location to blanks. A blank record in a direct file is an available record.

You cannot delete records from a direct file by using the Disk Copy/Dump program. Because the DELETE parameter of the COPYFILE control statement causes physical deletion of identified records, the DELETE parameter would destroy the relative positions on which direct file organization depends.

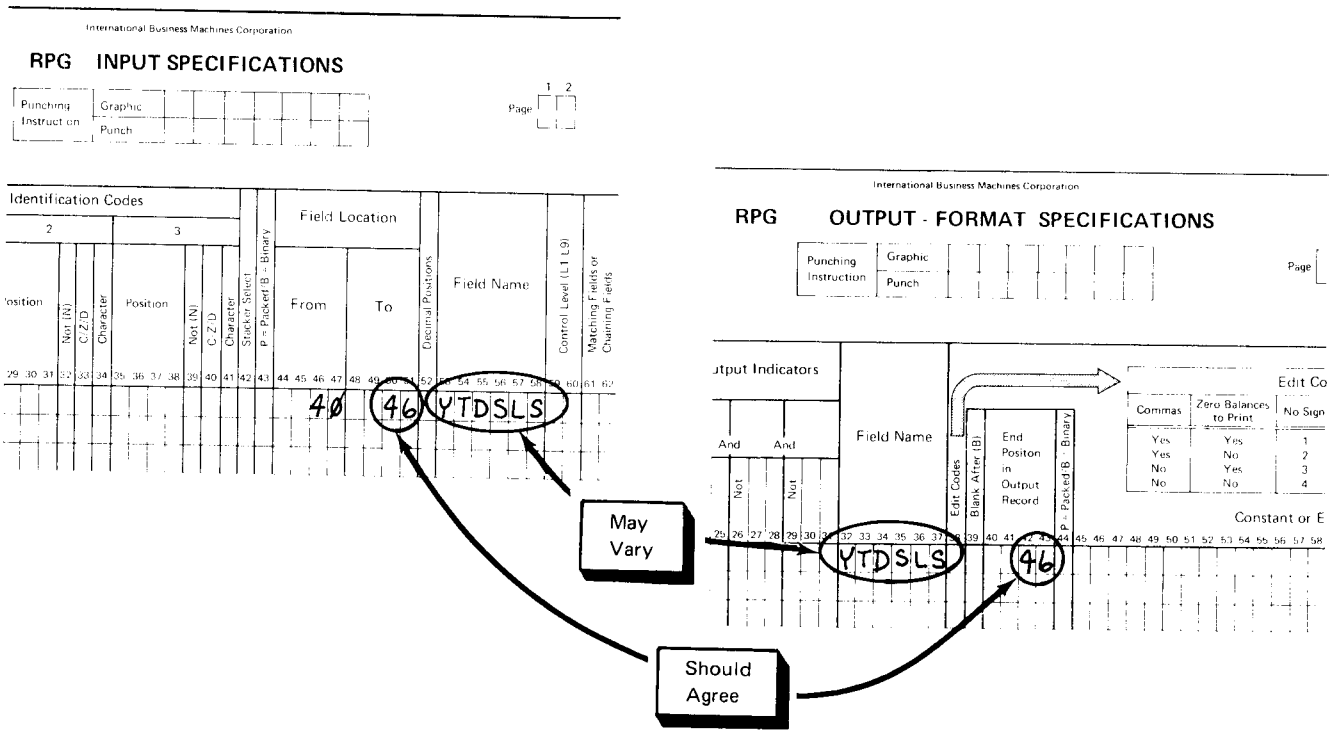
Example of Tagging Records for Deletion

You tag direct file records for deletion in the same way as sequential or indexed file records. For an example of this method, see the examples of tagging records for deletion in Chapters 1 and 2.

Updating Records

To modify certain data in the disk records, you must use the update function. Updating means getting a record from a disk file, changing some data, and putting the record back in its original location. Thus, an update file is like a combination input/output file.

The file to be updated must be specified on both the Input and Output-Format sheets. Field locations should agree between the two sheets. Field names may vary depending on the updating being done. Field lengths must agree.

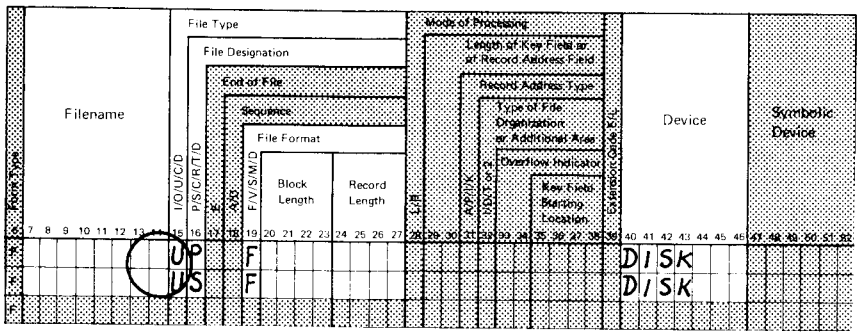


Records can be updated either consecutively or randomly.

Consecutive Updating of Records

If all or most of the records in a direct file are to be processed, you may want to update the file consecutively. Consecutive updating of records in a direct file requires the same File Description entries as consecutive retrieval of a record, with one exception: column 15 must contain a U to indicate that the file is an update file:

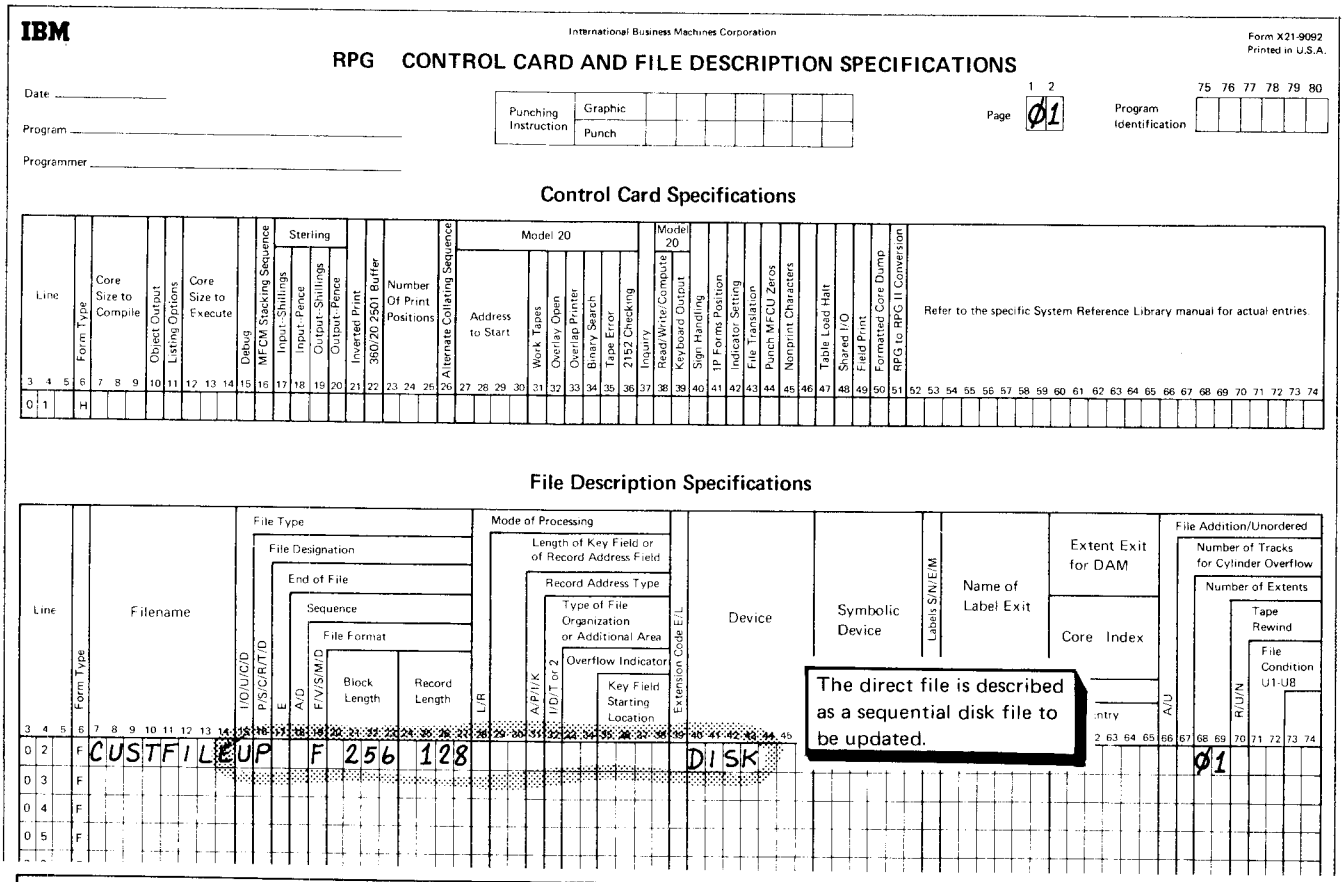
File Description Specifications



Example of Consecutive Updating

Suppose the direct customer file created in the *Example of Creating a Direct File without Synonyms* and retrieved consecutively in the *Example of Consecutive Retrieval* is to be updated. At the end of each sales period, when all reports are completed, the sales figures for that period must be adjusted. Sales amounts for the last period (LSTPER, Figure 21) are replaced by sales amounts from the current period (THSPER). The field containing the current sales amount is reset to zero, ready to accumulate the sales amount for the next selling period. Fields containing overdue amounts will be updated when the monthly accounts receivable statements are written.

Figure 27 shows the coding necessary to consecutively update CUSTFILE. As an update file, CUSTFILE must be defined by file description specifications, and the fields to be updated must be described by input and output specifications. Customer records are read, updated, and written out in the order they are stored in the direct file.



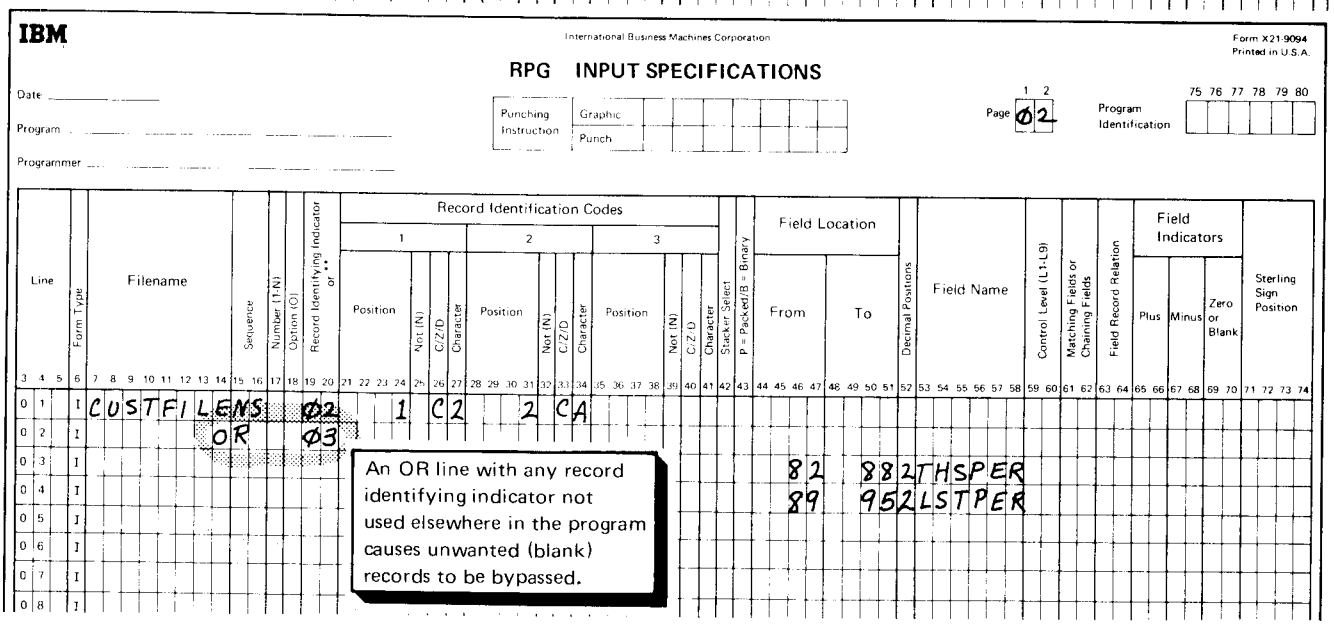


Figure 27 (Part 1 of 2). Consecutive Updating of Records in the Direct Customer File

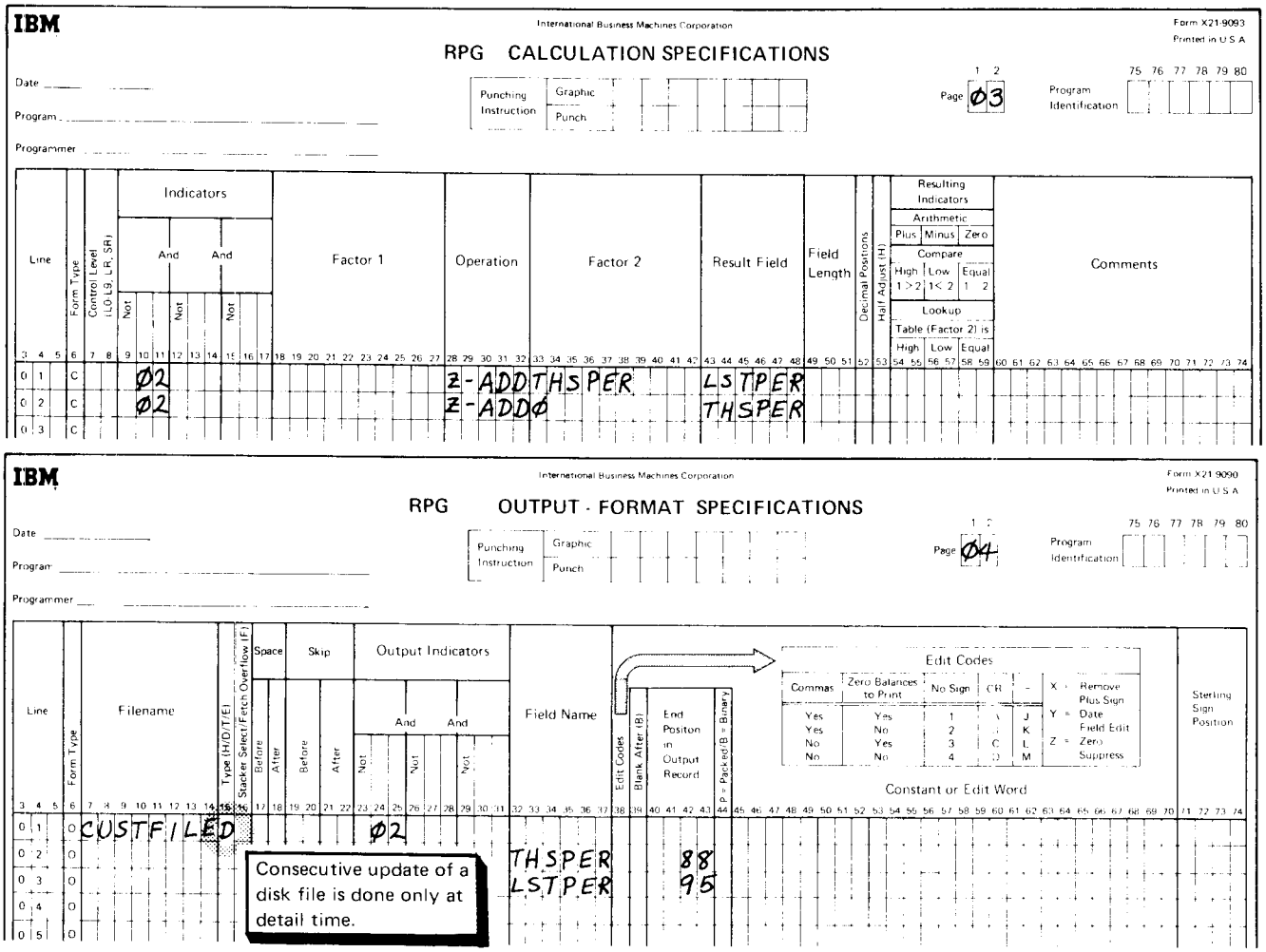
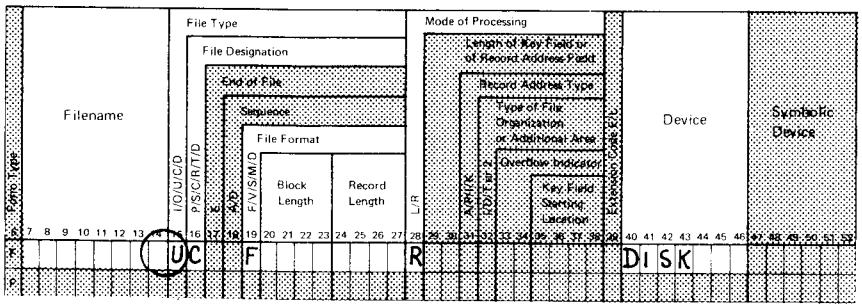


Figure 27 (Part 2 of 2). Consecutive Updating of Records in the Direct Customer File

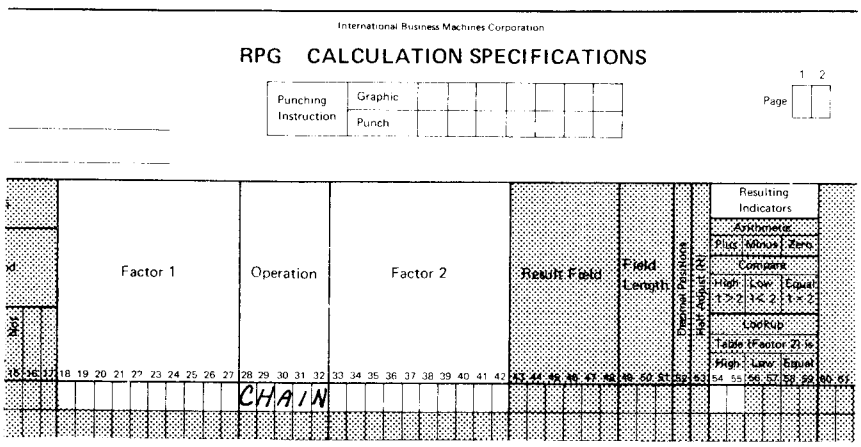
Random Updating of Records

Random updating of records in a direct file requires the same File Description entries as random retrieval of a record, with one exception: column 15 must contain a U to indicate that the file is an update file:

File Description Specifications



If the direct file being updated contains synonym records, calculations must be included in the program to test for synonyms and locate the desired record. Thus, the CHAIN operation code must be specified on the Calculation sheet. The entries on the sheet are the same ones discussed in this chapter under *Creating a Direct File:*



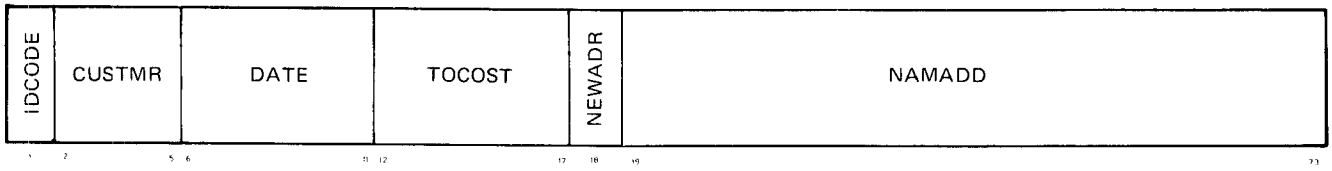
Example of Random Updating

Each day you want to prepare invoices for customer orders for the file described in the *Example of Consecutive Retrieval.* Information from the invoices is used to update the customer file, CUSTFILE. Since this information is read from records (Figure 28) in an unordered manner, a random update job is required.

The input records contain the date and total amount of the transactions for each customer. New addresses are also on this record when required. As each record is read, the customer number (CUSTMR) is used to chain to the direct file. The amount of the transaction is added to total sales for the period (THSPER) and to the accounts receivable amount (ARLT30). The transaction date is placed in the date of last order field (LSTORD) in the customer record.

If an address change is indicated (an X in column 18 of the input record), the new customer address replaces the old. If a record is not found in CUSTFILE because of an invalid relative record number, the input record is printed, followed by the statement "Above record not found—invalid customer number."

CUSTFILE, described as a chained update file, must be described on both Input and Output-Format sheets because data is read from and written on the file. The specifications are shown in Figure 29.



Key

IDCODE	=	Identification code
CUSTMR	=	Customer number
DATE	=	Date
TOCOST	=	Total transaction amount
NEWADR	=	New address code
NAMADD	=	Customer name and new address

Figure 28. Daily Invoicing Record

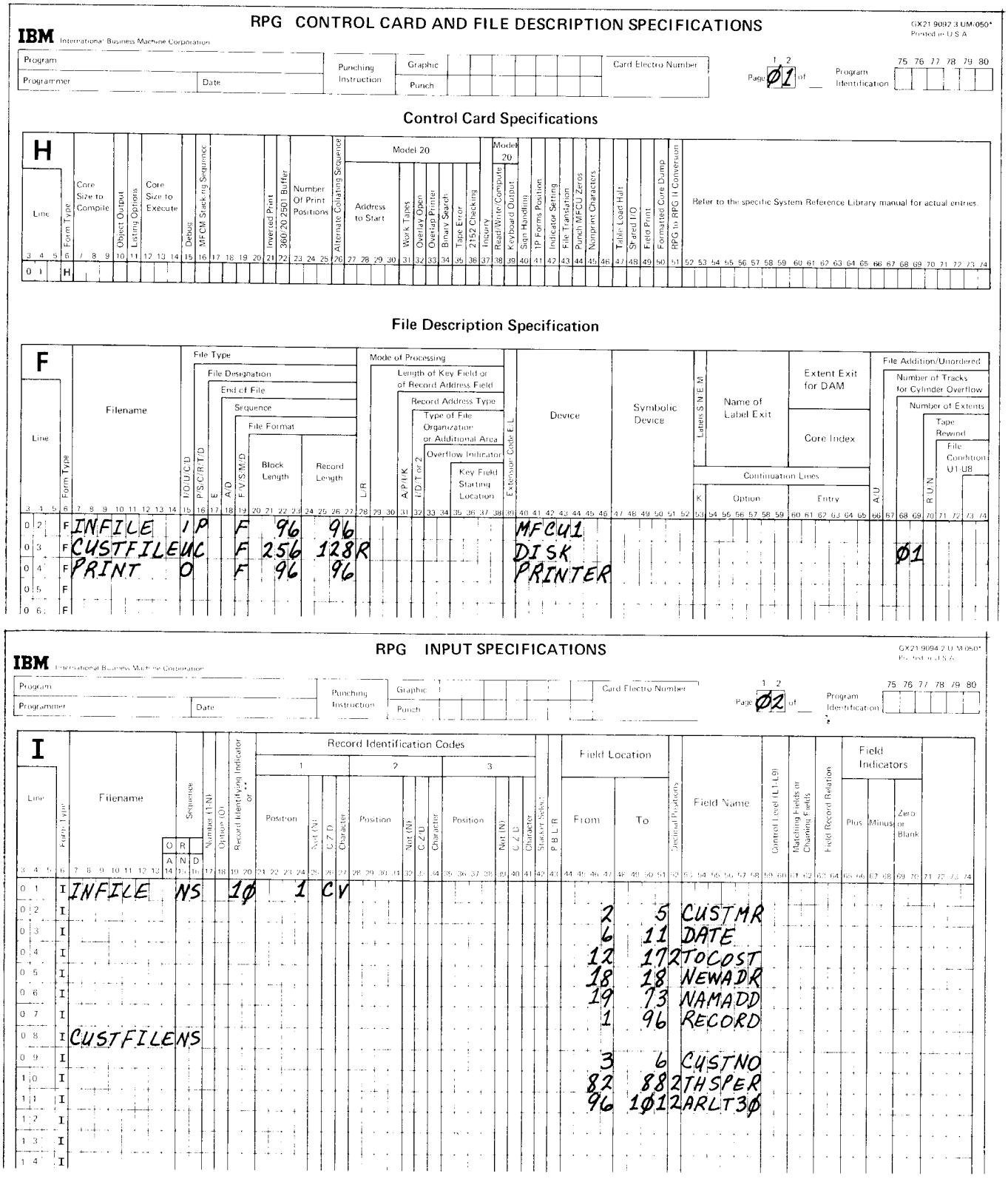


Figure 29 (Part 1 of 2). Random Updating of Records in the Direct Customer File

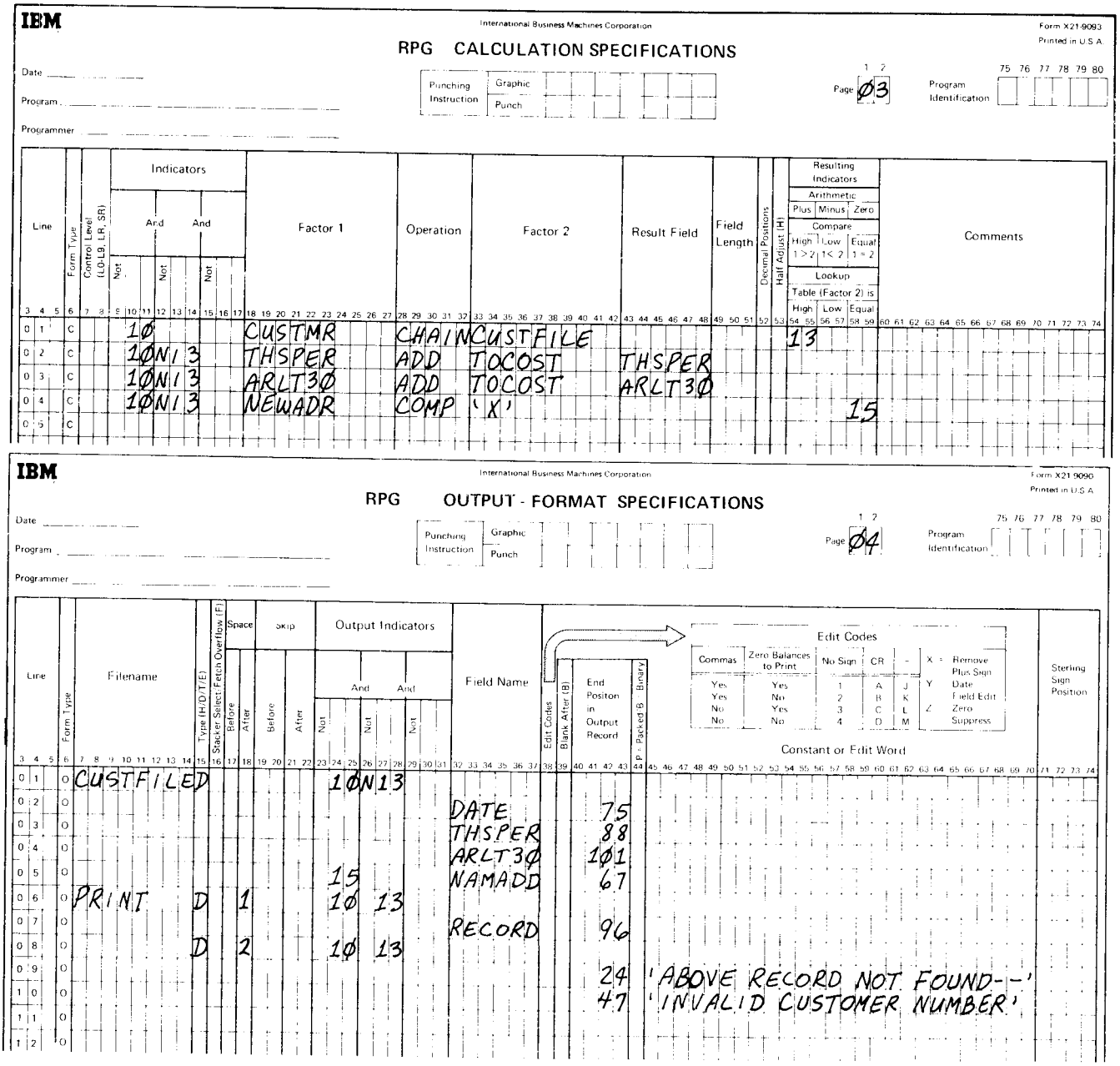


Figure 29 (Part 2 of 2). Random Updating of Records in the Direct Customer File

Chapter 4. Record Address Files

Record address files are input files that indicate (1) which records are to be read from disk files and (2) the order in which the records are to be read from the disk file. There are two types of record address files:

1. Files containing relative record numbers.

2. Files containing record key limits.

This chapter explains how to use the RPG II language to process these two types of record address files. Sample jobs are used to illustrate the functions.

To understand the sample jobs, a basic knowledge of RPG II is necessary. If you do not fully recall some of the coding used in the sample jobs, you should refer to the *IBM System/3 RPG II Reference Manual,* SC21-7504 (for Model 10 Disk System or the Model 15), or the *IBM System/3 Model 6 RPG II Reference Manual,* SC21-7517, depending on the system you are using.

FILES CONTAINING RELATIVE RECORD NUMBERS (ADDROUT FILES)

A record address file that contains relative record numbers is called an ADDROUT file. ADDROUT files are comprised of 3-byte, binary, relative-record numbers that indicate the relative position (first, twentieth, ninety-ninth, etc.) of records in the file to be processed.

An ADDROUT file can only be a disk file. All types of file organization (sequential, indexed, or direct) for primary or secondary files can be processed by an ADDROUT file. When an RPG II program uses an ADDROUT file to process a file, it reads a relative record number from the ADDROUT file, then locates and reads the record situated at that relative position in the file being processed. Only records with relative record numbers in the ADDROUT file are processed.

Creating an ADDROUT File

An ADDROUT file is created by the Disk Sort program. For an explanation on how to create an ADDROUT file, see the *IBM System/3 Disk Sort Reference Manual,* SC21-7522.

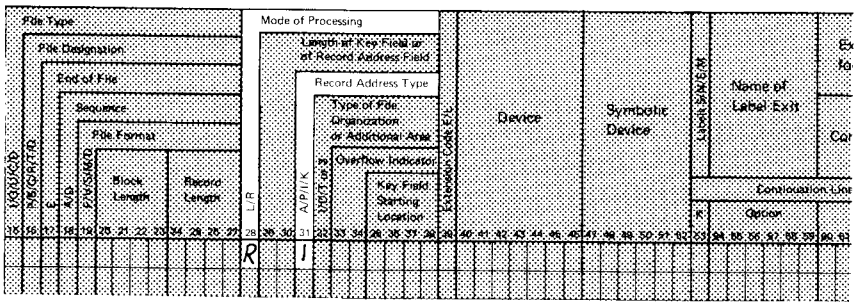
Processing by an ADDROUT File

To use an ADDROUT file to process a file in an RPG II program, entries must be made on the File Description and Extension sheets. Input specifications are not required.

File Description Specifications

The File Description sheet must describe both the file to be processed and the ADDROUT file. The description of the file to be processed must contain the following entries in addition to the usual entries necessary to describe the file:

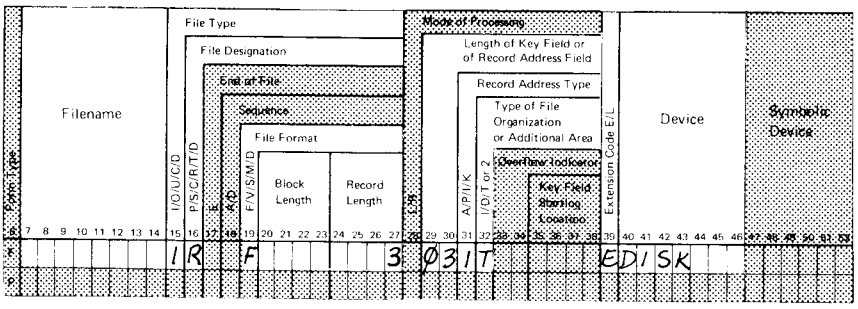
File Description Specifications



Column 28 must contain an R to indicate that the file is to be processed randomly. Column 31 must contain an I to indicate that the file is to be processed by relative record numbers from an ADDROUT file.

The ADDROUT file must be described with the following entries:

File Description Specifications



The ADDROUT filename must be entered in columns 7 through 14. Column 15 must contain an I to indicate that the file is an input file. Columns 16, 31, and 32 contain an R, I, and T respectively; these three columns indicate that the file is a record address file consisting of relative record numbers.

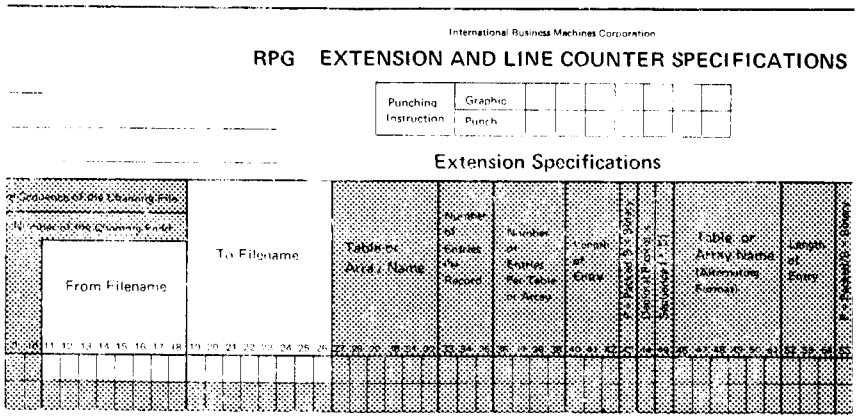
Because all records in a file must be the same length, column 19 contains an F to indicate that the record length is fixed.

The record length (columns 24 through 27) and the length of the record address field (columns 29 through 30) are always 3 because the ADDROUT file consists of 3-byte, relative record numbers. The block length should be equal to or a multiple of the record length.

Whenever a disk file is described, DISK (Model 6, 10, 15), DISK40 (Model 12 or 15), or DISK45 (Model 10, 12, or 15) is required in the Device columns. Since the file must be further defined on the Extension sheet, column 39 must contain an E.

Extension Specifications

Two entries are needed on the Extension sheet if you want to process by an ADDROUT file:



Columns 11 through 18 must contain the name of the record address file. This name must be the same one given to the record address file on the File Description sheet.

Columns 19 through 26 must contain the name of the file to be processed. This name must be the same filename defined on the File Description sheet. This entry indicates that the file is to be processed by the ADDROUT file coded in columns 11 through 18.

Example of Processing by an ADDROUT File

You have a customer master file containing:

- Customer numbers and addresses
- Salesman numbers
- Accounts receivable amounts

The file is an indexed file processed by customer number. It is processed throughout the month for invoicing and at the end of every month for customer statements. Along with the monthly statements, the sales department wants the accounts receivable, printed by customer number within salesman number, for all customers owing more than $2500.00.

In order to do this, the records of customers who owe more than $2500.00 must be sorted according to salesman number. You sort the file using an ADDROUT sort because there is not enough disk storage to use a tag-along sort. After the file is sorted, you have an ADDROUT file of records to be printed. The output of the sort becomes input to an RPG II program. Figure 30 shows the RPG II entries required to use the ADDROUT file as input to an RPG II program.

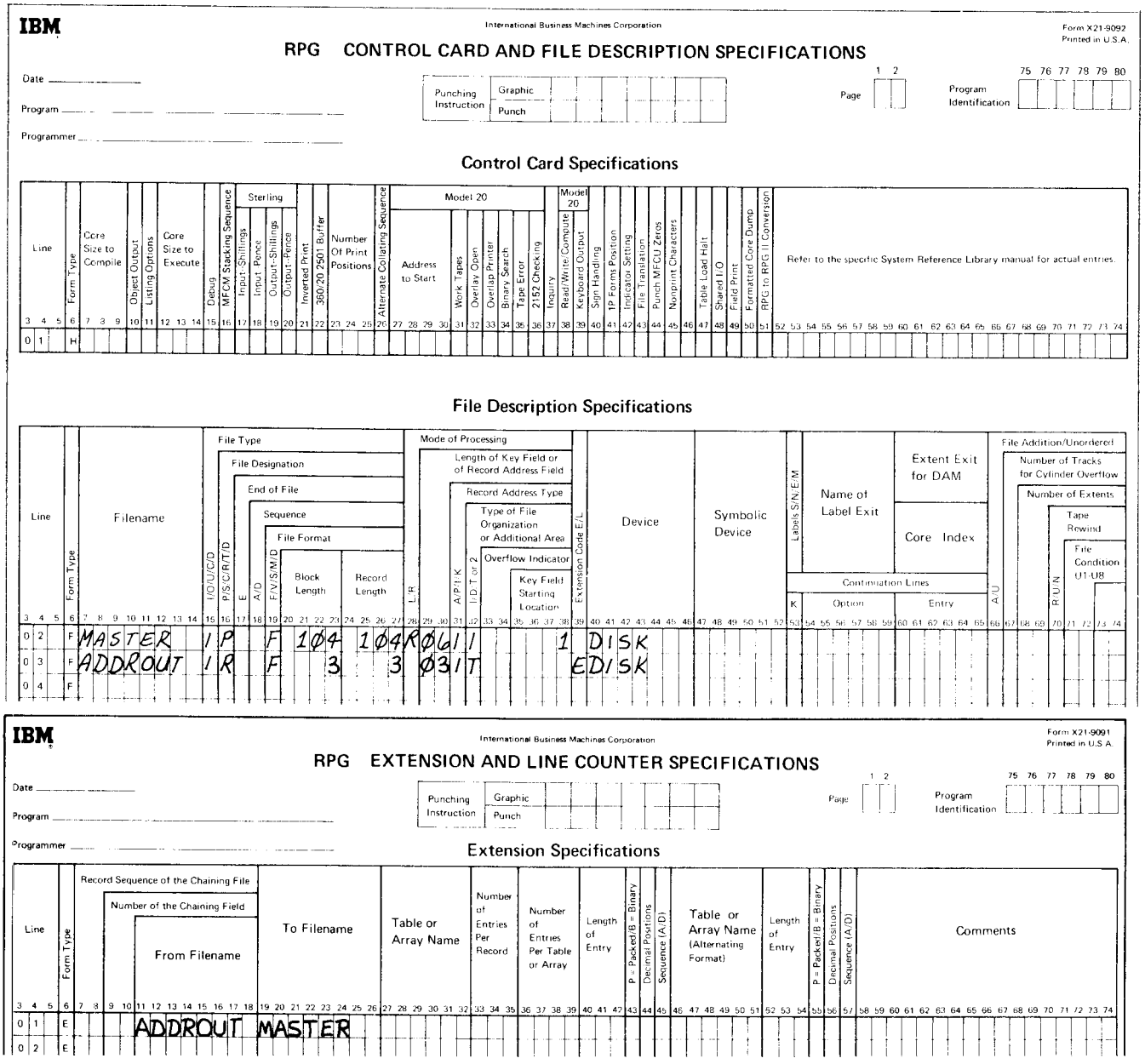


Figure 30. Processing by an ADDROUT File

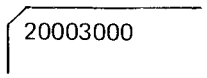
FILES CONTAINING RECORD KEY LIMITS

A record address file with record key limits contains the lowest and the highest key fields for a specified section of an indexed file. Record address files containing record key limits can be entered from disk, card, or a printer-keyboard. They are used to process only indexed files. When a section of an indexed file is processed using record key limits, the processing method is known as sequential within limits.

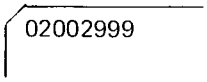
Creating a File with Record Key Limits

The following rules must be observed when you create a record address file with record key limits:

- Only one record address file can be used for each RPG II program, but the record address file can contain several sets of limits.
- Only one set of limits is allowed on each record in a record address file. Since a set of limits is comprised of two keys, the length of each record in a record address file is twice as long as the length of the record key.
- The low record key must begin in position 1 of the record.
- The high record key must immediately follow the low record key. No spaces are allowed between the two keys. If the key field were four bytes long and the low record key were 2000 and the high record key 3000, the record would look like this:

- Each record key can be from 1 through 29 characters long.
- An alphameric record key can contain blanks.
- The length of the keys must equal the length of the key field in the indexed file. To make the length of the keys equal, leading zeros may have to be placed in a numeric record key or blanks in an alphameric record key. For example, if the low record key were three positions (200), the high record key four positions (2999), and the length of the key field in the indexed file four positions, a zero must be placed before the 200 to make it a 4-position number. The record would look like this:

  Each key length must also equal the key field length you specify in columns 29 and 30 of the File Description sheet.
- The same set of limits can appear on more than one record in a record address file. Therefore, records within a set of limits can be processed as many times as needed.
- The two record keys in a set of limits can be identical. For example, both the low and high record keys can be 2999. In this case, only one record (2999) will be processed.

Processing Sequentially within Limits

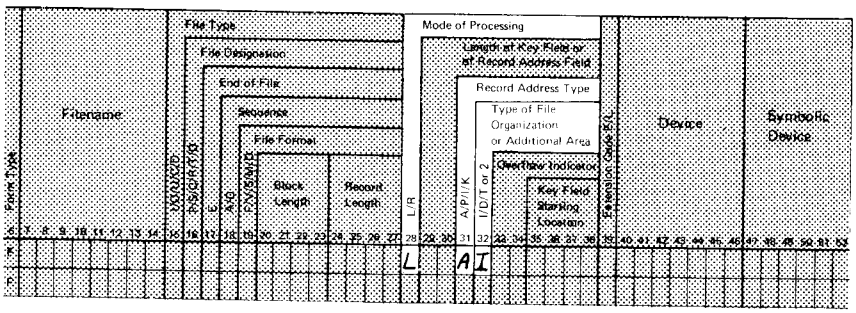
Processing a section of an indexed file sequentially by record keys is known as sequential within limits processing. The RPG II program uses one set of limits (one record in a record address file) at a time. Records are read from the section of the indexed file specified by the limits. When the records specified by one set of limits have been read, the program reads another set of limits from the record address file. The program continues reading records in this manner until the end of the record address file is reached.

To process a file by a record address file using RPG II, entries must be made on the File Description and Extension sheets. Input specifications are not required.

File Description Specifications

The File Description sheet must describe both the indexed file to be processed and the record address file. The description of the indexed file to be processed must contain the following entries in addition to the other entries necessary to describe the file:

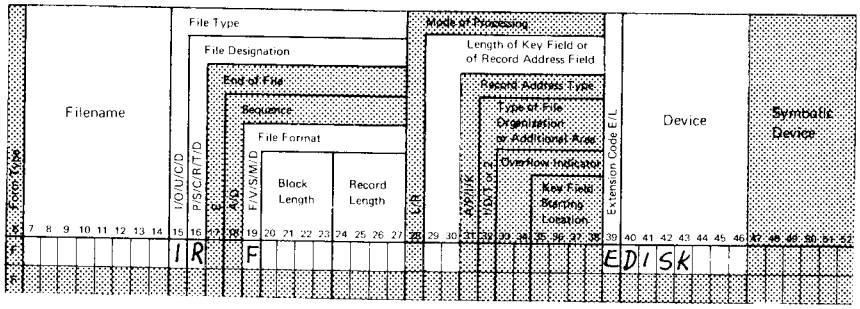
File Description Specifications



Column 28 must contain an L to indicate that the records are to be read from this file sequentially within limits. Enter an A in column 31 to indicate that record keys are used in processing the file. Enter an I in column 32 to indicate an indexed file.

The record address file must be described with the following entries:

File Description Specifications



The name of the record address file must be entered in columns 7 through 14. Column 15 must contain an I to indicate that the file is an input file. Enter an R in column 16 to indicate that the file is a record address file.

All records in a file must be the same length. Thus, column 19 contains an F to indicate that the record length is fixed.

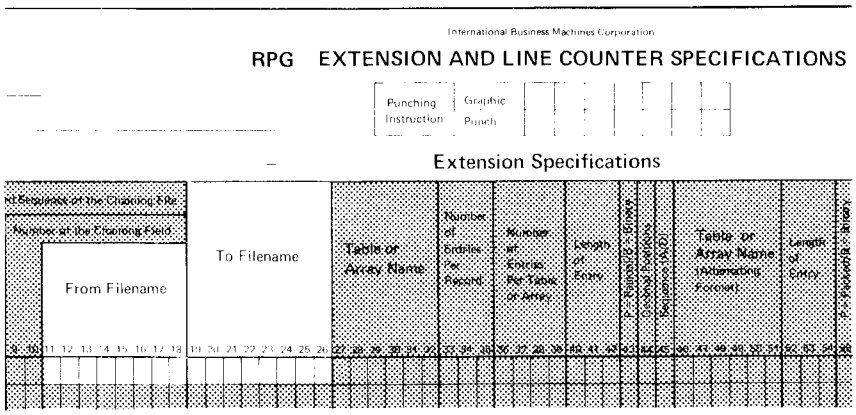
A number equal to or a multiple of the disk record length must be entered in columns 20 through 23. This entry determines the size of the input/output area allocated by RPG II. For an explanation on block length calculation, see the *IBM System/3 Disk Concepts and Planning Guide,* GC21-7571. If you want block length calculated for you by RPG II, assign a block length equal to the record length. By blocking disk records you can increase the input/output efficiency of your program by reducing the number of accesses. You must be sure, however, that enough main storage is available for your input/output area.

Columns 24 through 27 must contain the length of the disk record. Key length, entered in columns 29 and 30, is the same length for all records in a file. The maximum record key length is 29 characters.

Whenever a disk file is described, DISK (Models 6, 10, and 15), DISK40 (Model 12 or 15), or DISK45 (Model 10, 12, or 15) is required in the Device columns. Since additional information is given on the Extension sheet, column 39 must contain an E.

Extension Specifications

To process a file using a record address file, two entries must be made on the Extension sheet:



Columns 11 through 18 must contain the name of the record address file. The name must be the same as the record address file described on the File Description sheet.

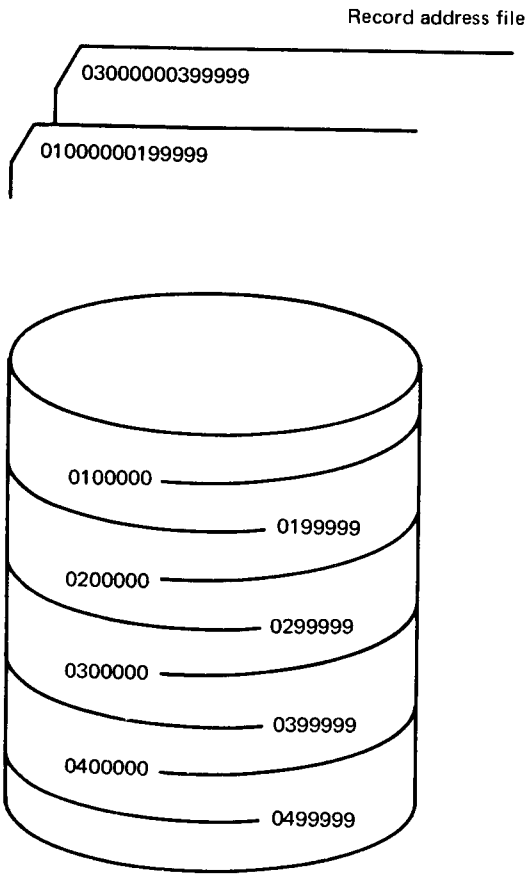
Columns 19 through 26 must contain the name of the file to be processed. This name must be the same filename defined on the File Description sheet. This entry indicates that the file is to be processed by the record address file coded in columns 11 through 18.

Example of Processing Sequentially within Limits

You have a master customer file on disk consisting of 128-character records. The file is organized by customer number within customer class. Customers are separated into such classes as wholesalers or retailers. Together the customer number and class form a composite customer account number (key) in the form: ccnnnnn.

The customer class is cc and the customer number is nnnnn. Customer classes begin at 01 and are in ascending order. Within each customer class, customer numbers range from 00000-99999.

You must prepare separate reports for each customer class for sales analysis purposes. A record address file can be used to supply the particular class categories and customer number ranges as shown in Figure 31. The key in each disk record begins in position 2, and the record address file is loaded in MFCU1. Figure 32 shows the necessary File Description and Extension entries for this job.



Symbolic representation of customer ranges on the disk file.

Figure 31. Files for Processing Sequentially within Limits

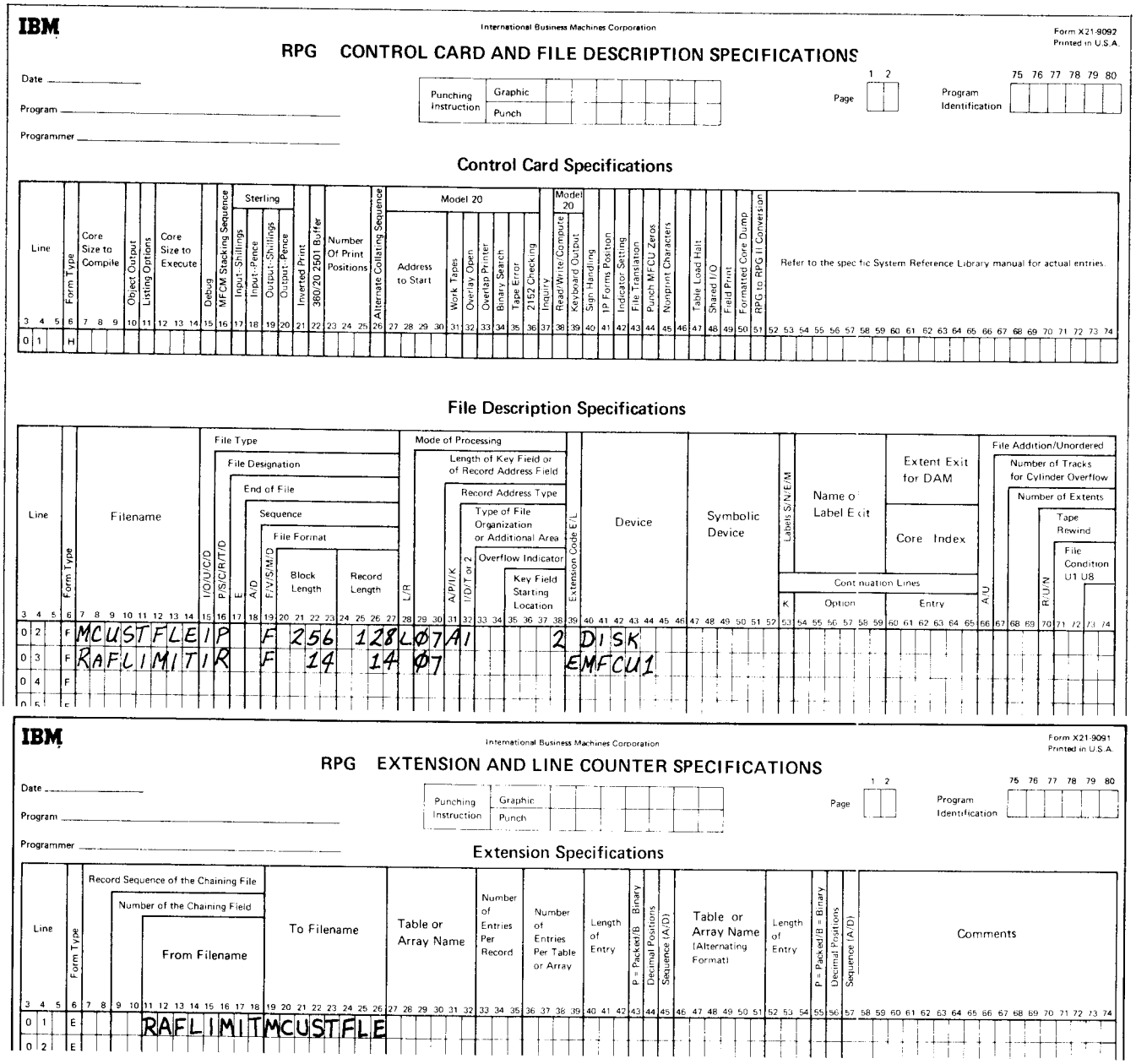


Figure 32. Processing Sequentially within Limits

adding records
direct file 70
indexed file
description 39
randomly by key using chaining 37
randomly by key without chaining 43
sequentially by key 44
sequential file
adding at the end of records in the file 7
description 7
merging between records in the file 12
ADDROUT file
creating 79
description 79
processing by 80

building a disk file v
CHAIN operation code
adding records to an indexed file 37
creating a direct file 53
consecutive processing
direct file 51, 61
indexed file 50
sequential file 1, 16
copy/dump (see reorganizing files)
DELETE parameter (disk copy/dump) 22

creating files
ADDROUT file 79
direct file
description 51
with synonyms 58
without synonyms 53
indexed file
description 26
ordered file 27
unordered file 28
record key limits file 83
sequential file 1

direct files
adding records 70
creating
description 51
with synonyms 58
without synonyms 53
description 51
record retrieval
consecutive 61
description 60
random 66
tagging records for deletion 70
updating records
consecutive 72
description 71
random 75
disk copy/dump (see reorganizing files)

indexed files
adding records
description 39
randomly by key using chaining 37
randomly by key without chaining 43
sequentially by key 44
consecutive processing 50
creating
description 26
ordered file 27
unordered file 28
description 26
random processing
by key 33, 37, 43
by relative record number 50
reorganizing a file 49
tagging records for deletion 49
updating records
description 32
randomly by key 33
sequentially by key 33
sequentially within limits 49, 84
inquiry request 67

master file vi
merging records between records in a sequential file 12
multivolume disk files (see note in preface)
OMIT parameter (disk copy/dump) 22

ordered indexed file
creating 27
description 26

processing files
by ADDROUT file 80
sequentially within limits 84

random processing
direct files (see direct files)
indexed files (see indexed files)
record address files
ADDROUT file
creating 79
description 79
processing by 80
description 79
with record key limits
creating 83
description 83
processing 84
record key limits file
creating 83
description 83
processing 84
record retrieval for direct files
consecutive 61
description 60
random 66
relative record number
direct files 52
indexed files 50
reorganizing files
indexed file 49
sequential file 22
retrieval of records (see record retrieval for direct files)
sequential files
adding records
adding at the end of records in the file 7
description 7
merging between records in the file 12
creating 1
description 1
reorganizing a file 22
tagging records for deletion 16
updating records 16
sequential within limits processing 84
synonyms (direct files) 58, 53
tagging records for deletion
direct file 6
indexed file 49
sequential file 16
transaction file v
unordered indexed file
creating 28
description 26
updating records
direct file
consecutive 72
description 71
random 75
indexed file
description 32
randomly by key 33
sequentially by key 33
sequentially within limits 49, 84
sequential file 16

Technical Newsletter

This Newsletter No.	GN21-5427
Date	24 September 1976
Base Publication No.	GC21-7566-1
File No.	S3-38
Previous Newsletters	None

IBM System/3
RPG II Disk File Processing
Programmer's Guide

This technical newsletter, a part of version 05, modification 00, of IBM System/3 Model 15 RPG II, applies to the current versions and modifications of RPG II for all System/3 models, and to subsequent versions and modifications unless specifically altered. This TNL provides replacement pages for the subject publication. Pages to be inserted and/or removed are:

v through viii
1, 2
27, 28
37, 38
43, 44
51, 52
81, 82
85, 86

Changes to text and illustrations are indicated by a vertical line at the left of the change.

Summary of Amendments

- Changes effected by 5704-RG2 have been added.
- Information about 5448 Disk Storage Drive has been added.

Note: Please file this cover letter at the back of the manual to provide a record of changes.

IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901

Printed in U.S.A.

Technical Newsletter

This Newsletter No. GN21-5625
Date 30 June 1978

Base Publication No. GC21-7566-1
File No. S3-28

Previous Newsletters GN21-5427

IBM System/3
RPG II Disk File Processing
Programmer's Guide

This technical newsletter applies to the current versions of RPG II for all System/3 models, and to subsequent versions and modifications unless specifically altered. This TNL provides replacement pages for the subject publication. Pages to be inserted and/or removed are:

Title Page, Edition Notice
iii, iv
29, 30
33, 34
34.1, 34.2 (added to accommodate moved text)
35, 36 (text rearranged)
37 through 40

Changes to text and illustrations are indicated by a vertical line at the left of the change.

Summary of Amendments

Miscellaneous technical changes

Note: Please file this cover letter at the back of the manual to provide a record of changes.

IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901

Printed in U.S.A.

READER'S COMMENT FORM

IBM System/3
RPG II Disk File Processing
Programmer's Guide

GC21-7566-1

YOUR COMMENTS, PLEASE . . .

Your comments assist us in improving the usefulness of our publications; they are an important part of the input used in preparing updates to the publications. All comments and suggestions become the property of IBM.

Please do not use this form for technical questions about the system or for requests for additional publications; this only delays the response. Instead, direct your inquiries or requests to your IBM representative or to the IBM branch office serving your locality.

Corrections or clarifications needed:

Page *Comment*

Please include your name and address in the space below if you wish a reply.

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

Cut Along Line

Fold

Fold

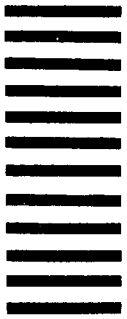
FIRST CLASS
PERMIT NO. 387
ROCHESTER, MINN.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

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



Fold

Fold

IBM Sy , RPG II (File No. S3-28) Pr d in U.S.A. GC21-7566-1

International Business Machines Corporation
General Systems Division
875 Johnson Ferry Road N. E.
Atlanta, Ga. 30342
(USA Only)

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

GC21-7566-1

IBM System/3 RPG II (File No. S3-28) Printed in U.S.A. GC21-7566-1

International Business Machines Corporation
General Systems Division
875 Johnson Ferry Road N. E.
Atlanta, Ga. 30342
(USA Only)

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