File Number 1130-28
Form C21-5002-1

12 MAY 1969

IBM Systems Reference Library

IBM 1130 RPG Language

This reference publication contains fundamentals of RPG programming and language specifications for the IBM 1130 Computing System. For information on RPG that is beyond the purpose of this language publication, see IBM 1130 Disk Monitor System, Version 2, Programming and Operator's Guide.

For the titles and abstracts of associated publications, see the IBM 1130 Bibliography, Form A26-5916.

Preface

This publication is intended for users of the IBM 1130 Computing System. IBM 1130 RPG is a problem-oriented language designed to provide users with an efficient, easy-to-use technique for generating programs that can:

1. Obtain data records from single or multiple-input files,
2. Perform calculations on data taken from input records or RPG literals,
3. Write printed reports,
4. Use table lookup,
5. Exit to a user's subroutine wirtten in a language other than RPG,
6. Branch within the calculations, and
7. Sequence-check input records.

RPG uses a set of specifications sheets on which the user makes entries. The forms are simple, and the headings on the sheets are largely self-explanatory.

Although many reports use only one input file, RPG can combine data from multiple input files to create a report. The output may be a single report, or it may be several reports created simultaneously on different devices.

Prerequisites: The reader must be familiar with the operation of the components of his 1130 System and with basic programming concepts. For titles and abstracts of associated publications, refer to the *IBM 1130 Bibliography.*

Second Edition

This is a major revision of, and obsoletes, C21-5002-0.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form is provided at the back of this publication for reader's comments. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publications, Department 425, Rochester, Minnesota 55901.

RPG consists of a problem-oriented, symbolic programming language and a compiler program. The RPG language provides an easy-to-use technique for writing a wide variety of 1130 programs. The compiler program reads the program specifications written in the RPG symbolic language and produces an object program in machine language that can be used to perform a particular application. Both compilation and execution of object programs are under control of the IBM 1130 Disk Monitor System, Version 2.

The 1130 RPG language provides the capability for writing programs that can:

1. Obtain data records from single or multiple-input files.
2. Perform calculations on data taken from input records.
3. Produce printed reports.
4. Use table lookup.
5. Exit to a user's subroutine written in Assembler language.
6. Branch within the calculations.
7. Sequence-check input records.
8. Maintain files.

RPG uses a set of specifications sheets on which the user makes entries. The forms are simple, and the headings on the sheets are largely self-explanatory.

Although many reports use only one input file, RPG can combine data from multiple input files to create a report. The output may be a single report, or it may be several reports created simultaneously on different devices.

MACHINE REQUIREMENTS

The minimum machine requirements for generating an RPG object program are:

- An IBM 1131 Central Processing Unit, Model 2B or 3B.
- One card-reading device.
- One printer (IBM 1132, 1403, or console printer).

The minimum machine requirements for the execution of an RPG program are:

- An IBM 1131 Central Processing Unit, Model 2B or 3B
- Input/output devices as required by the object program. These may be any combination of the following:
  IBM 1403 Printer, Model 6 or 7
  IBM 1442-5 Card Punch
  IBM 1132 Printer
  IBM 2501 Card Reader, Model A1 or A2
  IBM 1442 Card Read/Punch, Model 6 or 7
  IBM 2310 Disk Storage Drive, Models B1 and B2

The following additional system features are supported for RPG program generation:

- An IBM 1131 Central Processing Unit, Model 2C, 2D, 3C, or 3D.
- A card-punching device
- One additional IBM 2310 Disk Unit

PROGRAMMING REQUIREMENTS

1130 RPG can only be used in association with the IBM 1130 Disk Monitor System, Version 2.

FUNCTION OF RPG

When RPG is used, the IBM 1130 performs two separate functions:

1. program generation, and
2. program execution.

During program generation, program specifications defined by the user produce machine-language instructions. Storage areas are automatically assigned, constants or other reference factors are included, and program routines for checking, for input/output operations, and for other functions are produced.

During program execution, the machine-language instructions are combined with the input data files and both are processed by the system to produce the desired reports and output files.

USE OF RPG

Figure 1 depicts the general operations involved in using RPG to prepare a report. The circled numbers in Figure 1 refer to the numbers in the following text.

1. The programmer evaluates the report requirements to determine the format of the input files and the layout of the finished report. For example, he determines what fields in the input records are to be used, what calculations are to take place, where the data is to be located in the output records, and how many and what kind of totals must be accumulated.
2. After the programmer has evaluated the requirements of the report, he provides the RPG program with information about these requirements as follows:

   - He describes all files used by the object program (input files, output files, table files, etc.) by making entries on the File Description Specifications and on the Extension Specifications forms.
   - He describes his input (record layout, fields used, etc.) by making entries on the Input Specifications form.
   - He states what processing is to be done (add, subtract, multiply, divide, etc.) by making entries on the Calculation Specifications form.
   - He defines the layout of the desired report (print positions, carriage control, etc.) by making entries on the Output-Format Specifications form.

3. After the specifications have been written on the appropriate forms, cards are punched with the data from the forms. Each line on the form is punched into a separate card.
4. These punched cards (called a source deck) are combined with the RPG control card. The source deck and the control card are placed into a card reader and processed by the RPG compiler under control of the disk monitor system. At the end of this processing run (known as the compilation run), a program capable of preparing the report specified by the programmer has been produced and stored in the working storage area of the disk. This program, known as the object program, contains all the machine instructions required to prepare the desired report.
5. The programmer may now have the object program:
   a. executed immediately,
   b. punched into cards for storage, or
   c. stored on disk for later execution.
6. The input files are then read into the system and production of the report begins. This is known as the object run.

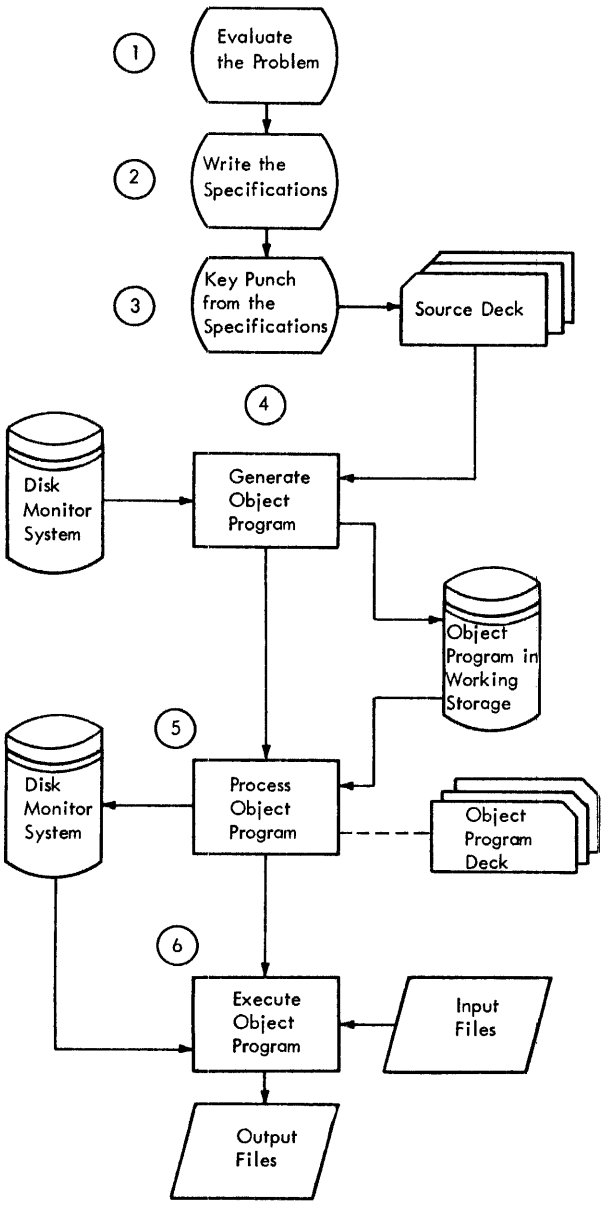


Figure 1. Producing Reports Using the RPG Program

UNIQUE 1130 RPG FILE ORGANIZATION

1130 RPG uses unique file organization techniques. Disk files not created by 1130 RPG cannot be processed by 1130 RPG.

USE OF 1130 RPG WITH OTHER SYSTEMS

RPG for the IBM 1130 System may differ in some respects from those of other systems. Check the following language features when considering system compatability.

1. Items supported by 1130 RPG include:
   - Halt indicators H1 through H9.
   - Matching record fields M1 through M9.
   - Chaining fields C1 through C3.
   - Calculation operations CHAIN, BEGSR, ENDSR, EXSR, and EXCPT, among others.
   - Edit codes.
   - Exception output.
   - Numeric fields may be from one to 14 digits in length.
2. Reserved words under 1130 RPG are:
   - ALTSE
   - PAGExx
   - TAByyy
   - INxx in RLABL statements.

   The characters xx and yy represent any alphameric characters.
3. File Description Specification entries differ according to the devices used.
4. Blank/zero indicators are not initialized (set on) and are not reset on a blank-after condition.
5. Number and type of files supported may vary.
6. When moving data to numeric fields, the data is not checked for valid sign or digits. Non-numeric characters may result in unpredictable results. Signs are forced on all arithmetic operations.
7. An 1130 RPG program cannot both add and retrieve records from an indexed sequential file in the same program.
8. Packed data may only be specified on disk records.
9. Prior to performing arithmetic operations, no checks are made to ensure that valid numeric data is present in a field and that a valid sign exists. Any sign other than D is considered positive. When adding invalid numeric data, accuracy cannot be assured.

Alphabetic Characters

Include the 26 letters of the alphabet, A through Z.

Numeric Characters

Include the digits 0 through 9.

Special Characters

Are any Extended Binary-Coded-Decimal Interchange Code (EBCDIC) characters that are neither alphabetic nor numeric, e.g., #, $, &, ', %, etc.

Hexadecimal

Is a number system using the equivalent of the decimal number sixteen as a base. The sixteen values are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F.

Numeric Fields

Are fields containing only numeric characters and possibly a plus or minus sign over the rightmost position. In RPG, numeric fields are all fields that have a decimal-positions specification on the appropriate specification forms. The contents of numeric fields are placed into core storage in unpacked decimal format. A field must be defined as numeric if any arithmetic or editing (insertion of commas or decimals) is to be performed on it.

Alphameric Fields

Are fields that contain both letters and numerals, and usually other characters. In RPG, alphameric fields are all fields (regardless of content) for which a decimal-positions specification has not been made on the appropriate specifications forms. The contents of alphameric fields are placed into core storage in unpacked format.

Alphameric Literals

May consist of any of the 256 EBCDIC characters. They must be enclosed in apostrophes (').

Numeric Literals

May consist of the digits 0 through 9, and may contain one decimal symbol and/or a plus or minus sign. Numeric literals must not be enclosed in apostrophes.

The RPG compiler performs two types of error testing:

1. During a compilation run, the RPG source deck will be checked for errors and a message will be printed that describes the error and lists the source statement number. A further explanation of errors is contained in *IBM 1130 Disk Monitor System, Version 2: Programming and Operator's Guide,* Form C26-3717.
2. During the execution run, RPG may also discover errors (e.g., undefined record type, divide by zero, etc.) A halt will occur and the operator will be informed of the specific cause by the number in the console accumulator.

This section introduces the fundamental concepts of preparing RPG programs. It discusses how to describe files and records, and how to specify the most frequently used RPG functions.

These functions are used in the most typical reports produced by RPG and depend on a single sequential file only. Although RPG programs can process files contained on disk, only card input files are used for the purposes of this section.

Programmers familiar with the basic concepts of RPG may skip this section and concentrate on the detailed descriptions of the specifications forms in subsequent sections.

The numerous fields on the RPG specifications forms may make it appear that writing RPG specifications is a difficult task. However, few programs use all the specifications. Some programs may require entries on only one or two lines of the forms.

At the end of the description of basic RPG functions is a simple file-to-file listing application that illustrates how the specifications forms are related.

The following functions are described in this section:

- File Description
- Record and Field Description
- Addition and Subtraction
- Detail Printing
- Control Field Establishment
- Total Calculations
- Detail and Total Printing
- Group Printing
- Group Indication
- Page Headings
- Summary Punching
- Zero, Plus, and Minus Balance Test
- Field Comparison
- Multiplication and Division
- Sequence Checking

FILE DESCRIPTION

The File Description Specifications form illustrated in Figure 2 furnishes information about the files used in a job.

In the Figure 2 example, the first line provides information about the detail labor file. The file name DETLABOR identifies this file. The letter C in file type indicates that the file is a combined file. A combined file is a file that contains both cards read into the system and cards used for punching. The P in file designation is required to define the primary or only input file. The particular input/output device that is to be used with this file is specified in Device. In this example, the IBM 1442 Card Read/Punch is used.

The second line describes the output file. The letter O in file-type indicates that the file is an output file. PRINTER in the device column indicates that the file is to be printed on the IBM 1132 Printer.

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 REPORT

Control Card Specifications

Line	Form Type	Core Size to Compile	Object Output	Listing Options	Core Size to Execute	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	Refer to the specific System Reference Library manual for actual entries.
3 4 5	6	7 8 9	10	11	12 13 14	15	16	17	18 19	20	21	22	23 24 25	26	27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
0 1	Ø 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, or E)	Name of Label Exit	Extent Exit for DAM	File Addition	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind
			I/O/U/C	P/S/C/R/T	E	A/D	F/V			L/R		K/I	I/D/T or 1-9									A		N/U	
3 4 5	6	7 8 9 10 11 12 13 14	15	16	17	18	19	20 21 22 23	24 25 26 27	28	29 30	31	32	33 34	35 36 37 38	39	40 41 42 43 44 45 46	47 48 49 50 51 52	53	54 55 56 57 58 59	60 61 62 63 64 65	66	67 68 69	70	71 72 73 74
0 2	Ø F	DETLABORCP					F		8Ø								READ42								
0 3	Ø F	DAILYRPTO					F		12Ø								PRINTER								
0 4	F																								

Figure 2. File Description Specifications

RECORD AND FIELD DESCRIPTION

Entries on the Input Specifications form describe the data that is to be read into the system.

Example

A detail labor file (DETLABOR) contains card records as shown in Figure 3. Three fields from each card are to be read into storage. Field A is contained in columns 46-50, field B is contained in columns 56-60, field C is contained in columns 66-70 of each record.

Each card that contains fields A, B, and C is identified by two distinct attributes. It must contain a digit 5 in column 35 and no 11-punch in column 80. When these two conditions exist in the card record, the record is valid; that is, it contains fields A, B, and C.

Specifications: In Figure 4, the entries on the Input Specifications form that apply to this example are circled and identified by numbers referring to the items in the text that follows. The remaining fields on the form are described later in this publication.

1. Each file must be given a name. Each file name can be used to identify one file only. This file name must be the same as specified on the File Description Specifications form.
2. If several card types exist in a file, each card type must be identified by its particular card code. The identifying codes must be entered into the Record Identification Codes (columns 21-41) of the Input Specifications form.

   In this example, the card codes desired are a 5 punch in column 35 without an 11 zone in column 80. They are specified by writing a 5 in Character (column 27) and a 35 in Position (columns 21-24) for the first code; and by writing an 80 in Position, a minus in Character, and an N in Not (column 32) for the second code. The fields C/Z/D (columns 26 and 33) on the form indicate how the card code is to be compared:

   D = digit portion only.
   Z = zone portion only.
   C = zone and digit portions.

   After all the identification codes are established, the programmer assigns a two-digit number (from 01-99) to the card type identified. This code, known as a

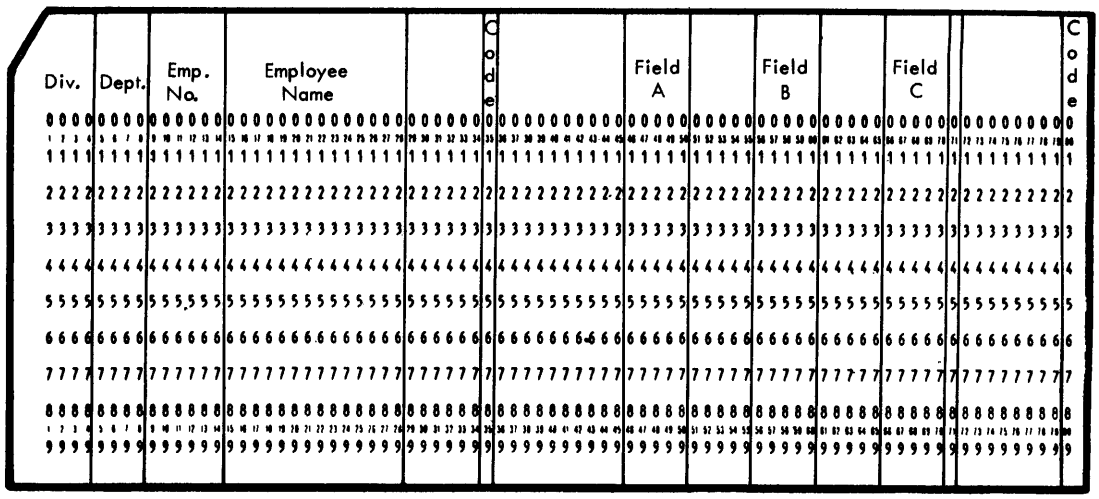


Figure 3. Input Card from a Detail Labor File

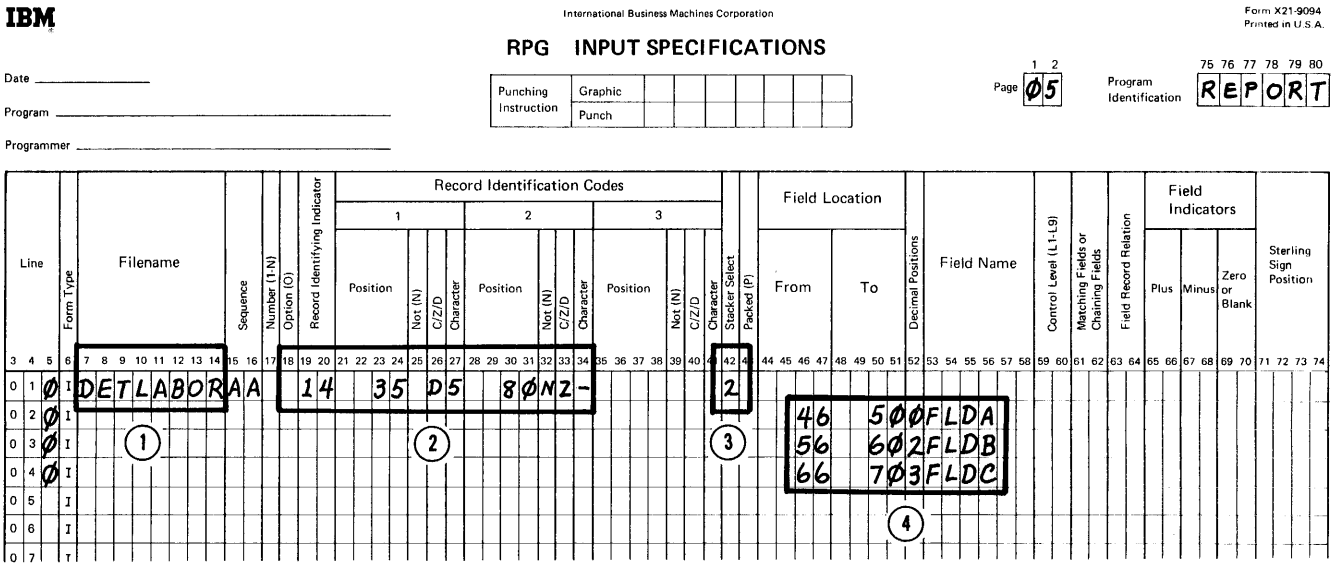


Figure 4. Input Specifications Form

Record Identifying Indicator (columns 19-20), is used as a switch or selector. Whenever a card of the specified type is being read, the associated indicator is turned on. If the card read is not one of the type specified, the indicator is turned off or remains off. The Record Identifying Indicator is used wherever it is necessary to refer to the specific card type on other RPG specification forms.

3. If the input card is to be placed into a stacker other than the one into which it would normally be selected, the stacker number is written in Stacker Select (column 42).
4. Each field of the card to be read must be defined as shown on the lines following the record identification line (line 010 using columns 7-42). The specifications shown in this example obtain the data from the input record and place it in fields A, B, and C in three separate locations in storage (named FLDA, FLDB, and FLDC). The entries in Decimal Position (column 52) define the associated fields as numeric.

The programmer need not know the actual location of these fields in core storage. To refer to these fields on other RPG forms, the programmer writes only the appropriate field name.

A field name may consist of six or fewer alphameric characters. The first character must be alphabetic. No blanks or special characters can be used in a name.

ADDITION AND SUBTRACTION

Once input data has been described on the Input Specifications form, calculations to be performed with that data can be defined on the Calculation Specifications form.

Example

In this example, the contents of fields A, B, and C from the previous example are used to calculate the contents of a new field D. The calculation A+B-C=D is to be performed. Figure 5 shows the entries necessary on the Calculation Specifications form.

Specifications: The first line of the Calculation Specifications form causes the contents of field B (FLDB) to be added to the contents of field A (FLDA) and the result of this arithmetic operation to be placed in field D (FLDD), thus replacing any previous information in this field.

The second line on the form causes the object program to subtract the contents of FLDC from the contents of FLDD, and to replace the contents of FLDD by the result of this subtraction.

The numbers in the following text refer to the items circled in Figure 5.

1. The 14 in Indicator (columns 9-11) specifies the condition under which the calculations are to be performed. This number was assigned as the Record Identifying Indicator on the Input Specifications form (see Figure 4).
2. The card columns for FLDA, FLDB, and FLDC were defined on the Input Specifications form. These fields are referred to on the Calculation Specifications form by their names.
3. The result field FLDD is defined for the program by writing the name FLDD in Result Field (columns 43-48) and indicating in Field Length (columns 49-51) the number of positions that must be reserved for this field in core storage. Just as in the case of FLDA and FLDB, which were defined on the input form, the name FLDD can now be used in other calculation operations or in output specifications.

Decimal Positions: If an entry is made in Decimal Positions, the field is considered to be a numeric field. In operations involving quantities that have decimal positions in each of the fields is frequently not the same. When the number of decimal positions is not the same in both factors of an arithmetic operation, the factors usually require shifting to align the decimal point.

RPG automatically shifts factors and aligns decimal points. The programmer need only indicate the number of decimal positions required in the result field in column 52. If a field is to be used in arithmetic operations, the number of decimal positions must be specified. If there are no decimal fractions, a 0 must be entered in column 52. The fields in Figure 4 have decimal positions as indicated in column 52 of the Input Specifications form.

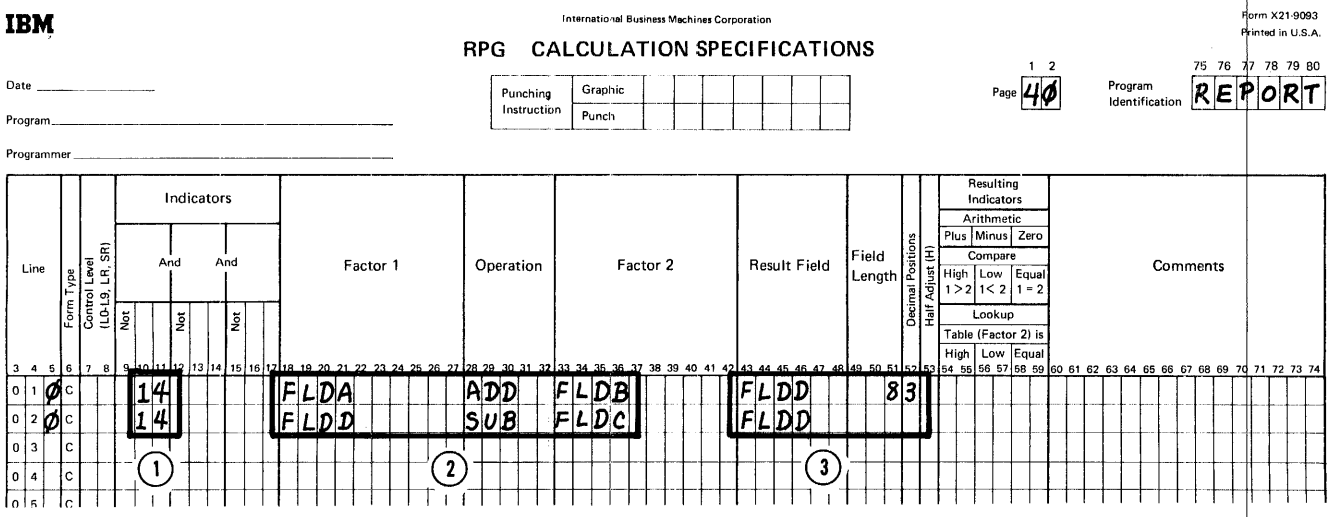


Figure 5. Calculation Specifications Form

A set of values for these fields might appear in the program as follows.

FIELD	DECIMAL POSITIONS	CONTAINED IN CARD	ACTUAL VALUE
FLDA	0	00126	126.
FLDB	2	01123	11.23
FLDC	3	04264	4.264

If, as in Figure 5, the programmer had specified three decimal positions for the result field in Decimal Positions (column 52), the calculation A+B-C would result in a shifting of the values as follows, (result field has three decimal positions):

```
00126.000 = FLDA
 +011.230 = FLDB
00137.230   FLDD
 - 04.264 = FLDC
00132.966 = FLDD
```

The result 00132.966 is stored as the value of FLDD.

DETAIL PRINTING

The printing of information obtained from each record as it is read, is called detail printing.

Example

This example illustrates how the input fields FLDA, FLDB, and FLDC and the calculated result (FLDD) can be specified for listing.

Specifications: Figure 6 shows the output specifications required. The numbers in the following text refer to the numbers circled in Figure 6.

1. A filename must be assigned to the output listing. In this example, it is named DAILYRPT. This is the same name used on the File Description Specifications in Figure 2.

   The D in Type H/D/T/E (column 15) indicates that the line to be printed is a detail line. (The other possible entries, H, T, and E are described later.)

   Space After (column 18) contains a 2 and provides a double-spaced listing. (Each printed line is followed by one blank line.)
2. The 14 in Output Indicator (columns 24-25) governs the printing of the line.
3. Each field to be printed must be specified in Field Name (columns 32-37). The Z in column 38 means that all zeros to the left of significant digits and the sign are not printed (suppressed). For example, if the value 00126 is stored in FLDA, it is printed at 126.

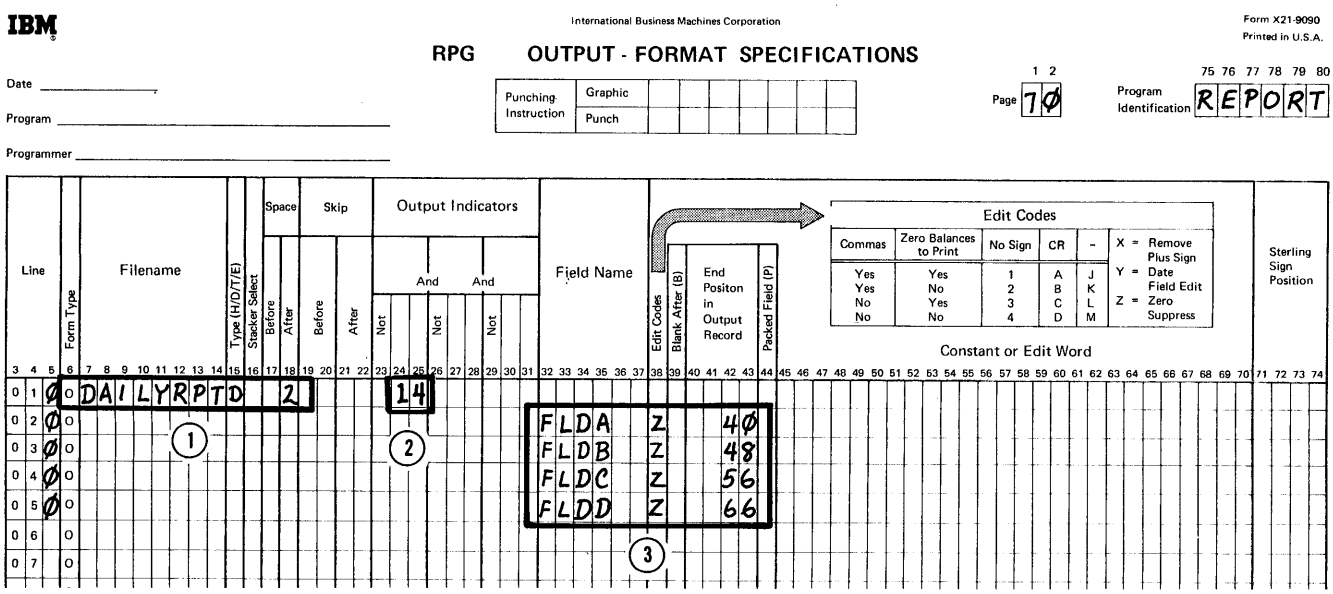


Figure 6. Output-Format Specifications Form

The positions where the data is to be printed on the output report are specified in End Position in Output Record (columns 40-43). All specifications in these columns must be right justified. The programmer indicates only the last (or right-most) print position of each field. The length of each field has been established either as part of the input specifications, or in the Result Field of the Calculation Specifications form.

When preparing a simple listing such as the one in this example, the programmer would probably mentally select print positions based upon the number of positions in each field plus a number of positions for spaces between two fields.

More elaborate reports are laid out before the pertinent program is written. Usually, this indicates both the location on the form where data is to be printed and the source of the data (the card columns of the input data or the name of data fields developed in the program). This is normally a function of job definition, which is discussed later under *Problem Definition.*

Figures 2 through 6 show how to read data into the system, how calculations can be performed upon the data, and how the original data and the data developed in the program can be printed. The examples so far have shown the items required to prepare a report. A significant item not yet described is how to specify a control field.

CONTROL FIELD ESTABLISHMENT

A field containing information to be compared from record to record is called a control field. A control break occurs when the information in the control field of a record is different from the information in the control field of the preceding record. A control level establishes the relative importance of the control fields.

Example

This example shows how to specify three control level indicators for three control fields.

Specifications: Figure 7 contains the specifications shown in Figure 4 with the addition of the appropriate control data (the items inside the circle). However, in this case, the numeric fields are specified as having no decimal fractions.

This example shows three levels of totals:

- The lowest level is the employee number (EMPNO).
- The next highest level is the department (DEPT).
- The highest level is division (DIVSON).

IBM International Business Machines Corporation

RPG INPUT SPECIFICATIONS

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

Date _____ Program _____ Programmer _____

Punching Instruction: Graphic / Punch

Page 05 Program Identification REPORT

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank	Sterling Sign Position
01	0	DETLABORAA					14				35		D	5		80	N	Z-		Z											
02	0																				1	4		DIVSON	L3						
03	0																				5	8		DEPT	L2						
04	0																				9	14		EMPNO	L1						
05	0																				15	28		NAME							
06	0																				46	50	0	FLDA							
07	0																				56	60	0	FLDB							
08	0																				66	70	0	FLDC							

Figure 7. Specifying Control Levels on the Input Specifications Form

These levels are designated by the control level codes L1, L2, and L3 placed in columns 59-60.

A maximum of nine control levels plus a final total can be used in RPG. Level 1 is the lowest control level; level 9 is the highest. A control break at one level forces control breaks for all lower levels.

To designate a control field and to establish a level of control on the Input Specifications form:

- Enter the card columns of the field in Field Location (columns 44-51).
- Enter the appropriate name in Field Name (columns 53-58).
- Enter the correct control level (L1, L2,. . .L9) in Control Level (columns 59-60).

TOTAL CALCULATIONS

When a control break occurs, certain special operations are normally performed before processing the record that caused the control break. These operations are called total calculations.

Example

This example shows how totals can be accumulated at each control level.

Specifications: Figure 8 shows calculations that can be performed on the input data shown in Figure 7. The examples in the first and second lines of Figure 8 are the same as in Figure 5. The third line adds the result, which is contained in FLDD, to a field called TOTE. Thus, TOTE is the accumulated amount for each employee group.

The operations specified in the first three lines are performed each time a detail card is read.

When the level 1 (L1) control break occurs (i.e., a card containing a different employee number has been read), specifications line 040 adds the contents of TOTE (accumulated from each detail card) into a field named TOTF. This is used to accumulate the total for the employee group of each department.

When the level 2 (L2) control break occurs (change in department) in addition to the previous calculation, the contents to TOTF are added to a field named TOTG.

When the level 3 (L3) control break occurs, all three total calculations specified with L1, L2, and L3 are performed.

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page 40

Program Identification: REPORT

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
010	C		14	FLDA	ADD	FLDB	FLDD	70				
020	C		14	FLDD	SUB	FLDC	FLDD	70				
030	C		14	TOTE	ADD	FLDD	TOTE	80				
040	C	L1		TOTF	ADD	TOTE	TOTF	80				
050	C	L2		TOTG	ADD	TOTF	TOTG	80				
060	C	L3		FINTOT	ADD	TOTG	FINTOT	80				
070	C											

Figure 8. Specifying Control Levels on the Calculation Specifications Form

DETAIL AND TOTAL PRINTING

Example

This example shows the specifications required to print the following:

- Three control fields.
- The name.
- The amount accumulated in FLDD.
- The three accumulated totals (when a control break occurs at level 1, level 2, and level 3).
- The final total at the end of the report.

Specifications: Figure 9 illustrates the specifications required for the printed report shown in Figure 10. The numbers in the following text refer to the circled items in Figure 9.

1. This information is similar to the specifications in Figure 6. The 1 in Space After (line 010, column 18) provides one space between the printed lines of the report; i.e., there are no blank lines between the printed lines of the report.
2. The data fields named DIVSON, DEPT, EMPNO, NAME, and FLDD are specified for printing.
3. The specifications to print the total for the first control level shown on lines 070 and 080 are:
   - T in Type H/D/T/E (column 15) indicates that the line is a total line.
   - 3 in Space After (column 18) provides two blank lines after each total line to make the report easier to read.
   - L1 in Output Indicator (columns 23-25) indicates that the line is to be printed each time a level 1 or any higher control break occurs.
   - The field to be printed (TOTE) is indicated under Field Name (columns 32-37).

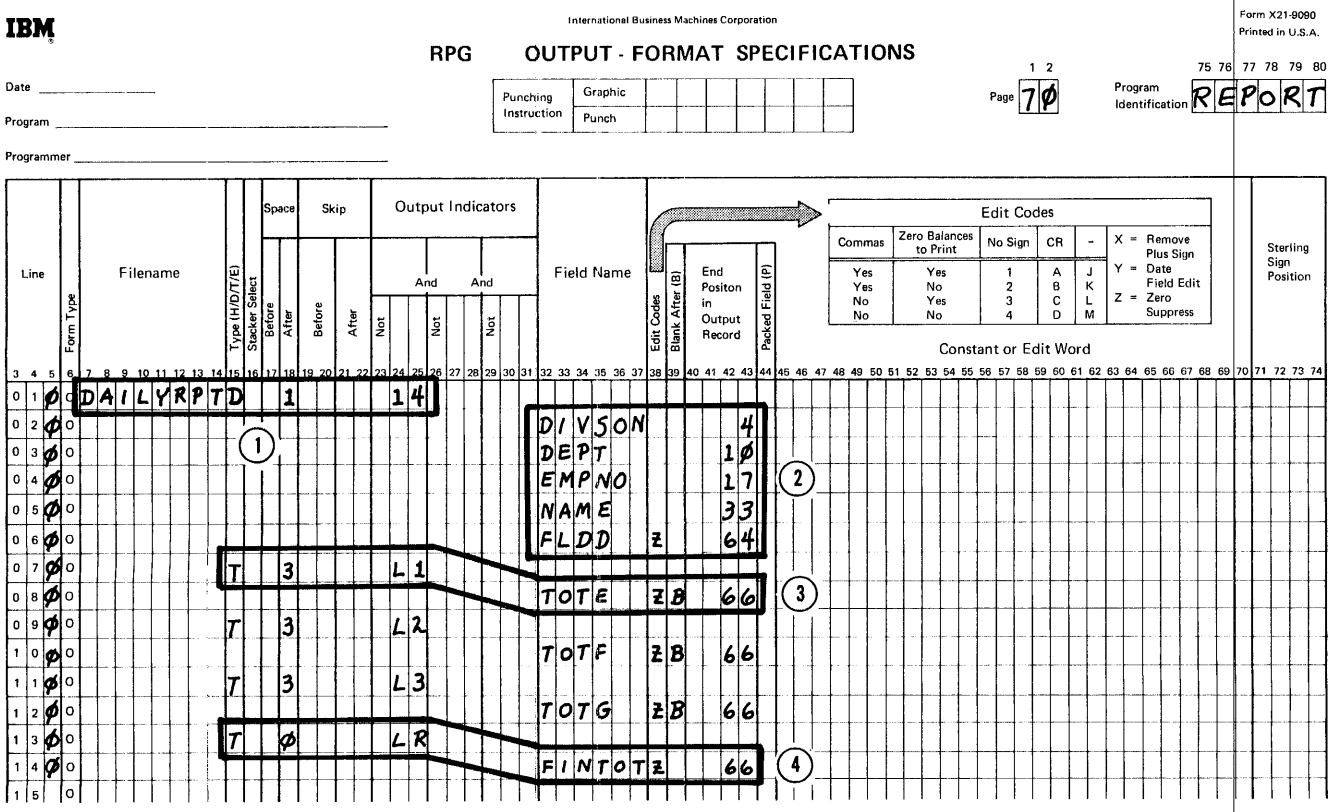


Figure 9. Specifications on the Output-Format Specifications Form

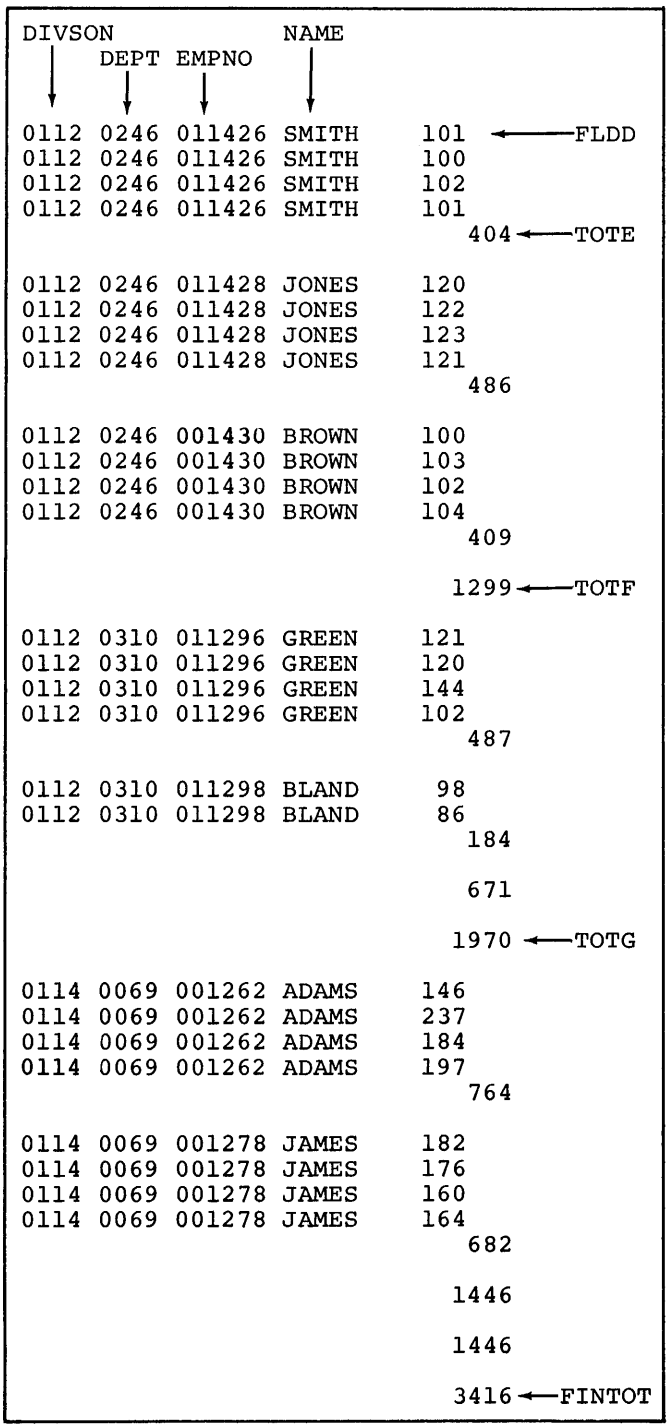


Figure 10. Detail-Printed Report

- Z in Edit Codes (column 38) indicates that zeros to the left of significant digits and the sign are *not* to be printed.

- B in Blank After (column 39) causes the core-storage positions containing the field TOTE to be set to zeros after the total has been transferred to the print line. This is done to avoid adding the total for one group to the total for the previous groups.

The specifications to print the second and third control levels are essentially the same as those for the first level.

4. The specifications for printing the final total contain the code LR (Last Record) in Output Indicators (columns 23-25). LR is turned on by the program at the end of the job, and causes the final total to be printed. At this time, all control-level indicators (L1-L9) are turned on and all total lines are printed in the specified sequence.

   The end of the job is denoted by an end-of-job card; that is, a card with a /* in columns 1 and 2. This card must follow the last data card in the card reader.

GROUP PRINTING

In group-printing operations, only one line is printed for each group of detail cards. This line usually contains the control fields and the totals of the quantity fields. An example of a group-printed report is shown in Figure 11.

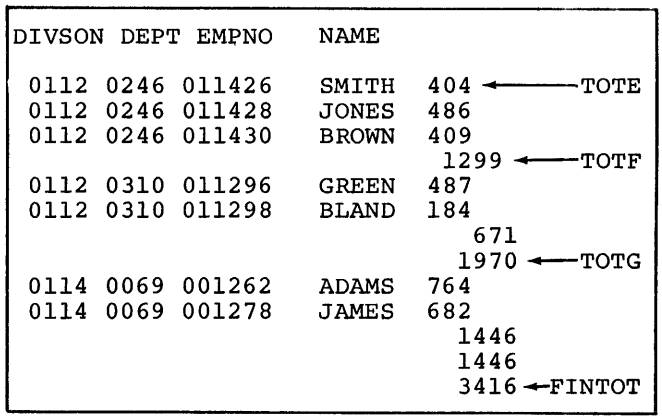


Figure 11. Group-Printed Report

The specifications for the detail-printed report shown in Figure 9 could be altered as illustrated in Figure 12 to provide a group-printed report. The difference between Figure 9 and Figure 12 is:

1. The detail line specification in Figure 9 (line 010) has been changed to a total line specification on an L1 control break (Figure 12, line 010).
2. The total line in Figure 9 (line 070) has been combined with the first total line 010 of Figure 12.
3. The space-after specification on lines 06090 and 06110 of Figure 9 has been changed in Figure 12 from three spaces to one space after printing.

GROUP INDICATION

In group-indication operations, each detail card is processed. However, only the control fields that identify the specific detail card are printed.

Figure 13 shows a group-indicated report. In this example, name and number of each employee are printed at each control change (level 1). The fields for division and department are printed only when a control break occurs at L2 or L3, respectively. Figure 14 illustrates how the specification for the detail-printed report shown in Figure 9 could be altered to specify a group-indicated report.

Control level indicators can be used to condition fields so that the field will print from the first card of each control group.

PAGE HEADINGS

Page headings can be easily prepared with RPG. Headings at the top of each page improve the readability of the report. Including headings on pages other than the first page is called overflow printing.

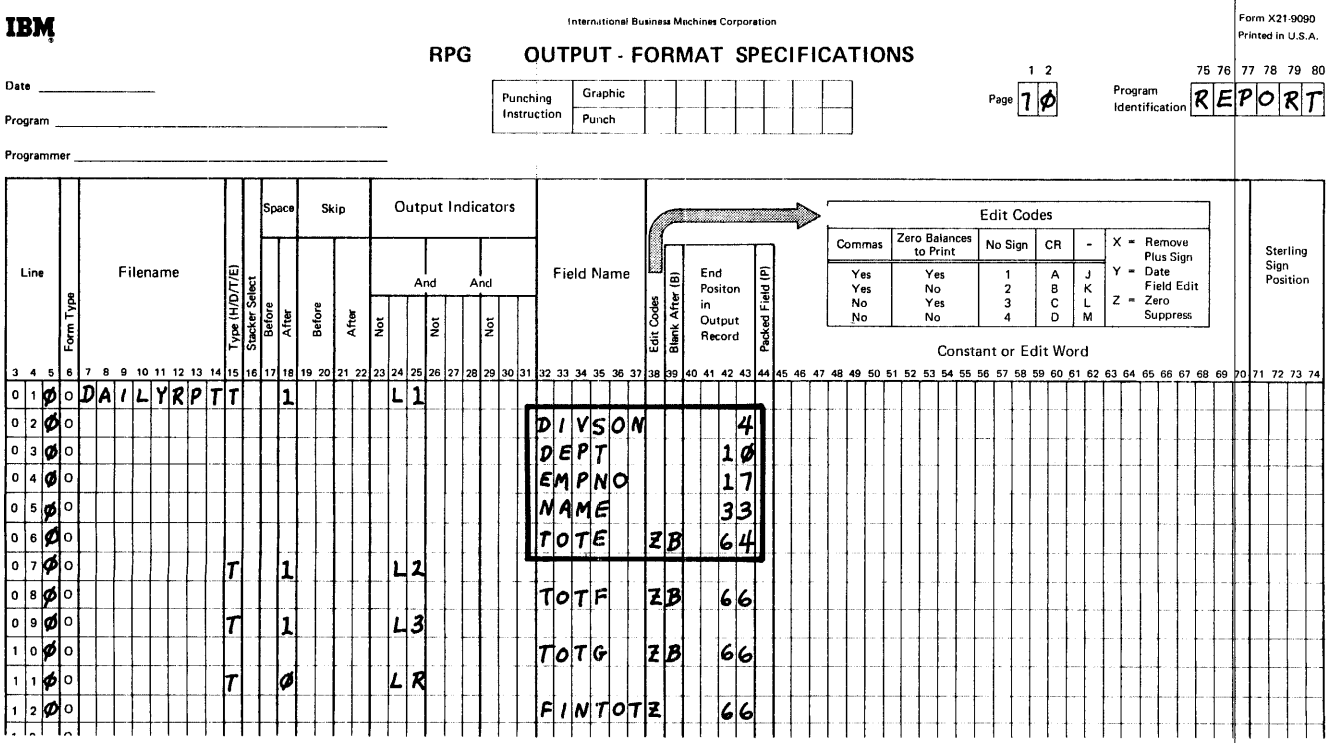


Figure 12. Specifying a Group-Printed Report

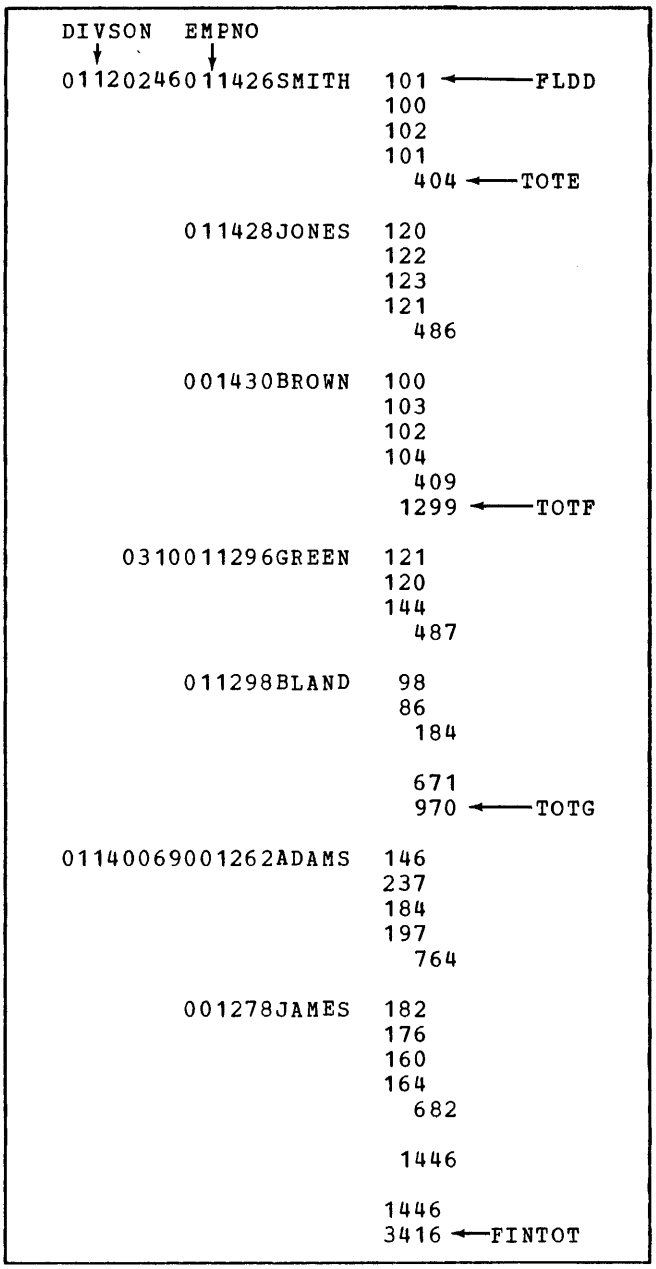


Figure 13. A Group-Indicated Report

OVERFLOW PRINTING

Overflow printing, another function used in preparing reports, can be performed by RPG. An overflow occurs when a punch in channel 12 of the carriage-control tape is encountered.

Example

This example shows the specifications required to print eight column headings at the top of each printed form. Heading lines may contain information from an input record (e.g., the date) or constant information that is defined on the Output-Format Specifications form.

Specifications: The output specifications required for this operation are illustrated in Figure 15. The numbers in the margin of the text refer to the circled numbers in Figure 15. The filename (DAILYRPT) is the same as in the other specifications required for the report.

1. On the first line of the form, the H in Type H/D/T/E (column 15) indicates that the line to be printed is a heading line. The 3 in Space After (column 18) provides two blank lines after each printed heading. The 01 in Skip Before (columns 19-20) causes the printer carriage to skip to a punch in channel 1 of the carriage-control tape before printing the line.

   The OF in Output Indicator (columns 23-25) causes the heading line to be printed each time an overflow condition occurs. The OF indicator must have been specified on the File Description Specifications form for the device associated with the filename DAILYRPT.

   The letters OR on the second line (columns 14-15) of Figure 15 indicate a second condition that can cause the heading line to be printed. This is specified by means of the entry 1P in columns 23 and 25. The entry 1P is required to have the heading printed on the first page of the report because the overflow condition does not occur until after at least one page has been printed. Indicator 1P is on only at the beginning of an RPG program.
2. The entries on lines 030-100 of Figure 15 specify the actual or constant information to be printed in the heading of the report. Note that an apostrophe must be placed before and after each constant defined.
3. The specifications for printing the contents of detail cards follow the last specifications for the heading line.

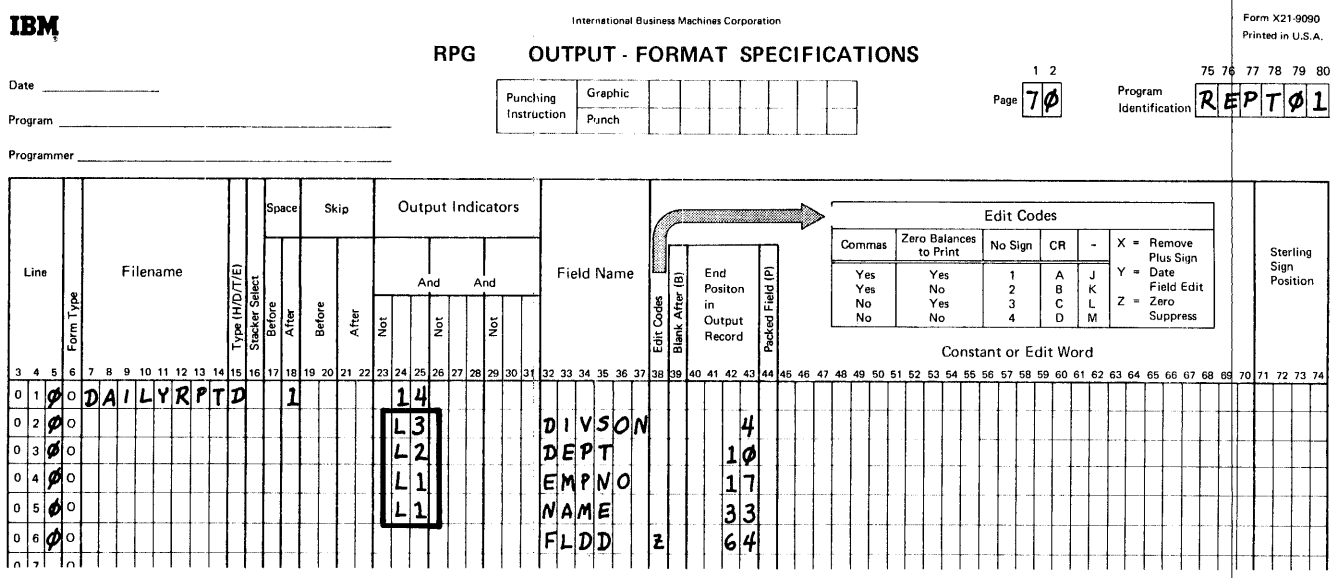


Figure 14. Specifying a Group-Indicated Report

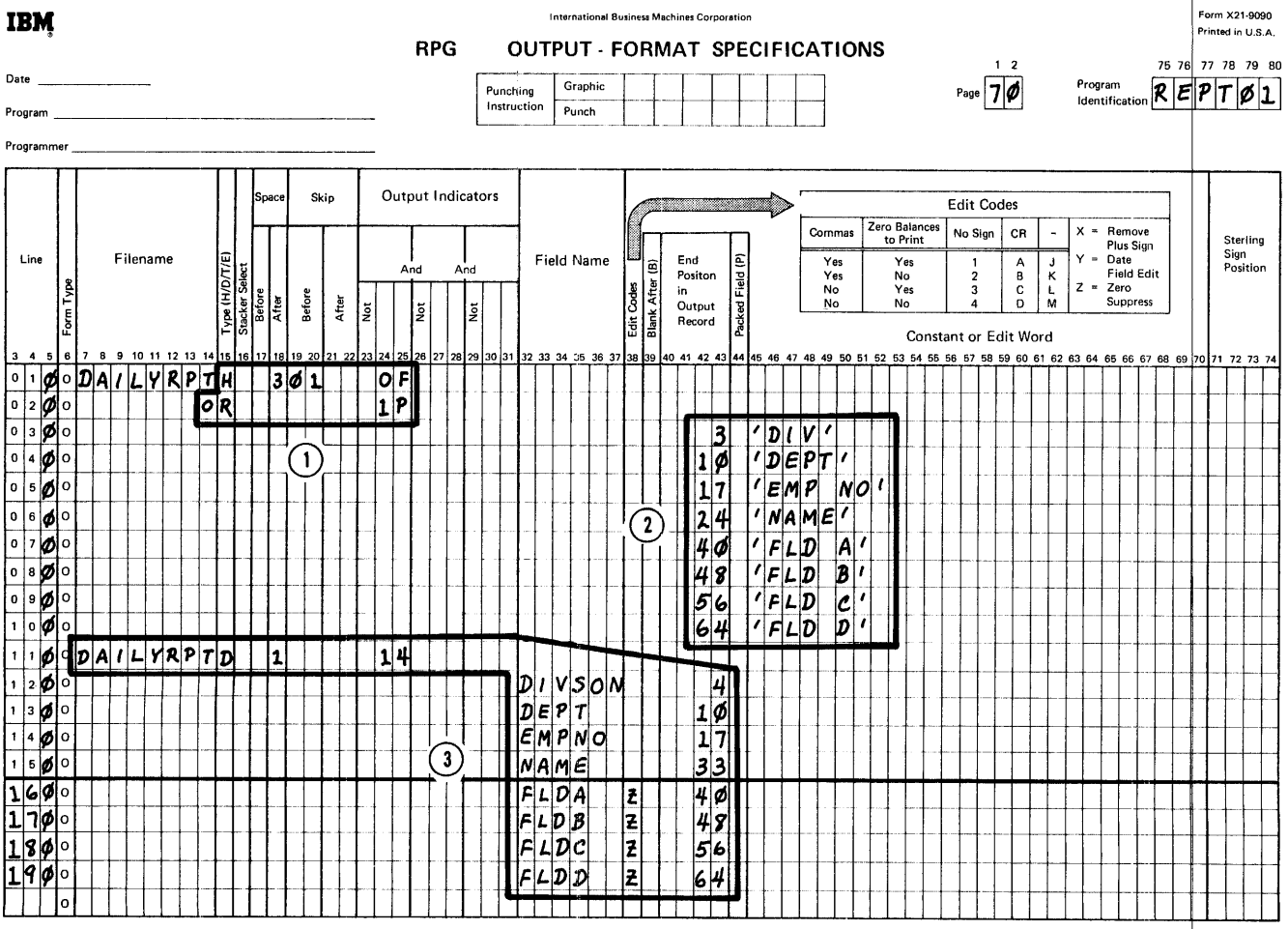


Figure 15. Specifications for Overflow Printing

SUMMARY PUNCHING

Summary punching is the function whereby one card is punched with information taken from many cards. For instance, information in one field of several input cards may be summed and punched into a separate card. This reduces card volume and simplifies further processing.

Example

In this example, a summary total of employee totals (TOTE) is to be punched for each department together with the appropriate department and division numbers.

Specifications: Before processing of the object program begins, a blank card is inserted behind each department group. The blank card is specified on the Input Specifications form as illustrated in Figure 16, line 090.

The calculation specifications for the summary totals to be punched are illustrated in Figure 17. Line 040 is processed when the blank card is read. It causes the accumulation of the level 2 total for the summary card.

This specification is required because the summary cards are inserted behind the control groups rather than obtained from a second file of blank cards. The level 2 control break does not occur until the summary card is punched and the first card of the next group is read. Therefore, a separate total is kept for punching.

The entries required for punching accumulated values are illustrated on the Output-Format Specifications form in Figure 18. The specifications required for this operation are circled. The overflow specifications of the former example have been omitted to make the example easier to read. The cards to be punched will become a new output file. Therefore, the combined file is named DETLABOR (line 120, columns 7-14 of Figure 18).

The D in column 15 indicates that the operation is to be performed at detail time.

The entry in Output Indicators (columns 23-25) indicates that the operation is to occur at the time the blank card is encountered.

IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic							
	Punch							

Page 1 2: 1Ø

Program Identification 75 76 77 78 79 80: REPTØ1

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	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
3 4 5	6	7 8 9 10 11 12 13 14	15 16	17	18	19 20	21 22 23 24	25	26	27	28 29 30 31	32	33	34	35 36 37 38	39	40	41	42	43	44 45 46 47	48 49 50 51	52	53 54 55 56 57 58	59 60	61 62	63 64	65 66	67 68	69 70	71 72 73 74
01 Ø	I	DETLABORAA				1A	35		D	5	8ØN	Z		K																	
02 Ø	I																				1	4		DIVSON	L3						
03 Ø	I																				5	8		DEPT	L2						
04 Ø	I																				9	14		EMPNO	L1						
05 Ø	I																				15	28		NAME							
06 Ø	I																				46	5ØØ		FLDA							
07 Ø	I																				56	6ØØ		FLDB							
08 Ø	I																				66	7ØØ		FLDC							
09 Ø	I		NS			16	35		C																						
10	I																														

Figure 16. Summary Punching Example, Input Specifications

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic							
	Punch							

Page 40

Program Identification REPT01

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And	Indicators: And	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
03 04 05	06	07 08	09 10 11	12 13 14	18 19 20 21 22 23 24 25 26 27	28 29 30 31 32	33 34 35 36 37 38 39 40 41 42	43 44 45 46 47 48	49 50 51	52	53	54 55	56 57	58 59	60 61 62 63 64 65 66 67 68 69	70 71 72 73 74					
01	C		14		FLDA	ADD	FLDB	FLDD	70												
02	C		14		FLDD	SUB	FLDC	FLDD													
03	C		14		FLDD	ADD	TOTE	TOTE	80												
04	C		14		TOTE	ADD	TOTF	TOTPCH	80												
05	C	L1			TOTE	ADD	TOTF	TOTF	80												
06	C	L2			TOTF	ADD	TOTG	TOTG	80												
07	C	L3			TOTG	ADD	FINTOT	FINTOT	80												
08	C																				

Figure 17. Summary Punching Example, Calculation Specifications

To identify the card as a summary card, an 8 is to be punched in column 35. This is specified on line 011. The remaining three fields are specified as described before. The format of the summary card is shown in Figure 19.

TOTPCH is blanked after punching to reset the field to zeros for the next group.

ZERO, PLUS, AND MINUS BALANCE TEST

During the execution of an object program, operations are performed in the same sequence in which the specifications are written on the specifications form, unless a different sequence is specified.

If specifications had to be followed sequentially in a fixed pattern, a program would follow a single path of operation without any possibility of dealing with predefined exceptions to the procedure. Moreover, the program could not be coded to follow a predefined alternative to the procedure, based upon conditions encountered during execution.

For example, assume that a program contains five consecutive specifications known to be meaningless whenever the quantities they refer to are zero. In this case, execution time could be reduced if the program could be made to bypass these specifications whenever it detects that the quantity to be processed is zero.

An entry on the Input Specifications form can be used to test a quantity and, depending upon the value of that quantity, set an indicator that will direct the program to some other specification. Three types of tests can be specified:

1. Testing an input field to determine if it contains a positive value.
2. Testing an input field to determine if it contains a positive value.
3. Testing an input field to determine if it is blank or contains zero.

Three types of tests can also be made on the Calculation Specifications form to determine whether the result of a calculation is positve, negative, or zero.

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 7Ø

Program Identification REPTØ1

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators And Not	And Not	And Not	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
01Ø	O	DAILYRPTD			1				14									
02Ø	O											DIVSON			4			
03Ø	O											DEPT			1Ø			
04Ø	O											EMPNO			17			
05Ø	O											NAME			33			
06Ø	O											FLDA	Z		4Ø			
07Ø	O											FLDB	Z		48			
08Ø	O											FLDC	Z		56			
09Ø	O											FLDD	Z		64			
10Ø	O	DETLABORD							16									
11Ø	O														35		'8'	
12Ø	O											DIVSON			4			
13Ø	O											DEPT			8			
14Ø	O											TOTPCH	B		79			
15Ø	O	DAILYRPTT			3				L1									
16Ø	O											TOTE	ZB		66			
17Ø	O	DAILYRPTT			3				L2									
18Ø	O											TOTF	ZB		66			
19Ø	O	DAILYRPTT			3				L3									
2ØØ	O											TOTG	ZB		66			
21Ø	O	DAILYRPTT			Ø				LR									
22Ø	O											FINTOTZ			66			

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 18. Summary Punching Example, Output Format Specifications

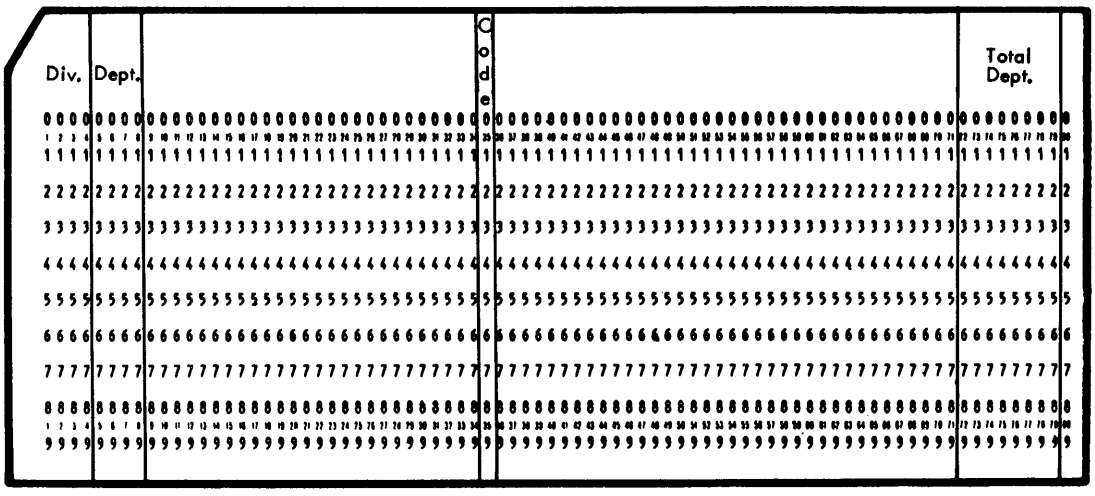


Figure 19. Summary Punching Example, Summary Card Format

Example

Figure 20 illustrates a test ot determine if FLDA contains zero. Figure 22 shows the specifications for a test to determine if FLDD (as the result field) contains a negative value.

Specifications: Figure 20 illustrates a typical test of the contents of an input field. To make this test, the programmer places a number from 01 through 99 in Field Indicators (columns 69-70). In this example, indicator 18 is used. If a card that is read into the system contains zeros or blanks in columns 46-50 (FLDA) indicator 18 is turned on. Thus, a special condition has been created.

The program must consider this condition during the execution of calculation and/or output specifications. This indicator condition has the same effect as having a record identifying indicator turned on by a record identification code from an input record.

The entries circled in Figure 21 are the additional specifications that bypass calculations for detail cards that contain zeros or blanks in FLDA. The specifications 14N18 in columns 10-14 of lines 010-030 cause the calculation to be performed if indicator 14 is on and indicator 18 is not on. (The N in column 12 stands for *Not.*)

If desired, the same indicator specification, N18, can be used on the Output Format Specifications form to prevent the printing of detail cards when FLDA is zero.

Example

The calculation specifications in Figure 22 illustrate the use of a field indicator in testing a value to determine if it is negative. The result of this test can be used to bypass certain specifications whenever the tested field contains a negative value.

Specifications: The specifications for this example are shown in Figure 22. The program logic for this example is shown in Figure 23.

Line 010 specifies that FLDB is to be added to FLDA and that the result is to be placed into FLDD. Line 020 specifies that FLDC is to be subtracted from FLDD and that the result is to be placed into FLDD. In addition, line 020 causes FLDD to be tested to determine if its contents are negative. This example assumes that, if the value in FLDD is negative, the calculations involving FLDD are meaningless and should be bypassed. If FLDD contains a negative value, the following is required:

1. The result must not be added into the field TOTE.
2. The contents of F1DD must be reset to zeros. (This step is required whenever the contents of a result field are used in a subsequent operation in which a negative quantity would produce incorrect results.)

This is done on the specifications form (see Figure 22) by placing an indicator (code 19) into Resulting Indicators (columns 56-57). Indicator 19 is turned on whenever FLDD contains a negative value.

IBM

Internat onal Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page 05

Program Identification: REPT01

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	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
010	I	DETLABORAA				14	35		D	5	8		N	Z-					2													
020	I																				1	4		DIVSON	L3						
030	I																				5	8		DEPT	L2						
040	I																				9	14		EMPNO	L1						
050	I																				15	28		NAME							
060	I																				46	50		FLDA						18	
070	I																				56	60		FLDB							
080	I																				66	70		FLDC							
090	I																														

Figure 20. Specifying a Test for a Zero Balance

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page 40

Program Identification: REPT01

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, Minus, Zero)	Compare (High 1>2, Low 1<2, Equal 1=2)	Lookup Table (Factor 2) is (High, Low, Equal)	Comments
010	C		14 N18	FLDA	ADD	FLDB	FLDD	70						
020	C		14 N18	FLDD	SUB	FLDC	FLDD	70						
030	C		14 N18	FLDD	ADD	TOTE	TOTE	80						
040	C	L1		TOTE	ADD	TOTF	TOTF	80						
050	C	L2		TOTF	ADD	TOTG	TOTG	80						
060	C	L3		TOTG	ADD	FINTOT	FINTOT	80						
070	C													

Figure 21. Calculations Based on a Test for a Zero-Balance

The operation of adding FLDD into TOTE is specified on line 030. It is performed only if indicator 19 is off. This is specified by entering N19 in columns 12-14.

The 0 in Factor 2 on line 040 is a numeric literal and is used to set FLDD to zeros. This is done by the zero-and-add (Z-ADD) operation. A literal is an actual value (constant) to be used in a calculation. The Z-ADD operation sets FLDD to zero and adds a value to it. In this case, the value added to FLDD is zero. The remaining specifications in Figure 22 are the same as those from previous examples.

If detail cards were not to be printed whenever the result of the specification in line 020 was negative, indicator 19 would also have been used on the pertinent Output-Format Specifications form.

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page 40

Program Identification: REPORT

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, Minus, Zero)	Compare (High 1>2, Low 1<2, Equal 1=2)	Lookup Table (Factor 2) is (High, Low, Equal)	Comments
010	C		14	FLDA	ADD	FLDB	FLDD	70						
020	C		14	FLDD	SUB	FLDC	FLDD					19		
030	C		14N19	FLDD	ADD	TOTE	TOTE	80						
040	C		19 14		Z-ADD	0	FLDD	70						
050	C	L1		TOTE	ADD	TOTF	TOTF	80						
060	C	L2		TOTF	ADD	TOTG	TOTG	80						
070	C	L3		TOTG	ADD	FINTOT	FINTOT	80						
080	C													

Figure 22. Testing for a Minus Condition

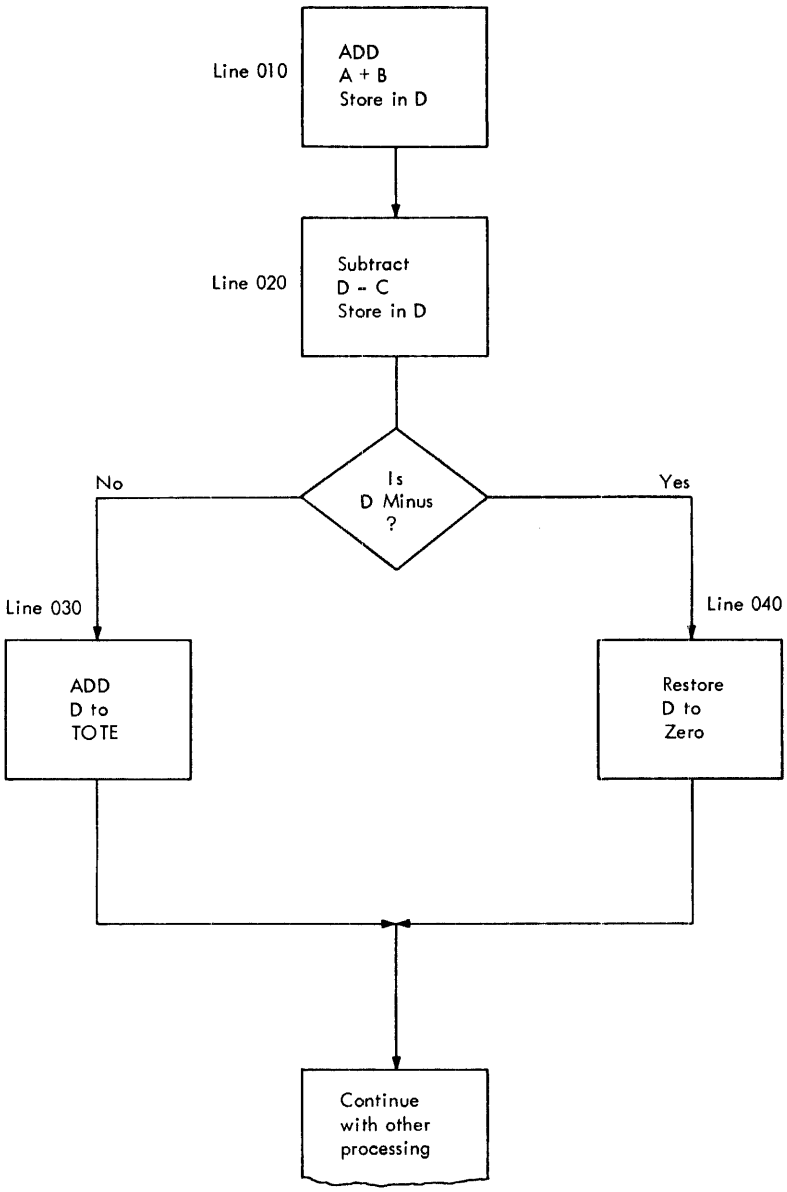


Figure 23. Testing Indicators to Govern Processing

FIELD COMPARISON

The calculation specifications in this example illustrate a comparison of two fields and the subsequent processing governed by the result of this comparison. The COMP operation is used to compare two fields and to determine whether the contents of the field in Factor 1 are greater than (high), equal to (equal), or smaller than (low) those in Factor 2. The COMP specification turns on the appropriate indicators specified in Resulting Indicators. These indicators can be used to bypass subsequent specifications or to have some specific operations performed whenever a particular condition has occurred.

Specifications: The calculation specifications for a compare problem are shown in Figure 24. The program logic for the problem is shown in Figure 25.

Line 010 specifies that FLDA is to be compared with FLDB. If the two fields are equal, indicator 18 is turned on and the contents of FLDA are placed in HOLD (line 020). If the two fields are not equal (line 030), FLDB is subtracted from FLDA and the result is placed in SAVE.

A literal may also be used in a compare specification. In the lower half of Figure 24, the contents of the input field DATE are compared to 06-09-65. Indicator 19 is turned on if they are equal.

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page 40

Program Identification: REPORT

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And / Not	Indicators: And / Not	Indicators: 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
010	C					FLDA	COMP	FLDB					18	
020	C		18				MOVE	FLDA	HOLD	102				
030	C		N18			FLDA	SUB	FLDB	SAVE	102				
070	C					DATE	COMP	'06-09-65'					19	
080	C													
090	C													

Figure 24. COMP Operation, Coding

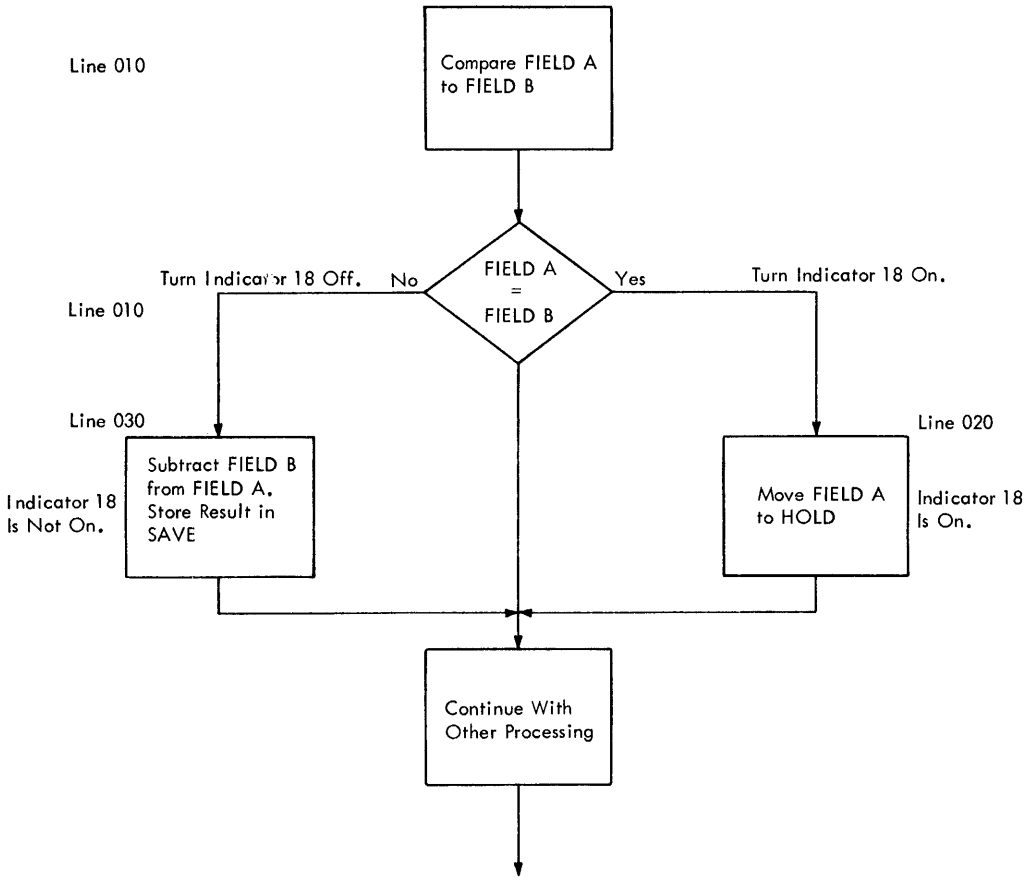


Figure 25. COMP Operation, Logic

MULTIPLICATION AND DIVISION

Multiply and divide operations are easily accomplished with RPG. Moreover, two problems associated with these functions (decimal-point alignment and half-adjusting) are easily specified.

Example

This example illustrates an inventory problem. Input consists of inventory cards whose format is shown in Figure 26. For each part number, the quantity (QTY) is to be multiplied by the price to obtain the total cost per week.

The second part of the problem is to update the price each week so that the price used in the subsequent weekly calculation reflects the average fabrication cost and scrap losses on a cumulative (year-to-date) basis.

Specifications: The specifications to solve this problem are shown in Figures 27 and 28. Figure 27 shows the input specifications for all five fields. The programmer must be certain of the exact length of each field. He must also know the number of decimal positions within each field. The fields in this example are:

PARTN	XXXXX	No decimals (alphameric)
PRICE	XX.XXX	3 decimals
QUANTY	XXXX.	0 decimals
YDCOST	XXXXXX.XX	2 decimals
YDUSE	XXXXXX.	0 decimals

As shown in Figure 27, the number of decimal positions in each field is entered under Decimal Positions on the Input Specifications form.

The calculation specifications for this example are shown in Figure 28.

PRICE is multiplied by QUANTY and the product is placed in the result field named COST. COST is specified an nine positions long. This includes two decimal positions, specified in column 52. The H in column 53 specifies that the product of PRICE and QUANTY is to be half adjusted (rounded).

The following illustrates this operation.

```
PRICE     1.213
QUANTY  x 216
         7278
        1213
       2426
       262.008
```

Since half adjustment is always made in the position to the right of the last decimal position to be retained in the result field, the half adjustment in this example is made to the digit 8. To do this:

```
262.008
+     5
262.013
```

The rightmost decimal position (3) is truncated and the value 262.01 is placed in the result field COST.

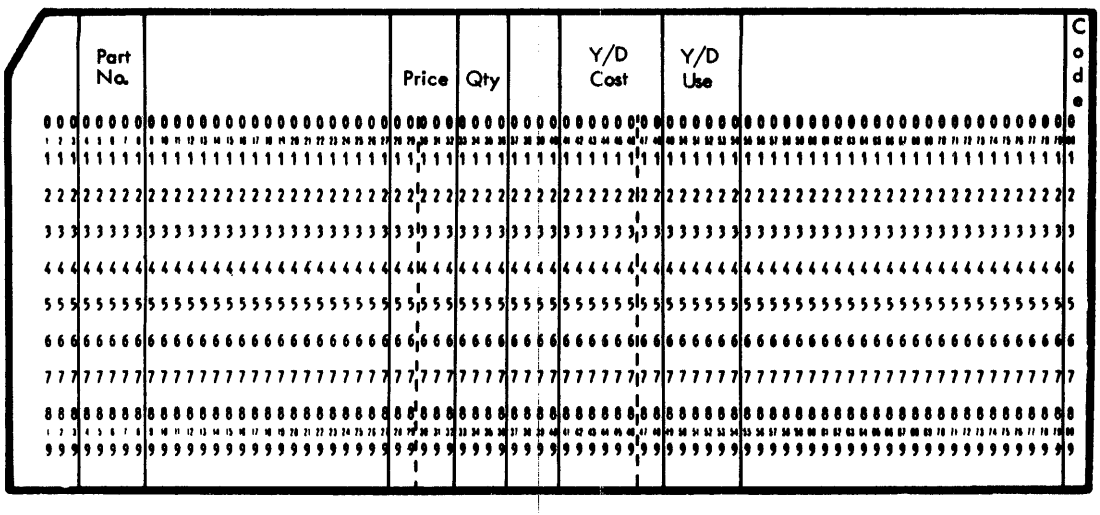


Figure 26. Inventory Card

IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page 10

Program Identification: REPORT

Line	Form Type	Filename	Sequence	Number (1-N) / Option (O)	Record Identifying Indicator	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	Packed (P)	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank	Sterling Sign Position
01	I	INVENTRYAA				29		80		Z-																				
02	I																			4	8		PARTN							
03	I																			28	32	3	PRICE							
04	I																			33	36	0	QUANTY							
05	I																			41	48	2	YDCOST							
06	I																			49	54	0	YDUSE							
07	I																													

Figure 27. Input Specifications for Inventory Multiplication Problem

The second line of the Calculation Specifications form in Figure 28 provides the specifications for dividing the year-to-date cost by the year-to-date usage. The result is placed in a field named PRICE. PRICE is specified as five positions long, including three decimal positions. The result is to be half adjusted as specified by the H in Half Adjust. The following is an example of the arithmetic of this operation, using actual values:

$$\text{PRICE} = \frac{\text{YDCOST}}{\text{YDUSE}} = \frac{1739.23}{1296} = 1.3419$$

The result field was specified to have three decimal positions. Therefore, the half adjustment is made to the digit 9 in the rightmost position since this is in the position to the right of the last position to be retained in the result field.

$$\begin{array}{r} 1.3419 \\ +\quad 5 \\ \hline 1.3424 \end{array} \quad \text{(half adjust)}$$

The result (1.342) is stored in PRICE.

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page 40

Program Identification: REPORT

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And Not	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		29		PRICE	MULT	QUANTY	COST		9	2H										
02	C		29		YDCOST	DIV	YDUSE	PRICE		5	3H										

Figure 28. Calculation Specifications for Inventory Multiplication Problem

SEQUENCE CHECKING

Two types of sequence checking can be performed by RPG:

1. Record types.
2. Match fields. (See the section *"Using the Matching Fields Specification for Sequence Checking".*)

Sequence Checking of Different Record Types

This example shows how to update an invoice file that contains two, three, or four of the following card types for each part number:

Card	*Code*
Master customer	5 in column 78
Shipped via	3 in column 78
Part number	2 in column 78
Comments	8 in column 78

Figure 29 illustrates the specifications required to check the sequence of a group of four record types. The item numbers in the following text refer to the circled numbers in Figure 29.

1. The record-identification codes for the four card types are specified in the same manner as in previous examples. The C in C/Z/D (column 26) specifies that the entire character punched in the card is to be examined to establish the record-identification code.

   In the specifications for the last card-type (line 130), a D is written in C/Z/D because the D specifies that only the digit punches in the card are examined to identify the card type. Thus, zone punches which could cause unequal comparisons are ignored.
2. The sequence established for the file is determined by the sequence in which the specifications for each record type are written on the form. The numbers must be in ascending sequence. In this example, the digits 01, 02, 03, and 04 are used. For each file, the numbers assigned must begin with 01.

IBM — International Business Machines Corporation — Form X21-9094 Printed in U.S.A.

RPG INPUT SPECIFICATIONS

Date _____ Program _____ Programmer _____

Punching Instruction: Graphic / Punch

Page 10 — Program Identification: REPORT

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank	Sterling Sign Position
010	I	INVOICE	01	1		21	78		C	5																					
020	I																				1	16		ACOUNT							
030	I																				17	20		CUSNO							
040	I																				40	46		ADDRS							
050	I		02	1	O	22	78		C	3																					
060	I																				1	6		SHIPER							
070	I																				28	31		LOCATN							
080	I																				60	66		RTE							
090	I		03	N		23	78		C	2																					
100	I																				1	6		PARTNO							
110	I																				7	10		NUMORD							
120	I																				40	46		DESC							
130	I		04	N	O	24	78		D	8																					
140	I																				1	6		DISCNT							
150	I																				28	31		NEWSAL							
160	I																				60	66		DUEDAT							

(2) (3) (1)

Figure 29. Input Specifications, Sequence Checking

Alphabetic characters under Sequence in preceding examples indicate that no sequence checking is to take place. Alphabetic specifications (indicating no sequence checking) must always be written before numeric specifications for any one file.

When a sequence error occurs, the operator is given the option of either (1) ending the job or (2) bypassing the record which caused the error and continuing the job.

3. If a numeric specification is provided in Sequence, a specification must be provided in Number.

On the first two record identification lines, the digit 1 in Number indicates that only one record of that type must be present before the next record type is checked. In this example, only one master record and one shipped-via record must be present before the next record type is read. If there are more than one of either of these records, the program assumes an error in the input file.

The letter N in Number of the specifications for the last two record types means that one or more records of this type may be present before the next record type is read. In this example, one or more part number and comments records may be present.

The letter O in Option specifies that the presence of record(s) of this type is optional.

If the letter O is not specified, at least one record of the particular type must be present. This requirement applies only if Sequence contains a numeric specification.

While sequence checking of record types is normally done to assure the proper makeup of a control group, the checking is independent of control levels. It only ensures that the record types are in sequence; it does not ensure that they all belong to the same control group.

Figure 30 shows a file-to-file sample program that illustrates the correlation of the RPG specification forms.

Assume that the contents of an input file are to be transferred to another file. In this example, the input file is a card file and the contents of the cards are to be listed on the printer. Three specification forms are required: Control Card and File Description, Input, and Output-format.

CONTROL CARD FORM

Only an H is needed in column 6.

FILE DESCRIPTION FORM

The card input file is named INPUT. The printed output file is named OUTPUT. Page and line sequence numbers are entered in columns 1-5, and the program identification is entered in columns 75-80. An F in column 6 indicates that each entry is a File Description entry. The file name is entered in columns 7-14. Column 15 contains an I or an O to indicate whether the file is an input or an output file.

Column 16 on line 010 contains a P to denote the primary input file for the job.

Column 19 contains an F in line 010 and in line 020 to indicate that the file formats are fixed length. (1130 RPG records are always fixed length.) Record length (columns 24-27) is 80 for the card input file and 120 for the printer output file.

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 RP0001

Control Card Specifications

Line	Form Type	Core Size to Compile	Object Output	Listing Options	Core Size to Execute	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	Refer to the specific System Reference Library manual for actual entries.
3 4 5	6	7 8 9	10	11	12 13 14	15	16	17	18	19	20	21	22 23 24 25	26	27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
01	0	H													

File Description Specifications

Line	Form Type	Filename	I/O/U/C	P/S/C/R/T	E	A/D	F/V	Block Length	Record Length	L/R	K/I	I/D/T or 1-9	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, or E)	Name of Label Exit	Extent Exit for DAM	A	N/U	Tape Rewind	Number of Extents	Number of Tracks for Cylinder Overflow
3 4 5	6	7 8 9 10 11 12 13 14	15	16	17	18	19	20 21 22 23	24 25 26 27	28	29 30	31	32	33 34	35 36 37 38		39	40 41 42 43 44 45 46	47 48 49 50 51 52	53	54 55 56 57 58 59	60 61 62 63 64 65	66	67	68 69	70	71 72 73 74	
02	F	INPUT	I	P			F		80									READ01									
03	F	OUTPUT	O				F		120									PRINTER									
04	F	* CARD TO PRINT PROGRAM																									
05	F																										

Figure 30. File-to-File Program (Part 1 of 2)

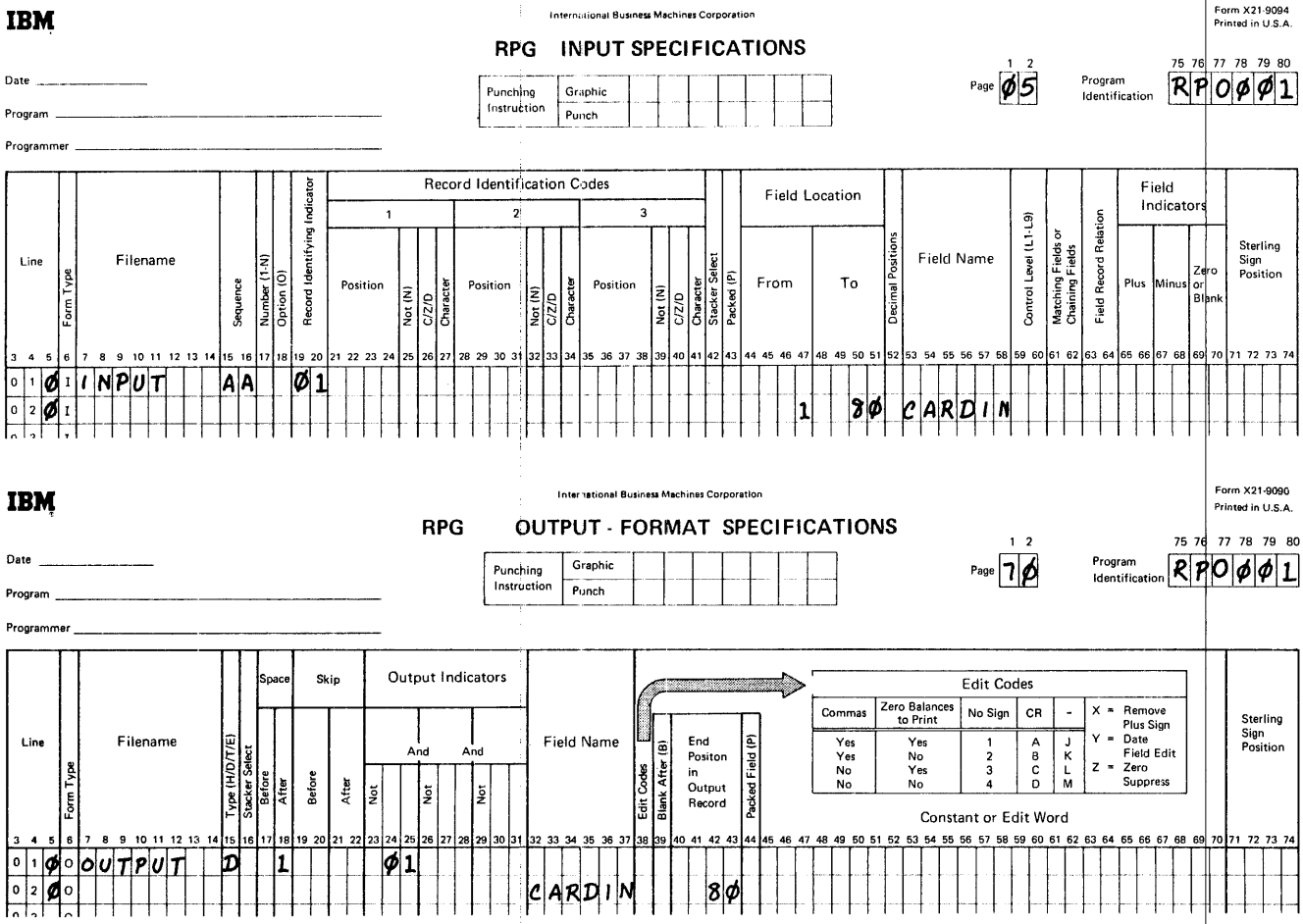


Figure 30. File-to-File Program (Part 2 of 2)

The entry READ01 in Device specifies that the input file is to be read from an IBM 2501 Card Reader, and the entry PRINTER specifies that the output is to be printed on an IBM 1132 Printer.

INPUT SPECIFICATIONS FORM

The Input Specifications form also has the program identification entered in columns 75-80 and the page and line numbers in columns 1-5. The I in column 6 of each line identifies this as an input specification.

The filename is entered in columns 7-14. Columns 15-16 contain the sequence code AA. Indicator 01, entered into columns 19-20, is on throughout the job showing that records from the associated file (the only file designated) are being processed. The second line describes the card field name CARDIN. The field comprises card columns 1-80.

OUTPUT-FORMAT SPECIFICATIONS FORM

The name of the output file is entered in columns 7-14. The D in column 15 indicates that each line printed is a detail line. A single space after printing is specified by the entry in column 18. Indicator 01 from the Input Specifications form is specified in columns 23-25. Since indicator 01 is on for each card read, all records are printed.

Each object program generated by RPG uses the same general logic. For each record to be processed, the program goes through the same general cycle of operations. Within that cycle, operations specified on the Calculation and Output-Format specifications forms are performed at two different times. These are called detail time and total time.

Familiarity with the logic flow of an RPG object program is helpful in writing RPG source programs. Figure 31 shows the general flow of the object program. The statements to the right of the chart expand upon the information in the chart. A program cycle begins with item 1 and continues through item 11. Steps 1 and 11 are referred to as detail time. Steps 6 and 7 are referred to as total time.

Knowledge of Figure 31 is adequate when performing relatively simple operations on a single input file as described in the preceding section *Fundamentals of RPG Programming.* However, a more detailed understanding of object program flow is necessary when performing complex operations on multiple input files. This detailed flow is contained in Appendix D.

1. Before the first record is read, the program prepares and writes all heading and detail lines are printed if the conditioning indicators are satisfied (1P, L, NOT conditions, and no indicators).
2. The program tests for any halt indicators. If any are on, a message is placed in the accumulator to notify the operator and he has the option of continuing or terminating the program.
3. The program sets off all record identifying indicators, L1-L9, H1-H9, 1P, and OF and OV.
4. The program tests for the end-of-file condition. If the end-of-file condition has occurred, the program branches to item 13.
5. Get an input record.
6. The record type is determined and the appropriate record identifying indicator is turned on. If control fields are specified, they are checked and the appropriate control level indicators are turned on.
7. Total calculations are performed if the conditioning indicators L0, L1-L9, and LR are satisfied. Exception output and/or closed subroutines may be performed. (This step is bypassed for the first control break.)
8. Total output lines are performed if the conditioning indicators are satisfied. (This step is bypassed for the first control break).
9. The program tests for an overflow condition. If an overflow condition has occurred in either step 1 or step 8, the program branches to item 14. Overflow is defined as the condition existing whenever OF or OV is on. (This step is also bypassed for the first control break.)
10. The last record (LR) indicator is tested. If it is on, execution of the program ends.
11. The data fields described on the Input Specifications for this record type are moved to the assigned core storage locations for these data fields. Any field indicators are set at this time.
12. Detail calculations (blanks in columns 7-8 of the Calculations Specifications form) are performed if the conditioning indicators are satisfied. Exception output and/or closed subroutines may be performed.
13. The Last-Record Indicator (LR) and all control-level indicators (L1-L9) are set on. Then, the program branches to item 7.
14. If overflow has occurred, total lines, heading lines, and detail lines (in that order) conditioned by overflow are printed. The program then branches to item 10.
15. If exception output is called for, exception output lines are performed if the conditioning indicators are satisfied. The program returns to the next calculation statement following the EXCPT operation that was performed.
16. If a closed subroutine is called for, the individual calculations within that closed subroutine are performed if the conditioning indicators are satisfied. The closed subroutine may also call other closed subroutines or exception output. The program returns to the next calculation statement following the EXSR operation that was performed.

Four conclusions can be drawn from the program logic:

1. At the beginning of execution of the RPG object program, all headings and detail output records whose conditions are satisfied are processed. At this time, indicators L0 and 1P are turned on. All other indicators are turned off.

   The user must ensure that only the output conditions for those records that are to be printed as heading information are satisfied before an input record is read into the system. First page heading information is usually specified with the indicator 1P. Other heading records should be specified with indicator N1P, OF, OV, or record identifying indicators whose conditions are not satisfied at the beginning of the object program run.

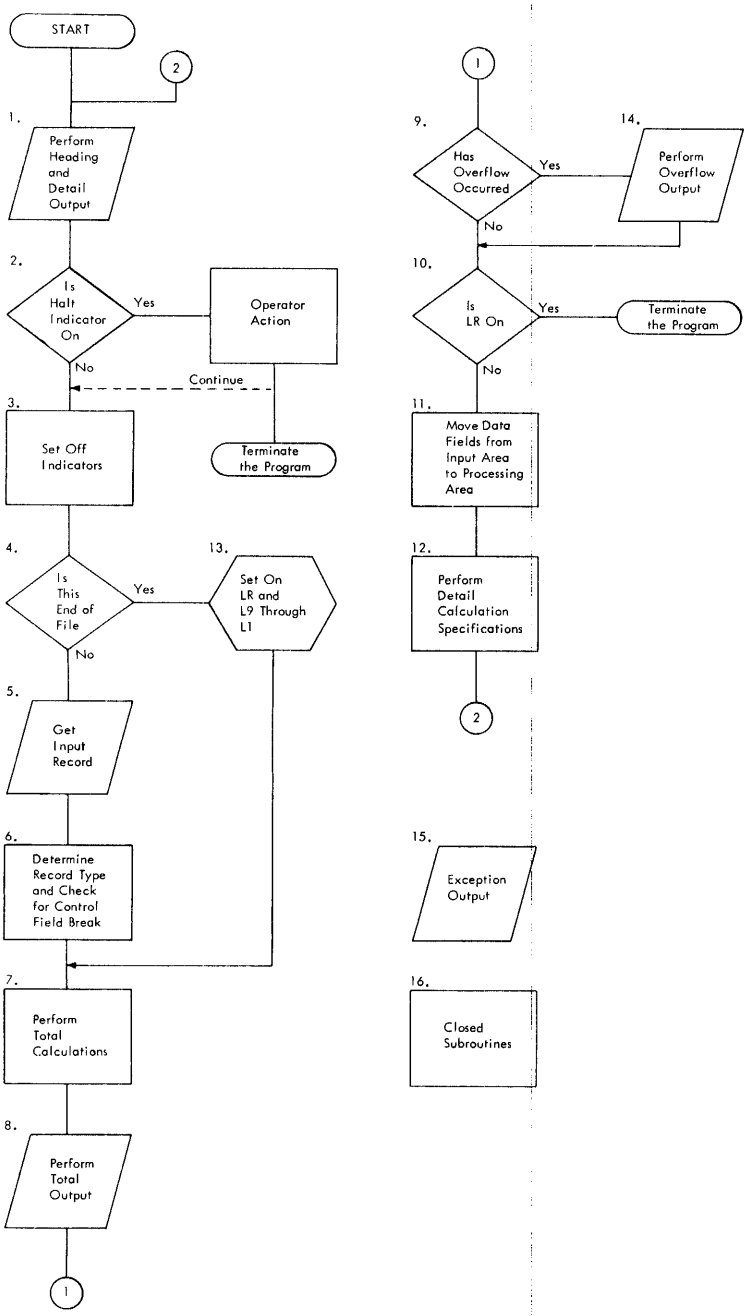


Figure 31. General Logic Flow of a Program Generated by RPG

As soon as the first input record is read into the system, indicator 1P is turned off. Therefore, all records specified with indicator 1P are not printed or punched again during the object program run.

Ensure that the punching of records in a combined file is specified in such a manner that their conditions are not satisfied at the beginning of the object program run. Otherwise, the first card of a combined file is punched prior to being read and the data in that card is lost.

If a heading or detail line is conditioned on a NOT condition (e. g., N15), it will be put out with 1P lines.

2. When a record is read that causes a control break, the record identifying indicator for the record is set on and all previous record identifying indicators are set off. The data from the record is not made available until all total calculations and total output for the previous group have been performed.

   Control fields are initialized to hexadecimal zeros. Consequently, a record with a blank or zero punched (hexadecimal F0) control field causes a control break.

3. If a hole was sensed in channel 12 of the carriage-control tape during the execution of total output specifications, all remaining total output specifications are executed whose conditions are satisfied. After the total lines are printed, overflow lines are printed.

   If a hole in channel 12 is sensed during the printing of a detail line, all the remaining detail and total lines specified (whose output conditions are satisfied) are printed before the printing of overflow lines. Therefore, the hole in channel 12 must be correctly placed to permit printing of these lines on the page.

4. If a specified halt indicator is turned on, processing is interrupted following detail output. The operator is notified and has the option of continuing or terminating the program.

   If a halt indicator is set on during the processing of total calculations, the object program continues processing until the end of the next detail output is reached. The execution of any specifications can be conditioned or prevented by placing H1 through H9 or their negation NH1 through NH9 in Indicators of the appropriate specifications form.

The programming examples in the preceding section were intended to introduce the reader to the use of RPG and were, therefore, elementary. More complex applications may require a thorough analysis of the existing or proposed problem before the program is written. This analysis should include a description of source data and its format, and the processing required to develop the report and other desired output information.

The following types of information must be defined before coding the program:

1. Available data.
2. Input/output formats to be used.
3. Information required in the input and output formats.
4. Codes to be used to identify the various types of input and output and their elements.
5. Handling of the various transactions and exceptions.

After all application data has been collected, it should be documented for easy reference during the writing of the specification forms. One method of documenting an application is to produce a layout of the complete format of the report on a Printer Spacing Chart. If the output is to be punched into cards, card layout should be used instead of a Printer Spacing Chart to define the output fields. These methods provide a pictorial representation of the final output.

PRINTER SPACING CHART (Form X24-6436)

Before the report specifications are written, the programmer should have a clear picture of what he wants as the final output. If the report is to be printed, he must know the number of fields to be printed on each line of the report, the spacing between lines, and the positioning of the information within each line of the report. The pictorial representation in this chart serves as a guide for completing the Output-Format Specifications form. It plays an important role in writing report specifications.

Layout of Lines and Fields

Figure 32 illustrates the layout of a report on the Printer Spacing Chart. Its two most important functions are:

1. To establish the positions of the data to be printed and to indicate the spacing between printed lines.
2. To assign each line an identification code representing the type of line.

The numbers across the top and bottom of the spacing chart refer to the print positions. The numbers down the left side are the line numbers. The programmer selects the line number and print positions for a particular field and makes his notation in the selected positions.

Headings and other constant information should be spelled out completely in the print positions assigned to them. Fields of variable information should be identified by X's. Where applicable, such fields should include credit symbols, punctuation, etc. The position in an amount field where zero suppression ends should be indicated by a zero rather than an X.

Line Identification Code

The line identification code specifies the type of line to be printed. The identification codes are: H for a heading line, D for a detail line, E for an exception line, and T for a total line. All lines should be identified as belonging to one of these categories.

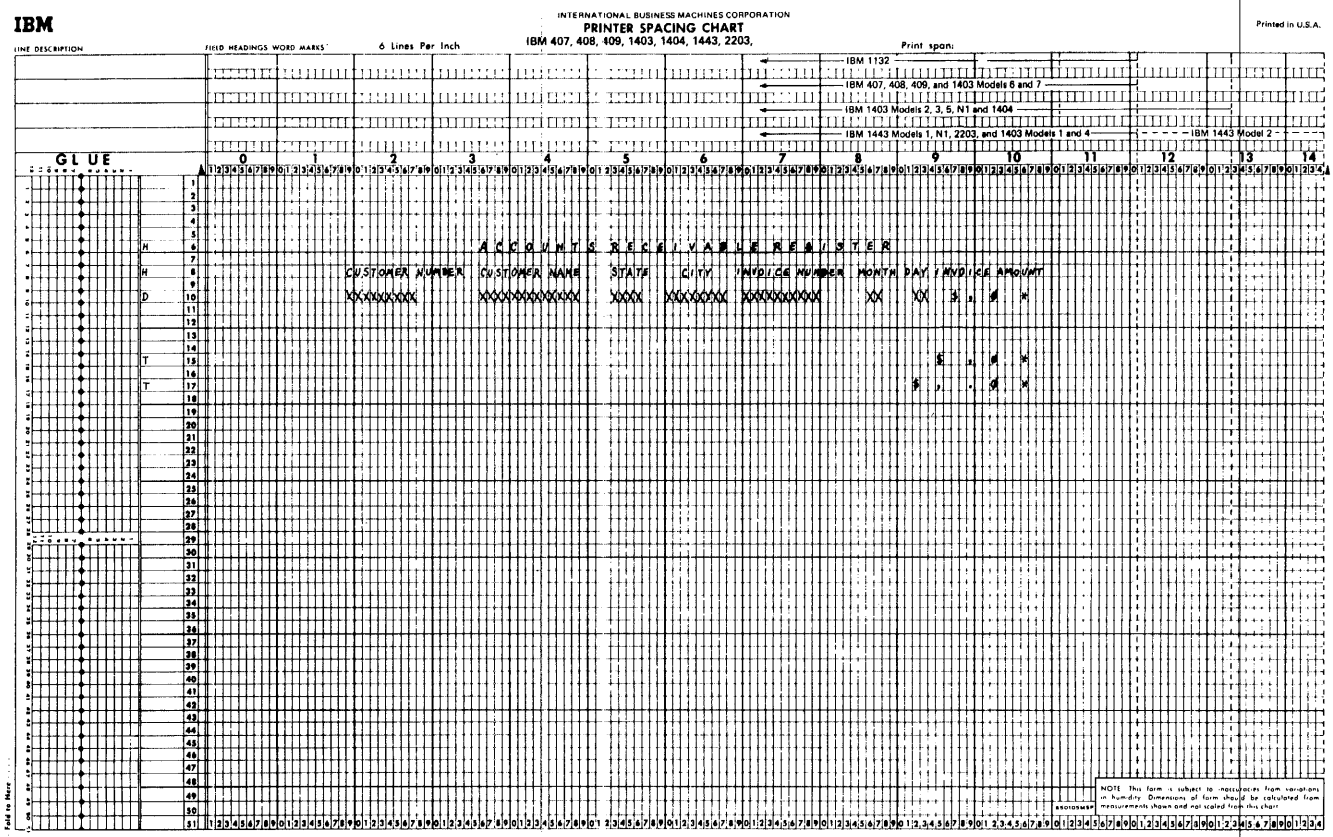


Figure 32. Printer Spacing Chart

RPG specification forms supplied by IBM are used when coding an RPG program for the 1130. The format and column headings on each of the forms guide the programmer in making entries in the appropriate places. The forms are designed so that one card is punched from each specification line. More than one sheet of a particular specifications form may be used in a program.

The RPG specification forms are shown in Figure 33. They are:

1. *Control Card and File Description Specifications,* which is used to define the RPG control card and the characteristics of all files in the user's program. The RPG control card must be the first card in the user's RPG source program deck. It must follow the necessary Monitor System Control Records.
2. *Extension and Line Counter Specifications,* which is used to provide additional information about files used by the user's program such as tables, chaining files, and record address files. The Line Counter Specifications part of this form is not used by the 1130 programmer.
3. *Input Specifications,* which is used to define the input records, to identify the various types of records in each input file, and to describe the location of the data fields in each input record.
4. *Calculation Specifications,* which is used to specify the operations to be performed upon the input data, upon data obtained as the result of previous calculations, and upon data obtained from tables.
5. *Output-Format Specifications,* which is used to specify the type of data that will be written or punched and the arrangement of that data on printed reports or cards, or both.

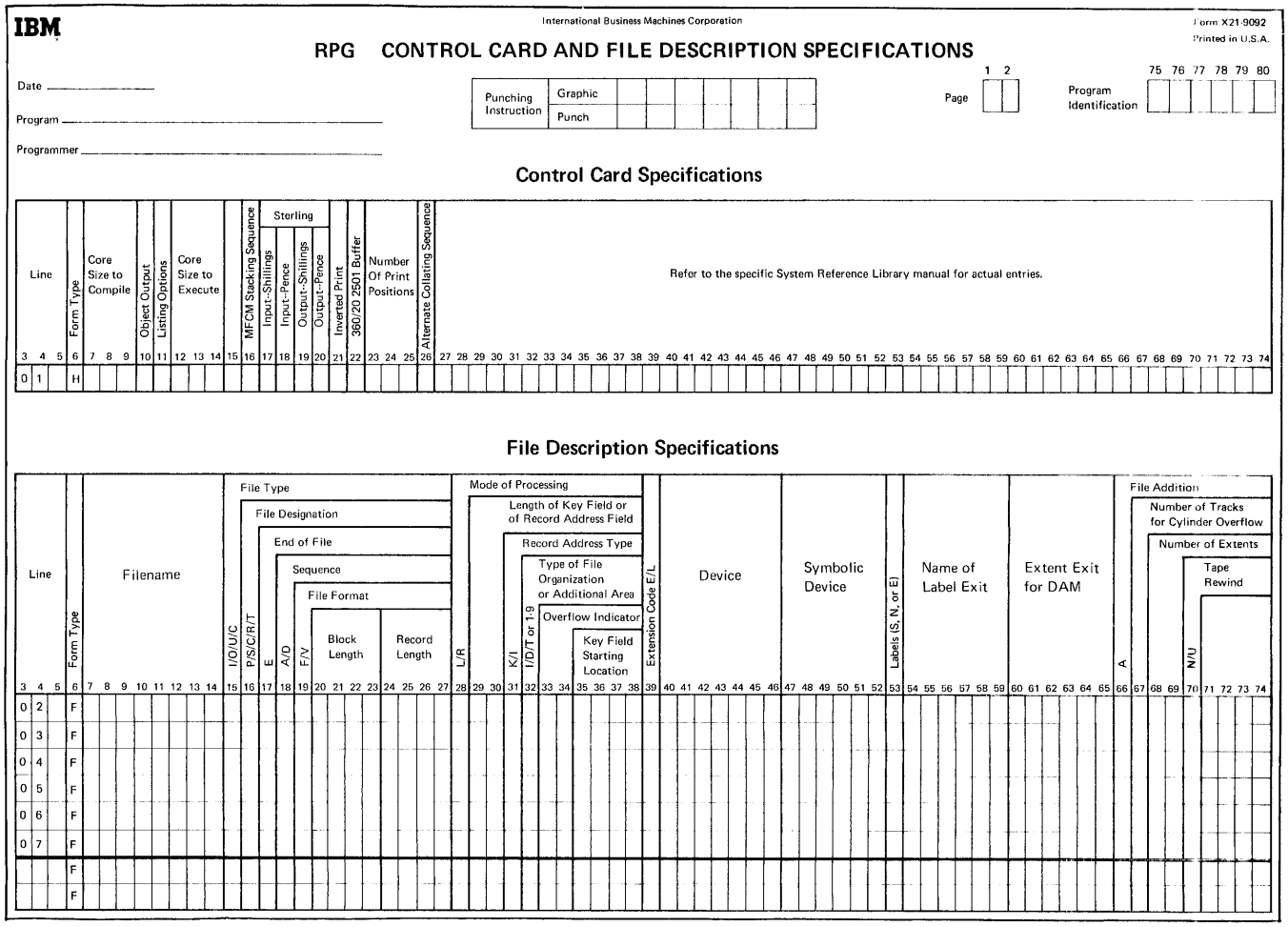


Figure 33. RPG Specification Forms (Part 1 of 3)

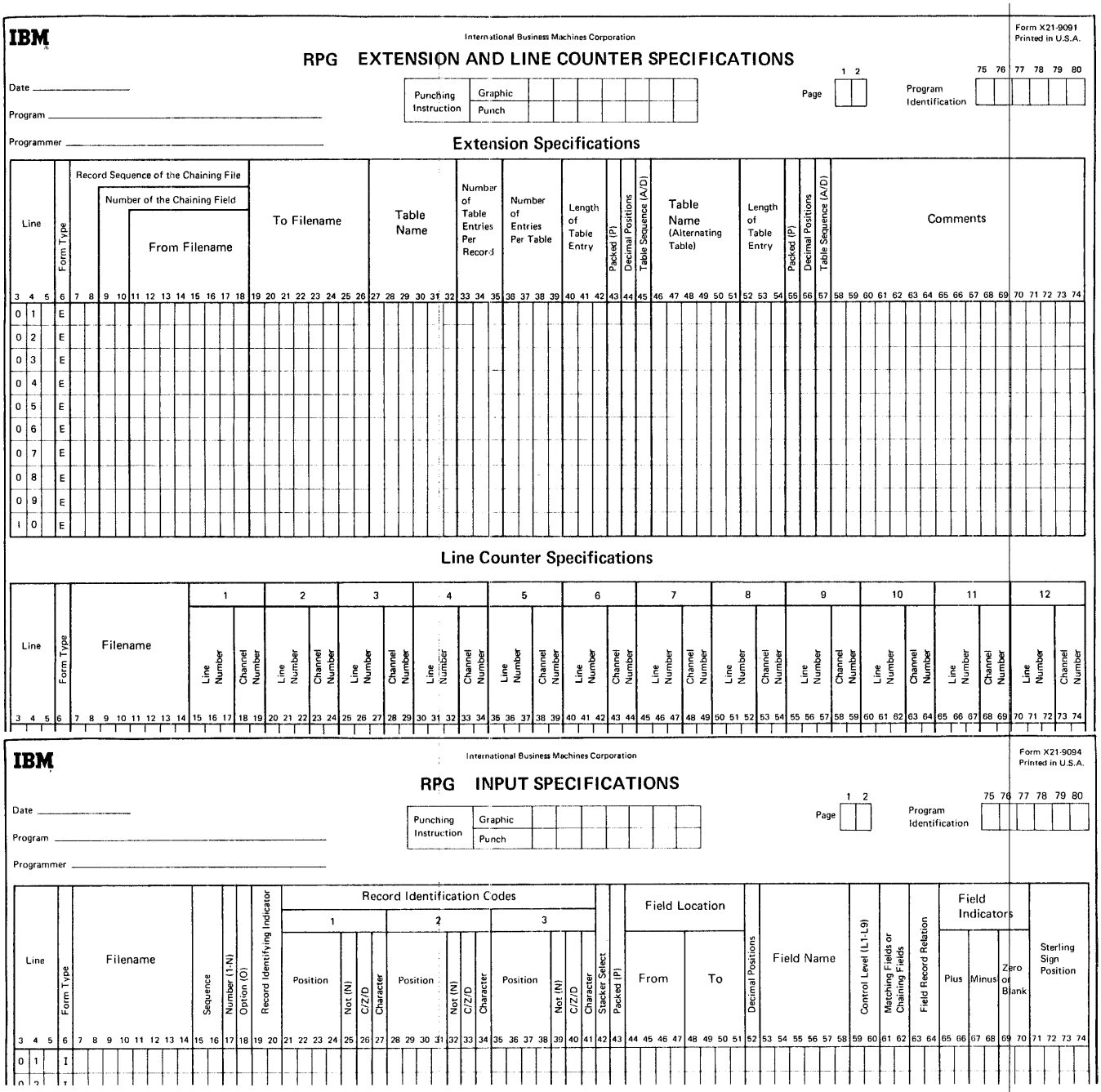


Figure 33. RPG Specification Forms (Part 2 of 3)

The RPG specifications forms are used in the sequence in which they are noted in the preceding list. The method of coding each form is described in detail in subsequent sections of this publication.

In addition to the five RPG specification forms, IBM provides a Summary of Indicators form (see the section *Program Documentation*). This form is designed to aid the programmer in keeping track of when indicators are turned on or off in the source program. Comment cards can be punched directly from the Summary of Indicators form.

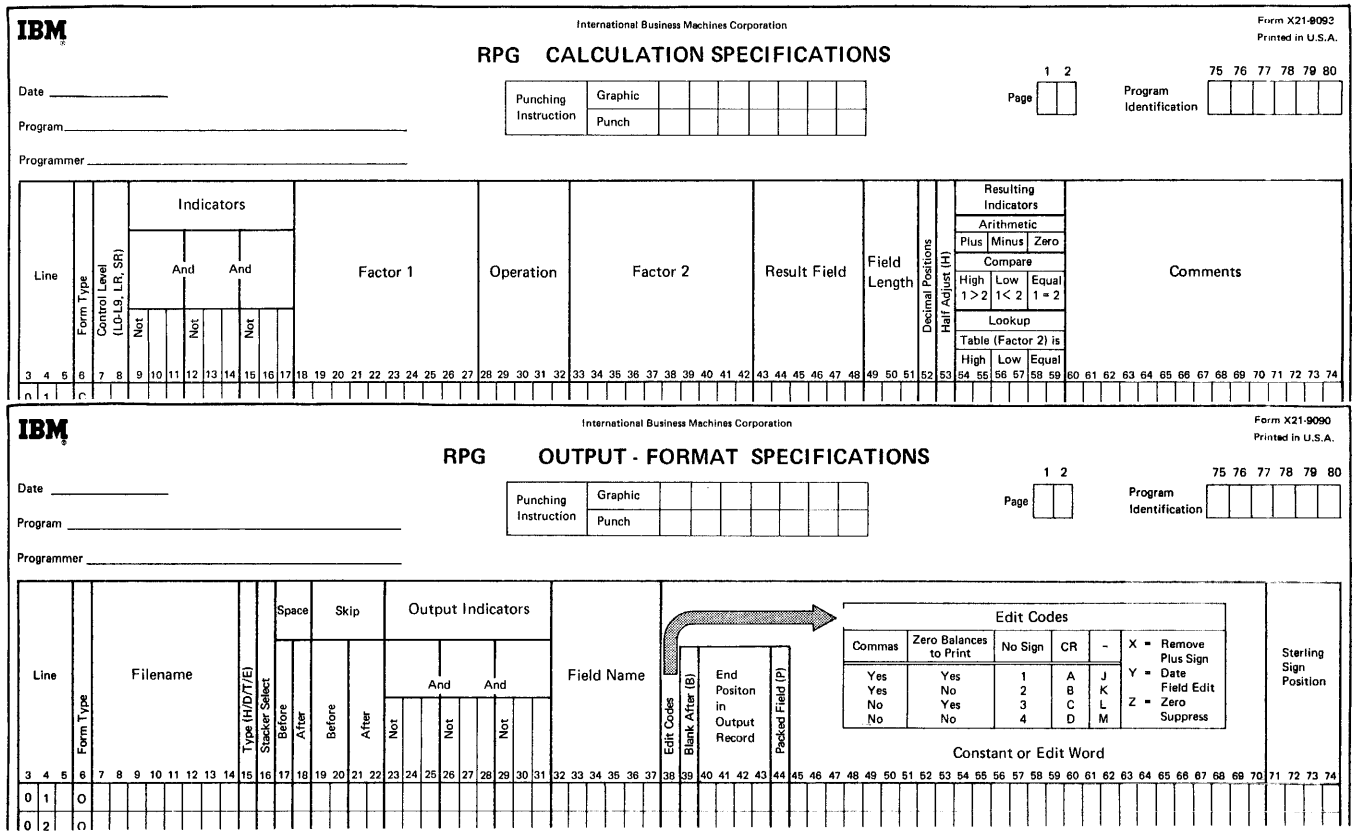


Figure 33. RPG Specification Forms (Part 3 of 3)

Common Fields

Five entries have the same function in each of the RPG specification forms. These common entries are:

- Page number (columns 1-2)
- Line number (columns 3-5)
- Form type (column 6)
- Comments (asterisk * in column 7)
- Program identification (columns 75-80)

The meaning and contents of each of these entries are described in Figure 34.

Each card in the user's RPG source program must be in numeric sequence according to the values in columns 1-5. Otherwise, an out-of-sequence warning notation is printed on the program listing.

Cross References

Detailed descriptions of each of the RPG Specification forms are tailored to provide basic information necessary to complete the forms. Supplementary topics such as disk file organization and processing, matching field processing, and chaining are described at the back of this publication. Refer to the table of contents or index for the location of these topics.

Column No.	Entry Name	Meaning and Contents
1-2	Page Number	Is a 2-digit number that uniquely identifies a specification page of the source program. This number aids in sorting the forms and in referring to a particular form. The page numbers should be in ascending order, but need not be consecutive. In order to insert new pages readily, the following range is suggested. Page No. — Specification 1-4 — Control Card & File Description 5-9 — Extension 10-39 — Input 40-69 — Calculation 70-99 — Output-Format
3-5	Line Number	Is a 3-digit number that uniquely identifies a specification line on a form. The first two digits are preprinted on the form. The third position is used when inserting a line between two previously written lines. For example, a line to be inserted between lines 14 and 15 must have the number 14 in columns 3 and 4, and a subnumber in column 5. This places the line in the proper numeric sequence.
6	Form Type	Is a preprinted code which identifies the type of specification form being used as follows: H for Control Card F for File Description E for Extension I for Input C for Calculation O for Output-Format The appropriate code must be punched into all specification cards.
7	Comments	Is an asterisk placed in column 7 indicating that the information on this line is a comment, and not a specification. The line is not processed, but is merely printed on the program listing.
75-80	Program Identification	Is an alphameric entry that identifies the program or section of a larger program with which these specification cards are associated.

Figure 34. Common Entries, Meanings and Contents of

This form is used to define the RPG control card and the characteristics of all files in the user's program. It is divided into two parts. The top portion is used for the RPG Control Card. The bottom portion is used for File Description Specifications. Each of these parts is shown in Figure 35 and described in the sections that follow.

RPG CONTROL CARD SPECIFICATIONS

The user's source program deck must be preceded by an RPG control card. This card must be prepared according to the functions required of the RPG compiler. Entries are to be made on the top of this form as described in the following text.

Columns	*Contents*	*Meaning*
1-5	xxxxx	Page and line number assigned to this control card.
6	H	Identifies this card as on RPG control card.
7-10	Blank	Not used.
11	B, D, blank	Listing options during compilation. B – compilation only. D – listing only. blank – Compilation with listing.
12-16	Blank	Not used.

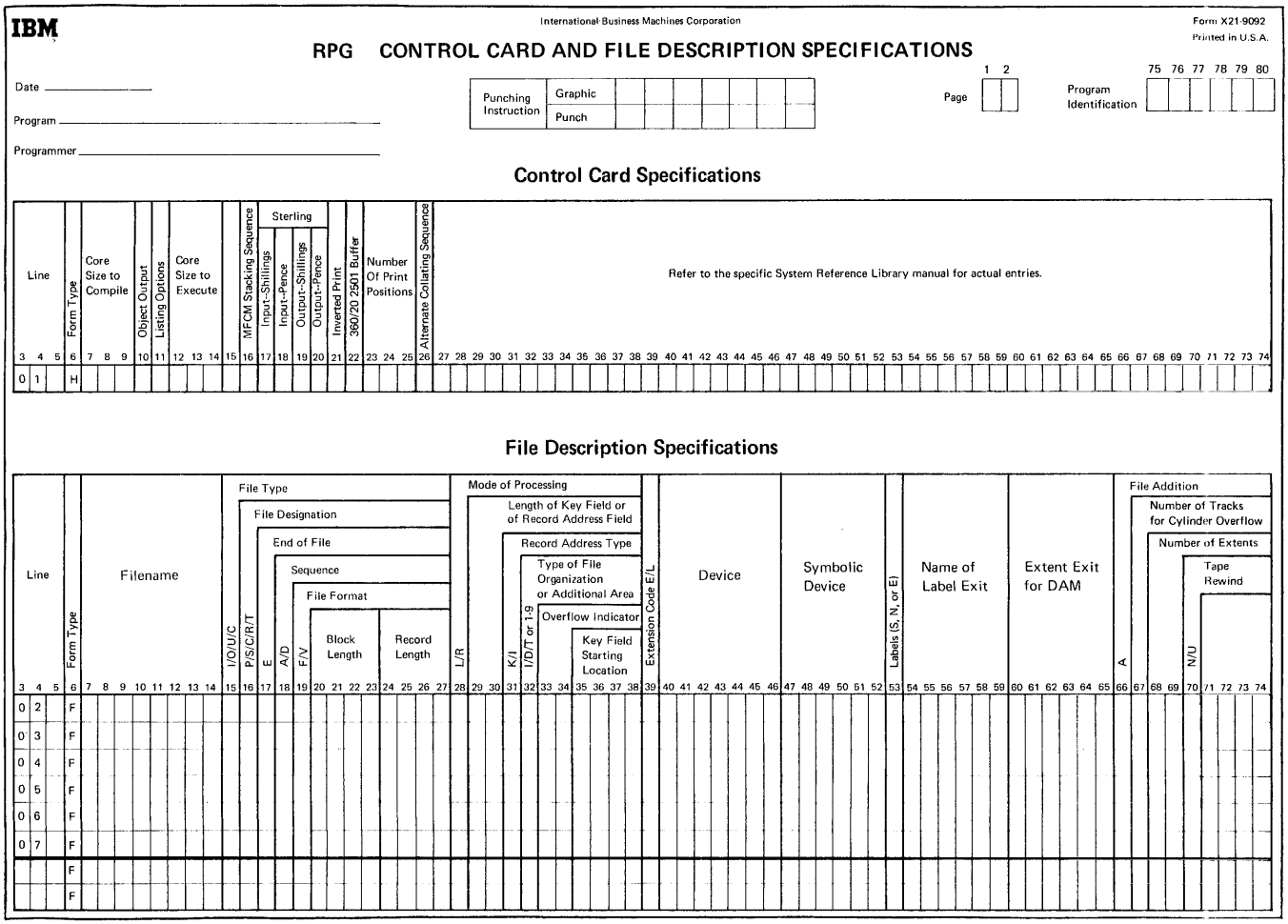


Figure 35. Control Card and File Description Specification Form

Columns	Contents	Meaning
17	1, 2, blank	Format of shillings portion of sterling-currency input field. 1 – IBM format. 2 – BSI format. blank – The input does not contain sterling-currency fields.
18	1, 2, blank	Format of pence portion of sterling-currency input field. 1 – IBM format. 2 – BSI format. blank – The input does not contain sterling-currency fields.
19	0, 1, 2, blank	Format of shillings portion of sterling-currency output field. 0 – Printer format 1 – IBM format 2 – BSI format blank – The output does not not contain sterling-currency fields.
21	0, 1, 2, blank	Format of pence portion of sterling-currency output field. 0 – Printer format 1 – IBM format 2 – BSI format blank – The output does not contain sterling-currency fields.
21	I, blank	Inverted print option. blank – Inverted print is not used. I – Inverted print is used. Commas must be entered by the user instead of periods in numeric literals to denote the decimal point. Edit codes 1-4, A-D, and J-M will substitute commas for decimals and vice-versa. Periods are printed instead of slashes with the Y edit code. *Note:* The inverted print option has no effect on edit words. Normally, the user who specifies this option will invert commas for decimals and vice-versa in the edit words he uses. Editing will be performed according to the edit word provided.
22-25	Blank	Not used.
26	A, blank	Alternate collating sequence. blank – An alternate collating sequence is not required. A – An external user-written subroutine named ALTSE stored in the subroutine library is used to translate the sequence of a matching field to the EBCDIC collating sequence. Refer to the section on *Exit to a Translate Subroutine.*
27-74	Blank	Not used.
75-80	XXXXXX	Program name. These columns have no other meaning than to appear on the program listing. To store this program on disk, the first five characters of this name can be supplied in the proper control record for a disk utility program. Refer to the publication *IBM 1130 Disk Monitor System, Version 2: Programming and Operator's Guide,* Form C26-3717.

FILE DESCRIPTION SPECIFICATIONS

Each file used by the object program must be defined. Each line of the File Description form is used to define one file. This form is used to furnish information about:

1. The input files, specified on the Input Specifications form, from which the object program obtains data records.
2. Input table files, and record address (RA) files.
3. The output files, specified on the Output-Format Specifications form on which the object program writes data records or output table files.

A maximum of ten files may be specified in an 1130 RPG program. Figure 36 defines the maximum number of the various types of files that can be used. Any combination of these is permitted.

A table file is defined as one line on the extension specifications. Eight table files can be loaded, each of which contains an alternating entry. Therefore, 16 different table names are possible.

Type of File	Max. No of Files
Primary	1
Secondary	8
Record Address	1
Chaining:	
C1-C3	3
Chain Operation	9
Table	8
Output:	
Printer	2
Other Output Units	9

Figure 36. Maximum Number of Files

Filename (Columns 7-14)

Is a name used to identify the file being specified. This same name must appear on other specifications that refer to this file.

A filename may consist of eight alphameric characters or less. The first character must be alphabetic and must appear in column 7. No special characters or blanks can be used.

Even though the filename can contain eight characters, 1130 RPG uses only the first five characters; the last three characters are ignored. Therefore, the first five characters of each filename in a program must be unique.

File Type (Column 15)

Specifies the type of file defined on this line, as follows:

Entry	*Type of File*	*Explanation*
I	Input	Identifies the file as an input file. It may be a table file, a record address file, or a file containing input data records which are read only.
O	Output	Identifies the file as an output file. It may be an output data file or an updated table file to be printed, punched, or written.
U	Update	Identifies the file as an update file (disk only).

Entry	*Type of File*	*Explanation*
		An update file is both an input and an output file. It must be specified similarly to an input file on the Input Specifications form. The file is an update file if the object program alters the data in one or more fields of each record contained in the file without changing: 1. The nature of the data. 2. The length of the field in which the data is found. 3. The location of the field. A chained file may be updated at detail time or at total time. All other disk files can be updated only at detail time.
C	Combined	Identifies the file as a combined card file (1442 Card Reader only). A combined file is a card file containing cards that are read into the system and cards used for punching. Reading and punching into the same file is accomplished in two different ways: 1. Punching into cards that contain input data. 2. Punching into a blank trailer card in the same file. For combined file cards that are read only, Stacker Select entries should be specified on the Input Specifications form. This will reduce the time required to run the object program. For combined file cards that are punched, Stacker Select entries must not be specified on the Input Specifications form. A combined file differs from an update file in that the update file input fields are revised or updated. A combined file can only refer to cards, and an update file can only refer to disk.

Sequence (Column 18)

Indicates whether matching fields are in ascending or descending sequence, as follows:

A, if the matching fields are in ascending order.
D, if the matching fields are in descending order.

An entry is made in column 18 if the matching fields specification (columns 61-62 of the Input Specifications for) is used. Otherwise, column 18 is left blank.

File Format (Column 19)

Must be F to indicate fixed length format.

Block Length and Record Length (Columns 20-27)

Identify the record and block length. Enter right-justified the length of one record in Record Length (columns 24-27). Leave Block Length (columns 20-23) blank. Disk files are automatically blocked so the maximum number of records will fit on a disk sector. (Refer to the section *Disk File Space Calculation* for file capacity).

If the record length entry is missing, a record length of 80 is assumed.

Maximum record lengths are as follows:

2501 and 1442 I/O units	80 characters
1132, 1403, or console printers	120 characters
sequential disk files	640 characters
indexed-sequential disk files	636 characters

Mode of File Processing (Column 28)

Indicates the method or mode by which disk files are processed, as follows:

Entry	*Explanation*
blank	If the disk file is to be processed sequentially or created. If an indexed sequential file is to be added to, this column must be blank.
R	If the user's records are to be processed randomly. In this case, the records of an indexed sequential file are to be processed by the CHAIN operation (or C1-C3) using key fields. The records of a sequential disk file are processed randomly by using the CHAIN operation (or record address file) and relative record numbers.
L	If a segment of an indexed sequential file is to be processed using limits specified by a record address file. Only an indexed sequential file can be processed within limits.

Length of Key Field or of Record Address Field (Columns 29-30)

If the file defined on this line is a Record Address file (RA file), enter the number of positions that each entry in the RA file occupies.

If the file defined on this line is an indexed sequential file (I in column 32), enter the length of the key. The maximum key length is 50 characters.

Type of Record Address (Column 31)

Indicates how records will be retrieved for processing, as follows:

K	If the file is an indexed sequential file. The file defined on this line will be processed by use of the record key.
blank	If the file is a sequential file.

Type of File Organization (Column 32)

Designates how the file is organized, as follows:

I	For indexed sequential organization. (There must also be a K in column 31 and 1 in column 38).
blank	For sequential organization.

2 If two input/output areas are to be assigned to a card reader, or a 1403 or 1132 printer file. Stacker select (Input Specification, column 42) should not be specified. Any entry of one through nine may be used, but the digit 2 is preferred. (Two I/O areas may not be specified for a table file.)

Correlation Between File Processing and Entries in Columns 28, 31, 32

Columns 28, 31, and 32 are used in conjunction with one another to govern file processing. Records within an indexed sequential file can be processed sequentially, randomly, or sequentially within limits. Records within a sequential file can be processed sequentially or randomly. The entries in columns 28, 31, and 32 describe the type of file being used and the way the file is to be processed. Figure 38 summarizes the entries necessary for each type of file organization and processing mode.

Overflow Indicator (Columns 33-34)

If the file defined on the line is a 1132 or 1403 printer file and overflow indicators are used, enter the overflow indicator associated with the file. The allowable indicators are OF and OV. Do not specify an overflow indicator for the console printer or any non-printer.

Key Field Starting Location (Columns 35-38)

Defines the starting location of the key field within a record. There must be a 1 in column 38 if indexed sequential files are used. Otherwise, the entry should be blank. The key field for 1130 indexed sequential files always begins in position 1.

Extension Code (Column 39)

If the file defined on the line is a chaining file (using C1-C3 codes), a record address file, or a table file, enter an E in column 39. The E specifies that additional information about the file will be coded on the Extension Specifications form.

Device (Columns 40-46)

Specify the input/output units used by each file, as follows:

Entry	*Input/Output Unit*
PRINTER	IBM 1132 Printer
PRINT03	IBM 1403 Printer
PUNCH42	IBM 1442 Punch
READ42	1442 Reader/Punch
READ01	IBM 2501 Reader
CONSOLE	Console Printer
DISK	2310 Disk Unit
DISK	IBM 1131 Internal Disk

Type of File Organization (Column 32)	Type of Record Addresses (Column 31)	Mode of Processing (Column 28)
Indexed-Sequential (I)	Record Key (K)	The entire file will be processed. (blank)
		A segment of the file will be processed. The limits to be processed are supplied by a Record Address File (RAF). (L)
		The records will be processed randomly. (R) Keys are supplied by Factor 1 of the CHAIN operation, by a chaining field (C1-C3), or by a record address file.
Sequential (blank)	Not applicable (blank)	The entire file will be processed sequentially. (blank)
		The records will be processed randomly. (R)

Figure 38. Processing Direct Access Storage Files

Symbolic Device (Columns 47-52)

Are only used when the file described is indexed sequential and is to be loaded.

Enter *left-justified* the maximum number of records in the file. This number may not exceed 99999. A message will be printed during compilation which describes how many disk sectors are needed for a file of this size.

A chart for determining file size requirements is contained under *Disk File* Space Calculation in the section entitled *Disk Storage Concepts.*

Columns 53-65

Must be blank.

File Addition (Column 66)

Enter an A in this column if the file defined on this line is an indexed sequential file and new records are to be added. The records to be added are described on the Output-Format Specifications form with an entry of the characters ADD in Columns 16-18.

For all other file description specifications, this column must be left blank.

Columns 67-74

Must be blank.

ENTRIES ON THE FILE DESCRIPTION SPECIFICATIONS FORM

Figure 39 shows examples of File Description Specfications entries. The following numbers and descriptions correspond to the circled numbers in figure 39.

1. INPUT is the primary file (P in column 16). The end of job condition will occur when the end of the file is reached (E in column 17). The file is in ascending sequence (A in column 18). Each record is 80 characters long. The file is read from an IBM 1442 Reader/Punch (Device code READ42).
2. RAFFILE is an input file (I in column 15) designated as a record address file (R in column 16). Each record is 80 characters long. The file will be further described on the Extension Specifications form (E in column 39). It is read from an IBM 2501 Card Reader (Device code READ01). The length of each Record Address field is 8 characters (8 in columns 29-30).
3. TABFILE is an input file designated as a table file (T in column 16). Each record is 100 characters long. The file will be further described on the Extension Specifications form (E in column 39). It is read from disk (Device code DISK).

File Description Specifications

Line	Form Type	Filename	I/O/U/C	File Type: P/S/C/R/T	File Designation: E	End of File: A/D	Sequence: F/V	File Format: Block Length	Record Length	Mode of Processing: L/R	Length of Key Field or of Record Address Field	Record Address Type	Type of File Organization or Additional Area: I/D/T or 1-9	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels (S, N, or E)	Name of Label Exit	Extent Exit for DAM	File Addition: A	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind: N/U
020	F	INPUT	I	P	E	A	F		80								READ42								
030	F	RAFFILE	I	R			F		80		8					E	READ01								
040	F	TABFILE	I	T			F		100							E	DISK								
050	F	MASTCUST	I	C			F		100	R	8	K	I			1	DISK								
060	F	DISKUPDT	U	C			F		200	R	6	K	I			1	DISK								
070	F	CARDREC	C	S		A	F		80								READ42								
080	F	OUTPUT	O				F		120					OF			PRINTER								
090	F	ISAM	O				F		75		6	K	I			1	DISK	5000							
100	F	ISAM	O				F		75		6	K	I			1	DISK					A			
110	F	SEQDISK	I	P			F		60								DISK								
120	F	SEQDISK	U	C			F		60	R							DISK								
130	F	ISAMLIMT	I	S		A	F		130	L	3	K	I			1	DISK								
140	F	MESSAGE	O				F		60								CONSOLE								

Figure 39. File Description Specifications Form, Use of

4. MASTCUST is an input file designated as a chained file (C in column 16) that will be processed randomly (R in column 28). Records are addressed through keys (K in column 31). The file has an indexed sequential organization (I in column 32), and is located on disk (Device code DISK). The key field location for indexed sequential files always begins in position 1 (1 in columns 35-38).
5. DISKUPDT is an update file (U in column 15) used for input and updated after each record has been processed. It is an indexed sequential file (I in column 32) that will be processed randomly (R in column 28). It is a chained file (C in column 16). Each record is 200 characters long. Records are addressed through keys (K in column 31). The file is located on disk (Device code DISK).
6. CARDREC is a combined file (C in column 15) used as input and designated as a secondary file (S in column 16). It is in ascending sequence (A in column 18). Each record is 80 characters long. Additional information will be punched in the input cards during processing. The CARDREC file is read and punched on an IBM 1442 Reader/Punch (Device code READ42).
7. OUTPUT is to be printed on an IBM 1132 Printer (Device code PRINTER). Each record to be printed is 120 characters long. Overflow indicator OF is assigned to the printer.
8. ISAM is an indexed sequential file (I in column 32) with a record length of 75 characters. The file is to be initially loaded (0 in column 15). The key field is six characters long (columns 29-30) and begins in position 1 (columns 35-38). Space for 5000 records is required for this file (columns 47-52). RPG automatically prints the number of sectors that will be required for this file. The index will be organized using the first six positions of the output record as the key field.
9. ISAM created in line 090 is added to (A in column 66). The entries are identical to line 090 except that the number of records to be allowed is not specified. The letters ADD are required on the Output-Format Specifications form (columns 16-18) for the record to be added. No retrieval or updating is allowed on the file during the addition operation.
10. SEQDISK is a sequential disk file (blank in column 32) of 60 characters. No sequence (blank in column 18) is specified since it is the only input file for the job and no sequence checking is to be done.
11. The same sequential file described in line 100 is now being processed randomly (R in column 28) as a chained file (C in column 16). A relative record number will be used to randomly retrieve records from this file.
12. ISAMLIMT is an indexed sequential file being processed by limits. It is treated as an input file (I in column 15) and is a secondary file (S in column 16) being matched against a primary file. The match fields are in ascending sequence (A in column 18). The records are 130 characters long. The file is limited (L in column 28) by an RA file which will supply the limits. The key is three characters long (Columns 29-30) and must begin in position 1 (Columns 35-38). Entries K-I (Columns 31-32) are required for an ISAM file. The file resides on disk (DISK in columns 40-46).
13. MESSAGE is used for records to be printed on the system console (device entry CONSOLE). It is an output file (0 in column 15). No overflow indicator can be assigned. Each record is 60 characters long.

This form consists of two parts (see Figure 40). The top portion is for Extension Specifications. The bottom portion is for Line Counter Specifications. Line Counter Specifications are not used in an 1130 RPG program.

Entries made on the Extension Specifications part of the form provide information to RPG about tables used in the object program, record address files, and chaining files (except when processed with the CHAIN operation). Detailed descriptions of these topics is contained in the sections *Using Tables in the Object Program* and *Processing Multiple Input Files*. A description of the entries to be made on this part follows.

Record Sequence of Chaining File (Columns 7-8)

Is the record sequence code from columns 15-16 of the Input Specifications form for the input file that is to supply the chaining field. This entry is only used when chaining by the alternate (C1, C2, C3) method described in the section *Input Specification Form.* It must be blank at all other times.

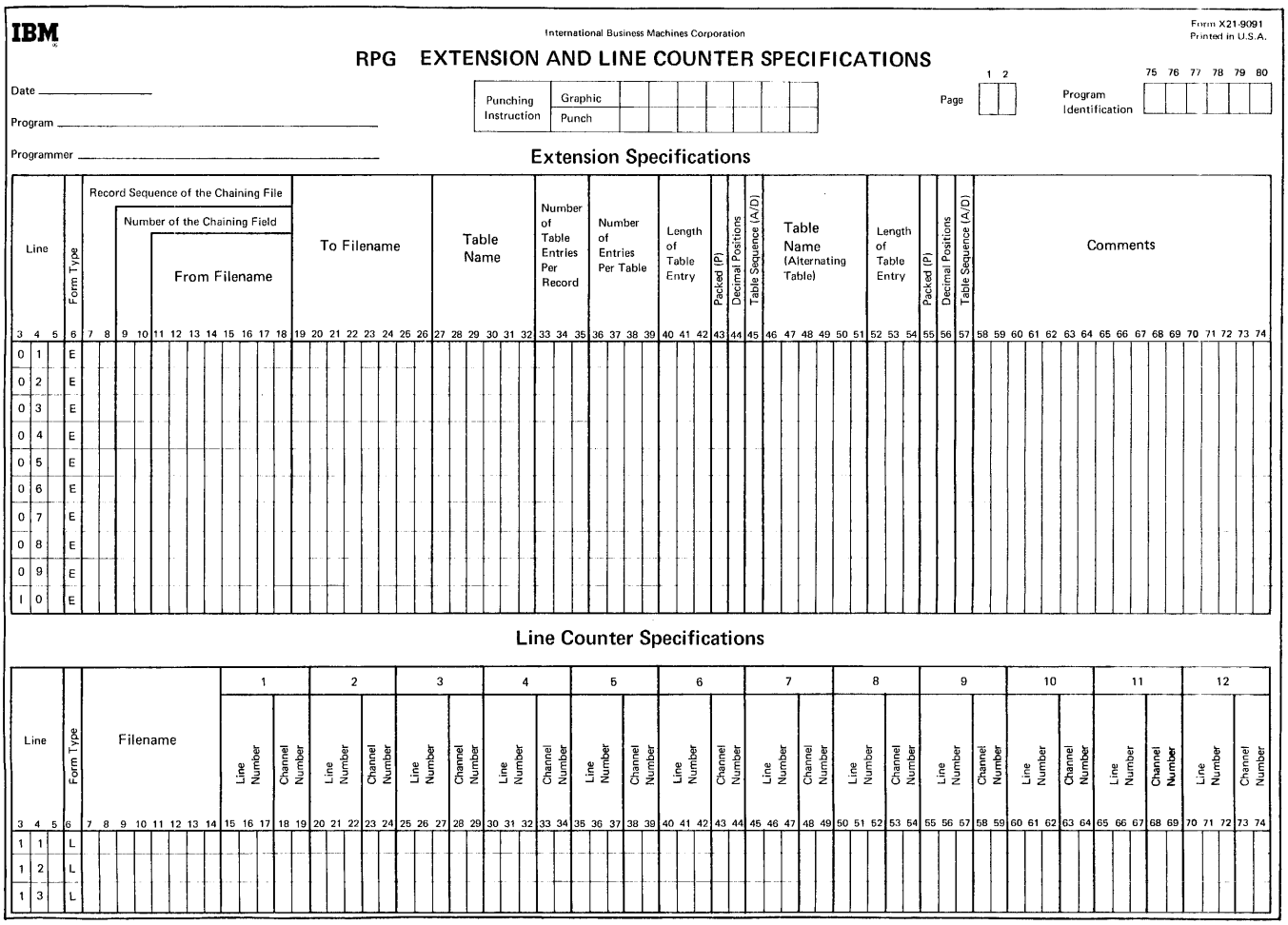


Figure 40. Extension and Line Counter Specifications Form

Number of Chaining Field (Columns 9-10)

Is the identifying code (C1-C3) that is entered in columns 61-62 of the Input Specifications form to identify the chaining field for this file. This entry is only used when chaining by the alternate method (C1, C2, C3) described in the section *Input Specifications Form.* It must be blank at all other times.

From Filename (Columns 11-18) and to Filename (Columns 19-26)

Are used in conjunction to identify the relationship between two files. For example, they may provide the name of an input table file and the name of the table file that is to be written at end of job. Both entries are taken from Filename (columns 7-14) of the related line on the File Description Specifications form. Entries depend on the type of file being used. Refer to Figure 41 for a description of this pair of entries.

Table Name (Columns 27-32)

Is the name of the table being defined on this line. The table name must contain four to six characters and the first character must appear in column 27. The name must be of the form TABnnn, where "nnn" are any alphameric characters supplied by the user. No special characters can be used. Table names must be unique throughout a program. Tables are described in detail in the section *Using Tables in the Object Program.*

Number of Table Entries Per Record (Columns 33-35)

Is the maximum number of table entries (i.e., the number of arguments or functions) contained in each input record. This entry must be right-justified.

Number of Entries Per Table (Columns 36-39)

Is the number of entries contained in the table(s) defined on this line. If an alternating table is specified, this number is the sum of the pairs of arguments and functions, i.e., one argument and one function equals one entry. This entry must be right-justified.

Length of Table Entry (Columns 40-42)

Enter in columns 40-42 the length of each table entry. The maximum lengths are 248 alphameric characters or 14 numeric digits. For a packed field enter the number of digits of the field. The entry must be right-justified.

Packed (Column 43)

Indicates whether or not the data is packed, as follows:

P The data is packed.

blank The data is not packed.

Packed data may only come from disk files.

Type of File	*From Filename (11-18)	*To Filename (19-26)
Chaining Files	Name of Chaining File. This is the file that has the data record containing the chaining field. The name of the file is taken from columns 7-14 of the File Description Specifications form.	Name of the Chained File. This is the file from which the data record is obtained. The name of the file is taken from columns 7-14 of the File Description Specifications form for the chained file.
Record Address File	The name of the record address file.	The name of the file that contains the data to be processed.
Table Files	If a table is being defined (columns 27-57 of the Extension Specifications) enter the name of the file that contains the table data.	If the table being defined will be printed, punched, or written after it is updated, enter the name assigned to it on the Output-Format Specifications form. Otherwise, leave the field blank.
* All entries must be left-justified.		

Figure 41. From and To Filenames

Decimal Positions (Column 44)

Indicates the number of decimal positions in the numeric data in the table, as follows:

blank	The field is alphameric.
0	The data contains no decimal positions.
1-9	The specified value (1-9) indicates the number of decimal positions in the data. For example, 3 denotes that the data contains three decimal positions.

The number of decimal positions cannot exceed the length of the table entry defined in columns 40-42.

Table Sequence (Column 45)

Defines the sequence in which data has been placed in the table named in columns 27-32, as follows:

A	Ascending sequence.
D	Descending sequence.
blank	Unsequenced.

Data must be in either ascending or descending sequence if it is used in a LOKUP operation as Factor 2 (argument table) and high or low indicators are used. For additional information on the LOKUP operation, refer to the section *Using Tables in the Object Program.*

Columns 46-57

Are used only if a table file consists of alternating arguments and functions.

Table Name (Columns 46-51)

Is the name of the second table if two tables are being used. The table name must contain four to six characters and the first character must appear in column 46. The name must be the form TABnnn, where "nnn" are any alphameric characters supplied by the user. No special characters can be used. Table names must be unique throughout a program.

Length of Table Entry (Columns 52-54), Packed (Column 55), Decimal Positions (Column 56), and Table Sequence (Column 57)

Are attributes of entries in the second table named in columns 46-51. For a description of the format and content of each of these specifications, refer to the appropriate specifications of the same name for the first table defined on this line (columns 40-45).

Comments (Columns 58-74)

Is optional explanatory information supplied by the user. This information is merely printed on the source program listing and is not interpreted by the RPG compiler.

ENTRIES ON THE EXTENSION SPECIFICATIONS FORM

Figure 42 shows examples of Extension Specifications entried. The following numbers and descriptions correspond to the circled numbers in figure 42.

1. Line 010 uses the alternate method of chaining. INPUT is a card file containing the record key that will be used to process records in the file MASTCUST which is located on a direct access device. INPUT is the chaining file; i.e., it is the file that links or chains to another file. The field in INPUT that is used to link the two files is the chaining field. AA in columns 7-8 denotes the record sequence entry of the chaining file record. C1 in columns 9-10 is the number of the chaining field. Thus, INPUT is chained to MASTCUST by using a field defined on the Input Specifications form that contains C1 in columns 61-62. Chaining is described in the section *Processing Multiple Input Files.*
2. RAFFILE is a record address file that supplies the record number of the records to be processed in the indexed sequential file DISKUPDT. In this case, RAFFILE must supply the actual keys to be processed or the limits of the file to be processed.
3. TABFIL is a table file that contains both a table of arguments and a table of functions.

   The arguments are identified by the table named TABARG described in columns 33-45. Each record contains ten arguments. TABARG contains 150 arguments, each of which is ten characters long. The arguments are numeric (column 44). The table is in ascending sequence (column 45). The table look-up operation can then specify whether the search should be for a high, low, or equal argument.

   The functions in the file are identified by the table named TABFUN described in columns 52-57. Each function is ten characters long and is numeric. TABFIL is organized in the form: argument-function. Therefore, TABARG was specified first.
4. Line 040 shows the specifications for a table file that contains only arguments. After the table of arguments is updated, the table is to be written or punched.

   OLDTAB is the name of the input table file. NEWTAB is the name of the output table file.

   The arguments in the file are contained in the table named TABREC described in columns 33-45. Each record contains five table entries. TABREC contains ten entries, each of which is twelve characters long. The data is numeric. TABREC is the name used on the Calculation Specifications form when specifying a table look-up or update operation.

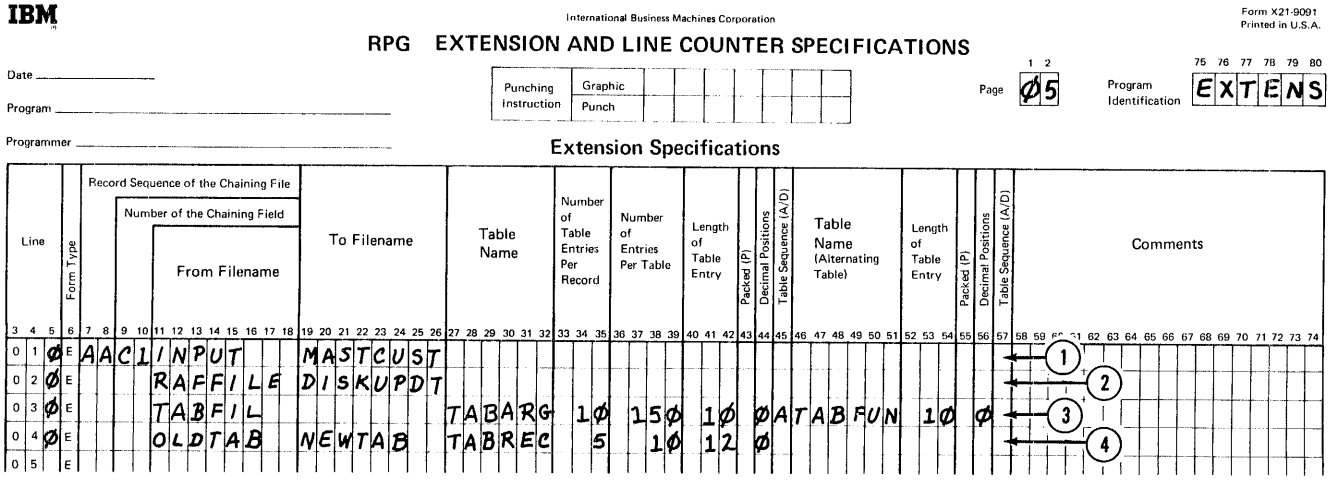


Figure 42. Extension Specifications Form, Use of

The specifications that are made on this form are divided into two categories (Figure 43):

1. Record Identification (columns 7-42). These specifications identify the input record by specifying the identifying record codes it contains. They also specify the relationship of the record to other records in the file. Every RPG program must contain at least one record identification entry.
2. Field Description (columns 43-74). These entries describe the fields of the input record used in the particular application. Each field is described on a separate line below its corresponding record identification entry.

Filename (Columns 7-14)

Is the name that was used on the File Description Specifications form to define the input file associated with the record type described on this line. A filename must be specified for the first record identification line of the appropriate file. The same filename may be specified for another record type. All record types for one file must be specified consecutively. For the format of filename, refer to the detailed description of filename in the section *Control Card and File Description Specifications Form.*

Sequence (Columns 15-16)

Is either a 2-character alphabetic code or a 2-digit numeric code that designates the sequence of this input record type in the file.

Alphabetic Code

A unique 2-character alphabetic code (e.g., AA) should be used when:

- The file only has one record type.
- The input records are not in a predetermined sequence.
- Processing is not to be ended when input records are out of sequence.
- The specific record type is not part of a predetermined sequence.

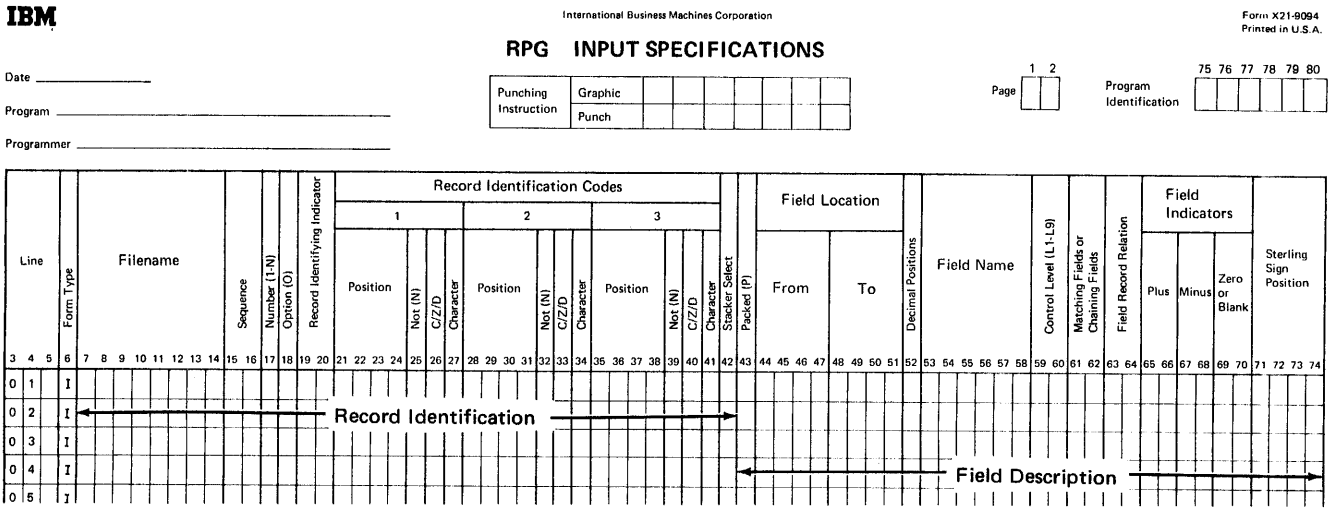


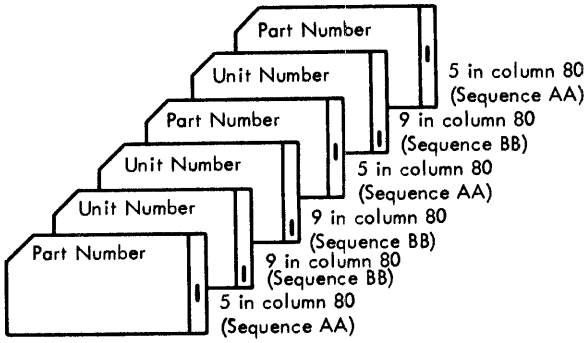
Figure 43. Input Specifications Form

In figure 44, the input data records, which contain part numbers and unit numbers, do not appear in a predetermined sequence. Consequently, each input record type is assigned an alphabetic code because the order of processing is not important to the program.

Numeric Code

A unique 2-digit numeric code must be used when the records of the input file are to be processed in a predetermined sequence. When this numeric code is used, column 17 must contain an appropriate non-blank character as described under the next subtopic *Number.*

To process input data records in a predetermined sequence, they are specified on the Input Specifications form in the same sequence in which they are to be read by the object program. For example, assume that the input records appear as shown in Figure 45. The Name record is assigned 01 because it is the first type. The Street record is the second type, and is assigned 02. The City-State record is assigned 03, and the Item Number Record is assigned 04.



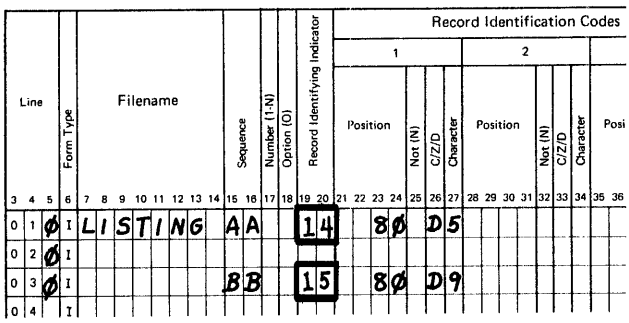


Figure 44. Input Data Records

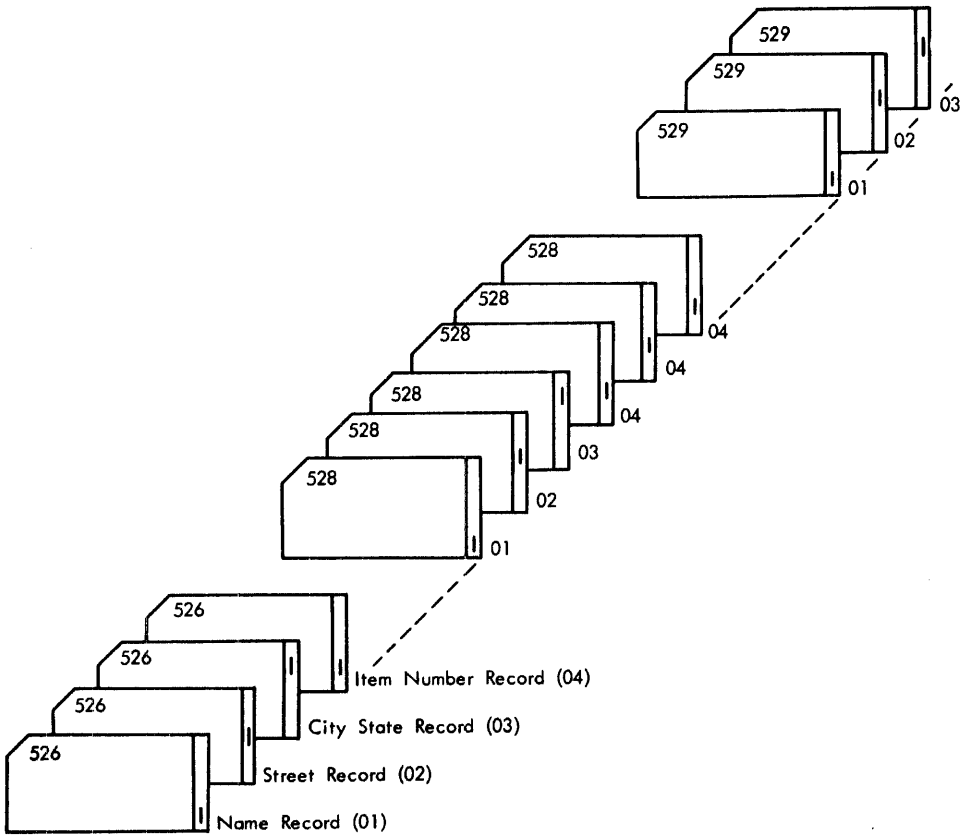


Figure 45. Sample of Input Records in Sequence

If the input records do not appear in this sequence, the processing of the program is discontinued (information out of sequence). Figure 46 illustrates the sequence-checking specifications for this example.

Restrictions

- Entries for checking a predetermined sequence (columns 15-16) must begin with 01 for each file.
- Entries for checking a predetermined sequence must be in ascending order.
- Header cards or other cards that are not in sequence must be specified on the Input Specifications form before any specifications for sequence-checking. Enter alphabetic record types first, followed by numeric sequence record types for any one file. Figure 47 illustrates the same file as Figure 46 but with a header card defined on line 010.

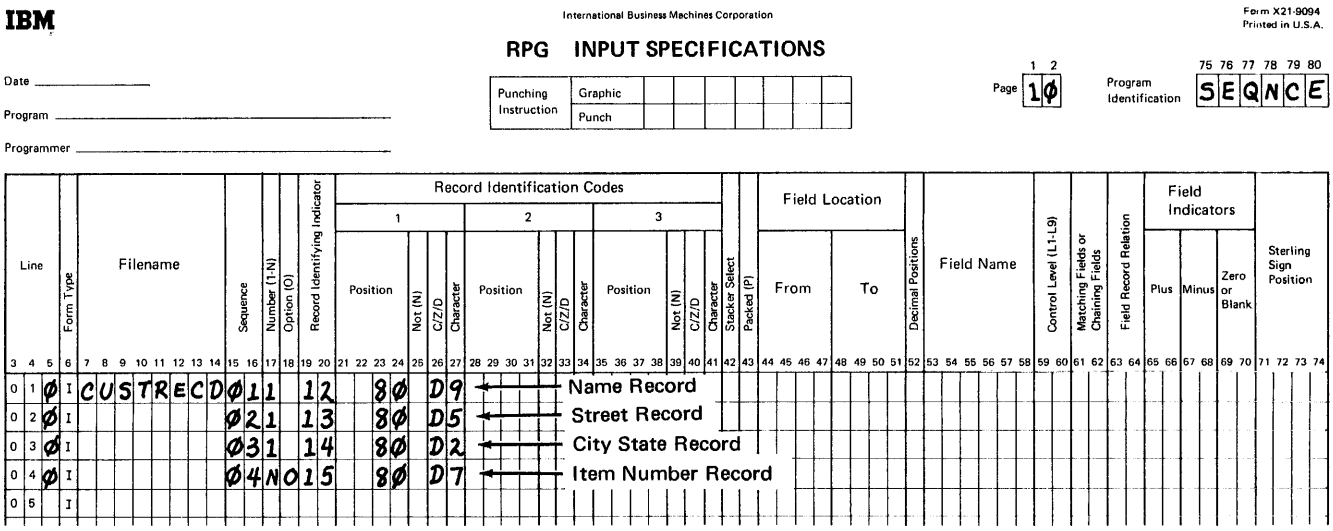


Figure 46. Sequence Checking

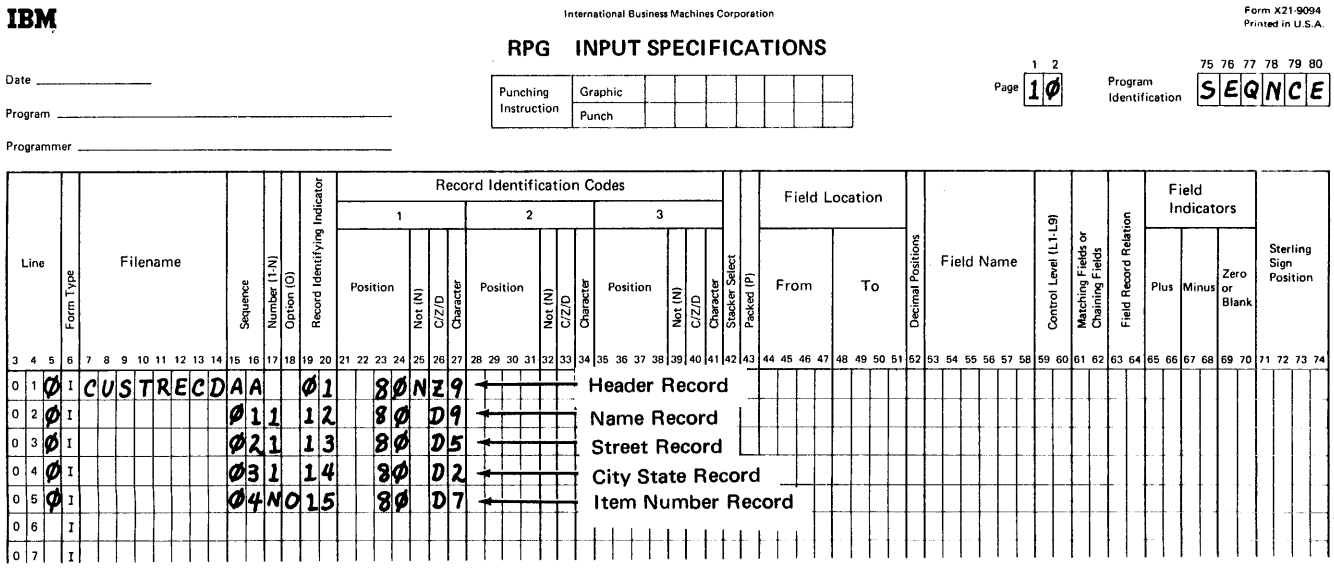


Figure 47. Sequence Checking with a Header Card

Number (Column 17)

Specifies the number of records of one type that may be present before another record type is read, as follows:

1 One and only one record of a type must be present before another record type is read.

N One or more records of a type may be present before another record type is read.

Blank The records do not appear in a predetermined sequence.

If a numeric code is assigned under Sequence (columns 15-16), a 1 or an N must be entered. If an alphabetic code has been assigned under Sequence, this column must be blank.

In Figure 46, there must be only one Name record, one Street record, and one City-State record, but there may be several Item Number records.

Optional (Column 18)

Specifies whether or not a particular record type is optional, as follows:

letter O A record of the specified type need not be present.

blank Either (1) a record of the specified type must be present, or (2) the records are not sequential.

This specification is used only with records that are sequenced by means of a numeric code.

In Figure 46, the letter O, in column 18 indicates there may be one, several, or no Item Number records in each group. Name, Street, and City-State records (one per type) must be present.

Figure 48 is an example of erroneous sequence checking. In this case, RPG ensures that the first record processed is sequence 01. Any record type can appear after this, including additional sequence 01 records.

Detection of Errors in Columns 15-18

If RPG detects a data card that does not satisfy the entries made in columns 15-18, processing is discontinued. The operator is given the option of ending the job or bypassing the record in error.

Record Identifying Indicator (Columns 19-20)

Is a 2-digit indicator used in conjunction with the entry in record identification codes (columns 21-41). The record identifying indicator:

1. Establishes a unique two-digit code for each input-record type.
2. Turns on a special indicator in the object program each time the input record is present. Thus, the object program may test for this condition during the processing of the calculation and output specifications.

IBM International Business Machines Corporation

RPG INPUT SPECIFICATIONS

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

Date _____ Program _____ Programmer _____

Punching Instruction: Graphic / Punch

Page 10 Program Identification SEQERR

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	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
3 4 5	6	7 8 9 10 11 12 13 14	15 16	17	18	19 20	21 22 23 24	25	26	27	28 29 30 31	32	33	34	35 36 37 38	39	40	41	42	43	44 45 46 47	48 49 50 51	52	53 54 55 56 57 58	59 60	61 62	63 64	65 66	67 68	69 70	71 72 73 74
010	I	* ERRONEOUS SEQUENCE CHECKING																													
020	I	RECORD	011			20		80		C1																					
030	I		02NO25					80		C2																					
040	I																														

Figure 48. Example of Erroneous Sequence Checking

To understand the use of a 2-digit record identifier, assume that an input record is identified by all of the following:

1. An X in column 80.
2. A 5 in column 79.
3. No R in column 78.

By assigning one 2-digit record identifier to represent all these codes, it is easier to refer to this type of input record during the writing of the calculation and output specifications.

The function of record-identifying indicators can be considered analogous to the function of selectors in unit record equipment, or to internal and external switches on electronic data processing equipment. The use of indicators, like the use of selectors and switches, permits the user to have certain operations occur only on specific conditions.

In Figure 46 the four record types have been assigned the record-identifying indicators 12, 13, 14, and 15. When one record-identifying indicator is turned on, all specifications pertaining to this type of record are performed. Specifications associated with other record types are not performed.

Record-identifying indicators are turned on and off during the processing of the object program, as the various record types are read. However, only the record-identifying indicator for one specified record type can be on at a time. When a record-identifying indicator is turned on, all other record-identifying indicators are turned off. This does not apply to record-identifying indicators for chained records.

Other indicator conditions that can be established in the program are field indicators (columns 65-70 of the Input Specifications form) and resulting indicators (columns 54-59 of the Calculation Specifications form). These functions are described later, but one aspect of their use is of interest at this time.

A unique two-digit code (01-99) must be assigned for each indicator used in the program. The indicator codes do not have to be assigned in any sequence. For example, four different record types could be assigned the record-identifying indicator codes 40, 62, 98, 02. The codes 01-99 can be assigned to any one of the three types of indicators. Different types of indicators with the same code are considered to be the same indicator. Therefore, care should be taken when assigning the same code to different types of indicators.

An Indicator Summary form is available to help the programmer keep track of the indicators he has used and the purpose for which each indicator was used. This form is described in the section *Program Documentation.*

Record Identification Codes (Columns 21-41)

These entries provide a means of identifying each different record type used in the object program. As mentioned previously, once the record type has been defined on the Input Specifications form, references to the record type are made by its record-identifying indicator.

These columns provide for the entry of one to three identifying codes as indicated by the numbers 1, 2, and 3 on the Input Specifications form. A record type can also be identified by specifying more than three codes in more than one line; i.e., the fourth code would be entered in columns 21-24 of the next line, the fifth code in columns 28-31, etc.

If only one record type is in the input file, record identification codes can be left blank, but a record-identifying indicator and a sequence entry must still be specified.

The three sets of entries in record identification codes are identical. Therefore, only the first set (columns 21-27) is described. The set consists of four entries as follows:

Position (Columns 21-24)
Not (column 25)
C/Z/D (column 26)
Character (column 27)

Position (Columns 21-24)

Position is the number of the position in a record type that contains the identifying code. This entry must be right-justified. Leading zeros can be omitted.

Not (column 25)

Is an N if the code described in columns 26 and 27 is not to be present in the specified position of the associated record type. Otherwise, this column should be blank.

C/Z/D (Column 26)

The object program identifies different types of input records by comparing the character written in Character (column 27) with codes contained in the records. The entry in column 26 specifies whether the object program is to compare against the

- Entire character (C).
- Zone portion (Z), or
- Digit portion (D).

Enter a C, Z, or D in column 26, as applicable.

Using the C specification saves core storage and reduces the time required for execution of the object program.

Character (Column 27)

Specifies the character to be used by the object program to identify different types of input records.

The particular character allowed for record identification (column 27) depends on the particular specification used in column 26.

C – Character: If column 26 is C, any of the 256 extended binary-coded-decimal interchange code (EBCDIC) characters may be specified in column 27.

Z – Zone: The zone portion of the punched card column consists of the 11 and 12 punch positions. If column 26 is Z, tests can be made for card zones or for core zones.

1. *Testing card* zones
   To test for a typical 12 zone, one of the characters A through I, &, or one of the other 7 EBCDIC characters that have the same hexadecimal zone as the letter A (hexadecimal C) must be specified in column 27. The presence of a 12 zone is recognized only if one of these characters is present in the appropriate position of the record.

   To test for a typical 11 zone, one of the letters J through R, the minus sign (-), or one of the other 7 EBCDIC characters that have the same hexadecimal zone as the letter J (hexadecimal D) must be specified in column 27. The presence of an 11 zone is recognized only if one of these characters is present in the appropriate position of the record.

   To test for the absence of both the 11 and 12 zones (no zone), any of the 16 EBCDIC characters having the same hexadecimal zone as 1 (hexadecimal F) or a blank, must be specified in column 27, or that column must be left blank. The absence of zones is recognized only if one of these characters is present in the appropriate record position, or if this record position is blank.

2. *Testing Core Zones*
   Any specification in column 27 other than those used in testing for card zones can be used to identify record codes with the same zone in core storage as the character specified in column 27.

D – Digit: The digit portion of the punched card column consists of the 0 through 9 punch positions. If column 26 is D, the digit portion of any of the 256 EBCDIC characters can be tested by placing any one of these EBCDIC characters in column 27.

Examples of Record-Identification Codes

Examples of record-identification codes are shown in Figure 49.

Circled numbers in the figure correspond to the item numbers in the following text:

1. Line 010 causes the specified record-identifying indicator to be set on if the minus sign (-) or any one of the 16 EBCDIC characters that have the same zone as the letter K is contained in record position 48. A Z must be placed in column 26 so that only the zone portion is checked. (If a C were placed in this column, the object program would compare the letter K, instead of an 11-punch, against the input-record code.)
2. Line 020 turns on the specified record-identifying indicator if a character whose digit portion is 5 is contained in record position 62 (e.g., a 5, N, V, or E would meet this requirement).
3. Line 030 turns on the associated record-identifying indicator if the letter T is contained in record position 49.
4. Line 040 specifies that three conditions must be met in the one input record. This is referred to as an AND relationship. In the example, the associated record-identifying indicator is turned on if a 4 appears in column 16 of the input record, column 40 does not contain a 5 and column 80 contains an ampersand (&) or any one of the 16 EBCDIC characters has the same zone as the letter A.
5. Lines 060 and 070 show the use of more than one code on more than one line to identify one type of input record. This again is termed an AND relationship, but in this case two lines are used to specify the conditions and the letters AND are written in columns 14-16 of the second line. In this example, the specified record-identifying indicator is set on if an input record is read that contains all of the following:

   - A 4 in position 16.
   - No 5 in position 40.
   - An 11-punch in position 80.
   - A 2 in position 20.
   - A 3 in position 25.

IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic
	Punch

Page 1Ø

Program Identification IDENT

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank	Sterling Sign Position
3 4 5	6	7 8 9 10 11 12 13 14	15 16	17	18	19 20	21 22 23 24	25	26	27	28 29 30 31	32	33	34	35 36 37 38	39	40	41	42	43	44 45 46 47	48 49 50 51	52	53 54 55 56 57 58	59 60	61 62	63 64	65 66	67 68	69 70	71 72 73 74
0 1 Ø	I	INPUT	AA			Ø1	48			ZK																					
0 2 Ø	I		BB			Ø2	62			D5																					
0 3 Ø	I		CC			Ø3	49			CT																					
0 4 Ø	I		DD			Ø4	16			C4	4Ø	N	C	5	8Ø		Z	A													
0 5 Ø	I																														
0 6 Ø	I	PAYREPT	AA			26	16			C4	4Ø	N	C	5																	
0 7 Ø	I		AND				2Ø			C2	25		C	3	8Ø		Z	K													
0 8	I																														

Figure 49. Specifying Record Identification Codes

Additional specification lines can be used to specify as many record-identification codes as required. Each additional line must begin with the word AND in columns 14-16. Columns 17-20 of each AND line must be blank.

An AND relationship is concerned with specifying more than three record identification codes for one record type, whereas an OR relationship is used to specify two *different record types* with just one set of field description specifications. The fields in the two record types may be in the same or in different positions.

To specify an OR-relationship, enter the word OR in columns 14-15. Columns 16-18 of OR lines must be left blank. For detailed information, refer to *Records in an OR-Relationship* contained later in this section.

Omitting Record-Identification Codes

When all input records are to be processed alike, columns 21-41 may be left blank. However, a sequence must be specified in columns 15-16.

If the different record types are not in a predetermined sequence and if only cetrain record types within a group are to be processed, they must be listed first with their record identification codes. The remaining record types can be bypassed or processed as a group. In either case, they are specified as the last record type and they must be assigned a sequence. Columns 21-41 of such lines are left blank. If the records are to be bypassed, they should not be referred to on the Calculation or Output-Format Specifications forms.

Duplicate Identification

If the program reads a record which satisfies two or more record types, only the first record type is recognized. In Figure 50 record-identifying indicator 02 will never be turned on, because the first record type (AA) is satisfied.

IBM

International Bus

RPG INPUT

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic
	Punch

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Position
3 4 5	6	7 8 9 10 11 12 13 14	15 16	17	18	19 20	21 22 23 24	25	26	27	28 29 30 31	32	33	34	35 36 37
0 1 Ø	I	CARDS	AA			Ø1		1	C	5					
0 2 Ø	I														
0 3 Ø	1		BB			Ø2		1	C	5		2	C	4	
0 4	I														

Figure 50. Duplicate Identification of Record Types

Undetermined Record Type

If the program cannot detect a record type, a halt occurs and the operator is given a choice of continuing with the next record or ending the job. In many applications, the user may wish to program for undetermined record types and process them differently.

This can be accomplished by defining a record type with no record identification. This record type must be specified following all valid types, as illustrated in Figure 51.

Stacker Select (Column 42)

Specifies that input cards are to be selectively placed into a particular stacker of a multi-stacker input unit attached to the system. The entry in column 42 may be one of the following:

- Number of the stacker into which the cards are to be placed after they are read.
- Blank, if the cards are to be placed in the normal stacker after they are read, or if the input unit has only one stacker.

For example, for the IBM 1442 Card Reader, the normal stacker, 1, may be specified with either a 1 or a blank in column 42. A 2 specifies stacker number 2.

IBM International

RPG INPU

Date _____

Program _____

Programmer _____

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Posi
3 4 5	6	7 8 9 10 11 12 13 14	15 16	17	18	19 20	21 22 23 24	25	26	27	28 29 30 31	32	33	34	35 36
0 1 Ø	I	INPUT	AA			Ø1	1		CA						
0 2 Ø	I														
0 3 Ø	I		BB			Ø2	1		CB						
0 4 Ø	I														
0 5 Ø	I		CC			Ø3	1		CC						
0 6 Ø	I														
0 7 Ø	I		DD			Ø4	1		CD						
0 8 Ø	I														
0 9 Ø	I		EE			Ø5									
1 0	I														

Punching Instruction: Graphic / Punch

Figure 51. Detecting Undetermined Record Types

A different stacker may be specified for each OR line. If only one stacker is required in an OR relationship, this stacker must be specified on each of the OR lines.

Two input/output areas are specified by means of a digit 2 in column 32 of the File Description Specifications form. If two I/O areas are specified for the file, any stacker select specification is ignored.

Using two input/output areas requires that a second card be read before stacker selection can occur for the first card. This prevents the device from allowing a stacker select and forces the card into stacker 1.

FIELD DESCRIPTION ENTRIES

As mentioned previously, the Input Specifications form consists of two categories of entries: record identification and field description entries.

In record identification entries (columns 7-42), a line contains the specifications for one record type.

In field description entries (columns 43-74), one line represents the specifications for one field of an input record type.

Field description specifications are always written on the line immediately below the specifications that identify the record type.

Field description entries describe those fields of the input record that are to be used in the application. Each field of the record requires one line on the Input Specifications form. Columns 7-42 of each field description line must be left blank. Fields are always removed in the same sequence by which they are specified.

Packed (Column 43)

Indicates whether or not the data is packed, as follows:

P The data is packed.
blank The data is not packed.

Packed decimal format may only be used in disk files. When data is packed, two decimal digits are represented in one record position.

Field Location (Columns 44-51)

Are numeric values that define the beginning and ending location of a field within the input record.

The maximum length of a numeric field is 14 digits. The maximum length of an alphameric field is 256 characters.

IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic							
	Punch							

Page 05

Program Identification SEQNCE

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	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
010	I	REPORT	AA			14	35		C	5	80	NZ		K																	
020	I		OR				35		C	6																					
030	I																				1	4		DIVSON							
040	I																				5	8		DEPT							
050	I																				9	14		EMPNO							
060	I																				15	28		NAME							
070	I																				46	50	Ø	FLDA							
080	I																				56	60	Ø	FLDB							
090	I																				66	70	Ø	FLDC							
100	I																				71	71		FLDE							
110	I																														

Figure 52. Specifying Input Fields in Numerical Sequence

From (Columns 44-47)

Specifies the location of the leftmost position (i.e., the high-order position) of the field. This entry must be right-justified; leading zeros may be omitted.

To (Columns 48-51)

Specifies the location of the rightmost position (the low-order position) of the field. This entry must be right-justified; leading zeros may be omitted.

The fields of an input record may be described in any sequence. In Figure 52, they are shown in numerical sequence (by field location) to make the example easier to understand. A one-position field may be defined by placing its location into both From and To fields.

A field location may be defined twice with two different names. A field may be defined within another field (e.g., CUSTMR in positions 1 to 27 with NUMBER in positions 1 to 10 and NAME in positions 11 to 27). Two or more fields may overlap each other.

Records in an OR Relationship

In many applications, records may contain almost identical fields or the same record type may be denoted by two or more record identification codes.

This could be described as two entirely different record types. However, the programmer would then have to describe the same information two or three times, and more core storage would be necessary. To simplify this, RPG provides for records in an OR relationship.

The OR relationship allows one set of specifications to be used to describe fields that are contained either in the same locations of two or more different records, or in different locations of two or more records. Record identifying indicators need not be respecified. However, a different record identifying indicator may be specified in each OR line.

Figure 52 shows the OR relationship used to describe fields contained in the same locations of two or more record types. OR is entered in columns 14-15 of line 020. Columns 16-18 are blank as they must be when an OR relationship

is used. This specifies that the fields described in lines 030-100 are in input records that have a 5 in column 35 and no 11 punch in column 80, and in input records that have a 6 in column 35. Identifying indicator 14 has been previously specified. Therefore, it is not entered in the OR line.

Figure 53 also shows the OR relationship used to describe fields contained in different locations of two or more record types.

Record identifying indicator 14 is turned on whenever an input record contains a 5 in column 35 and no 11-punch in column 80. Record identifying indicator 16 is turned on whenever an input record contains a 6 in column 35.

The OR line indicates that there is a second type of input record that contains the required information. However, field FLDC is not contained in the same locations. For this reason, FLDC must be described twice. Each of the two entries for FLDC must be related to the appropriate record type. This is accomplished by entering the pertinent record identifying indicator in Field-Record Relation (columns 63-64). (An indicator entered in Field-Record Relation is referred to as a field-record relation indicator).

Thus, when record identifying indicator 14 is on, FLDC is taken from locations 66-70 of the input record. When record identifying indicator 16 is on, FLDC is taken from locations 61-65 of the input record.

Decimal Positions (Column 52)

Is a value 0-9 that is used:

- To specify the number of decimal positions in the field specified in Field Location.
- To cause all zones of the field specified in Field Location to be ignored during comparisons caused by a control-level or a matching-fields specification.
- To cause the field specified in Field Location to be numeric which is essential for arithmetic or editing functions.

One of the following entries can be specified in column 52:

- 0, if the input data field contains a whole number; that is, a non-fractional number.
- A value 1 through 9, which denotes the number of decimal positions in the input data field. This value must not exceed the field length described in columns 44-51.
- Blank, if the input data field is to be treated as alphameric data.

IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page 10

Program Identification: REPORT

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	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
010	I	REPORT	AA			14	35		C5		80WEK																				
020	I		OR			16	35		C6																						
030	I																				1	4		DIVSONL3							
040	I																				5	8		DEPT	L2						
050	I																				9	14		EMPNO	L1						
060	I																				15	28		NAME							
070	I																				46	50		FLDA							
080	I																				56	60		FLDB							
090	I																				66	70		FLDC			14				
100	I																				61	65		FLDC			16				
110	I																														

Figure 53. Specifying Input Fields

A value from 0 through 9 must be specified if column 43 contains a P or if the input data field is to be used in arithmetic or editing operations.

In Figure 53, the blank in column 52 adjacent to the fields named DIVSON, DEPT, EMPNO, and NAME defines those fields as alphameric fields. A zero in column 52 adjacent to the remaining fields in the figure defines those remaining fields as numeric fields that contain whole-number quantities.

Field Name (Columns 53-58)

Is a name used to identify a data field. Once a name has been assigned to a field, it can be referred to in calculation and output-format specifications.

The field name can consist of one through six alphameric characters. It must begin with an alphabetic character in column 53. It cannot contain special characters or embedded blanks.

The same field name may be used for any number of fields of different records, provided that all these fields have the same length and the same number of decimal positions as the first field given this name. A field name is assigned only once by RPG. The same storage location is used for fields with identical names.

Defining the Same Data Field as Alphameric and Numeric

If a field from an input record is to be used as both an alphameric and a numeric field, that field must be defined twice by assigning two different field names to the same location.

Using Input Data Fields for Constant Data

Constant data represents information that is not changed with each record. It is usually not altered during the object run. Examples of constants are date cards, or other data that may be changed for each run.

Constant data is supplied by defining a special record type and by assigning a field name that is not otherwise used. This technique also permits the entering of constants that are too large to be specified as literals in the source program.

When only disk files are being processed, specifications for an additional card file are required for the constant information. The file must be specified as follows:

1. In multifile programs using the matching technique, the additional card file must be designated as a secondary file without matching fields. Thus, the additional card file will be processed immediately as the cards are read.
2. In programs that do not use the matching technique (e.g., single-file programs or programs using the chaining technique), the additional card file must be designated as a primary file. For detailed information about matching and chaining techniques, see *Processing Multiple Input Files.*

Control Level (Columns 59-60)

Assigns a control level to the field specified on this line. One of nine control level indicators (L1, L2, L3. . .L7, L8, or L9) can be specified in any sequence. That is, L2 may appear before L1 on the Input Specifications form. These indicators are used to control when operations specified on the Calculation and Output-Format Specifications forms are to be performed.

Whenever a field assigned a control level indicator is encountered by the object program, the data in the control field is compared with data in the same control field from the previous card. If these differ, a control break occurs and the control level indicator assigned to the field is turned on.

Control level indicators are ranked in order of importance. L1 designates the lowest control level; L9 the highest. When a certain control level indicator is turned on, all control level indicators lower than it are also turned on. For example, if control level indicator 3 is turned on, control level indicators 1 and 2 are also turned on.

Thus, when a control break occurs, the operations on the Calculation and Output-Format Specifications forms that are associated with the new control level are performed in addition to the operations associated with all control levels lower than the new control level.

As mentioned in the description of the Decimal Positions entry, fields can be designated as being numeric or alphameric by the use of column 52. When an alphameric field is used as a control field, the zone portion of the field is used in the comparison of one record with another to determine a control break.

When a numeric field is used as a control field, the zone portion and the sign of the field are not used in the comparison. However, the sign is used in calculations performed on the data in that field.

Since the zone portion of a numeric field is not used when checking such a field for a control break, a +100 and a -100 compare equally and do not cause a control break. Furthermore, since a numeric control field contains only the digits 0 through 9, field values 0001 and bbb1 (where b denotes a blank) also compare equally and do not cause a control break.

Figure 53 shows the entries for three control fields. Indicator L3 is the highest control level used in this example. Therefore, the field named DIVSON is the highest level of control. Since DIVSON is a numeric field (denoted by the 0 in column 52), the zone portion of the data in that field is not used when checking for a control break. In this example, indicator L3 can be used to specify when the calculations for DIVSON are to occur, or when the totals for DIVSON are to be printed or punched. However, when indicator L3 is on, all operations associated with indicators L1 and L2 are also performed.

Split Control Fields

Several fields in an input record type can be specified as one control field. That is, the same control level indicator can be assigned to several fields. Such a field is called a split control field.

Split control fields of any one level within a card type must be specified in subsequent field description lines. All fields are placed according to the sequence in which they are defined on the Input Specifications form. For example, the first field defined on the input specifications form is placed in the high-order position, and the last field is placed in the low-order position.

An example of using split control fields is given following the rules for using control fields.

Rules for Using Control Fields

The following must be considered when using control field specifications:

1, Control field specifications for a chained file are ignored.
2. Within each control level, the length of the control fields or the overall length of the split control fields must be the same.
3. Split control fields: Several fields in an input record type can be specified as one control field to form a split control field.

Split control fields of any one level must be specified in subsequent field-description lines. All fields are placed according to the sequence in which they are defined on the Input Specifications form. For example, the first field defined in the input specifications is placed in the high-order position, and the last field is placed in the low-order position.

4. Control fields of one level can be specified as numeric or alphameric alternately. If a control field is specified once as a numeric field, all control fields of the same level will be treated as numeric control fields throughout, even if they are specified as alphameric in other input specifications.
5. The length of a packed numeric field is calculated as 2n-1, where n is the number of characters contained in the input field concerned. If control fields of one level are specified as packed and unpacked fields alternately, the unpacked fields must always have the same (odd) length as the packed numeric fields.

   If the length of the unpacked field is even (i.e., the first digit of the packed field is not to be used by the object program), the programmer should specify (as split control fields) the first column of the packed field as unpacked numeric or alphameric and the remainder as packed numeric in two subsequent lines on the specifications form.

   Packed data can occur only in disk records.
6. If multiple record types are specified in an OR relationship, an indicator entered in Field-Record Relation can be used to relate control fields to the pertinent record types.

   The following rules apply:

   - The control-field specifications must be grouped by field-record relation indicators.
   - The control fields with no such indicators must be specified first within one record.
   - Fields other than control fields may be interspersed anywhere within the set of field descriptions.
   - Only a record identifying indicator from the main record or an OR relation record type can be used in field record relationship of a control level.

   The control fields that have no field-record relation indicators are assumed to be contained in all records specified in the OR relation group described in the preceding paragraph. If control fields of one level are specified with and without field-record relation

indicators, only the control fields conditioned by field-record relation indicators are used when the appropriate indicator is turned on. The control-field specification without a field-record relation indicator is then disregarded.

7. Since all zone bits (other than the units position) are stripped from numeric control fields (and an 'F' sign is forced) a +100 will compare equally with a -100 and will not cause a control break.
8. If a control field is specified as numeric, it will contain only digits 0 to 9. Consequently, field values of 0001 and bbb1 (where b = blank) will compare equally and not cause a control break.
9. Control fields are initialized to hexadecimal zeros. Thus, the first record with control fields specified that is read will cause a control break. This is true of a field of all blanks or all zeros (EBCDIC). The only exception would be if an alphameric field containing hexadecimal zeros were used as a control field.

Figure 54 (Part 1) shows an example of using three levels of control fields contained in four different records in an OR-relationship.

Figure 54 (Part 2) shows the record locations from which the control fields will be taken when the different record types are read.

Record Type 91: The L1 and L3 control fields are specified and conditioned by the field-record relation indicator 91 in lines 110-130. The entries with indicators L1 and L3 that are not conditioned by field-record relation indicators are disregarded. Field-record relation indicator 91 is not entered for the L2 control field specified in line 070. Therefore, this L2 control field is used by the object program for records of type 91.

Record Type	L3		L2	L1	
91	51-70		11-20	6-10	46-50
92	51-60	31-40	61-70	1-10	46-50
93	51-60	31-40	11-20	1-5	46-50
94	51-60	31-40	11-20	1-5	46-50

Figure 54. Split Control Field Specifications and Record Locations (Part 2 of 2)

IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic							
	Punch							

Page 10

Program Identification REPORT

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	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
010	I	SPLTTCTLAA				91	80		C1																					
020	I		OR			92	80		C2																					
030	I		OR			93	80		C3																					
040	I		OR			94	80		C4																					
050	I																				1	5		FLDA	L1					
060	I																				46	50		FLDB	L1					
070	I																				11	20		FLDC	L2					
080	I																				76	77		DAY						
090	I																				51	60		FLDD	L3					
100	I																				31	40		FLDE	L3					
110	I																				6	10		FLDA1	L1		91			
120	I																				46	50		FLDB1	L1		91			
130	I																				51	70		FLDF	L3		91			
140	I																				6	10		FLDA1	L1		92			
150	I																				46	50		FLDB1	L1		92			
160	I																				74	75		MONTH			92			
170	I																				61	70		FLDC2	L2		92			
180	I																				78	79		YEAR						

Figure 54. Split Control-Field Specifications and Record Locations (Part 1 of 2)

Record type 92: L1 and L2 indicators are specified in lines 140-170. The specifications with the L3 indicators entry in lines 090-100 are used by the object program.

Record types 93 and 94: Control-field entries conditioned by field-record relation indicators are not specified for these records. The contents of the fields specified without a field-record relation indicator entry (lines 050-100) are obtained and processed by the object program for all three control levels.

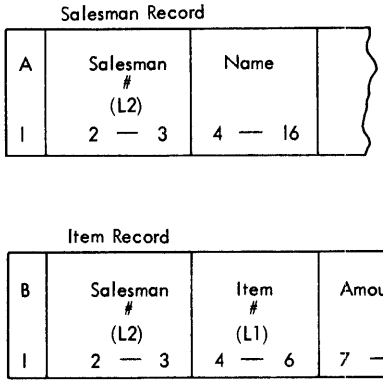
False Control Levels

Different record types normally contain the same number of control fields. However, some applications require a different number of control fields in some records. This is shown in Figure 55. The salesman records contain only the L2 control field. The item records contain both L2 and L1 control fields.

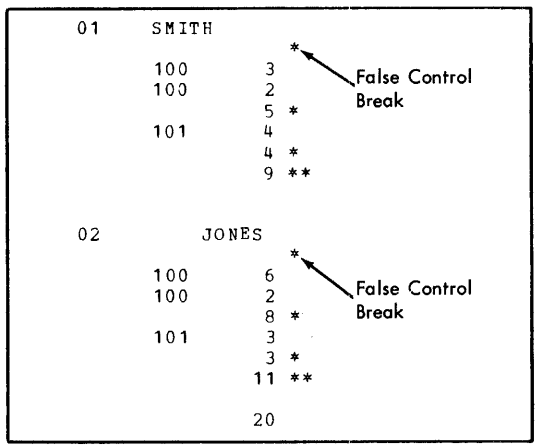
With normal RPG coding, a false control break is created by the first item record following the salesman record. This is recognized by an L1 control break immediately following the salesman record, and results in an asterisk being printed on the line below the salesman record (see Figure 55).

Figure 56 contains excerpts from a program that processes the input shown in Figure 55 and prevents the false control break from occurring, thus producing the corrected output shown in Figure 55.

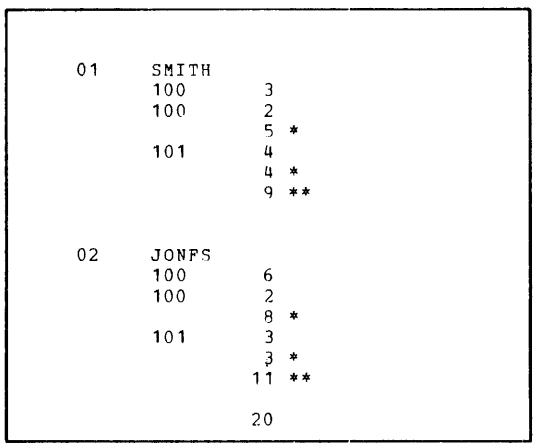
Line 040 of the Calculation-Specifications form sets on indicator 11 when the salesman record is read. When the next item record causes an L1 control break, no total output is printed because indicator 11 is on (line 090 of Output-Format Specifications form). Detail calculations are then processed for the item record and line 050 of the Calculation Specifications form sets indicator 11 off. This allows the normal L1 control breaks to occur.



A. Input Records

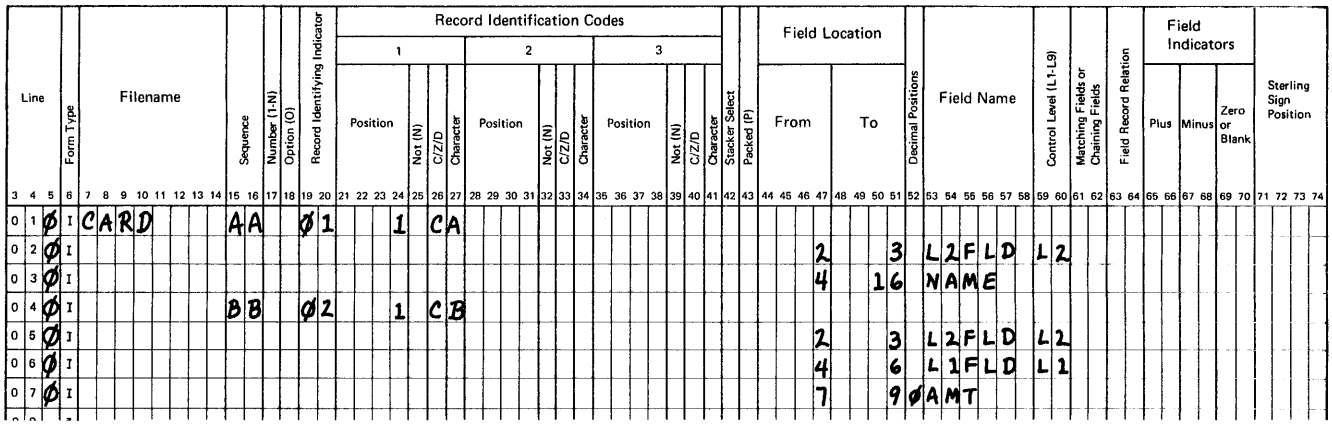


B. False Control Level Break Output

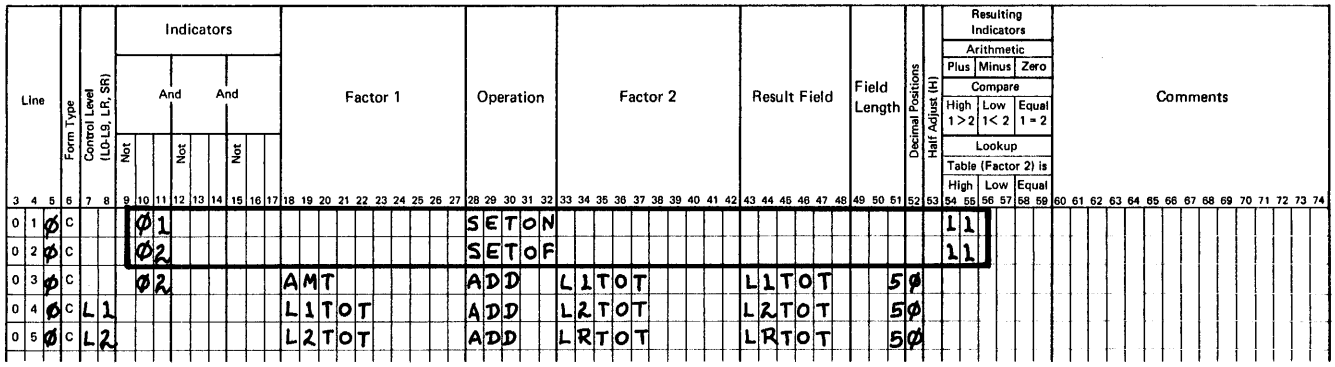


C. Correct Output

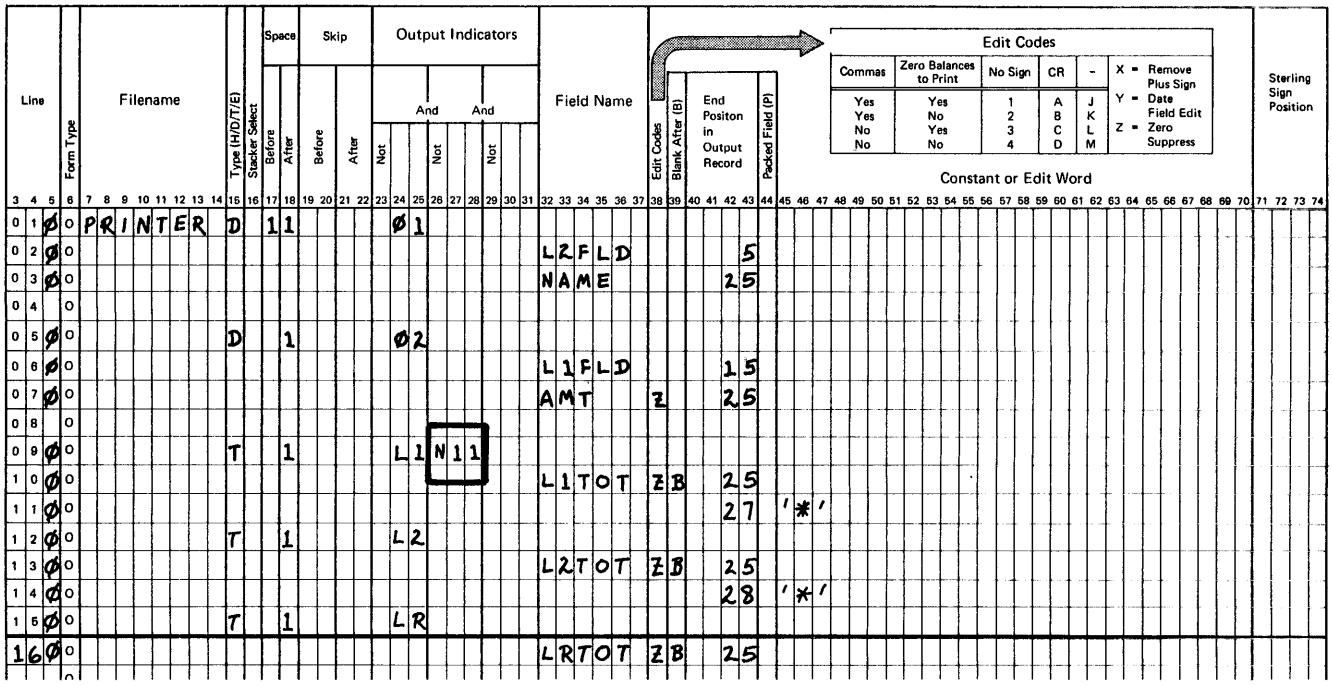
Figure 55. False Control Level Break



A. Input Specifications



B. Calculation Specifications



C. Output-Format Specifications

Figure 56. Specifications to Avoid False Control Level Breaks

Matching Fields or Chaining Fields (Columns 61-62)

This specification provides the ability to match or to chain the records of one file with those of another file.

Matching Fields

Matching fields within a record are designated by entering a notation M1, M2, . . . M8, or M9 (these notations are not to be confused with indicators) in columns 61-62 of the appropriate field description specifications line. These entries can be used for (1) matching fields with one input file, or (2) matching fields with several input files. A maximum of nine matching fields can be specified.

1. *Matching fields used with one input file:*

   If only one input file is specified, fields within the file may be sequence checked by using the matching field entries. The MR indicator is not turned on.

   If the fields are out-of-sequence, the program halts and an appropriate message number is placed in the accumulator. The user next has the option to terminate the job, or to bypass the record and continue processing.

   Figure 57 shows the specifications for sequence checking three input fields: DEPT, REGION, and DIVSON. When the second record is read, data from the three fields is moved to the matching field holding area as shown in Figure 58. The three fields are treated as one; M3 is placed in the high-order position and M1 is placed in the low-order position. The compare operation between the matching fields of the two subsequent records of the file is made on all three fields at the same time.

   This example checks to ensure that the records are in ascending sequence. It is assumed that the programmer has indicated that the file is in ascending sequence by the value specified in column 18 of the associated File Description Specifications form.

2. *Matching fields used with several input files:*

   When more than one input file is specified with matching field entries, matching fields are used to match records of the primary file with those of the secondary file. The entries in columns 61-62 determine which fields are to be matched. The comparison is made as shown in Figure 58. For details, refer to *Processing Multiple Input Files.*

Rules for Using Matching Fields

The following must be observed when specifying matching fields:

1. The locations of the matching fields within a record type of a file must remain fixed, except when field record relation is used.
2. When several record types are in an input file, the locations of the matching fields in the various types of records need not be the same.

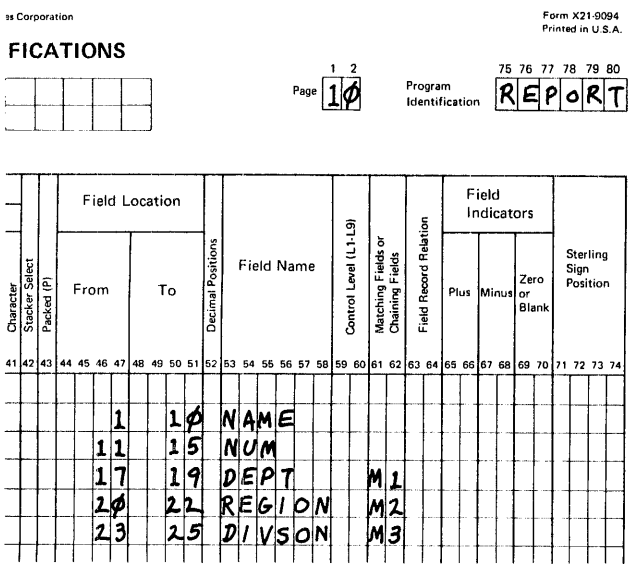


Figure 57. Using Matching Fields in a Single Input File

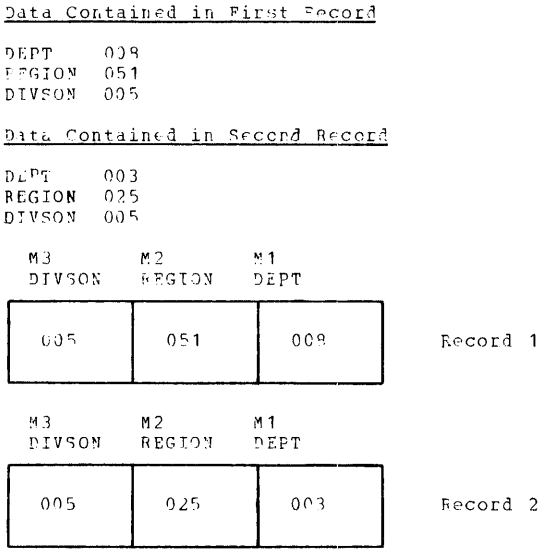


Figure 58. Comparing Matching Fields

3. The same number of matching fields must be specified on both the primary file and any secondary file. For example, if M1, M2, and M3 are specified on the primary file, all three notations must also be specified on a secondary file.
4. Matching field notations M1 through M9 must not be made for a chained file. These notations can be used for a file retrieved with a record address file.
5. If matching fields are specified for several records in a program, the same number and level of matching fields must be specified for each of these records. They must also have the same field length for the same level. Thus, it is not permitted to specify a record with only a matching field M1, and another record with only a matching field M2, or both M1 and M2. It is also not permitted to specify the same matching field, for example M1, with different lengths on two field description lines.
6. Split matching fields must not bespecified.
7. Matching fields of the same level can be specified as numeric in packed or unpacked decimal format alternately, or as alphameric fields. If a matching field is once specified as numeric, all matching fields of the same level will be treated as numeric matching fields throughout, even if they are specified as alphameric in other specifications. That is, all zones are ignored by the internal compare operation for matching and/or sequence checking.
8. The length of a pack matching field is calculated as 2n-1, where n is the number of characters contained in the input field concerned. The calculated length is always an odd number. Packed format can occur only with disk records.

   If matching fields of one level are specified as unpacked and packed alternately, the lengths of the unpacked matching fields must equal the calculated lengths of the packed numeric matching fields.

   When matching fields have an even length in unpacked decimal format, the only way to specify these fields in packed and unpacked decimal format alternately is to define the first (high-order) column of both fields as unpacked numeric fields M2, and the remainder of the fields (i.e., the odd-length portion) alternately as the lower-level matching field M1.
9. Field-Record relations for matching fields must be specified with record identifying indicators from the main specification line or an OR relation line to which the matching field refers. If multiple record types are specified in an OR relationship, an indicator entered in Field-Record Relation can be used to relate matching fields to the pertinent record types. The following rules apply:

- The matching field specifications must be grouped by field-record relation indicators.
- Matching fields with no such indicators must be specified first within one record.
- Fields other than matching fields may be interspersed anywhere within the set of field description.

Matching fields without field-record relation indicators are assumed to be contained in all records specified in the OR relation group described in the preceding paragraph. If matching fields of one level are specified with and without field-record relation indicators, only the matching fields conditioned by field-record relation indicators are used when the appropriate indicator is turned on. The matching field specification without a field-record relation indicator specified is then disregarded.

Figure 59 lists invalid matching field and/or field-record relation entries, then explains how these entries can be made valid. Following are valid examples of field and/or field-record relation entries; the first entry is the match field and the second entry is the Field-record relationship indicator.

Valid Field	Field-Record Relation Entries				
1.	2.	3.	4.	5.	6.
M1 01	M1 01 M1 02	M1 M1 01	M1 M2 M1 01	M1 M2 M1 01 M1 02	M1 M2 M3 M1 01 M2 01 M1 02

Chaining Fields (Disk Only)

Two methods of chaining are allowed with 1130 RPG. The preferred method uses the CHAIN operation discussed in the section called *Calculation Specification*. The following discussion concerns the alternate method of chaining and the required entries on the Input Specification sheet.

As many as three chaining fields are permitted in a record. To specify chaining, enter the code that identifies the chaining field C1 to C3 in columns 61-62.

Invalid Entries*	Reason	Method of Correction	Corrected Entry*
M1 01 M1	Entry with field record relation precedes entry without field record relation.	Switch the entries	M1 M1 01
M1 M2 01 M2 02	Entry without field record relation is specified, but some match fields contain a field record relation.	Add dummy M2 field without field record relation after the M1 field.	M1 M2 M2 01 M2 02
M1 01 M2 01 M1 02 M2 02 M3	Entry without a field record relation is specified, but some match fields contain a field record relation.	Either of two ways: (1) Delete M3 statement; add two M3 statements conditioned by 01 and 02 in proper sequence. (2) Define M1 and M2 without field record relation and place first followed by M3.	(1) M1 01 M2 01 M3 01 M1 02 M2 02 M3 02 (2) M1 M2 M3 M1 01 M2 01 M1 02 M2 02
M1 M2 M1 01 M1 02 M2 01 M2 02	Match fields with field record relation are not grouped together by indicator.	Group the match fields by indicators.	M1 M2 M1 01 M2 01 M1 02 M2 02

* First entry is the match field (columns 61-62). Second entry is the field record relationship indicator (columns 63-64). Assume that indicators 01 and 02 are record identifying indicators in an OR relationship.

Figure 59. Specifying Matching Fields With Field Record Relation Indicators

A chaining field can be specified as an alphameric, or an unpacked numeric field. The format of the key field on disk can be alphameric, therefore:

- Alphameric fields are moved unchanged into the appropriate chaining fields (C1 to C3).
- Unpacked numeric fields are moved in unpacked format, but with 'F' zones, into the appropriate chaining fields.
- If a chaining field is specified alternately as numeric and alphameric, it is treated as a numeric field. In other words, the zone parts of the alphameric field are always set to 'F'. Note that all positions of the key field on disk must also have 'F' zones if a chaining field is specified as numeric.

The use of file chaining is explained in the section, *Chaining.* A chaining file can also be sequence checked by entering M1-M9 in Chaining Fields (columns 61-62) on the line containing the specification for the appropriate field.

If the same field is used for chaining and sequence checking, it must be defined twice using two different field names.

Note: Matching field and control-field specifications are not permitted on the chained file input records.

Field Record Relation (Columns 63-64)

Is used in conjunction with specifications for records in an OR relationship. The columns contain the indicator that associates the field named on this line with the proper input record of an OR relationship.

Field-record relation can be used in connection with:

- Control levels (see the section, *Rules for Using Control Fields*, item 6).
- Matching fields (see the section, *Rules for Using Matching Fields*, item 9).
- Chaining fields (see the paragraphs that follow).

Instead of using the CHAIN operation on calculation specifications, field-record relation can be used to selectively control chaining operations. This method is described in the next two paragraphs. The description assumes a knowledge of chaining operations described in the section, *Chaining*.

If a record identifying indicator is placed in Field-Record Relation, the chained record is obtained only if any one of the record types specified in an OR relationship is present, and the record identifying indicator (in Field-Record Relation) is on.

However, if a chaining field is specified on a field description line and no entry is made in Field-Record Relation, the chained record is obtained when any of the record types specified in an OR relationship is present. Any indicator may be used to condition chaining.

Field Indicators (Columns 65-70)

This specification is used to test the status of a field when it is read into the system. Depending upon the status of the field – plus, minus, or zero or blank – it turns on an indicator that can be used to control calculation and output specifications to stop processing of the object program.

The entry for the specification is an indicator which is turned on when the field specified on the line is plus, minus, or zero or blank.

Plus (Columns 65-66) occurs whenever the value in the associated numeric input field is greater than zero. This entry is used only with numeric fields.

Minus (Columns 67-68) occurs whenever the value of the associated numeric input field is smaller than zero. This entry is used only with numeric fields.

Zero or Blank (Columns 69-70) occurs whenever either (1) the associated numeric input field is zero or blank, or (2) the associated alphameric field is blank. Fields that are all zeros turn on blank indicators even though a plus or minus sign is present.

Note: Alphameric input fields must not be assigned field indicators in Plus or Minus (i.e., Columns 65-68 must be blank).

Codes Available for Use as Field Indicators

Codes used as field indicators may be either numeric (01-99) or halt indicators (H1-H9).

Numeric Indicators: A 2-digit, field-indicator code is used for this specification. These codes, ranging from 01 through 99 can be defined one or more times on the form. If they are defined more than once, the second specification of this indicator resets it from the status it may have had by the previous specification for it.

Note: Defining these indicators means specifying them in columns 65-70. Using these indicators means specifying them in Indicators (columns 7-17) in the Calculation specifications or in Output Indicators (columns 23-31) in the Output-Format specifications as many times as required. In the latter case, they are merely tested to determine their status but are *not* reset by the test.

Halt Indicators: Nine additional indicator codes, called halt indicators, can be used in the RPG program. These indicators, designated as H1 through H9, halt the processing of the object program when error conditions (as determined by the programmer) have been detected. The user then has the option of either terminating the job or continuing processing. For example, if H1 is placed in columns 69-70 (Zero or Blank), it is only turned on whenever the associated field of the current input record is blank or contains zero(s).

These indicators can also be used to control processing in Calculation and Output-Format specifications.

If H1-H9 has been turned on during the processing of a record, execution of the object program stops after that record is processed. However, processing will not be interrupted if a halt indicator that has been turned on is turned off by another specification before the program attempts to process the next input record.

How Field Indicators are Turned On and Off

Field indicators entered in Plus, Minus or Zero or Blank are turned on if the value in the tested input field is positive or negative, respectively. They are turned off if the associated input field of the next record of the same type contains a negative or positive value, respectively.

Each field indicator is related to one record type. Therefore, a field indicator cannot be turned off until the associated record type is read again or until it is defined in some other specification. One or more field indicators can be on at the same time.

Zero or Blank indicators are initially turned off by the program and are not turned on if the associated field is blanked-after on Output-Format Specifications.

Figure 60 illustrates the setting of field indicators. Indicators 21-25 will be set on/off each time a record with a 1 in position one is read from the file DISKIN. If the next record read from the same file has a 2 or 3 in position one, the status of indicators 21-25 will remain unchanged. They will not be changed until another record with a 1 in position one is read.

Only numeric fields can be tested for a plus or minus setting. The same indicator can be used to test for either of two conditions (line 050). Different indicators can be used to test all of the possible conditions (line 080). If FLDF contains a zero amount, only indicator 28 is turned on. Lines 100 and 110 show an improper coding technique. When a record with a 3 in position one is read, indicator 29 will be set to reflect the status of FLDG then reset to reflect the status of FLDH. Consequently, the status of FLDG is lost and indicator 29 reflects only the status of FLDH.

Sterling Sign Position (Columns 71-74)

Is the position in the record that contains the sign of the sterling field. If the sign is in the normal position, enter an S in column 74. Leave these columns blank if the field is not a sterling field. For further information, refer to Appendix B.

IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page 10

Program Identification: REPORT

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	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
010	I	DISKIN	AA			01	1		C	1																					
020	I																				1	50	FLDA					21			
030	I																				6	80	FLDB						22		
040	I																				9	11	FLDC							23	
050	I																				12	140	FLDD					24		24	
060	I																				15	20	FLDE							25	
070	I		BB			02	1		C	2																					
080	I																				1	50	FLDF					26	27	28	
090	I		CC			03	1		C	3																					
100	I																				1	5	FLDG							29	
110	I																				6	9	FLDH							29	
120	I	*INDICATOR 29 IS SET ERRONEOUSLY																													
130	I																														

Figure 60. Setting Field Indicators

This form specifies the operations to be performed by the object program upon the input data and upon data obtained as a result of other calculations. Four rules govern the writing of calculation specifications:

1. Each operation is specified on one line of the form.
2. Detail calculations must precede all total calculations.
3. Calculations are specified in the order they are to be performed.
4. Subroutines must follow all other calculations, but need not be specified in the order in which they are used.

The Calculation Specifications form is divided into three parts as illustrated in Figure 61. These parts are:

- *Conditioning fields,* which determine under what conditions the calculations are to be performed.
- *Calculating fields,* which define the kind of calculation to be performed, the data to be used in the calculation, and the result field of the calculation.
- *Testing fields,* which test result fields and set indicators that can be used to condition subsequent calculation or output-format specifications.

CONDITIONING FIELDS (COLUMNS 7-17)

Entries in columns 7-17 determine the conditions under which calculations are to be performed. These conditioning fields are:

- Control Level (7-8)
- Indicators (9-17)

Control Level (Columns 7-8)

Indicates that calculations are to be performed at total time and determines the control level of the calculation specified in columns 18-59.

Possible entries are:

- L1 through L9, as defined for Control Level (columns 59-60) of the Input Specifications form.
- LR, which is the last record indicator.
- L0, which is the level zero indicator.
- SR, which identifies the line as a specification with a subroutine.

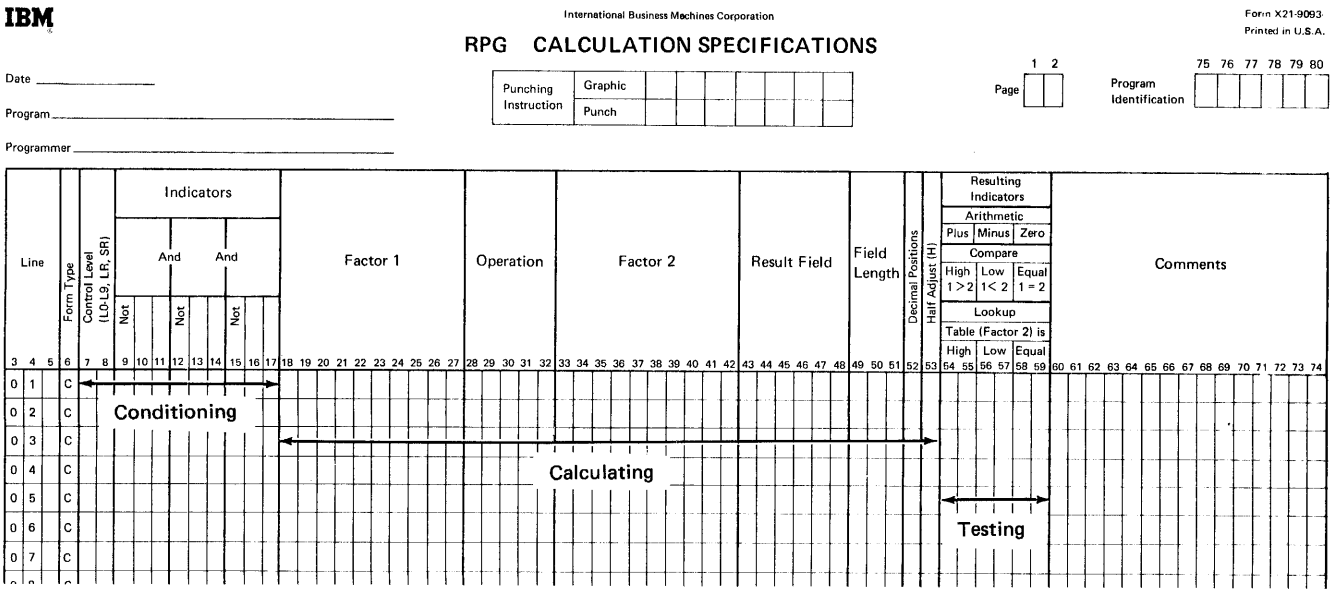


Figure 61. Calculation Specifications Form

A control-level indicator is turned on by a control break. It is on during total time and remains on for the following detail time, which includes both calculating and printing the detail record.

A control break for a specific control level forces all lower control-level indicators to be turned on. For example, an L3 break causes L3, L2, and L1 indicators to be turned on.

When an input record has been read, a test is made to determine if a control break has occurred. Total calculations are performed when this test is completed and before the record that caused the control break is processed.

Total calculations and total output lines are processed at the beginning of a program run as follows:

1. If no control fields are specified, total calculations and total output lines are bypassed for the first record read.
2. If control fields are specified, total calculations and total output lines are bypassed for all records read until the first record that contains control-field information has been processed. This applies also to calculations specified with an L0 indicator.
3. The control fields are initially set to hexadecimal zero. If the first three records are not tested for a control field change and the fourth is, all total calculations will be tested on the fifth cycle. Therefore, if the first record has a non-hexadecimal zero-control field, it causes a control break. Normal keypunched characters will cause a control break.

LR (Last Record) Indicator: LR is turned on after the last input record has been read and calculated, and after the specified detail output records have been processed. At this time, the control-level indicators L1 through L9 are also turned on and final totals can be performed.

L0 (Level Zero) Indicator: L0 is turned on at the beginning of a job and before a record is read. It remains on throughout processing by the object program. It is used to accumulate totals when no control break has occurred.

SR (Subroutine Identification): SR identifies this line as a specification with a subroutine (see *Subroutines).* Control levels are not used in subroutines. However, a total calculation may call a subroutine.

Turning Indicators On and Off: Each of the indicators L1 through L9 and LR may be turned on or off individually by means of the SETON and SETOF operations. These operations affect only the specified indicator. Lower level indicators remain as they were at the time the SETON or SETOF operation is invoked. For example, if L3 is turned on, L2 and L1 are not automatically turned on. The SETON and SETOF operations are described in detail later in this section.

Using Control Level Specifications

Figure 62 illustrates seven control-break specifications.

Indicators (Columns 9-17)

Define the conditions (if any) that control the calculations specified in columns 18-59. One, two, or three indicator codes can be specified. If the calculation specified in columns 18-59 of a line is not governed by an indicator condition, leave columns 9-17 blank.

If two or three indicators are specified in columns 9-17 on one line, these indicators are assumed to be in an AND relationship. That is, if three indicator codes are specified, all three indicator conditions must be satisfied before the calculation is performed. Enter in columns 10-11, 13-14, and 16-17 the appropriate indicators that must be checked before the associated calculation is performed. If an operation should only be performed when an indicator is off, enter an N in either column 9, 12, or 15 (whichever is appropriate).

IBM International E

RPG CALCULA

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic
	Punch

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And	Not	And	Not	And	Not	Factor 1	Operation
3 4 5	6	7 8	9 10 11	12	13 14	15	16 17		18 19 20 21 22 23 24 25 26 27	28 29 30 31 32
0 1 Ø	C	L Ø	98						FIELDA	ADD
0 2 Ø	C	L 1							DEPTOT	SUB
0 3 Ø	C	L 1							FIELDB	ADD
0 4 Ø	C	L 2							FIELDC	ADD
0 5 Ø	C	L 2							FIELDN	ADD
0 6 Ø	C	L 2							FIELDR	MULT
0 7 Ø	C	L R							TOTAL	MULT

Figure 62. Control Level in Calculation Specifications

The following considerations apply to the use of indicators in columns 9-17:

1. If columns 9-17 and columns 7-8 are blank, the calculation is performed each time a detail record is read.
2. If a record-identifying indicator that has been defined in columns 19-20 of the Input Specification form is placed in Indicators, the calculation is performed only if a record of the type associated with the record-identifying indicator is read.
3. A field-indicator code controls the calculation according to the status of the associated input data field.
4. A resulting indicator specified in columns 54-59 of a preceding calculation line controls the current calculation according to the condition that occurred in the result field of the preceding calculation.
5. A control-level indicator (L1-L9) used with a record identifying indicator defined on the Input Specifications form permits the calculation to be performed at detail time, but only on the first record of the control group specified. The specification of L0 is not allowed in columns 9-17.
6. If the matching-record indicator (MR) is placed in Indicators, the calculation is performed only if there is a matching record in a secondary input file.
7. Halt indicators H1 through H9 are normally used to suppress a calculation when an error has been detected in the input data or during a previous calculation.
8. Overflow indicators OF and OV permit the calculation to be performed only if a page overflow has occurred.
9. If columns 9-17 and columns 7-8 of the specification line contain entries identical to the preceding line, no repetitive test is made. The core storage requirements for testing indicators are also dropped. Consequently, the programmer should group his calculations by identical conditioning indicators to reduce core and improve throughput.

Columns 9-17 may contain a combination of the type of indicators discussed in the preceding text.

Figure 63 shows the use of indicators in columns 9-17 of the calculation lines. The numbers to the right of the entries refer to the item numbers in the text below.

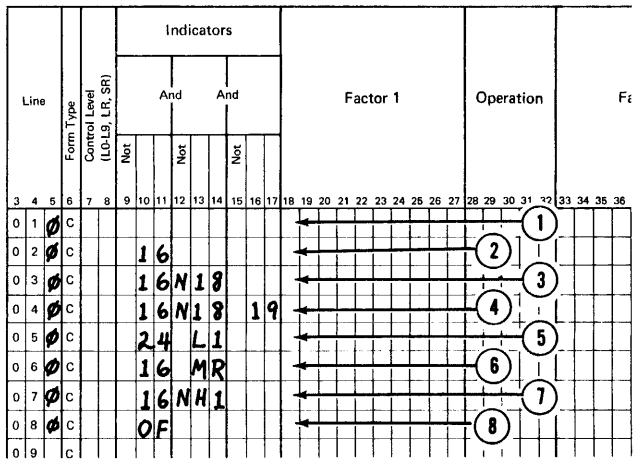


Figure 63. Indicators in Calculation Specifications

1. Columns 7-17 are blank, indicating that the calculation is to be performed on each detail record read.
2. The associated calculation is performed only if indicator 16 is on.
3. Assume that indicator 16 is a record-identifying indicator defined in columns 19-20 of the Input Specifications form, and indicator 18 is a field indicator defined in zero or Blank on the Input Specifications form. The associated calculation is performed only if record-type 16 is present and the field assigned to indicator 18 is not blank.
4. The calculation is performed only if indicators 16 and 19 are on and indicator 18 is off.
5. The calculation is performed at detail time whenever control-level indicator L1 and indicator 24 are on for the first card of a new group.
6. The calculation is performed only if indicator 16 is on and there is a matching record condition. (For example, when fields from detail records are to be multiplied by a factor contained in a master record, the program must have a means of ensuring that the detail record has been matched with the appropriate master record.)
7. The calculation is not done if an error condition has occurred that resulted in H1 being turned on. This prevents the calculation of erroneous data.
8. The calculation is performed during the detail cycle following page overflow and printing of the overflow lines.

Calculations in an AND/OR Relationship

AND/OR relationship is specified differently on calculation specifications than on input and output specifications. An OR condition is defined by means of separate entries. An AND condition is defined by setting an intermediate intermediate indicator. Figure 64 shows an example of how this can be done when four indicators (11, 12, 13, 14) must all be on before an addition is made. Indicator 21 must be turned off each time or the addition will not be conditioned correctly.

CALCULATING FIELDS

Entries in columns 18-53 define:

- The data fields to be used (Factors 1 and 2)
- The type of operation to be performed (operation), and
- The result field (columns 43-53).

General Description of Data Fields

1. Field Name is to be used in calculations and must be defined in either columns 53-58 (field name) of the Input Specifications form, or columns 43-48 (result field) of the Calculations Specifications form. The first character of a field name must be alphabetic. The length must not exceed six characters. Special characters and embedded blanks must not be used.
2. Literals are the actual data to be operated on, rather than a name representing the location of data in storage. Literals may be numeric or alphameric and must be left-justified. Alphameric literals must be enclosed in apostrophes.
3. A Label is the name of a subroutine BEGIN or END statement, or a TAG statement. The first character must be alphabetic. The length must not exceed six characters. Special characters and embedded blanks must not be used. A label cannot have the same name as a field.

Numeric Literals

A numeric literal consists of any combination of the digits 0 through 9. One decimal symbol and/or one plus or minus sign may be used. Other separators (e.g., a comma between the thousands and hundreds positions) may not be used. The European format of commas to denote the decimal point may be used if the inverted print option is selected in the RPG control card.

Rules for Forming Numeric Literals

1. A numeric literal may be ten characters or less.
2. Blanks must not appear within a numeric literal.
3. The sign, if present, must be the left-most character. An unsigned literal is considered positive.
4. The decimal point may appear anywhere in the literal.
5. Numeric literals must not be enclosed in apostrophes.

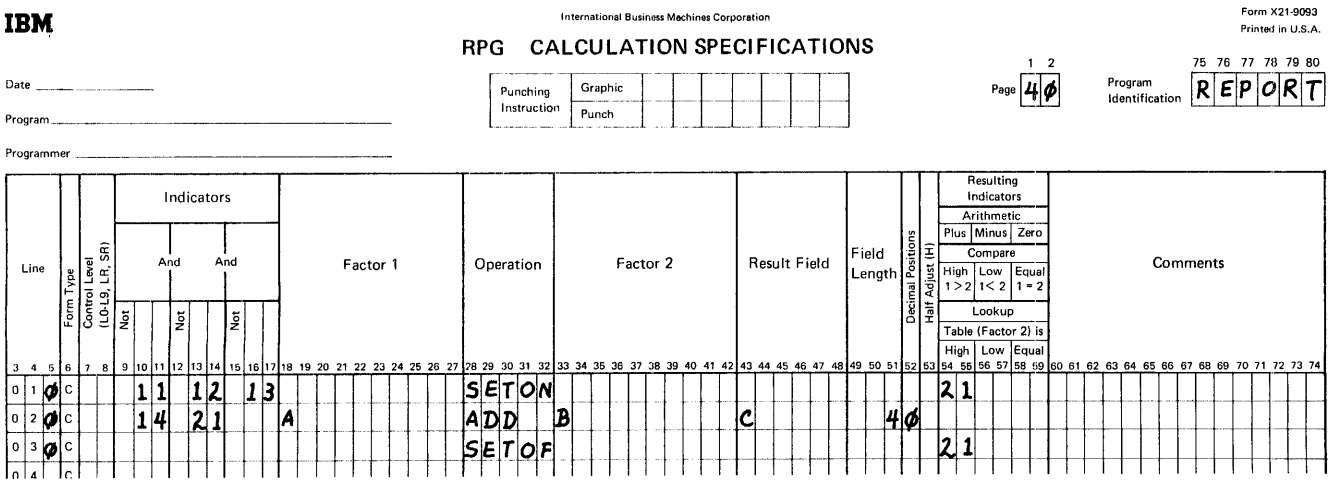


Figure 64. Calculations Specifying an AND Condition

Alphameric Literals

An alphameric literal consists of characters enclosed in apostrophes. Alphameric literals may be used for compare, move, and table-lookup operations, but must not be used in arithmetic operations.

Rules for Forming Alphameric Literals

1. The maximum length of alphameric literals is eight characters, excluding the two enclosing apostrophes.
2. Alphameric literals must be enclosed in apostrophes.
3. Any characters of the EBCDIC character set may be used in an alphameric literal. Blanks in the body of the literal are treated as valid characters.
4. An apostrophe may be included in a literal by entering two consecutive apostrophes. For example, the literal o'clock would be coded as 'o'clock'.

Factor 1 (Columns 18-27) and Factor 2 (Columns 33-42)

Factors 1 and 2 may be field names, labels, or literals, depending upon the operations being performed. They must be left-justified.

Figure 65 illustrates Factor 1 entries. The numbers in the right-hand margin refer to the item numbers in the following text. Factor 2 entries are specified in the same manner.

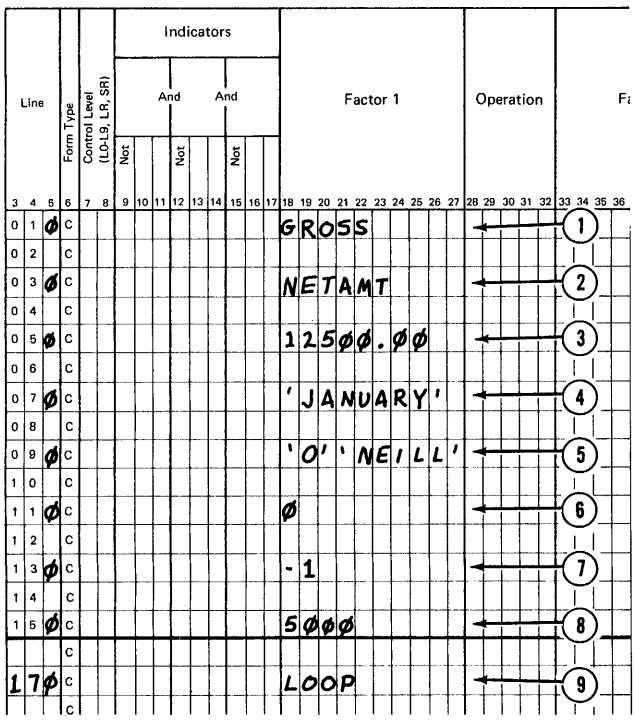


Figure 65. Factor 1 on Calculation Specifications Form

1. GROSS is the field name that may have been defined on the Input Specifications form.
2. NETAMT may be the name of a result field of another calculation, or of an input field defined on the Input Specifications form.
3. This numeric literal could be used to determine if the contents of a specific input field are higher or lower than this quantity. The position of the decimal point is indicated. No other punctuation is permitted.
4. This alphameric literal can be compared against the contents of a data field, or moved into a data field, or used in a table-lookup operation.
5. This shows an apostrophe within an alphameric literal. The literal would be printed as O'NEILL.
6. A numeric literal of 0 is created. This is normally used in a Z-ADD instruction to set a field to zero.
7. A numeric literal of minus 1 is created. This could be used in arithmetic or compare operations.
8. A numeric literal of 5000 is created. An implied decimal position of zero will be used since no decimal point is included.
9. A label name of LOOP is specified. This will be used with either a TAG, BEGSR, or ENDSR statement.

Operation (Columns 28-32)

Specifies the operation to be performed using the entries in Factor 1, Factor 2, and Result Field. The entry consists of the appropriate operation code beginning in column 28. For detailed information, refer to the section *Entries in the Operation Field.*

Result Field (Columns 43-48)

Entering a field name in Result Field causes the specified number of core storage positions to be reserved for calculation results. The name must be alphameric and left-justified. It must not exceed six characters. It must not contain embedded blanks, or special characters. The first character must be alphabetic.

The same name can be used several times in different calculations if the length of the field and the number of decimal positions are the same for all calculations.

Figure 66 illustrates result-field entries. On line 010, GROSS is multiplied by DRATE, and the result is placed in DISCNT. The contents of DISCNT are then used as Factor 2 on line 020 to calculate the net amount which is placed in NETAMT. The same field DISCNT is then used in Factor 1 on line 030 to calculate total discount, which is placed in TOTDIS.

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic							
	Punch							

Page 4Ø

Program Identification REPORT

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And	Indicators: And	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus	Resulting Indicators: Arithmetic Minus	Resulting Indicators: Arithmetic Zero	Resulting Indicators: Compare High 1>2	Resulting Indicators: Compare Low 1<2	Resulting Indicators: Compare Equal 1=2	Resulting Indicators: Lookup Table (Factor 2) is High	Resulting Indicators: Lookup Table (Factor 2) is Low	Resulting Indicators: Lookup Table (Factor 2) is Equal	Comments
01	Ø	C			GROSS	MULT	DRATE	DISCNT	82		H										
02	Ø	C			GROSS	SUB	DISCNT	NETAMT	82												
03	Ø	C			DISCNT	ADD	TOTDIS	TOTDIS	82												
04		C																			

Figure 66. Result Field Entries

Field Length (Columns 49-51)

Specifies the length of the result field; that is, the number of positions to be reserved for the field specified in Result Field. The unpacked length must be specified. The maximum length of numeric fields is 14 digits. The maximum length of alphameric fields is 256 characters.

If the length of an arithmetic result exceeds the specified length of the associated result field, the leftmost digits are truncated. Any resulting indicators are set according to the truncated field.

If the same result field name is used in more than one calculation, the field-length specification and the decimal position specification need only be specified once. If the length and the number of decimal positions in a result field have been established on the Input Specifications form, entries in Field Length and Decimal Position on the Calculation Specifications form are not required. However, if such entries are made, they must not contradict the related entries on the Input Specifications form.

If half-adjustment (rounding) is specified, the entries in Field Length and Decimal Position refer to the length of the result field after half-adjustment.

Decimal Position (Column 52)

An entry (0-9) in Decimal Posiiton identifies the associated result field as numeric and specifies the number of positions to the right of the decimal point in the result field. An entry must be made in this column for all arithmetic operations if the field is not specified elsewhere. If no decimal positions are to be retained in the result field, enter 0. Otherwise, enter one of the digits 1-9, as required. In Figure 66 each result field is specified to have two decimal positions.

Note: The number of decimal positions must not exceed the specified length of the result field.

If the result field is alphameric, leave this column blank.

Half-Adjust (Column 53)

An H in Half-Adjust specifies half-adjusting (or rounding) of the result.

If the number of decimal positions of the arithmetic result is less than or equal to the specified decimal positions of the result field, no half-adjusting is possible. If the arithmetic result has more decimal positions than the specified result field, an H in Half-Adjust causes an arithmetic result to be truncated (by dropping the rightmost digits) until the result is one digit longer than the specified result field. The absolute value of the low-order digit of the (truncated) result is added to the absolute value of the (truncated) result. The rightmost digit is then dropped, the original sign (plus or minus) is restored, and the adjusted result is moved into the specified result field. Because of this, rounding always occurs away from zero for both positive and negative fields.

Using Figure 66, assume the following values in line 010:

```
GROSS =  2,145.07
DRATE =        .09

          2,145.07
            x .09
          --------
          193.0563
            +  5
          --------
          193.0613

DISCNT=  193.06
```

TESTING FIELDS

Entries in columns 54-59 test the result of a calculation and set indicators that can be used to condition subsequent calculation or output-format specifications.

Resulting Indicators (Columns 54-59)

Are indicator codes used to test the value of a result field after the completion of an operation. The indicators entered in columns 54-59 are turned on or off according to the result of this test. They can be used to control subsequent calculation or output operations.

Any indicators can be specified except L0, 1P, and MR. The indicators can be defined one or more times on the form. If they are defined more than once, a subsequent specification of the indicator resets it from the status it may have had from a previous specification.

The resulting indicators specification can be used to:

1. Determine whether the result of an arithmetic operation is plus, minus, or zero. (In the case of half-adjustment, the resulting indicator refers to the result after half-adjustment.)
2. Test the result of a compare operation to determine if:
   Factor 1 is greater than Factor 2 – High
   Factor 1 is less than Factor 2 – Low
   Factor 1 equals Factor 2 – Equal
3. Define the type of LOKUP operation that determines:

   - If the table argument next-higher than the search argument is found.
   - If the table argument next-lower than the search argument is found.
   - If the table argument equal to the search argument is found.

   The table argument is always Factor 2.

   To place the indicator in the correct column, the programmer should think in terms of the table (Factor 2) rather than the search argument (Factor 1).

   An equal-search resulting indicator takes precedence over either high or low indicators if an equal-table value exists.
4. Define a TESTZ operation as to what type of zone is to be tested. (See the definition under TESTZ).
5. Define SETOF and SETON operations as to what indicators are to be turned off or on.
6. Determine the results of a CHAIN operation. (See *CHAIN Operations).*

Resulting indicators are not reset (turned on or off) until the next time a calculation is performed for which that indicator is specified. This means that one or more resulting indicators can be on at one time.

If the field is also a result field being tested for plus or minus, the resulting field indicators specified are not turned off when the field is set to zeros by the *Blank After* specification. Nor is the zero or Blank indicator turned on by a *Blank After* specification.

Figure 67 summarizes conditions that cause resulting indicators to be turned on.

Figure 68 shows the use of resulting indicators. On line 010, DISCNT is subtracted from GROSS and the result is stored in NETAMT. If the answer is negative, indicator 10 is set on. If the answer is zero, indicator 15 is set on.

On line 020, the literal JANUARY is compared against the contents of DATE. If the result is equal, indicator 24 is turned on.

Lines 030-100 compare 1000.00 against the contents of AMT and turn various indicators on as a result of the comparison. For example, line 030 turns on indicator 21 if 1000.00 is greater than AMT, line 040 turns on indicator 22 if 1000.00 is less than AMT, and line 060 turns on indicator 24 if 1000.00 does *not* equal AMT.

Lines 030 to 100 show the same calculation with various indicator tests. Lines 030-050 show a test for only one condition. Line 060-080 shows that one indicator will be set if either of the conditions specified is met. (The test

Condition		Columns 54-55	Columns 56-57	Columns 58-59	Remarks
Arithmetic Operation	If result field is	Plus	Minus	Zero	The same indicator may be used in any two columns, but not in all three. If different indicators are used and the result is zero, only the zero indicator will be on.
Compare Operation	If Factor 1 is	Greater than Factor 2	Less than Factor 2	Equal to Factor 2	The same indicator may be used in any two columns, but not in all three. If the same indicator is used in any two columns, the indicator will be set on if either condition exists.
Table Lookup	If table argument in Factor 2 is	Greater than Search Argument in Factor 1	Less than Search Argument in Factor 1	Equal to Search Argument in Factor 1	The same indicator may be used in High-Equal, or Low-Equal, but not in High-Low. If different indicators are used, the equal indicator takes precedence if an equal condition is found.
CHAIN Operation	If no record is found	Specified Indicator	Specified Indicator	Not Used.	The same indicator must be used in both entries. If only one is specified or different indicators are specified, the indicator in columns 54-55 is used. Any indicators specified in columns 58-59 are ignored.
SETON/SETOF Operation	May be one, two, or three indicators	Specified Indicator	Specified Indicator	Specified Indicator	
Test Zone Operation	If high-order zone, zone is	Note 1	Note 2	Note 3	

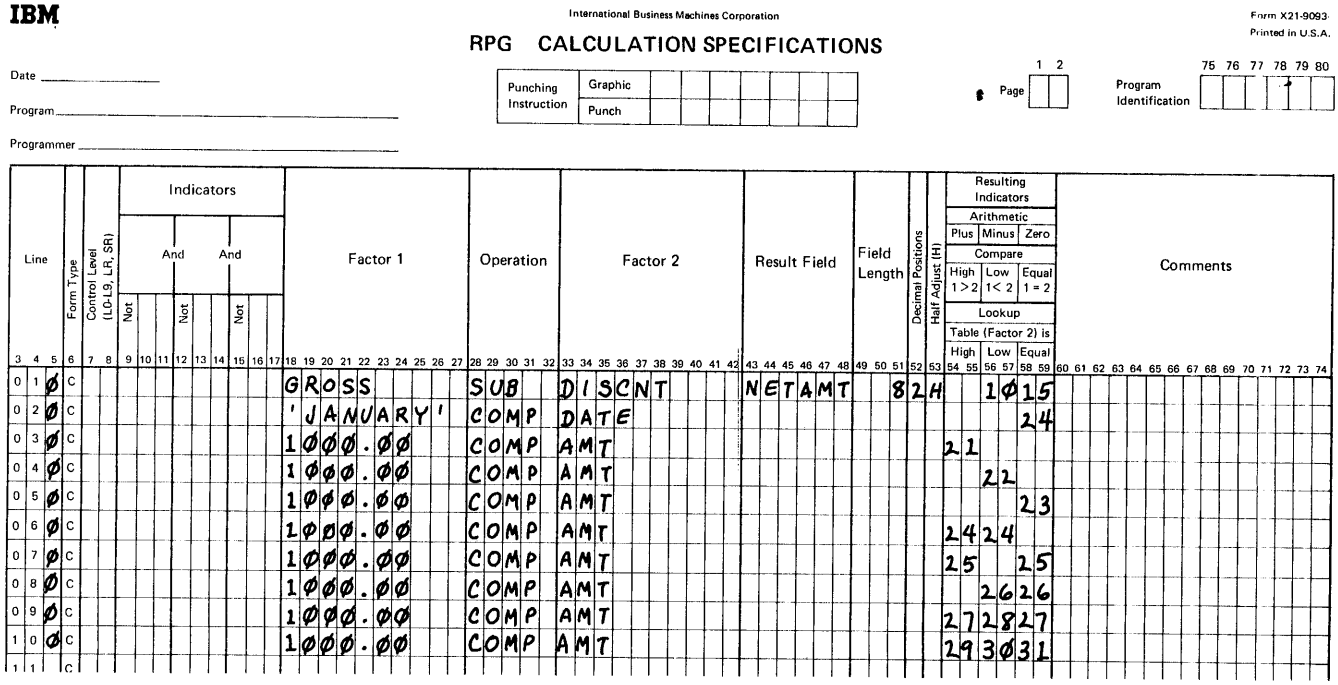
Note 1: The indicator in columns 54-55 is normally used to test for a 12-punch (letters A through I and &). It can be used to test for any hexadecimal character having the same zone as an A.
Note 2: The indicator in columns 56-57 is normally used to test for an 11-punch (letters J through R and -). It can be used to test for any hexadecimal character having the same zone as a J.
Note 3: If the zone is made up of any characters other than those mentioned in notes 1 & 2, the indicator in columns 58-59 is on.

Figure 67. Conditions That Turn Resulting Indicators On

is made for either condition. If the first condition is met, the second condition will not turn it off.) Line 090 shows indicator 27 being set for the high or equal condition and indicator 28 for the low condition. Line 100 shows an individual indicator being used for each condition.

Comments (Columns 60-74)

Is optional explanatory information supplied by the user. This information is merely printed on the source program listing and is not interpreted by the RPG compiler.



IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And / Not / And / Not / 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
010	C			GROSS	SUB	DISCNT	NETAMT	8	2	H	1015	
020	C			'JANUARY'	COMP	DATE					24	
030	C			1000.00	COMP	AMT					21	
040	C			1000.00	COMP	AMT					22	
050	C			1000.00	COMP	AMT					23	
060	C			1000.00	COMP	AMT					2424	
070	C			1000.00	COMP	AMT					25 25	
080	C			1000.00	COMP	AMT					2626	
090	C			1000.00	COMP	AMT					272827	
100	C			1000.00	COMP	AMT					293031	
110	C											

Figure 68. Using Resulting Indicators

As mentioned previously, codes for RPG operations are placed in columns 28-32 of the Calculation Specifications form. The operations are grouped according to type as follows:

- Arithmetic
- Move
- Compare and Test
- Indicator Setting
- Table
- Chain
- Branch and Exit

The operations are described in detail in the paragraphs that follow. They are summarized in Figure 69.

ARITHMETIC OPERATIONS

1130 RPG provides the following arithmetic operations:

- Add (ADD)
- Zero and ADD (Z-ADD)
- Subtract (SUB)
- Zero and Subtract (Z-SUB)
- Multiply (MULT)
- Divide (DIV)
- Move Remainder (MVR)

After an arithmetic operation is performed, the result is moved to the result field specified in columns 43-48. Truncation is done as required.

All airthmetic operations in RPG are done with decimal alignment. Decimal alignment means that if one field has fewer decimal positions, enough decimal positions are added to make the fields the same length. After the operation is performed, the result is moved to the result field specified. If truncation is required, it is done at this time.

Fields or literals involved in arithmetic operations must be numeric. Two of the fields involved in an arithmetic operation may be specified as the same field. For example, the literal 1 might be added to the field COUNT, and the resulting sum could replace the value previously in COUNT. In this case, COUNT is specified in both Factor 1 and Result.

RPG cannot detect an arithmetic overflow. Therefore, any field involved in arithmetic operations must not exceed 14 digits. This includes consideration for decimal alignment and half-adjusting. Resulting indicators (columns 54-59) may be used with all arithmetic operations.

RPG cannot detect an invalid numeric field. For example, the digit portion of the EBCDIC code for an asterisk is hexadecimal C. If an asterisk is mistakenly punched into a card, the value for a hexadecimal C is added to a valid digit, and an erroneous result may be produced.

Add (ADD)

Algebraically adds the contents of the field or the literal in Factor 2 to the literal or the contents of the field in Factor 1. The result is then placed in the result field.

Zero and Add (Z-ADD)

Sets the result field to zeros and adds algebraically the literal or the contents of the field in Factor 2 to that result field. Factor 1 is not used.

Subtract (SUB)

Algebraically subtracts the literal or the contents of the field in Factor 2 from the literal or the contents of the field in Factor 1. The result is placed in the result field.

Operation Codes Operation	Control Level	Conditioning Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust	Resulting Indicators
DECIMAL										
ADD	O	O	N	ADD	N	N	O	O	O	O
ZERO AND ADD	O	O		Z-ADD	N	N	O	O	O	O
SUBTRACT	O	O	N	SUB	N	N	O	O	O	O
ZERO AND SUBTRACT	O	O		Z-SUB	N	N	O	O	O	O
MULTIPLY	O	O	N	MULT	N	N	O	O	O	O
DIVIDE	O	O	N	DIV	N	N	O	O	O	O
MOVE REMAINDER	O	O		MVR		N	O	O		O
MOVES										
MOVE	O	O		MOVE	E	E	O	O		
MOVE LEFT	O	O		MOVEL	E	E	O	O		
MOVE HIGH-TO-LOW ZONE	O	O		MHLZO	A	E	O	O		
MOVE LOW-TO-HIGH ZONE	O	O		MLHZO	E	A	O			
MOVE HIGH-TO-HIGH ZONE	O	O		MHHZO	A	A	O			
MOVE LOW-TO-LOW ZONE	O	O		MLLZO	E	E	O	O		
LOGICAL										
COMPARE	O	O	E	COMP	S					R
TEST ZONE	O	O		TESTZ		A	O			R
TABLE LOOKUP	O	O	E	LOKUP	S	O	O	O		R
SET INDICATORS										
SET INDICATORS ON	O	O		SETON						R
SET INDICATORS OFF	O	O		SETOF						R
FILE										
CHAIN TO A FILE RANDOMLY	O	O	E	CHAIN	I					O
BRANCH TO EXCEPTION OUTPUT LINE	O	O		EXCPT						
BRANCHING										
BRANCHING OR GOTO	O	O		GOTO	L					
PROVIDING LABEL FOR GOTO	O		L	TAG						
BRANCH TO CLOSED SUBROUT	O	O		EXSR	L					
BEGIN CLOSED SUBROUT	SR		L	BEGSR						
END CLOSED SUBROUT	SR		OL	ENDSR						
EXIT TO A SUBROUT	O	O		EXIT	L					
RPG LABEL	O			RLABL		E	O	O		

KEY TO OPERATION CODE CHART

IF THE ENTRY IS BLANK IT CANNOT BE FILLED IN

E = REQUIRED FIELD MUST BE EITHER ALPHAMERIC OR NUMERIC
A = REQUIRED FIELD MUST BE ALPHAMERIC
I = REQUIRED FILE NAME
L = REQUIRED LABEL NAME
N = REQUIRED FIELD MUST BE NUMERIC
O = OPTIONAL ENTRY
OL= OPTIONAL LABEL ENTRY
R = REQUIRED ENTRY
S = REQUIRED FIELD MUST BE OF THE SAME TYPE (ALPHAMERIC OR NUMERIC) AS FACTOR 1
SR= REQUIRED ENTRY IN CONTROL LEVEL COLUMNS

Figure 69. Summary of RPG Operation Codes

Zero and Subtract (Z-SUB)

Sets result field to zero and causes the negative of the literal or the negative of the contents of the field in Factor 2 to be placed in that result field. Factor 1 is not used in this oepration. This operation is normally used to change the sign of a field.

Multiply (MULT)

Algebraically multiplies the literal or the contents of the field in Factor 1 by the literal or the contents of the field in Factor 2. The result is placed in the result field.

Only the number of decimal positions specified in column 52 is retained. Any excess low-order (rightmost) decimal

positions are dropped. If the arithmetic result is longer than 14 digits, the excess digits must be zeros or erroneous results will be achieved. If the result is less than 14 digits, but greater than the result field, the high-order digits are truncated. The remaining digits of the arithmetic result are placed in the result field.

Divide (DIV)

Algebraically divides the literal or the contents of the field in Factor 1 by the literal or the contents of the field in Factor 2. The result is placed in the result field. The literal or the contents of the field in Factor 2 must not be zero.

Any remainder from this operation is lost, unless the move-remainder (MVR) operation is specified as the next operation. If the MVR operation is used, half-adjusting (rounding) must not be specified for the divide operation.

The following formulas can be used to determine the largest number of positions available in any field. They can also be used to check in advance whether the fields are long enough to perform the division.

$$L_2 + D_1 - D_2 - D_r \leqslant 14; \text{ and}$$

$$L_1 - D_1 + D_2 + D_r \leqslant 14; \text{ and}$$

in the case of half-adjustment,

$$L_1 - D_1 + D_2 + D_r \leqslant 13,$$

where

L_1 = Length of Factor 1 (dividend).

D_1 = Number of decimal positions in Factor 1.

L_2 = Length of Factor 2 (divisor).

D_2 = Number of decimal positions in Factor 2.

D_r = Specified number of decimal positions in the result field (quotient).

The number of decimal positions in each of the three elements of a division should satisfy the following condition:

$$A = D_r - D_1 + D_2 = 0$$

If this is not the case, either the dividend or the divisor will be adjusted (padded) by adding zeros to the right, depending on the following conditions:

$A > 0$ implies padding of dividend
$A <$ implies padding of divisor

The number of zeros added is equal to the absolute value of A.

A divisor of zero causes an error wait. The operator may terminate the job or continue with a zero in the result field. If the programmer does not wish the operator to have this decision, the divisor should be tested for zero and appropriate steps taken. A dividend of zero always produces a zero result field.

Move Remainder (MVR)

Moves the remainder of a divide operation to a separate field, specified in result field, that has been set to zero by RPG. If MVR is used, it must immediately follow the divide operation.

The value of the remainder can be determined by the following formula:
R = Dividend - Divisor X Quotient

The result field of an MVR operation may be assigned resulting indicators in columns 54-59. Factor 1 and Factor 2 must not be used in an MVR specification. Decimal alignment is performed by the RPG program. Half-Adjust may not be specified.

The same conditions and indicators specified for a DIV operation must be specified for an associated MVR operation. The MVR line must contain all the indicators specified in the DIV line and may contain additional indicators. The MVR line must contain the same or a higher control level than specified in the DIV line. If the DIV operation is executed at detail time (columns 7-8 blank), the MVR operation must also be performed at detail time.

Figure 70 shows the use of the MVR operation. The remainder is placed in a field named STORE.

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic							
	Punch							

Page 05

Program Identification

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And	Indicators: And	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		12		FIELD1	DIV	FIELD2	SAVE		80											
02	C		12			MVR		STORE		40											

Figure 70. MVR Operation

Example of Divide and Move Remainder

Consider dividing the following:

11 ÷ .76 and requesting an answer in 2 decimals

```
      14.47
.76 ⌈11∧00.00
     7 6
     3 40
     3 04
       36 0
       30 4
        5 60
        5 32
          28 = Remainder
```

The number of decimal positions in the remainder can best be seen by multiplying the divisor times the quotient.

```
  14.47
x  .76
-------
10.9972
```

To get back to the value of the dividend, .0028 must be added.

```
  10.9972
   .0028
--------
  11.0000
```

Therefore, the remainder must have 4 decimal places.

The number of decimal places in the remainder is also the same number as in the adjusted dividend. This can be determined from the formula described in the Divide operation.

The MVR operation performs decimal alignment. Consequently, the result field could contain the following depending upon the field length and decimal positions described.

Field Length	Decimal Positions	Result Value
5	0	00000
4	1	000.0
3	2	0.00
4	3	0.002
4	4	.0028
5	5	.00280

Summary of Arithmetic Operations

Figure 71 illustrates some sample arithmetic operations. The actual result of each operation is shown in the Comments columns (60-74). The field values for these operations are A=100, B=32, C=20, D=-17, E=2.77, F=25, G=1, H=1.40, N=0, P=.00, Q=0, R=.0, and S=.000.

MOVE OPERATIONS

1130 RPG provides the following operations for moving data and zones:

- Move (MOVE)
- Move Left (MOVEL)
- Move Zone (MLHZO, MHLZO, MHHZO, or MLLZO)

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic								
	Punch								

Page 40

Program Identification TEST1

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 / Minus / Zero); Compare (High 1>2 / Low 1<2 / Equal 1=2); Lookup (Table (Factor 2) is High / Low / Equal)	Comments
01 05	C			A	ADD B		N	5	0			132
02 05	C			A	ADD C		C					120
03 05	C			A	ADD D		N					83
04 05	C			A	ADD E		N					102
05 05	C				Z-ADD A		N					100
06 05	C				Z-ADD A		P	7	2			100.00
07 05	C				Z-ADD 0		P					.00
08 05	C			A	SUB B		N					68
09 05	C			F	SUB A		F					-75
10 05	C			A	SUB E		N					97
11 05	C				Z-SUB A		N					-100
12 05	C				Z-SUB D		N					17
13 05	C			A	MULT F		Q	8	0			2500
14 05	C			A	MULT E		R	8	1			2.7
15 05	C			G	MULT E		R			H		2.8
16 05	C			A	DIV C		N					5
17 05	C			A	DIV H		P			H		71.43
18 05	C			A	DIV H		P					71.42
19 05	C				MVR		S	3	3			.012
	C											

Figure 71. Summary of Arithmetic Operations

MOVE and MOVEL instruct the RPG program to move data. The literal or the name of the field containing the data to be moved is placed in Factor 2. The name of the field to which the data is to be moved is placed in Result Field. If the result field is not defined elsewhere in the program, its length must be specified in columns 49-51. If the result field is numeric, an entry must be specified in column 52. Factor 1, Half-Adjust, and Resulting Indicators may not be used.

MLHZO, MHLZO, MHHZO, and MLLZO are four operations that instruct the RPG program to move zones.

Move (MOVE)

Moves the literal or the contents of the field in Factor 2 to the result field. The field or literal in Factor 2 may be alphameric or numeric.

If Factor 2 is shorter than the result field, Factor 2 is moved to the rightmost positions of the result field as illustrated in Figure 72. The excess lefthand positions of the result field are not disturbed.

If Factor 2 is longer than the result field, the excess leftmost positions of Factor 2 are not moved, as illustrated in Figure 72.

The MOVE operation can be performed on both numeric and alphameric fields. In addition, an alphameric field can be moved into a numeric field or vice versa.

Resulting indicators must not be used in MOVE specifications. Decimal alignment or sign checking is not performed.

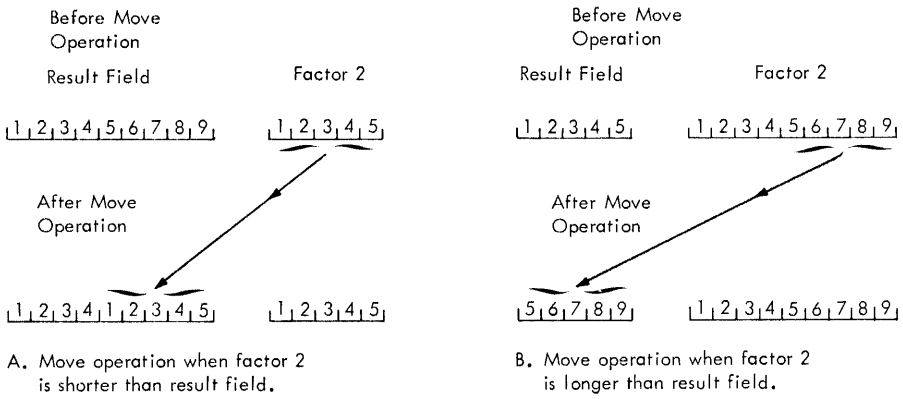


Figure 72. Move Operation

Move Left (MOVEL)

Causes the literal or the contents of the field specified in Factor 2 – beginning with the leftmost character – to be moved left-justified into the result field. Alphameric data can be moved into numeric fields, and vice versa. Decimal alignment is not performed for this operation.

If the result field is longer than Factor 2, the excess right-hand positions of the result field remain undisturbed. If the result field is shorter than Factor 2, the excess right-hand positions of Factor 2 are not moved.

If Factor 2 is shorter than the (numeric) result field, the sign of Factor 2 is not moved. If Factor 2 is equal to or longer than the (numeric) result field, the sign of Factor 2 is moved into the rightmost position of the result field.

If a numeric field is moved into an alphameric result field that is equal to or longer than Factor 2, the sign of the numeric field is moved into the position of the result field that contains the rightmost position of Factor 2. If the alphameric result field is shorter than the numeric Factor 2 field, the sign is not moved.

Three examples in Figure 73 illustrate the function of the MOVEL operation.

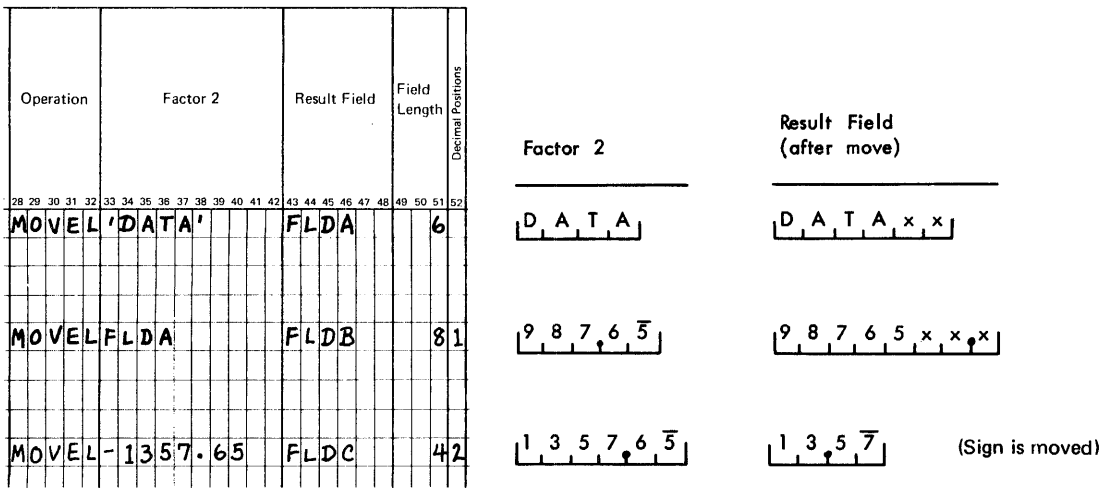


Figure 73. Functions of MOVEL Operation

Move Zone

This operation has four variations. In each case, the zone portion of the specified position (L: Low-order, H: High-order) in the field in Factor 2 is moved to the specified position (L: Low-order, H: High-order) of the result field.

Factor 1 is not used in this operation. The field which is specified to use the high zone (H) can only be an alphameric field.

Move Low-to-High Zone (MLHZO)

Moves the zone in the rightmost position of the field in Factor 2 to the left most zone position of the result field. Factor 2 can be numeric or alphameric, but the result field must be alphameric.

Move High-to-Low Zone (MHLZO)

Moves the zone in the leftmost position of the field in Factor 2 to the rightmost zone position of the result field. The field in Factor 2 must be alphameric. If the field in Factor 2 contains only numeric data, this operation can still be performed, provided that the field is defined as an alphameric field. However, it may contain either numeric or alphameric data. The result field may be numeric or alphameric.

Move High-to High Zone (MHHZO)

Moves the zone in the leftmost position of the field in Factor 2 into the leftmost zone position of the result field.

Factor 2 and the result field must be defined as alphameric.

Move Low-to-Low Zone (MLLZO)

Moves the zone in the rightmost position of the field in Factor 2 into the rightmost zone position of the result field. Factor 2 and the result field may be alphameric or numeric. A result field specified as numeric always contains the correct sign after this operation.

Figure 74 illustrates the functions of the move-zone operations. The zones are moved from Factor 2 to the result field.

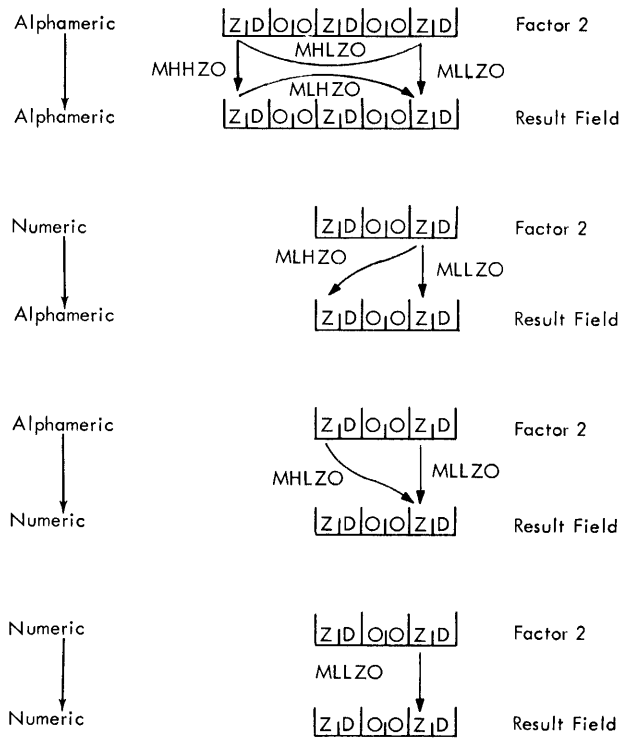


Figure 74. Function of Move Zone Operations

COMPARE AND TEST OPERATIONS

1130 RPG provides the following compare and test operations:

- Compare (COMP)
- Test Zone (TESTZ)

Compare (COMP)

Compares the literal or the contents of the field in Factor 1 against the literal or the contents of the field in Factor 2.

COMP turns on the appropriate indicator specified in columns 54-59. These resulting indicators can be used to condition calculation and/or output operations.

If the literal or the data in the field in Factor 1 is greater than the literal or the data in the field in Factor 2 (F1>F2), the indicator specified in High (columns 54-55) is turned on.

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic								
	Punch								

Page 40

Program Identification REPORT

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And	Indicators: And	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
010	C				FACT1	ADD	FACT2	FACT2	82										21		
020	C		21			Z-SUB	FACT2	CFLD	82												
030	C		N21			Z-ADD	FACT2	CFLD													
040	C				CFLD	COMP	10000.0											31	32	33	

Figure 75. Absolute Compare Routine

If the literal or the data in the field in Factor 1 is smaller than the literal or the data in the field in Factor 2 (F1<F2), the indicator specified in Low (columns 56-57) is turned on.

If the literal or the data in the field in Factor 1 equals the literal or the data in the field in Factor 2 (F1=F2), the indicator specified in Equal (columns 58-59) is turned on.

A result field must not be used in compare specifications.

The following considerations apply to the COMP operation:

1. Alignment of the Factor 1 and Factor 2 fields depends on whether they are numeric or alphameric.

   When *numeric* fields are compared, fields of unequal length are aligned at the implied decimal point. Excess positions in numeric fields are assumed to be filled with zeros.

   When *alphameric* fields are compared, fields of unequal length are aligned at their leftmost characters. The excess right-hand positions of the shorter field are assumed to be blank.
2. For numeric fields, the maximum length is 14 digits.
3. For alphameric fields, the maximum length is 256 characters.

All numeric comparisons are algebraic. An absolute comparison can be performed by means of a short routine programmed to meet the user's requirements. Figure 75 shows the example of comparing the absolute value of a sum to a literal. The ADD operation on line 010 places the arithmetic sum of the contents of FACT1 and FACT2 into the result field FACT2.

Indicator 21 is turned on if the sum is negative. In this case, line 020 contains the next operation to be performed: the negative of the contents of FACT2 is placed into the compare field CFLD.

If FACT2 is positive (or zero), the next operation to be executed is specified in line 030. The actual value contained in FACT2 is moved into CFLD.

Line 040 contains the COMP statement that compares CFLD with the literal 10000.0 and sets on one of the indicators 31, 32, or 33, depending on the result of the compare operation.

Test Zone (TESTZ)

Tests the zones of the leftmost (high-order) position of the field whose name is entered in Result Field. This operation is restricted to alphameric fields and can be used to test the zone portion of any of the 256 EBCDIC characters. The TESTZ operation sets on one of the indicators specified in columns 54-59, depending on what zone it has encountered. The TESTZ operation is normally used to test for an 11 or 12 zone that was present in the high order position of a field from an input card file record.

If a 12-zone has been encountered (&, A through I, or any of the other 7 EBCDIC characters that have the same hexadecimal zone as the letter A), the indicator in columns 54-55 is set on. If an 11-zone has been encountered (-, J through R, or any of the other 7 EBCDIC characters that have the same hexadecimal zone as the letter J), the indicator in columns 56-57 is set on. Any other zone causes the indicator in columns 58-59 to be set on.

Factor 1, Factor 2, and columns 52-53 are not used in test-zone operations.

Figure 76 shows an example of this operation. When indicator 25 is on, the field DATA is tested. If the high-order position has an 11-zone, indicator 02 is turned on. If the high-order position has a 12-zone, indicator 01 is set on.

INDICATOR SETTING

1130 RPG provides two operations for setting indicators on and off:

- Set Indicators ON (SETON)
- Set Indicators Off (SETOF)

Columns 54-59 are used for these operations. The headings on those columns have no meaning for SETON and SETOF and are to be ignored.

Set Indicators On (SETON)

Turns on the indicators specified in columns 54-59. All RPG indicators except L0, 1P, and MR can be turned on.

Figure 77 (line 010) shows an example of this facility. Indicator 01 is turned on whenever a specific record type is read. The L3 is a control-level indicator that is turned on when a level 3 control break has occurred. If a level 3 control break is caused by a record type that is not associated with the record identifying indicator 01, halt indicator H1 is turned on.

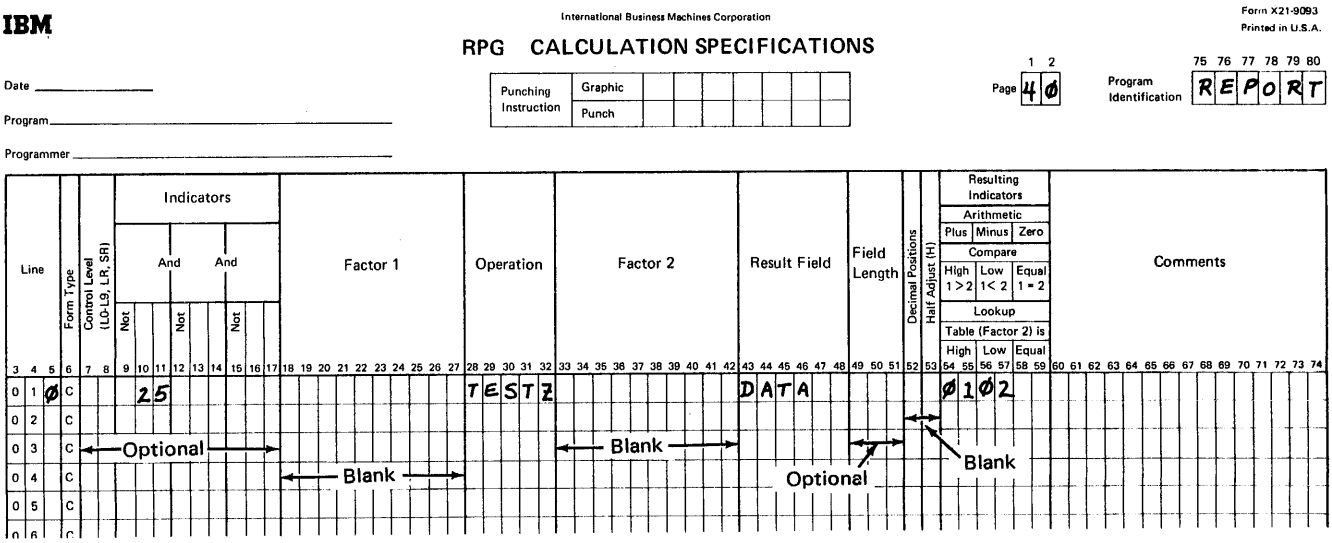


Figure 76. Using the TESTZ Operation

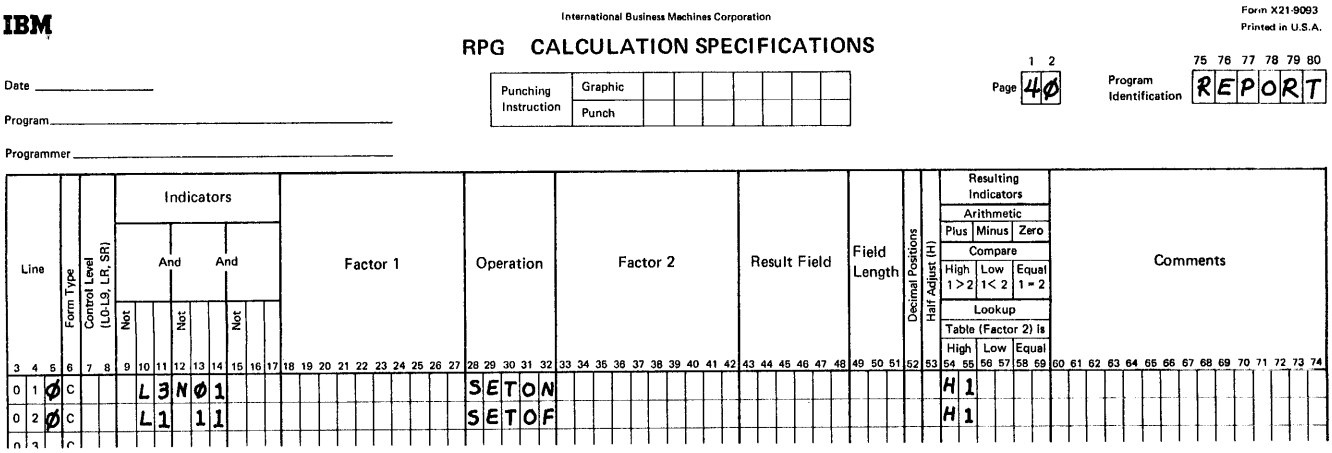


Figure 77. SETON and SETOF Operations

Set Indicators Off (SETOF)

Turns off the indicators specified in columns 54-59 to be turned off. All RPG indicators except L0, 1P, and MR can be turned off.

Figure 77 (line 020) shows an example of this facility. Indicator 11 is turned on at a specified time. L1 is a control level indicator that is turned on when a level 1 control break occurs. Whenever both the indicators L1 and 11 are on, the halt indicator H1 is turned off.

Considerations When Setting Indicators

1. If the LR indicator is turned on during total calculations by a SETON operation, processing is terminated after the output of the total lines.
2. If the OV or OF indicator is changed by a SETON or SETOF operation during total or detail calculations, the RPG object program resets the indicator at the end of the associated total detail, or exception output. Therefore, overflow printing will not be affected.
3. The indicators H1-H9, L1 through L9, and all record identifying indicators defined in columns 19-20 of of the Input Specifications form can be turned on by a SETON operation. However, they will all be turned off by the RPG object program following detail output.
4. Setting indicators L1 to L9 or LR on or off will not automatically set the lower-level indicators on or off.

TABLE OPERATIONS

1130 RPG provides a table lookup operation.

Table Look-Up (LOKUP)

Causes the literal or the field name in Factor 1 to be used as a search argument in a look-up operation. Factor 2 contains the name of the table to be searched, and the result field may contain the name of the table from which the associated function is to be obtained. Decimal alignment is performed for this operation if the table is numeric. The result field may be left blank if the associated function is not to be retrieved. Indicator setting refers to the table argument, rather than the search argument. Refer to the section *Using Tables in the Object Program.*

1130 RPG provides a CHAIN operation that makes it easy to retrieve records from a disk file. For detailed discussion of chaining, refer to the section, *Chaining*.

Chain (CHAIN)

Causes a record to be retrieved from a disk file during calculations. The file from which the record is retrieved is called a chained file. The process of retrieving a record is called chaining. The CHAIN operation may be executed at either detail time or total time.

Records can be retrieved from a sequential file or from an indexed sequential file. To retrieve a record from a sequential file, the record must be identified by relative record number. To retrieve a record from an indexed sequential file, the record must be identified by key.

The CHAIN statement consists of the entry CHAIN in Operation, and entries in Factor 1 and Factor 2. Factor 1 identifies the record to be retrieved. Factor 2 identifies the file from which the record is to be retrieved.

The entry in Factor 1 differs depending upon the organization of the chained file.

For sequential files, Factor 1 either contains the name of a numeric field which contains the relative record number of the file, or contains a literal which is the relative record number.

For indexed sequential files, Factor 1 either contains the name of the chaining field that contains the key for the record or contains a literal that is the key for the record. A key can be a customer number, an item number, or any other convenient means for identifying a record.

Factor 2 identifies the chained file. It must be a filename previously specified on the File Description Specifications form.

The CHAIN statement may be conditioned by control level in columns 7-8 and by indicators in columns 9-17.

The record identifying indicator for any chained file is turned off following detail output. If the user chains to the same file more than once in the same cycle, several record identifying indicators may be on for that file. This can be prevented by using a SETOF operation prior to chaining.

Resulting indicators may be used to determine when a record is not found. If these indicators are used, the same indicator must be specified in columns 54-55 and 56-57. Columns 58-59 must be blank.

When a record is not found, the specified resulting indicator is turned on. For indexed sequential files, a record is not found when the key specified in Factor 1 does not exist in the file. For sequential files, a record is not found when the relative record number is either negative, zero, or larger than the largest record number of the file.

If a resulting indicator is not specified and a record is not found, the program halts and an appropriate message number is placed in the accumulator. The user can then either end the job or bypass the record and retrieve the next record in the file. Bypassing a record will bypass heading and detail output and all remaining calculations for the bypassed record.

Figure 78 shows a typical use of the CHAIN operation. A chaining field called CUSTNO from a transaction record contains the customer number. It is used to chain to a customer file called MASTCUST. If the record is not found, indicator 20 is turned on. If indicator 20 is on, further calculations can be bypassed and an error message printed.

Figure 79 shows the use of the CHAIN operation to retrieve records from an indexed sequential file. The file, named ISAM, is made up of a variable number of records per customer.

Each customer record has a 6-digit key field. The first five digits are the actual customer number. The units position is a digit 0 through 9 depending upon how many records the customer has. Figure 80 contains an example of these records.

The field CHNFLD is set up for the first customer record with a units position of zero. Indicator 20 is used to determine when a record is not found. If 20 is off, an exception output line occurs to print the information from the first record. Then, a one is added to CHNFLD and GOTO is used to branch back to instructions for retrieving the next record.

This process continues until no more records are found for this customer and indicator 20 is turned on. The program then continues to other detail calculations. In this example, the user would have to use the record identifying indicator of a chained file to determine whether a record cannot be found when the units position of the chaining field is zero.

Figure 81 shows the use of the CHAIN operation to retrieve records from a sequential file by use of a relative record number.

Since no index relates to a sequential file, the programmer must have some means of obtaining the relative record number based on a customer number. This can be done by one of three ways:

1. The actual customer number is used as the relative record number (this method depends on the coding structure used).
2. A mathematical formula is used to convert the customer number to a relative record number.
3. A table look-up approach is used to locate the relative record number.

The example in Figure 81 uses the table look-up technique. Indicator 20 determines whether the customer number is in the table, and the appropriate processing that is necessary if that number is not in the table.

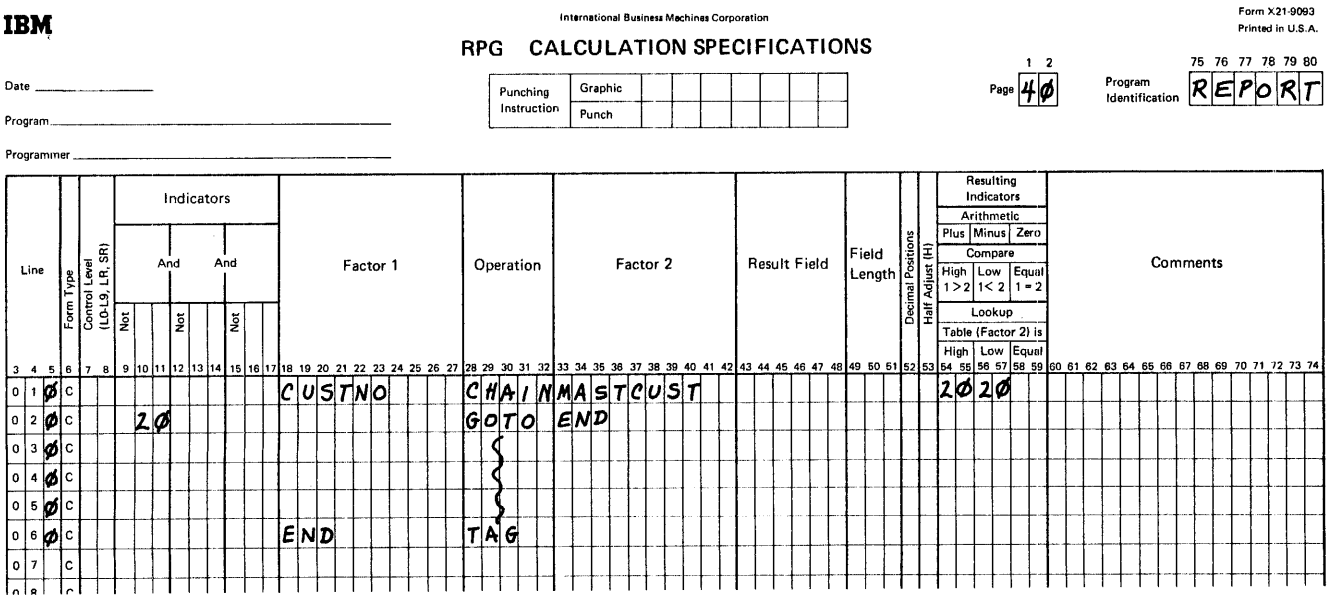


Figure 78. CHAINing Operation

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic							
	Punch							

Page 40

Program Identification: REPORT

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And / Not	Indicators: And / Not	Indicators: 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
010	C						Z-ADD	0														
							MOVEL	CUST	CHNFLD													
						LOOP	TAG															
						CHNFLD	CHAIN	ISAM											20	20		
			N20				EXCPT															
			N20			1	ADD	CHNFLD	CHNFLD													
			N20				GOTO	LOOP														

Figure 79. CHAIN Operation on an Indexed Sequential File

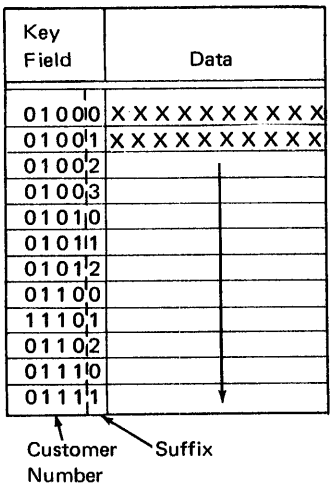


Figure 80. Records in the ISAM File

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic							
	Punch							

Page 40

Program Identification: REPORT

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 / Minus / Zero); Compare (High 1>2 / Low 1<2 / Equal 1=2); Lookup (Table (Factor 2) is: High / Low / Equal)	Comments
01	C			CUST	LOKUPTABCUS		TABNUM				20	
02	C		N20		GOTO END							
03	C			TABNUM	CHAINDISK							
04	C											
05	C											
06	C											
07	C			END	TAG							
08	C											

Figure 81. CHAIN Operation on a Sequential File

1130 RPG provides the following branch and exit operations:

- Branch (GOTO)
- Provide a Name for a Branch (TAG)
- Transfer to an RPG Subroutine (EXSR)
- Begin an RPG Subroutine (BEGSR)
- End an RPG Subroutine (ENDSR)
- Exit to an External Subroutine (EXIT)
- Define RPG Subroutine Labels (RLABL)
- Write Records During Calculations (EXCPT)

Branch (GOTO)

Causes a branch from one point in the program to another.

A GOTO statement consists of the entry GOTO in Operation and a name indicating the point-of-destination in Factor 2. A GOTO statement may have a control-level indicator, and/or one, two, or three indicators in columns 9-17. All other columns must be blank. The GOTO statement is executed only if the conditions established in the indicator and control-level specifications are satisfied.

The name in Factor 2 must be the same as the name in Factor 1 of a TAG statement. More than one GOTO statement may refer to this name.

Branches may be forward (skipping subsequent specifications) or backward (returning by skipping preceding specifications).

A GOTO from outside a subroutine cannot be issued to a tag which is within a subroutine.

Provide a Name for a Branch (TAG)

Marks the point of entry from a GOTO statement. A TAG statement may appear in detail or total calculations or within a closed subroutine.

A TAG statement consists of the entry TAG in Operation and a name in Factor 1. The same name must be used in Factor 2 of the associated GOTO statement. Columns 7 and 8 of a TAG statement may have any valid entry. Columns 9-17 and 33-59 must be blank.

The name in Factor 1 of a TAG statement may be six characters or less. It must begin in column 18. Its first character must be alphabetic. Remaining characters must be alphameric. No special characters or embedded blanks can be used. The name must be unique throughout the program.

Figure 82 shows the use of GOTO and TAG statements. The GOTO statement in line 010 causes control to be passed to ADDR01 on line 060 if indicator 10 is off. Otherwise, control proceeds to line 020. If indicators 05 and 06 are on, the GOTO statement in line 020 causes control to be passed to ADDR02 on line 100. If indicators 05 and 06 are off, control proceeds to line 030.

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date

Program

Programmer

Punching Instruction: Graphic / Punch

Page 40

Program Identification: REPORT

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And / Not / And / Not / 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		N10		GOTO	ADDR01						
02	C		06 05		GOTO	ADDR02						
03	C					Some Sequence of Specifications						
04	C											
05	C											
06	C			ADDR01	TAG							
07	C					Some Sequence of Specifications						
08	C											
09	C											
10	C			ADDR02	TAG							
11	C					Some Sequence of Specifications.						
12	C											
13	C											

Figure 82. GOTO and TAG Operations

Transfer to an RPG Subroutine (EXSR)

Causes a branch from any point in the program to an internal subroutine written in RPG as a part of this program. (See the section *Subroutines.*) After the subroutine has been performed, control is returned to the next specification following the EXSR operation.

EXSR statements consist of the entry EXSR in Operation and the name of the subroutine branched to in Factor 2. EXSR statements may have a control level indicator, and/or one, two, or three indicators in columns 9-17. All other columns must be blank. The EXSR operation may itself be part of an RPG subroutine and may refer to another subroutine. In this case, columns 7-8 would contain SR. However, the name in Factor 2 must not be the name of the subroutine of which the EXSR operation is a part. A subroutine may not call itself. See Figure 83 for an example of the EXSR statement.

Begin RPG Subroutine (BEGSR)

Defines the entry point of an RPG subroutine.

BEGSR statements consist of the entry BEGSR in Operation and the name of the subroutine in Factor 1. This name is the name referred to by EXSR statements. Columns 7-8 must contain SR. All other columns must be blank. See Figure 83 for an example of the BEGSR statement.

End RPG Subroutine (ENDSR)

Is the last statement of an RPG subroutine. ENDSR identifies the exit point from the subroutine. At this point, control is returned to the specification immediately following the EXSR statement that invoked the subroutine.

ENDSR statements consist of the entry ENDSR in Operation. Factor 1 may be used to provide a name for a GOTO statement within the subroutine. Columns 7-8 must contain SR. All other columns must be blank.

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

RPG CALCULATION SPECIFICATIONS

Date _____ Program _____ Programmer _____

Punching Instruction: Graphic / Punch

Page 40 Program Identification REPORT

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
010	C		05		EXSR	SUBR1						
020	C				•							
030	C		22		EXSR	SUBR2						
040	C				•							
050	C	L1			EXSR	SUBR1						
060	C				•							
070	C	SR		SUBR1	BEGSR							
080	C	SR			•							
090	C	SR			•							
100	C	SR			ENDSR							
110	C	SR		SUBR2	BEGSR							
120	C	SR			•							
130	C	SR	17		GOTO	END2						
140	C	SR			•							
150	C	SR		END2	ENDSR							
	C											

Figure 83. BEGSR, ENDSR, and EXSR Operations

Figure 83 shows two RPG subroutines: SUBR1 and SUBR2. Lines 010 and 030 show conditional calls to the subroutines at detail time. If the associated indicator is on, the subroutine is executed. Line 050 shows a call to SUBR1 at total time. The subroutines perform the desired function and return control to the statement following the EXSR statement associated with the subroutine. Line 130 shows a conditional GOTO to the ENDSR statement. If indicator 17 is set on, the GOTO transfers control to the ENDSR statement which returns control to the statement following the EXSR associated with the subroutine.

Exit to an External Subroutine (EXIT)

Causes control to be transferred from the RPG program to an external subroutine written by the user in the 1130 Assembler language. Factor 2 contains the name of this subroutine. For additional information, refer to the section *Subroutines.*

RPG Label (RLABL)

Defines fields or indicators that are to be referred to in external user's subroutines written in the 1130 Assembler Language. (Refer to the section *Subroutines.)* RLABL statements consist of the entry RLABL in Operation and a left-justified entry in Result Field. Factors 1 and 2 must not be used.

When RLABL is used to define a field, Result Field must contain the name of the field to be defined.

When RLABL is used to define an indicator, Result Field must contain the code IN followed by the indicator code.

RLABL statements must be specified after the EXIT statement which refers to the subroutine that uses the field or indicator defined by RLABL.

If the field length and decimal notation (if numeric) has not been defined in the program, they must be defined in the RLABL statement.

Write Records During Calculations (EXCPT)

Allows records to be written at the time calculations are being done. Such records are called exception records and are indicated by an E in column 15 of the Output-Format Specifications form.

A typical example of the use of EXCPT is to write a variable number of similar or identical records (either detail or total) in one program cycle, without repectively coding the output-format specifications for each record.

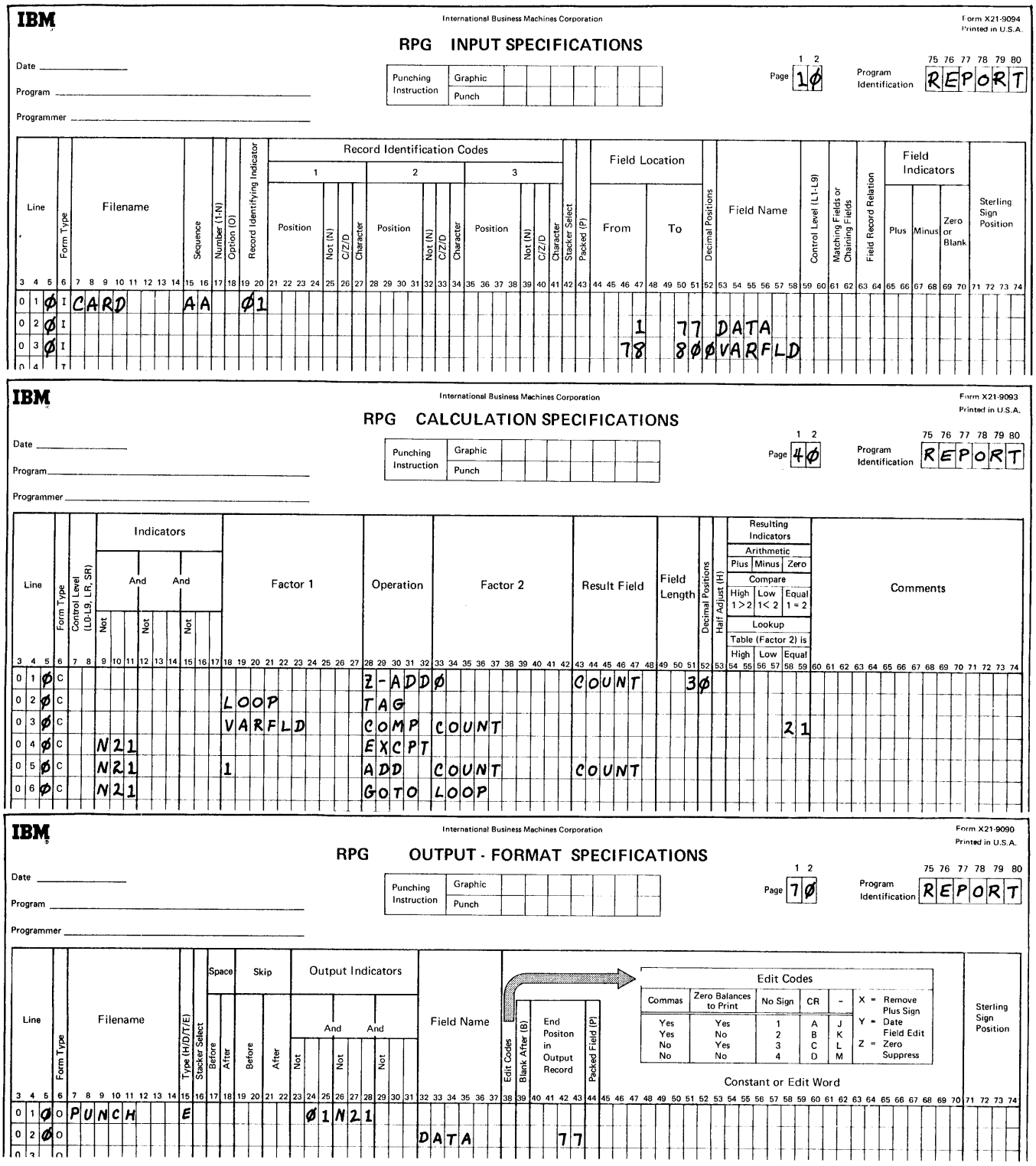
An EXCPT statement consists of the entry EXCPT in Operation. The statement may be conditioned by control levels (column 7-8), or indicators (columns 9-17). The remaining columns should be blank.

Execution of this statement causes all exception records whose indicator tests are met to be placed in the output file. These exception records are produced in the order of their specification on the Output-Format Specifications form.

In Figure 84, the EXCPT operation is used to create a variable number of file replenishment cards. The master card is read and contains data to be reproduced in columns 1-77. The variable number of cards to be punched is contained in a field called VARFLD. The Calculation Specifications show a series of steps that form a loop. A field called COUNT is created and set to zeros. It is then compared against the variable count field. If not equal, the exception output lines are requested by the EXCPT operation.

Exception Output (E in column 15) lines are tested for indicator conditions which are met. In this case, a card is to be punched with the contents of the field DATA. After all exception output lines are tested (only one in this case), control is transferred to the calculation specifications at the step immediately following EXCPT (line 050). A one is added to COUNT and a GOTO (branch) returns to the TAG operation labeled LOOP. The process is repeated again until the field COUNT is equal to the field VARFLD. Each time it is not equal, exception output is called for and another card is punched.

In Figure 85, exception output is called from two places in the program and under different conditions. Line 040 sets on indicator 12 which is used to condition the file called PRINTER. Line 060 sets off indicator 12 to prevent this same output from occurring at the wrong time. Lines 12 and 14 use indicator 13 to control a file called PUNCH in the same manner. The two EXCPT statements (lines 050 and 130) both call for exception output to occur. Line 050 is performed at detail calculations and line 130 at total calculations on a L1 control break.



IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date

Program

Programmer

Punching Instruction: Graphic / Punch

Page: 10

Program Identification: REPORT

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	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	CARD	AA			01																									
02	I																				1	77	DATA								
03	I																				78	800	VARFLD								

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date

Program

Programmer

Punching Instruction: Graphic / Punch

Page: 40

Program Identification: REPORT

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And Not	Indicators: And Not	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 Low	Lookup Equal	Comments
01	C						Z-ADD	0	COUNT		30											
02	C					LOOP	TAG															
03	C					VARFLD	COMP	COUNT													21	
04	C		N21				EXCPT															
05	C		N21			1	ADD	COUNT	COUNT													
06	C		N21				GOTO	LOOP														

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: 70

Program Identification: REPORT

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators: And Not	Output Indicators: And Not	Output Indicators: And Not	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
01	O	PUNCH	E						01	N21								
02	O											DATA			77			
03	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 84. EXCPT Operation to Create Variable Number of File Replenishment Cards

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic							
	Punch							

Page 40

Program Identification: REPORT

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And / Not	Indicators: And / Not	Indicators: And / Not	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus / Minus / Zero)	Resulting Indicators: Compare (High 1>2 / Low 1<2 / Equal 1=2)	Resulting Indicators: Lookup Table (Factor 2) (High / Low / Equal)	Comments
010	C															
020	C															
030	C															
040	C						SETON								12	
050	C						EXCPT									
060	C						SETOF								12	
070	C															
080	C															
090	C															
100	C															
110	C															
120	C	LR					SETON								13	
130	C	LR					EXCPT									
140	C	LR					SETOF								13	

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 70

Program Identification: REPORT

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select	Space: Before	Space: After	Skip: Before	Skip: After	Output Indicators: And / Not	Output Indicators: And / Not	Output Indicators: And / Not	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
010	O	PRINTER	E			1			12									
020	O																	
030	O																	
040	O																	
050	O	PUNCH	E						13									
060	O																	
070	O																	
080	O																	
090	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 85. EXCPT Operation From Two Places and Under Different Conditions

Figure 86 illustrates entries on the Calculation Specifications form that are used in part of a payroll application. The entries are discussed by line number.

Page-Line Explanation

010 The number of hours worked by an employee is compared with zero. If they are equal, indicator 15 is turned on.

020 The number of hours the employee worked is compared with 40. If the employee worked more than 40 hours, indicator 20 is turned on.

030-050 If indicator 20 is on, three calculations are performed. 40 is subtracted from the number of hours worked. Then the overtime hours are placed in the field OVERHR. OVERHR is a 3-position field with one decimal position. RATE is multiplied by OVERHR and the result is placed in the field SAVE. SAVE is a 6-position field with two decimal positions. SAVE is multiplied by 1.5 (the overtime premium rate). The result is half-adjusted and placed back in SAVE.

060 RATE is multiplied by 40, and the result is stored in GROSS. This operation is performed whether or not the employee worked any overtime. This operation is not performed if the employee did not work any hours during the pay period.

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

RPG CALCULATION SPECIFICATIONS

Date _____ Program _____ Programmer _____

Punching Instruction: Graphic / Punch

Page 40 Program Identification REPORT

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: Not	And	Not	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
010	C							HOURS	COMP	0										15			
020	C							HOURS	COMP	40								20					
030	C		20					HOURS	SUB	40	OVERHR	3	1										
040	C		20					RATE	MULT	OVERHR	SAVE	6	2	H									
050	C		20					SAVE	MULT	1.5	SAVE			H									
060	C		N15					RATE	MULT	40	GROSS	6	2	H									
070	C		20					SAVE	ADD	GROSS	GROSS												
080	C		N15						GOTO	FICA													
090	C							(Additional Calculations)															
100	C							FICA	TAG														
110	C							GROSS	MULT	.048	DFICA	6	2	H									
120	C							YDFICA	ADD	DFICA	HOLD	6	2	H									
130	C							HOLD	COMP	374.40									21	21			
140	C		21						GOTO	ADFICA													
150	C							374.40	SUB	YDFICA	DFICA												
160	C							ADFICA	TAG														
170	C							YDFICA	ADD	DFICA	YDFICA												
180	C							(Additional Calculations)															
190	C							WHTAX	TAG														
	C																						

Figure 86. Using The Calculation Specifications Form

070 SAVE is added to GROSS.
080 The program branches to FICA if indicator 15 is off.
090 Additional calculations not used in this example.
100 TAG provides the label FICA.
110 GROSS is multiplied by the 0.048 and the result is placed in the field DFICA. DFICA is a 6-position field with two decimal positions. The result is half-adjusted.
120 DFICA is added to YDFICA and the result is placed in HOLD.
130 The contents of HOLD are compared with 374.40. If HOLD does not exceed 374.40, indicator 21 is turned on.
140 If indicator 21is on, the program branches to the ADFICA.
150 YDFICA is subtracted from 374.40. The result is placed in DFICA.
160 TAG provides the label ADFICA.
170 DFICA is added to YDFICA and the result is stored in YDFICA.
180 Additional calculations not used in this example.
190 TAG provides the label WHTAX to which the program may branch.

This form specifies the kind of output files to be produced and the location of the data fields in the output reports and records.

The specifications for this form are divided into two parts as illustrated in Figure 87. These parts are:

- *File Identification and Control,* which identify the output fields (disk, printer and/or punched card files), and records to be added to the file; direct cards to the appropriate stackers; provide for correct spacing on printed reports; and determine when the records are to be produced.
- *Field Description,* which indicate where and when the individual fields of the output record are to be punched, printed, or written on disk. The entries for these specifications are written on the lines below the file identification entries. Each field is described on a separate line.

Sequence of Specifications

Output operations should be specified in the following sequence:

1. Heading and/or Detail lines.
2. Total lines.
3. Exception lines.

Output operations are performed in the sequence they are specified on the Output-Format Specifications form. In a printed report, for example, the specifications for a level 1 total would be listed first followed by the specifications for the level 2 totals, etc. If this data is to be both printed and punched, specifications for punching and printing lines should be listed alternately. This enables the object program to overlap the punching and printing operations.

If a detail card is to be punched, and selected into a stacker, all of the specifications for these operations must be given as part of one file identification line.

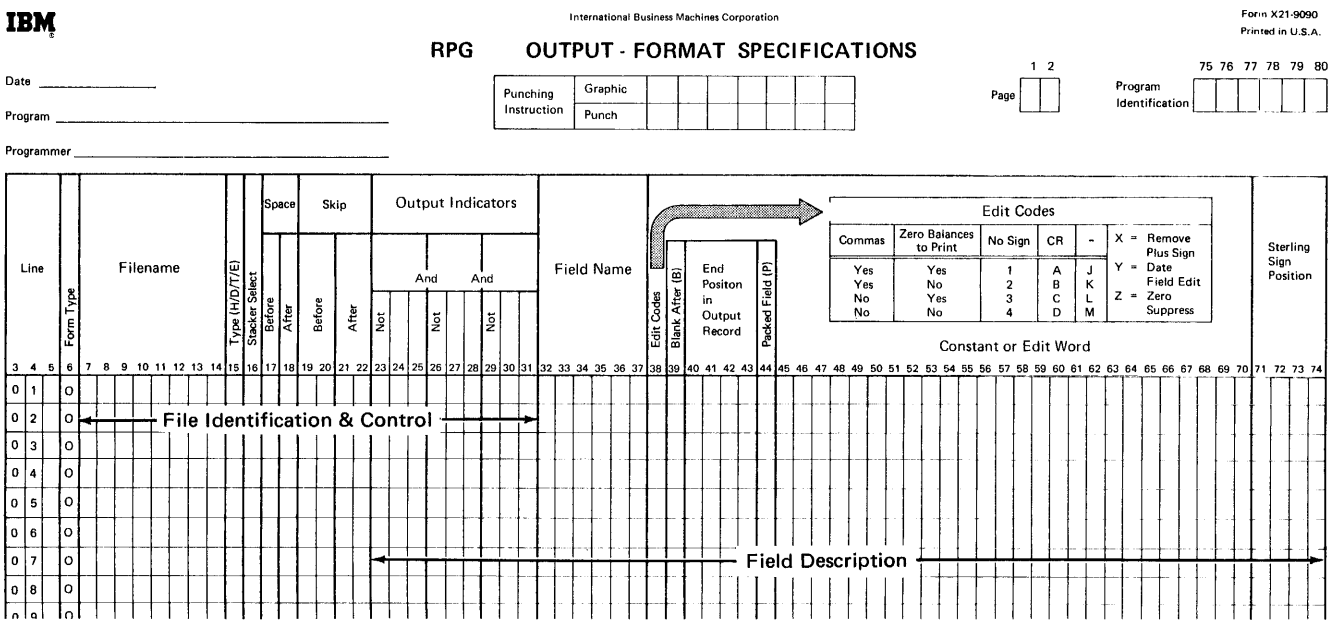


Figure 87. The Output-Format Specifications Form

Specifying Output Units

When writing the output-format specifications, it is not necessary to indicate the specific output units used in the program. The input/output unit used for each file is specified in the File Description Specifications form. Each file name, therefore, is related to a specific output unit. By merely writing the file name on the output form, the output unit has, in effect, been specified.

Writing the specifications in the proper sequence should not be difficult if (1) the layout of a printed report has been correctly made on a Printer Spacing Chart or proportional record layout form, and (2) reference is made to the spacing chart or layout form as the specifications are written.

FILE IDENTIFICATION AND CONTROL

Filename (Columns 7-14)

A filename must be assigned to each output file. The name may contain up to eight alphabetic and/or numeric characters and must be left-justified. The first character must be alphabetic. The name must not contain embedded blanks or special characters. 1130 RPG utilizes only the first five characters.

In file maintenance operations where an input record is updated on the basis of newly calculated results (update or combined files), the same filename must be used for both input and output specifications.

When writing the specifications for output records, the filename must be given only on the first line for the file. On subsequent lines for the same file, the filename need not be specified. However, the filename must be repeated each time another output file is specified on the preceding lines. Figure 88 illustrates this point. (Field specifications are not shown.) The first and second specification lines cause the printing of the overflow heading and detail lines. The third specification line causes the punching of a summary card; this line causes a new file to be created and must therefore be given a separate filename. The remaining specification lines cause the control level 1, exception output, and final total lines to be printed.

Type H/D/T/E (Column 15)

Identifies the type of record being specified, as follows:

H	Heading Record.
D	Detail Record.
T	Total Record.
E	Exception Record.

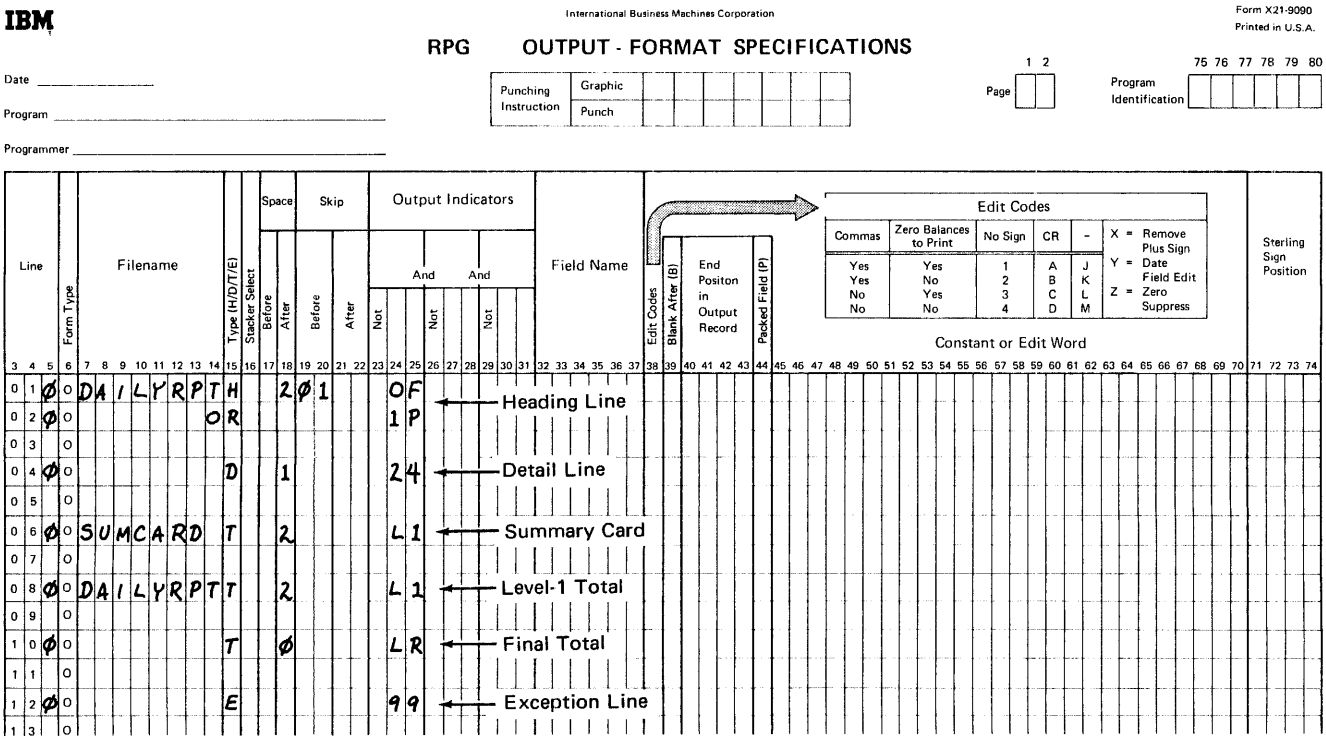


Figure 88. Sample of Filename and Type H/D/T/E Specifications

Figure 88 contains examples of entries for this type of specification.

Heading records usually contain unchanging information such as column headings.

Detail records have a direct relationship to the input records. Most of the data in a detail record comes from the input record or from calculations performed on data from the input record.

Total records usually contain data that is the end result of specific calculations on many detail records.

Exception records are produced whenever the EXCPT operation is executed.

Records must be specified in the following sequence:

1. Heading and Detail Records
2. Total Records
3. Exception Records

Heading and detail records are distinguished only for the user's convenience. RPG treats both record types in the same way.

Adding Records to an Indexed Sequential Organized File (Columns 16-18)

Columns 16-18 may be used to specify that an output record is to be added to an indexed sequential file. This is done by placing the characters ADD in columns 16-18, and ensuring that column 66 of the related file description specifications contains an A.

A record cannot be added to a file in the same program in which that file is retrieved or updated. However, one file can be added to and another can be retrieved in the same program.

Columns 16-18 must be blank if the output record is to be used to update indexed sequential file.

In 1130 RPG the key field for any indexed-sequential file must begin in position 1 of each record. This entry along with the length of the key field is described on the File Description Specifications. If the programmer describes the key field as being 5 positions long, the RPG program will use the first 5 positions of the record to be added as the new key field. In 1130 RPG, the key field can be either alphameric or numeric, but it cannot be packed.

For card and/or printer specifications use these columns as described in the following.

Stacker Select (Column 16)

Causes card records to be selected into stackers of a multi-stacker output unit attached to the system. It is used only for output fields or combined files. The entry in Column 16 may be one of the following:

- Number of the stacker into which the cards are to be placed.
- Blank or 1, if the cards are to be placed in the normal stacker.

New stackers may be specified in subsequent OR lines. If column 16 is left blank in an OR line, the card is directed to the normal stacker. If all OR cards are to go into the same stacker, the same specification must be made for each card. Figure 89 illustrates two examples of stacker select entries.

Space (Columns 17-18) and Skip (Columns 19-22)

Columns 17-22 are used to specify spacing and line skipping for a printer file. At least one entry must be made in these columns if a record is to be printed.

These entries are directly related to the operation of the 1132 or the 1403 printer carriage control tape. Space entries may be specified for the console printer. Skip entries may not be specified for the console printer, or for channels 7, 8, 10, and 11 with the 1132 printer.

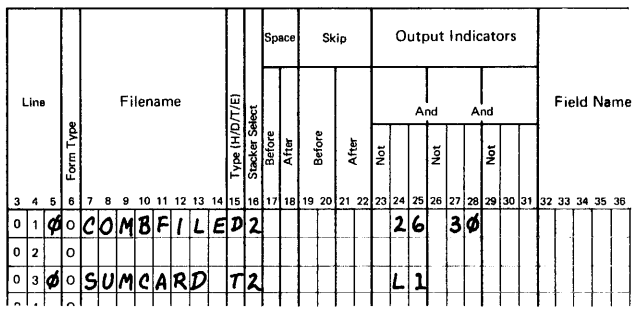


Figure 89. Sample of Stacker Select Specifications

The space and skip entries are:

- *Space Before (column 17),* which may be a 0, 1, 2, or 3 to indicate the number of spaces before printing a line.
- *Space After (column 18),* which may be 0, 1, 2, or 3 to indicate the number of spaces after a line is printed.
- *Skip Before (columns 19-20),* which may be a 01 through 12 to cause the printer carriage to skip to channels 1 through 12 of the carriage tape before the line is printed.
- *Skip After (columns 21-22),* which may be a 01 through 12 to cause the printer carriage to skip to channels 1 through 12 of the carriage tape after the line is printed.

A space before or a space after a zero may not be specified for the console printer.

If space after and skip after are specified on the same line, only skip after is executed.

Skip specifications may differ in various OR lines.

Overflow Indicators

Carriage overflow, indicated by a punch in channel 12 of the carriage control tape, causes the overflow indicator specified on the File Description Specifications to be turned on. This indicator remains on for one complete cycle of processing; it is turned off after the heading and detail lines of the next record are printed.

A test for carriage overflow is made immediately before each line of the report is printed, but after any space before or skip before specifications are executed.

If carriage overflow occurs at detail time, the remainder of the detail lines are printed. Then the overflow indicator is turned on and the next record is read. If the appropriate conditions are satisfied, total calculations and total output are performed. Then the overflow lines are printed or an automatic skip to the next page is executed if no overflow lines are specified. When the next detail calculations and detail output have been performed, the overflow indicator is turned off.

If a carriage overflow occurs at total time, the remainder of the total lines are printed. Then the overflow indicator is turned on. Overflow lines are printed if specified. Otherwise, an automatic skip to the next page is executed. When the next detail calculations and detail output have been performed, the overflow indicator is turned off.

Overflow Lines: An overflow line is any line conditioned by OF or OV. It is printed during overflow output time in the object flow.

A line conditioned by NOF or NOV is not an overflow line. It is printed at either heading/detail, total, or exception time.

In any program, total overflow lines are printed, followed by heading and detail overflow lines.

If overflow lines are not specified on the report, a punched in channel 12 of the carriage control tape causes the carriage to skip to a punched in channel 1.

Indicators OF and OV cannot be specified to condition an Exception Output line.

Fields Conditioned by Overflow: A field description may be conditioned by an overflow indicator. This field is treated as any other field conditioned by an indicator. It is put out only if the field indicator is on.

Multiple Printers: Two printer files may be used for each object program. A unique overflow indicator must be specified for the second printer.

Example of Overflow Printing Specifications: Figure 90 shows how overflow printing specifications may be coded.

In the top half of the figure, the line is printed whenever the OF or L1 indicator is on. If L1 and OF are on the same time, the line is printed twice. This might cause an error in the printed report.

To prevent this, the conditions must be made mutually exclusive, that is, so that both conditions can cause the skip, but not at the same time. Then the line is not printed twice. This is coded as shown in the bottom half of the figure.

Figure 91 shows the possible conditions under which the overflow indicator will be set and overflow lines will occur.

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 7Ø

Program Identification REPORT

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

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators And Not	Output Indicators And Not	Output Indicators And Not	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
3 4 5	6	7 8 9 10 11 12 13 14	15	16	17	18	19 20	21 22	23 24	25 26	27 28	29 30 31	32 33 34 35 36 37	38	39	40 41 42 43	44	45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70	71 72 73 74
0 1 Ø	O	* INCORRECT					Ø V	ERF	LOW			HANDLING						
0 2 Ø	O	OUTPUT	H				Ø 1		OF									
0 3 Ø	O		OR						L 1									
0 4 Ø	O														1ØØ		'ACCOUNT'	
0 5	O																	
0 6	O																	
0 7 Ø	O	* CORRECT					OV	ERFLOW				HANDLING						
0 8 Ø	O	OUTPUT	H				Ø 1		OFNL1									
0 9 Ø	O		OR						L 1									
1 0 Ø	O														1ØØ		'ACCOUNT'	
1 1	O																	

Figure 90. Overflow Printing Specifications

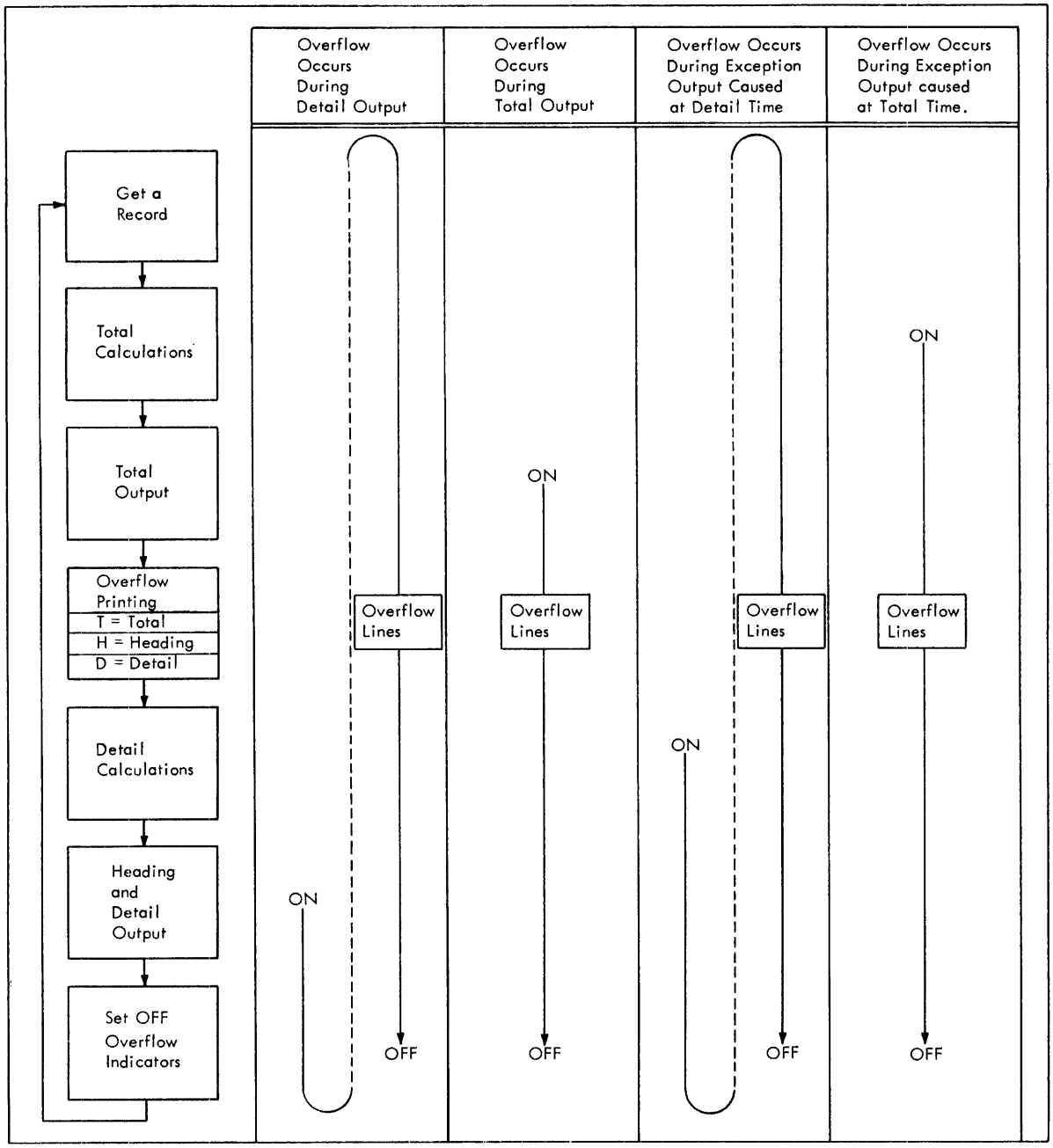


Figure 91. Setting of the Overflow Indicator and Overflow Printing

Figure 92 shows how the overflow indicator is actually set on. The indicator is not set until a line is printed on or after the carriage control tape's 12-punch is encountered. If multiple lines are printed during an output block (detail, total, or exception) and the 12-punch is passed by the first line:

a. the overflow indicator is set according to Figure 92.
b. Overflow time and all other lines for the particular block will be printed.

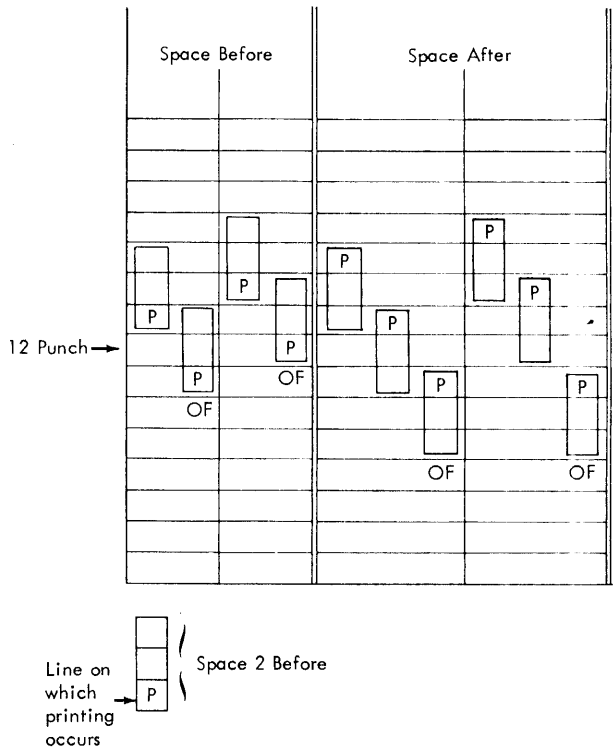


Figure 92. Setting of the Overflow Indicator

Output Indicators (Columns 23-31)

Control (1) when the record is to become output and (2) when a particular field is to be printed, punched, or written on disk. A maximum of three indicators may be specified on a line.

Several indicators specified on the same line are considered to be in AND relationship. Hence, all conditions specified must be satisfied before the specified output operation can be executed.

If an AND relationship requires more than three indicators, AND is entered in columns 14-16 of the following line, and the additional indicators are specified on that line.

If the output condition is executed in an OR relationship, OR is entered in columns 14-15 of the following specification lines, and the OR indicators are specified on that line.

Additional specifications lines can specify as many output indicators in either an AND or an OR relationship as required. Each additional line must begin with AND or OR in column 14. AND and OR can only be specified to condition records, not fields.

The following indicators can be specified in the Output Indicators columns:

1. *Record Identifying Indicators* which specify the particular input record type on which the output operation is to be performed. These indicators refer to entries in columns 19-20 of the Input Specifications form.
2. *Resulting Indicators* control the output operation by conditions that have occurred during calculations. These indicators refer to entries in columns 54-59 of the Calculation Specifications form.
3. *Field Indicators* control the output operation by the status of the input field. These indicators refer to entries in columns 65-70 of the Input Specifications form.
4. *Control Level (L0 . . . L9, LR)* cause the output operation to be performed only when the affected control break has occurred.
5. *Matching Record Indicators (MR)* causes the output operation to be performed only if there is a matching record in a secondary file.
6. *Halt Indicators* suppress the output operation when an error has been detected in the input data or during calculation.
7. *Overflow Indicators (OF, OV)* cause the output operation to take place only if a page overflow has occurred. These indicators do not apply to the first page of a report. They must be defined on the File Description Specifications.
8. *First Page Indicator (1P)* causes headings to be printed on the first page of a report. This indicator is turned on only at the beginning of the processing before any input records have been read.

If the user intends to print identical heading lines on each page, an overflow indicator (OF or OV) and the first page indicator (1P) must be specified in an OR-relationship, because the overflow condition does not occur until after the first page has been printed.

If any of the output indicators must be off, enter N in columns 23, 26, or 29 as appropriate.

The first page (1P) and level zero (LO) indicators are always on at the beginning of executing an object program.

Examples of Output Indicators

Figure 93 shows seven examples of output indicators used with the output file. The numbers to the right of the figure refer to the item numbers in the following text.

1. The carriage is skipped to channel 2 before printing. The heading line is printed only when an overflow condition occurs or when the 1P (first page) indicator is on.
2. The detail line is printed only if indicator 14 is on and indicators 26, 28, and 30 are off. Indicators 26, 28, and 30 could be field indicators or record identifying indicators from the input specifications, or resulting indicators from the calculation specifications.
3. The detail line is printed if indicator 40 or 46 is on.
4. The total line is printed and the carriage is skipped to channel 2 before printing only if the level 2 indicator is on, the MR indicator is off, and H2 is off. The specification NH2 prevents the object program from printing a line if an error condition has occurred. Although the program does not stop until all processing for the record has been completed, printing of erroneous data is prevented.
5. The summary record is printed at level 2 control time, but the MR (matching records) indicator must also be on.
6. A summary card is punched at level 2 control time and is selected into stacker 2.
7. An exception output line is written if indicators 03 and 98 are on.

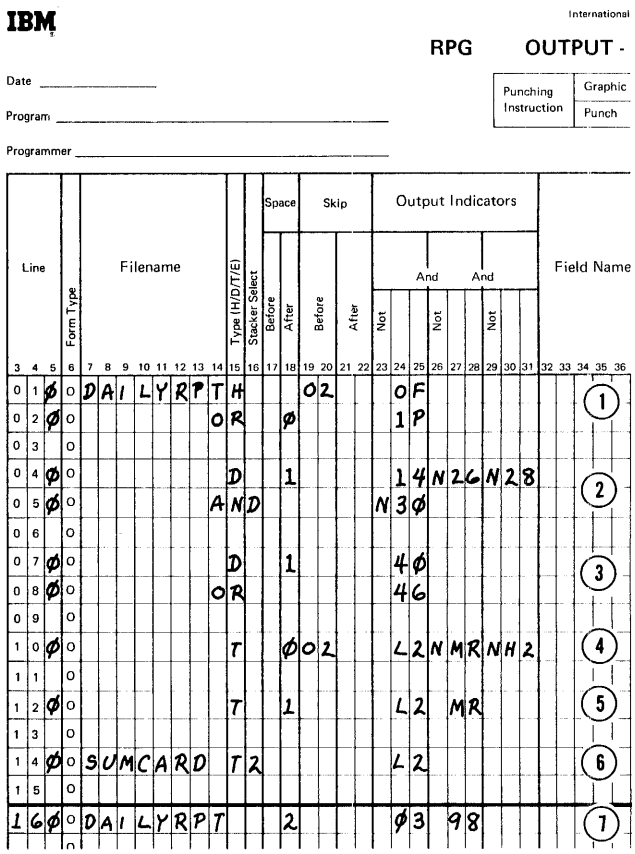


Figure 93. Specifying Output Indicators for Output Lines

FIELD DESCRIPTION

Field description entries indicate where and when the individual fields of the output record are to be punched, printed, or written on disk. They also define the output format of individual fields of a record.

The fields of a record are written on the lines below their corresponding file entries. Each field is described on a separate line, and no entries are permitted in columns 7-22. The fields are written in the sequence they are specified.

The rightmost position of a numeric field contains the sign. A 12-punch or a digit punch designates a positive value; an 11-punch a negative value. If a numeric field with a negative value is printed, the rightmost position in the output field will contain one of the letters J through R, or the special character – (minus). This can be avoided by means of an appropriate edit word, or edit code in the output specifications.

Output Indicators (Columns 23-31)

Follow the same considerations and format as described for Output Indicators for the File Identification and Control portion of this form, except that output indicators must not be specified in an OR-relationship, on a field description line.

Figure 94 shows four sets of indicators used as output indicators for field description lines. The numbers to the right of the figure correspond to the following:

1. Four fields are printed from a detail record identified by indicator 44: INVOIC, AMOUNT, CUSTR, and SALESM. The entry L1 causes the contents of the field SALESM to be printed only for the first detail record of each control group. Only the field named SALESM has group indication. (Note that control level indicators remain on during the following detail calculation and output cycle.)
2. Line 070 through 100 show how to prevent the printing of just one field of a record. The contents of the field AMOUNT is printed only if indicator 16 is turned off.
3. Lines 120 through 180 show selective printing of constant headings contained in cards for an invoice form. The specification prints all the heading information on the first form. If the information for one customer order continues on two or more forms, only the customer and invoice fields on succeeding forms are printed. Printing of the entire line is controlled by indicators 04 or OF. In the field description specifications, the OF indicator is also used to prevent printing of fields named ORDER, DATE, and SALESM when an overflow condition occurs. In this example, indicators OF and 04 are never on at the same time.
4. Lines 200 through 230 show how the printing of a field can be controlled by an AND relationship and an OR relationship. The printing of the field named

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 70 — Program Identification REPORT

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators And Not	And Not	And Not	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
015	O	PRINT	D		1				44									
020	O											INVOIC			16			
030	O											AMOUNT			26		(1)	
040	O											CUSTR			40			
050	O								L1			SALESM			59			
060	O																	
070	O	PRINT	D		1				24									
080	O											NAME			40			
090	O											DEPT			48		(2)	
100	O								N16			AMOUNT			59			
110	O																	
120	O	PRINT	H		1				04									
130	O		OR				03		OF									
140	O								NOF			ORDER			16			
150	O								NOF			DATE			26		(3)	
160	O								NOF			SALESM			39			
170	O											CUST			48			
180	O											INVOIC			61			
190	O																	
200	O	TOTAL	T		1				12	MR								
210	O								16	NH2	NL3	DIVSON			26			
220	O								26			AMOUNT			48		(4)	
230	O								28			AMOUNT			59			

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 94. Specifying Output Indicators for Fields

DIVSON is controlled by three AND indicators: 16, NH2 and NL3. The field named AMOUNT is controlled by two OR conditions. In the field description line, the OR relationship is used by writing the field name twice and specifying each appropriate OR indicator. OR cannot be specified in columns 14-15 of a field description line.

Fieldname (Columns 32-37)

Identifies each field of the record to be printed. The fields may be listed on the form in any sequence. The order in which they appear in the output record is determined by the entry in columns 40-43.

The field names must have been previously defined on either the Input Specifications or Calculation Specifications form. If a constant is to be written, it is specified in Constant or Edit Word (columns 45-70).

Page Numbering

The Output-Format Specifications form can be used to cause pages of a printed report to be numbered automatically. The page number is increased by one each time a page is printed.

To specify automatic page numbering, the word PAGE must be placed in columns 32-37 and the position where the last digit of the page number is to appear must be in columns 40-43.

When page numbering is specified solely on the Output-Format Specifications form, page numbering begins with one and PAGE is a 4-column numeric field without decimal positions.

To begin numbering pages with a number other than one, the Output-Format Specifications form must be used in conjunction with either the Input Specifications form or the Calculations Specifications form. In this case, the field PAGE must be defined either in Field Name of input specifications or in Result Field of calculation specifications. The field length and the number of decimal positions must also be defined on the Input Specifications or the Calculations Specification form as appropriate.

By using output indicators on the Output-Format Specifications form, a page number can be reset to zero and a new series of page numbers may be started during the processing of the object program. In this case, the indicator must be placed in Output Indicators on the same line as the field name specification PAGE. If the status of the indicator is as specified, the PAGE field is reset to zero. Remember that one is added to the field before it is printed.

Any field name which begins with PAGE is treated as a page numbering field. Therefore, the user could specify several page numbering fields for one or more files. A second page numbering field might be called PAGE1. A third might be called PAGE2.

Figure 95 shows the use of the Input Specifications form in conjunction with the Output-Format Specifications form to define automatic page numbering. Page numbering is defined on lines 010-020 of the Input Specifications form. Then on line 020 of the Output-Format Specifications form, PAGE is defined as a separate field whose last digit will print in position 100. If the page numbering were to begin with the number 500, 0499 would be entered in columns 2-5 of the input record that contains the character - (minus) in position 1.

Any field name that begins with PAGE will be treated as a page numbering field. Therefore, the user could specify several page numbering fields for one or more files. A second page numbering field might be called PAGE1. Figure 96 illustrates the use of two page counters.

Edit Code (Column 38)

Causes numeric fields to be edited. This code can provide comma insertion, zero suppression, sign removal, negative indication with either – or CR, blanking of zero fields, date field editing, and insertion of the decimal point.

Zero suppression means that zeros to the left of the significant digits and the sign on the units position do not appear in the output record.

Edit codes are divided into two categories:
simple and combination.

Simple edit codes cause no punctuation of numeric fields. These codes are:

- X Has no affect in 1130 RPG since the preferred sign is F.
- Y Causes the date field to be edited, as follows:
  - 3 digits (n)n/n
  - 4 digits (n)n/nn
  - 5 digits (n)n/nn/n
  - 6 digits (n)n/nn/nn

  The first digit is zero suppressed. If inverted print is specified on the RPG Control Card, a decimal point is printed instead of a slash.
- Z Causes leading zeros to be suppressed, and signs to be removed. Decimal positions are not considered.

IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic								
	Punch								

Page 1 Ø

Program Identification 75 76 77 78 79 80: R P T Ø Ø 1

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	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
3 4 5	6	7 8 9 10 11 12 13 14	15 16	17	18	19 20	21 22 23 24	25	26 27		28 29 30 31	32	33 34		35 36 37 38	39	40 41		42	43	44 45 46 47	48 49 50 51	52	53 54 55 56 57 58	59 60	61 62	63 64	65 66	67 68	69 70	71 72 73 74
0 1 Ø	1	A D D R F I L E N S				3 3	1		C -										1												
0 2 Ø	1																				2	5 Ø		P A G E							
0 3																															

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 7 Ø

Program Identification 75 76 77 78 79 80: R P T Ø Ø 1

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select	Space: Before	Space: After	Skip: Before	Skip: After	Output Indicators: And: Not	Output Indicators: And:	Output Indicators: And: Not	Output Indicators: And:	Output Indicators: And: Not	Output Indicators: And:	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
3 4 5	6	7 8 9 10 11 12 13 14	15	16	17	18	19 20	21 22	23 24	25 26	27 28	29 30	31		32 33 34 35 36 37	38	39	40 41 42 43	44	45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70	71 72 73 74
0 1 Ø	O	R E P O R T	H		1				O F												
0 2 Ø	O								L 3						P A G E	Z		1 Ø Ø			
0 3	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 95. Page Numbering

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 7 Ø

Program Identification 75 76 77 78 79 80: R P T Ø Ø 1

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select	Space: Before	Space: After	Skip: Before	Skip: After	Output Indicators: And: Not	Output Indicators: And:	Output Indicators: And: Not	Output Indicators: And:	Output Indicators: And: Not	Output Indicators: And:	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
3 4 5	6	7 8 9 10 11 12 13 14	15	16	17	18	19 20	21 22	23 24	25 26	27 28	29 30	31		32 33 34 35 36 37	38	39	40 41 42 43	44	45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70	71 72 73 74
0 1 Ø	O	L O W E R P R T	D				Ø 2		2 Ø												
0 2 Ø	O		O R						O F												
0 3 Ø	O														P A G E 1	Z		1 Ø Ø			
0 4 Ø	O	U P P E R P R T	D				Ø 1		O V												
0 5 Ø	O														P A G E	Z		5 Ø			
0 6	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 96. Page Numbering with Two Counters

Combination edit codes cause an amount field to be punctuated. Leading zeros are always suppressed. The decimal point is inserted for all non-zero values in fields with decimal positions. If a zero balance is to be suppressed, the field prints as blanks. If a zero balance is to be printed and the field has decimal positions, the decimal positions print as zero and the decimal point is inserted. If a zero balance is to be printed and the field does not have decimal positions, a zero is printed in the units position of the field and the remainder of the field is blank. These edit codes are summarized in Figure 97.

The following must be considered when using edit codes:

1. When inverted print is specified on the RPG Control Card, the use of a comma and period is reversed in punctuating amount fields.
2. When '*' is entered in Columns 45-47, each leading zero is replaced by *.
3. When '$' is entered in columns 45-57, the dollar sign appears immediately to the left of the high-order significant (non-zero) digit.
4. Column 38 must be left blank when an edit word is used in columns 45-57.
5. Edit codes A-D and J-M require extra print positions because of the 'CR' and '-' symbols. The end position (columns 40-43) refers to the field including the edit symbols.

Figure 98 illustrates the use of edit codes when printing numeric fields.

Blank After (Column 39)

Defines the contents of a field after that field is placed in the output record, as follows:

	Negative Balance Indicator		
	None	CR	-
Print with commas, print zero balance	1	A	J
Print with commas suppress zero balance	2	B	K
Print without commas print zero balance	3	C	L
Print without commas suppress zero balance	4	D	M

Figure 97. Summary of Edit Codes

B The output field is to be reset to blanks or zeros. Alphameric fields are set to blanks. Numeric fields are set to zeros.
blank The output field is to remain unchanged.

Note: A specification in Blank After also affects any constant (columns 45-70) that may be contained in the line. Since each constant is stored only once for every RPG program, no matter how often it is used, a constant affected by a blank-after specification is not available for use in any following operation. If a field is tested for indicators in the plus, minus or zero columns (Input or Calculation Specifications), a blank-after specification will not reset these indicators.

End Position in Output Record (Columns 40-43)

Is a decimal number that indicates the position in the output record where the rightmost (low-order) character of the field is to be located. Leading zeros may be omitted.

For example, if a 10-position amount field is to be printed in print positions 21 through 30, the entry in columns 42-43 would be 30.

When edit words or edit codes are being used, the end position must allow for the punctuation characters introduced by the editing process.

Packed Field (Column 44)

Indicates whether or not the output field is to be in packed decimal format, as follows:

P The field is to be packed.
blank The field is not to be packed.

This specification may only be used with disk files.

If the output is a constant or is to be interpreted or printed, leave this column blank.

The length of an output field in characters of packed format is:

$$\frac{n + B}{2}$$

where n is the number of digits used, and B=1 if n is odd, or B=2 if n is even.

Edit Codes	Positive Number-Two Decimal Positions	Positive Number-No Decimal Positions	Negative Number-Three Decimal Positions**	Negative Number-No Decimal Positions**	Zero Balance-Two Decimal Positions	Zero Balance-No Decimal Positions	Negative Number -Two Decimal Positions- End Position Specified as 10. Output Print Positions								
							3	4	5	6	7	8	9	10	11
Unedited	1234567	1234567	00012b	00012b	000000	000000				0	0	4	1	2	
1	12,345.67	1,234,567	.120	120	.00	0					4	.	1	2	
2	12,345.67	1,234,567	.120	120							4	.	1	2	
3	12345.67	1234567	.120	120	.00	0					4	.	1	2	
4	12345.67	1234567	.120	120							4	.	1	2	
A	12,345.67	1,234,567	.120 CR	120 CR	.00	0				4	.	1	2	C	R
B	12,345.67	1,234,567	.120 CR	120 CR						4	.	1	2	C	R
C	12345.67	1234567	.120 CR	120 CR	.00	0				4	.	1	2	C	R
D	12345.67	1234567	.120 CR	120 CR						4	.	1	2	C	R
J	12,345.67	1,234,567	.120 -	120 -	.00	0					4	.	1	2	-
K	12,345.67	1,234,567	.120 -	120 -							4	.	1	2	-
L	12345.67	1234567	.120 -	120 -	.00	0					4	.	1	2	-
M	12345.67	1234567	.120 -	120 -							4	.	1	2	-
X	1234567	1234567	00012b	00012b	000000	000000				0	0	4	1	2	
Y*			0/00/12	0/01/20	0/00/00	0/00/00			0	/	4	1	/	2	
Z	1234567	1234567	120	120								4	1	2	

*The Y code suppresses the leftmost zero only. It edits a three through six digit field according to the following pattern:
nn/n
nn/nn
nn/nn/n
nn/nn/nn

**The b represents a blank. This is caused by a negative zero not corresponding to a printable character.

Figure 98. Edit Code Usage

Constant or Edit Word (Columns 45-70)

Is used to:

- Include constants (literals) in output records.
- Specify the edit word for proper output format.
- Modify the function of the edit code (column 38). Permit editing of numeric fields.

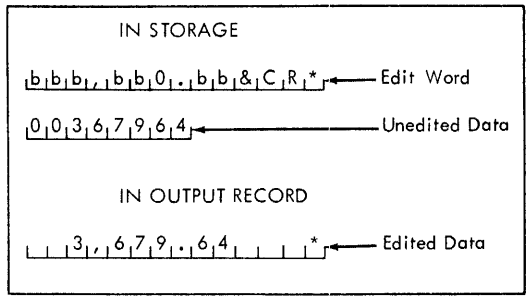


Figure 99. Functions of Edit Operation

Constants or Literals

An alphameric literal of 24 characters or less may be placed in columns 45-70. The literal must begin in column 45, and must be enclosed in apostrophes. The literal is placed on the output record as defined in End Position in Output Record (columns 40-43).

Any character in the EBCDIC character set may be used in an alphameric literal. Blanks are treated as valid characters.

Alphameric literals must be enclosed in apostrophes. An apostrophe (') is represented by two consecutive apostrophes. For example, the literal 5 o'clock would be entered at '5 o' 'clock'.

When an alphameric literal is defined on a line, Field Name (columns 32-37) must be blank.

Edit Code Options

The characters '*' and '$' may be entered in columns 45-47 to modify the edit code entered in column 38.

'*' indicates that all leading zeros are to be replaced by *.

'$' indicates that a floating dollar sign is to be used. The character $ prints immediately left of the high-order significant digit.

Edit Words

For typical fields, the use of edit codes will provide proper editing. Edit words may be used where unusual editing requirements are needed.

An edit word provides for the punctuation of amount fields, including the printing of dollar signs, commas, periods, and sign status. Edit words can also be used to suppress leading zeros, and to provide for asterisk protection up to a specific position as indicated (see items 3 and 4 under the next section). A sample edit operation is shown in Figure 99.

When an amount field is to be edited, its edit word is placed in columns 45-70 of the same output specification line where the field to be edited is specified.

An edit word consists of three parts (the body, the status, and the expansion) as shown in Figure 100.

The *body* governs the transfer of the data field to the output record. The body begins at the leftmost character of the edit word and contains the same number of blanks (one zero or an asterisk) as the number of digits of the data field.

The *status* position displays the status (positive or negative) of the data field. It is the portion continuing to the right from the body to the CR (credit) or - (minus) symbol. Edit words that contain no CR or - symbol have no status portion.

The *expansion* remains unchanged. It is the portion continuing to the right from the status portion (or body portion if there is no status portion) and ending with the rightmost character of the edit word.

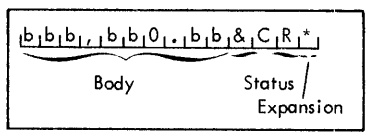


Figure 100. Body, Status, and Expansion of an Edit Word

Rules for Forming an Edit Word

1. An edit word must be enclosed in apostrophes.
2. A blank in the body of the edit word is replaced with the character from the corresponding position of the data field specified in *Field Name.*
3. An ampersand in the body or status portion causes a blank in the edited field. It remains unchanged in the expansion portion.
4. A zero is used for zero-suppression. It is placed in the rightmost position where zero suppression is to stop. It is replaced with the character from the corresponding position of the data field, unless that character is zero. Column 38 (edit codes) must be left blank.
5. If leading zeros are desired, the edit word must contain one more position than the field to be edited. A zero must be placed in the high-order position of the edit word.
6. An asterisk in the body of the edit word is used for asterisk protection and zero suppression. It is placed in the rightmost position where zero suppression is to stop. It is replaced with the non-zero character from the corresponding position of the data field. Each suppressed zero is replaced by an asterisk. An asterisk preceding a zero is interpreted as representing asterisk protection.
7. A dollar sign in the body of the edit word written immediately to the left of the zero-suppression code causes the insertion of a dollar sign in the position to the left of the first significant digit. This is the *floating dollar* sign. A dollar sign that is entered immediately after the initial single-quote mark is fixed (printed in the same location each time). This is the *fixed dollar sign.*
8. The decimal and commas are printed in the same relative positions they were written in the edit word. If they are to the left of significant digits, they are blanked out or replaced by an asterisk.
9. All other characters used in the body of the edit word are printed if they are to the right of significant digits in the data field. If they are to the left of high-order significant digits in the data word, they are blanked out. If asterisk protection is used, they are replaced by an asterisk.
10. The letters CR or the minus symbol in the status portion of the edit word are undisturbed if the sign in the data field is minus. If the sign is plus, CR and - are blanked out.
11. Characters to the right of the status portion of the edit word are undisturbed.
12. The edit word may be larger than the field to be edited.

Figure 101 illustrates the use of constants and edit words. The numbers to the left refer to the item numbers in the following text.

1. The constant 26.75 is in the output record ending in position 96. The fieldname specification must be blank.
2. The constant DEPARTMENT TOTAL is contained in the output record ending in position 96. The field name must be blank.
3. This example illustrates zero suppression to the left of significant digits. The letters CR are written because the amount field can contain a negative value.
4. The floating dollar-sign protection enters the $ to the left of the first significant digit.
5. Asterisk protection enters as many asterisks to the left of the first significant digit as required to fill out the number of positions specified in the edit word.

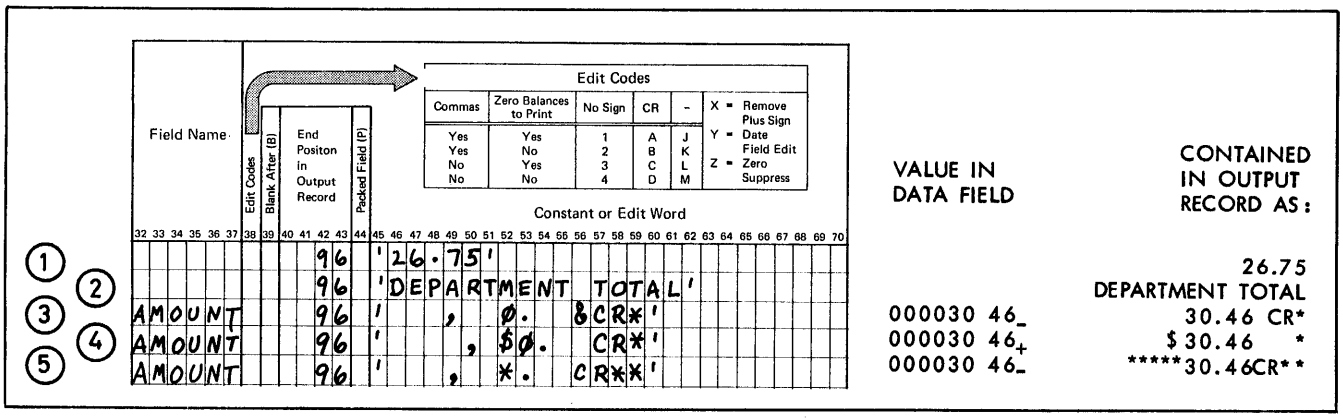


Figure 101. Using Constants and Edit Words

Sterling (Columns 71-74)

Is a decimal number that indicates the position in the record that will contain the sign of the sterling field. Leading zeros may be omitted. Enter an S in column 74 if the sign is in the normal position. If the sterling specification is not required, leave this column blank.

A table is an arrangement of data that is searched and used by the object program. Tables are loaded into storage by the object program before any files are processed.

A table may consist of two parts: arguments and functions.

In Figure 102, the table consists of part numbers (arguments) and prices (functions). The card file contains part numbers that have been ordered. The cards do not contain the prices of the parts. The part number is selected from the card by the RPG program, the table is searched, and the price is retrieved and made available for additional processing. If the price of part number 10 is wanted, the table is searched until part number 10 is found. The corresponding function of 10 in the table is 155. The number used to search the table is called the *search argument*.

Entries in a table may be:

- Arguments
- Functions
- Alternating arguments and functions
- Alternating functions and agruments

An example of each of these entries is shown in Figure 103.

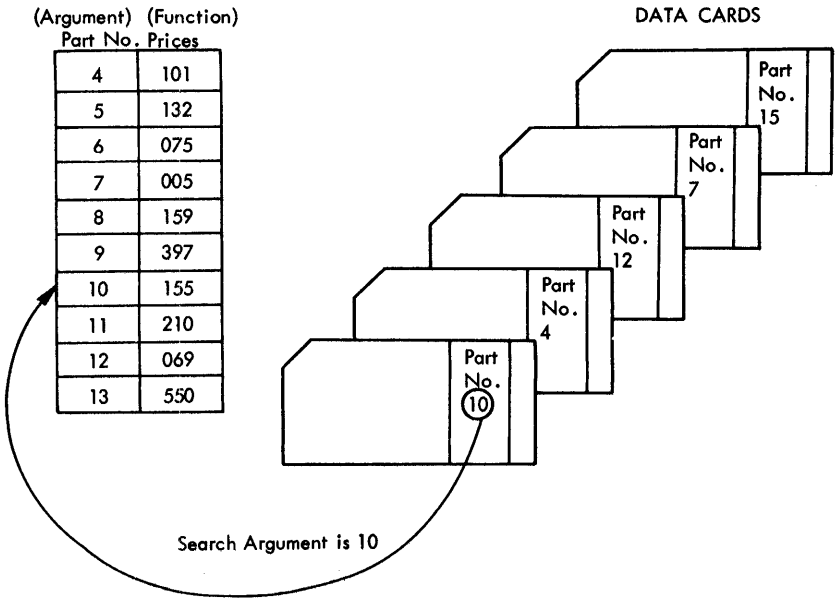


Figure 102. Using a Table

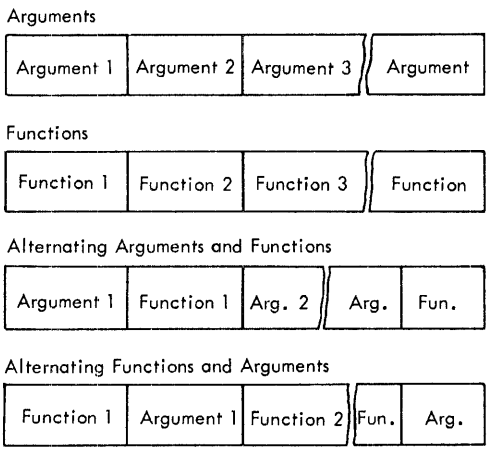


Figure 103. Types of Tables

RULES FOR FORMING TABLES

1. For tables consisting of arguments only, or functions only, each unit of table data is called a table entry. For tables consisting of alternating arguments and functions, one argument and one function are called one table entry.
2. The collection of all arguments is assigned a name. The collection of all function entries is assigned a name. The table name may consist of 4, 5, or 6 characters. The first three must be TAB. The rest may be any alphameric characters.
3. Although all of the tables may be loaded from the same device, a unique file name must be assigned in columns 7-14 for each line entry of the File Description Specifications form and repeated in columns 11-18 of the Extension Specifications form. Each line entry on the Extension Specifications form must have:

- A unique filename (columns 11-18)
- Entries in columns 27-45 if the table contains only arguments or functions.
- Entries in columns 27-45 and 46-57 if the table consists of alternating arguments and functions.

Rules for Creating Records Containing Table Data

1. The first table entry for each record must begin in column 1.
2. All records must have the same number of table entries, except the last (see Figure 104), record which may have any number of entries.
3. All entries must be continuous in every record. In Figure 104, the first entry begins in position 1 and the second entry begins in position 10. No blanks may be contained between the entries.
4. All function or argument entries belonging to a table must have the same length. In Figure 104; each argument is three positions long, and each function is six positions long.
5. When alternating tables are used, each record must begin with an entry of the same type; i.e., each record must always begin with an argument, or each record must always begin with a function as shown in Figure 104.
6. When alternating tables are used, the table entries in each record must not be split. For example, function 3 must be in the same record as argument 3.
7. If a table consists of all arguments or all functions, an argument or a function must not be split between records. Assume that argument 1, argument 2, argument 3, and argument 4 are contained in the first record. No part of argument 4 could overflow into the second record. Figure 105 illustrates the correct way to specify records containing arguments or functions.
8. The table may be ascending, descending, or in no sequence. If the LOKUP operation is to be performed on the table, and high or low indicators are specified, the table must be either ascending or descending. If an alternating table is used, normally the argument table will be in ascending sequence and the function table will have no sequence.
9. Alphameric entries must not exceed 248 characters. Numeric entries must not exceed 14 digits.
10. The records of a table must be on a sequentially organized file.
11. The table file to be loaded must contain the exact number of table entries as specified on the Extension Specifications form.
12. The record format for a table must be fixed length. However, there may be more than one table entry per record.
13. When tables are read from or written onto disk, each table is regarded as one file.
14. If multiple tables are read from a card reader, the same device name is used on the File Description Specifications form. However, the filenames must differ. The sequence of loading tables is based upon the sequence assigned on the Extension Specifications form.

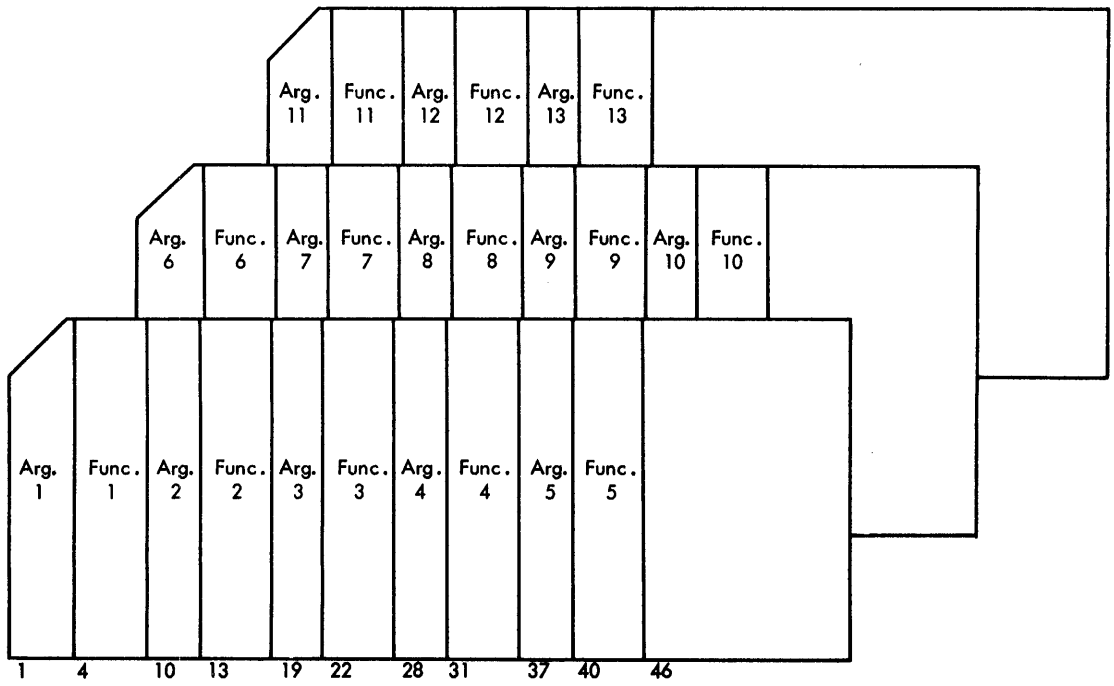


Figure 104. Sample Table File Containing Arguments and Functions

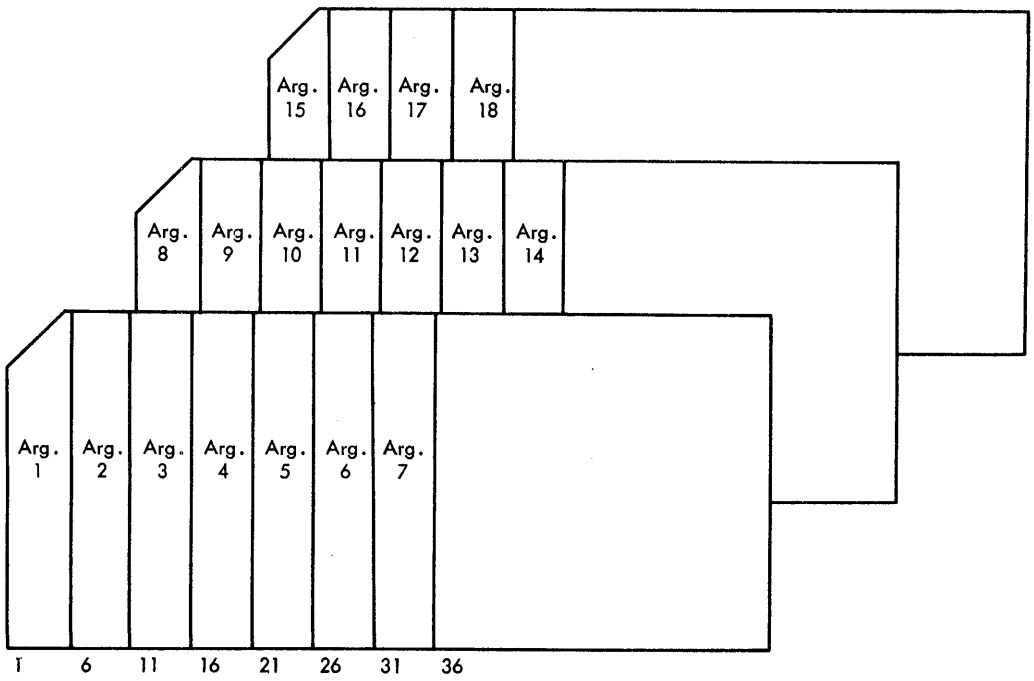


Figure 105. Sample Table File Containing Arguments Only

15. No more than eight lines for tables may be coded on the Extension Specifications form. If each table has an alternating table, there may be 16 different table names.
16. Numeric tables and numeric search arguments must contain the preferred signs: an F for plus and a D for minus.

METHODS OF PROCESSING TABLES

The operation code LOKUP entered on the Calculation Specifications form causes a table look-up operation to be performed.

Factor 1 contains the search argument. The search argument may be a literal or a field name. Factor 2 contains the name of the table which contains the arguments.

The length of the data in the argument table (table argument) must equal the length of the search argument. The length includes the decimal positions. Decimal alignment is performed.

If arguments and functions are used, the result field contains the name of the table where an associated function is located. The result field may be left blank if the user wants to determine if an argument is present in the table, but a corresponding function is not required.

When the table look-up operation is performed, resulting indicators (columns 54-59) must have an entry. The indicators indicate the type of look-up to be performed. They are turned on whenever the condition is satisfied. The program may search for the table argument next higher than the search argument (resulting indicator in columns 54-55). It may search for the table argument next lower than the search argument (resulting indicator in columns 56-57), or it may search for the table argument equal to the search argument (resulting indicator in columns 58-59). A high-equal or low-equal search may be specified by placing indicators in the appropriate two of the three fields (columns 54-59). Indicators cannot be specified in both the high and low columns.

Note: Indicators must not be placed in High or Low if the table is not in ascending or descending sequence.

The compare operations are logical for alphameric arguments and algebraic for numeric arguments. The search arguments must have the same format as the table entries compared with them; i.e., they must be numeric for numeric table entries, and alphameric for alphameric table entries. Decimal alignment is performed if numeric search arguments and table entries have differing decimal lengths.

The look-up operation is performed as follows:

1. The object program takes the field name or literal in Factor 1 and searches the table indicated by Factor 2. The kind of look-up is determined by the entries in the resulting indicators.
2. After the proper entry from the argument table has been found, the corresponding function from the function table (indicated by the entry in the result field) is located. Then, argument and function are placed in special holding areas for the function and argument tables. If the proper table argument is not found, the indicators in columns 54-59 are not turned on.

Data found by the table-lookup operation can be retrieved from the holding areas by using the name of the function table in either Factor 1 or Factor 2 of an operation.

Updating Tables

Calculations can be performed to change the argument or the function in the appropriate holding area as well as the table. This can be achieved by entering the defined name of the function or argument table in the Result Field and by entering an arithmetic operation code (such as ADD, SUB, MULT, DIV) or a move operation code (such as MOVE, MOVEL, MHHZO, etc.) in Operation. If a Blank After specification is entered for a table name, both the holding area and the specific table entry are blanked or zeroed out.

RPG returns the updated contents of the appropriate holding area to the location in core storage from where it was retrieved by the LOKUP operation. Table data can be updated only on an argument or the related function obtained by means of the last preceding LOKUP operation that refers to this table. Before updating, the programmer must ensure that an entry was found that satisfies the table search condition (specified by the resulting indicators). If no entry is found and updating occurs, the last entry found is updated.

Figure 106 illustrates several ways in which the data found by the LOKUP operation can be used. The numbers on the figure refer to the discussion that follows it.

1. Factor 1 identifies PERCNT as the field containing the search argument. The name of the table that contains the argument is TABCST. The name of the table that contains the corresponding function is TABAMT. The program will search for the value in the argument table that is equal to the search argument (columns 58-59).

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

RPG CALCULATION SPECIFICATIONS

Date _____ Punching Instruction: Graphic / Punch Page 40 Program Identification TABLES

Program _____

Programmer _____

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And / Not	And / Not	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	1				PERCNT	LOKUP	TABCST	TABAMT												15	
02	C	2		15		TABAMT	LOKUP	TABARG	TABFUN												10	
03	C	3		10			MOVE	TABFUN	WITHTX		52											
04	C	4				+25	LOKUP	TABFIL	TABLIT												20	
05	C			20			MOVE	+500	TABLIT													
06	C			20			MOVE	+30	TABFIL													
07	C	5				SEARCH	LOKUP	TABNUM													30	
08	C		N30				SETON									H1						
09	C	6		01		000	LOKUP	TABARG	TABFUN												35	
10	C			35	01		Z-ADD	NEWARG	TABARG													
11	C			35	01		MOVE	NEWFUN	TABFUN													
12	C																					

Figure 106. Using the LOKUP Operation Code

2. The function found in TABAMT from the previous operation is used as the search argument. TABAMT is the name of the table; however, Factor 1 is the function stored in the special holding area of the table TABAMT. The program searches for the value in the table TABARG that is equal to the search argument (columns 58-59).
3. The data obtained from the function table TABFUN from the previous look-up operation is moved to a field called WITHTX. It will be used for additional calculations.
4. The data found in the function and argument tables is updated. The literal +25 is the search argument. The table TABFIL is searched for +25 (columns 58-59). A new entry for the corresponding function of +25 is entered in TABLIT. The new function is +500; the new argument is +30. When updating a table, both the special holding area and the table entry are changed.
5. A look-up with only one argument table turns on indicator 30 if SEARCH is equal to an argument in TABNUM. If 30 is not on (N30), H1 is turned on by the SETON operation.
6. Entries are added to the table. The LOKUP operation is conditioned by indicator 01. (Indicator 01 is turned on when the input file contains records with additional table information. Each record contains the fields: NEWARG and NEWFUN.) To determine the first vacant argument, a field of zeros is used as the search argument. Zeros are used if the argument field is numeric. Blanks are used if the argument field is alphabetic. However, these zeros or blanks must be loaded as part of the table at the beginning of the job. If there is an equal compare, indicator 35 is turned on. Because the argument field of the table is vacant, the corresponding function field is also vacant. The new argument (NEWARG) is inserted in the TABARG field, and the function is inserted in the TABFUN field.

Figure 107 shows the actual contents of four tables. The Extension Specifications for entering them into a program are shown in Figure 108. TABLEC and TABLED are entered in an alternating format.

	First Entry	Second Entry	Third Entry	Fourth Entry	Fifth Entry
Table A	01	05	08	32	96
Table B	05.13	02.12	47.15	28.70	15.16
Table C	WWW	NNN	LLL	GGG	AAA
Table D	7	8	3	2	5

Figure 107. Contents of Four Tables

International Business Machines Corporation

EXTENSION AND LINE COUNTER SPECIFICATIONS

Punching Instruction: Graphic Punch — Page

Extension Specifications

Table Name	Number of Table Entries Per Record	Number of Entries Per Table	Length of Table Entry	Packed (P)	Decimal Positions	Table Sequence (A/D)	Table Name (Alternating Table)	Length of Table Entry	Packed (P)	Decimal Positions	Table Sequence (A/D)
TABLEA	2	5	2			A					
TABLEB	5	5	4	2							
TABLEC	1	5	3			D	TABLED	1	Ø		

Figure 108. Extension Specifications for Table Entries

The Calculation Specifications show various LOKUP possibilities with various indicator settings (see Figure 109).

Figure 110 describes the results of each of the LOKUP statements. The indicator is only set when the search is satisfied in the table. If the search is not satisfied, the table holding areas will contain the value of the last satisfied search.

Retrieving Tables

After a table has been updated, the table may be written or punched out for later use. On the File Description Specifications form, the programmer enters the name of the file that will contain the updated table. The file must be defined as a sequentially organized output file. On the Extension Specifications form, the programmer enters the name of the file on which the updated table will be placed under To Filename. The name of the table is entered in columns 27-32. If two tables are to be put out, the name of the second table is entered in columns 46-51. The updated table files will be put onto the output file after the program has reached the end-of-job condition (LR condition). This output file must have the same format and size as the input table file. If the updated table is to be printed, no automatic skip to a new page will be initiated by the RPG compiler. An overflow indicator must not be specified for a printed table file. No editing is possible for printing of tables. If editing is necessary, two approaches are possible:

1. The table is written onto disk and then read back in and edited in a second job.
2. The table is put out at LR time using the EXCPT operation and a loop on the Calculation Specifications.

IBM

International Business Machines Corporation

Form X21-9093 Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date

Program

Programmer

Punching Instruction: Graphic Punch — Page 40 — Program Identification: TABLES

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 / Minus / Zero; Compare High 1>2 / Low 1<2 / Equal 1=2; Lookup Table (Factor 2) is High / Low / Equal	Comments
01	C			'Ø8'	LOKUP TABLEA	TABLEB					Ø1	
02	C			'Ø8'	LOKUP TABLEA	TABLEB					Ø2	
03	C			'Ø8'	LOKUP TABLEA	TABLEB					Ø3	
04	C			'Ø8'	LOKUP TABLEA						Ø4 Ø5	
05	C			'Ø8'	LOKUP TABLEA						Ø6Ø7	
06	C			'ØØ'	LOKUP TABLEA						Ø8	
07	C			'97'	LOKUP TABLEA						Ø9	
08	C			'Ø9'	LOKUP TABLEA	TABLEC					10 11	
09	C			'Ø9'	LOKUP TABLEA	TABLEC					12 13	
10	C			'LLL'	LOKUP TABLEC	TABLED					14	
11	C			'JJJ'	LOKUP TABLEC						15	
12	C			'JJJ'	LOKUP TABLEC						16	
13	C			'JJJ'	LOKUP TABLEC	TABLED					17	

Figure 109. Table Lookup (LOKUP) Operations

Specification Line Number	Entry Found	Indicator On	Table Item Satisfying search Condition	Table Item Used From Related Table
01	yes	01	32	28.70
02	yes	02	05	02.12
03	yes	03	08	47.15
04	yes	05	08	
05	yes	06	05	
06	no			
07	no			
08	yes	10	32	GGG
09	yes	12	08	ILL
10	yes	14	NNN	8
11	yes	15	GGG	
12	no			
13	yes	17	LLL	3

Figure 110. Results of LOKUP Operations

EXAMPLE OF USING TABLES

Figures 111 and 112 illustrate an input data file, the way a table of alternating functions and arguments might appear, and the forms for a program that uses tables. In this example, a card input file contains the number of hours worked by each employee (columns 42-44) and the employee's number (columns 1-5). The RPG program takes the employee number and uses it as the search argument to find the salary rate for the employee. After the employee's rate has been determined, the rate is multiplied by the number of hours worked by the employee. The result of this operation is the amount earned for each employee. A description of the entries on the Specification forms in Figure 112 follows.

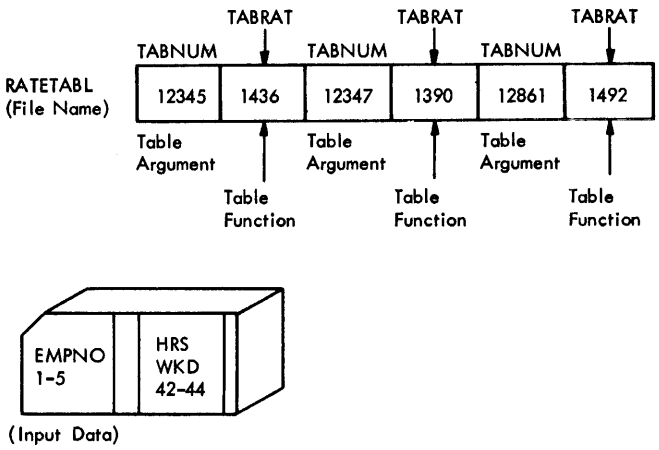


Figure 111. Table of Alternating Arguments and Functions

File Description Specifications Form

The file containing the input card records is called TIMECARD. It is an input file (I in column 15) and a primary file (P in column 16). When this file is depleted, processing is terminated (E in column 17). Each record is 80 characters long (columns 26-27). This file is entered from an IBM 2501 Card Reader.

The file that contains the arguments and functions is called RATETABL. It is an input table file (I in column 15, T in column 16). Each record is 80 characters long. The file is entered from the IBM 1442 Reader/Punch. Additional information about the file is on the Extension Specifications form.

Extension Specifications Form

This completes definition of the file RATETABL. The collection of arguments is called TABNUM (columns 27-32). There are eight arguments per record (columns 34-35) and 500 entries in the table (columns 36-39).

Sixty-three records are required for the table (500 ÷ 8 + 1). The first 62 records contain punches in columns 1-72 (5-position argument, plus 4-position function, times 8 entries per record). The 63rd record has only four entries and contains punches in columns 1-36.

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: RPT

Control Card Specifications

Line	Form Type	Core Size to Compile	Object Output	Listing Options	Core Size to Execute	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	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, or E)	Name of Label Exit	Extent Exit for DAM	File Addition	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind
		I/O/U/C	P/S/C/R/T	E	A/D	F/V				L/R		K/I	I/O/T or 1-9									A		N/U	
02	F	TIMECARDIPE				F			80							READ01									
03	F	RATETABLIT				F			80							EREAD42									
04	F																								

IBM

International Business Machines Corporation

Form X21-9091
Printed in U.S.A.

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page: 05

Program Identification: RPT

Extension Specifications

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table Name	Number of Table Entries Per Record	Number of Entries Per Table	Length of Table Entry	Packed (P)	Decimal Positions	Table Sequence (A/D)	Table Name (Alternating Table)	Length of Table Entry	Packed (P)	Decimal Positions	Table Sequence (A/D)	Comments
01	E			RATETABL		TABNUM	8	500	5			ØA	TABRAT	4		3		
02	E																	

IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page: 10

Program Identification: RPT

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	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	TIMECARDAA				01																									
02	I																				1	5		EMPNUM							
03	I																				42	44		1HRSWKD							

Figure 112. Alternating Arguments and Functions, Coding of (Part 1 of 2)

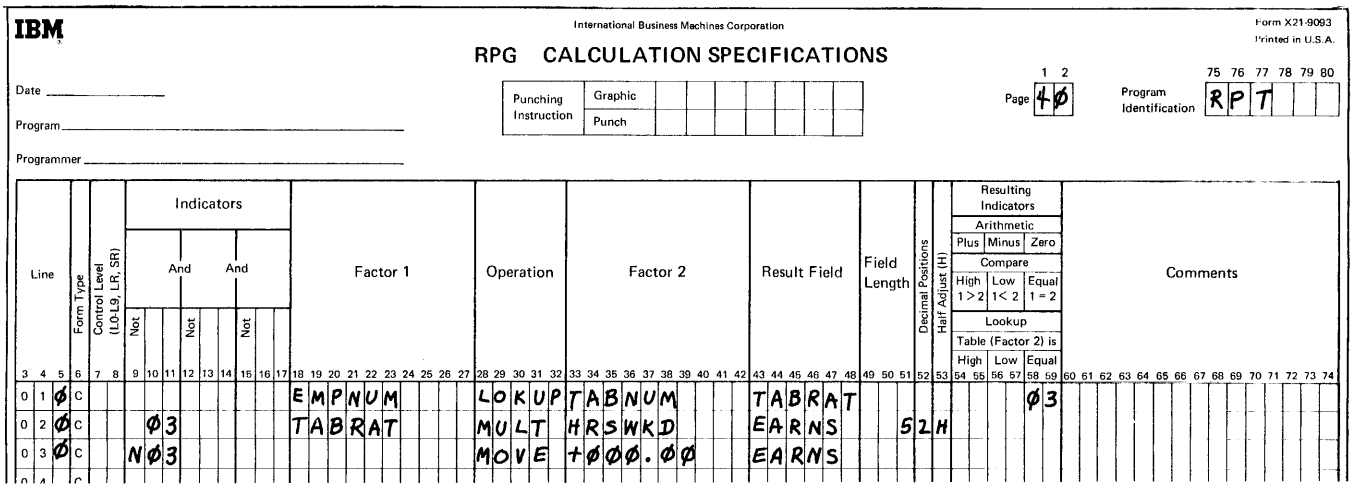


Figure 112. Alternating Arguments and Functions, Coding of (Part 2 of 2)

Each table entry is five positions long (column 42), and contains no decimal positions (column 44). The table is in ascending sequence (column 45).

The collection of functions is called TABRAT (columns 46-51). Each entry is four positions long (column 54), and contains three decimal positions (column 56).

Input Specifications Form

The input file (TIMECARD) is assigned a sequence of AA (columns 15-16). Record identifying indicator 01 is turned on whenever an input record is present for processing. No record identification codes are specified, because there is only one record type.

Lines 020 and 030 describe the locations of the two input fields used by the program. The employee number (EMPNUM) is located in columns 1-5 of the input record. The number of hours worked by the employee (HRSWKD) is found in columns 42-44 of the input record.

Calculation Specifications Form

On line 010, Factor 1 contains the search argument EMPNUM (employee number). LOKUP causes the lookup operation to be performed. Factor 2 contains the name of the collection of arguments (TABNUM) which is searched by the search argument. The result field contains the name of the collection of functions (TABRAT). Line 010 causes the employee number (EMPNUM) to be used as the search argument for the data contained in TABNUM. Indicator 03 is turned on when the program finds an entry in the argument table that is equal to the search argument. The entry in the argument table is placed in the TABRAT.

Line 020 is performed when indicator 03 is on. The rate for the employee (TABRAT, which is 4 positions long, with 3 decimal positions) is multiplied by the number of hours worked (HRSWKD, which is 3 positions long wtih 1 decimal position) and the result is stored (EARNS, which is 5 positions long with 2 decimal positions). The result is half-adjusted.

When the search argument does not find an equal entry in the argument table (indicator 03 is not on), line 030 is performed. The literal +000.00 is then moved to the field EARNS, specifying that the employee does not have an entry in the table.

EXAMPLE OF RETRIEVING TABLES

Occasionally, it is necessary to update a table during a program and to put out the table once (e.g., at LR time) or several times (e.g., at L1 time). If output is required only at LR time, it can be accomplished by letting RPG put out the table and specifying a second program to print the results.

Figure 113 shows a table being put out during a processing run. The same technique can be used at LR time. A card file shows records to be processed and totals accumulated for the items sold. The cards are in sequence but the items within each card are not in sequence. Consequently, a table-updating program has been choosen. The same job could be accomplished by sorting the cards by item and using control levels.

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: RPT

File Description Specifications

Line	Form Type	Filename	I/O/U/C	P/S/C/R/T	E	A/D	F/V	Block Length	Record Length	L/R	K/I	I/D/T or 1-9	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels (S, N, or E)	Name of Label Exit	Extent Exit for DAM	A	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	N/U
02	F	CARD	I	P			F		80								READ42									
03	F	TABL1	I	T			F		9							E	DISK									
04	F	TABL2	I	T			F		2							E	DISK									
05	F	PRINTER	O				F		120								PRINTER									

IBM

International Business Machines Corporation

Form X21-9091
Printed in U.S.A.

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page: 05

Program Identification: RPT

Extension Specifications

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table Name	Number of Table Entries Per Record	Number of Entries Per Table	Length of Table Entry	Packed (P)	Decimal Positions	Table Sequence (A/D)	Table Name (Alternating Table)	Length of Table Entry	Packed (P)	Decimal Positions	Table Sequence (A/D)	Comments
01	E			TABL1		TABCOD	1	20	3				TABAMT	6	2			
02	E			TABL2		TABNUM	1	20	2	0								

IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date _____

Program _____

Programmer _____

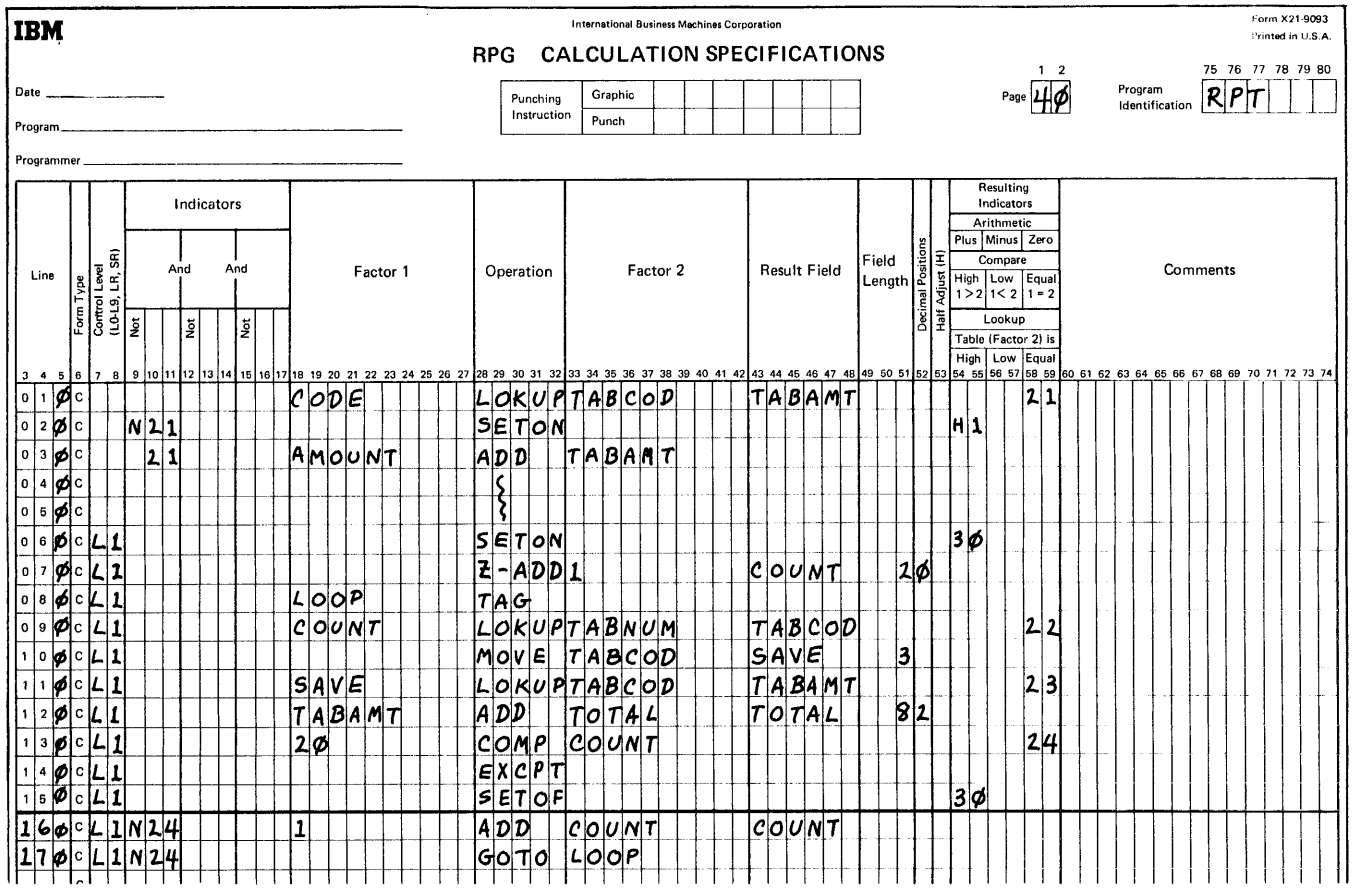
Punching Instruction: Graphic / Punch

Page: 10

Program Identification: RPT

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	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	CARD	AA			01																									
02	I																				1	2		SLSMANL1							
03	I																				3	5	0	CODE							
04	I																				6	9	2	AMOUNT							

Figure 113. Retrieving an Updated Table (Part 1 of 3)



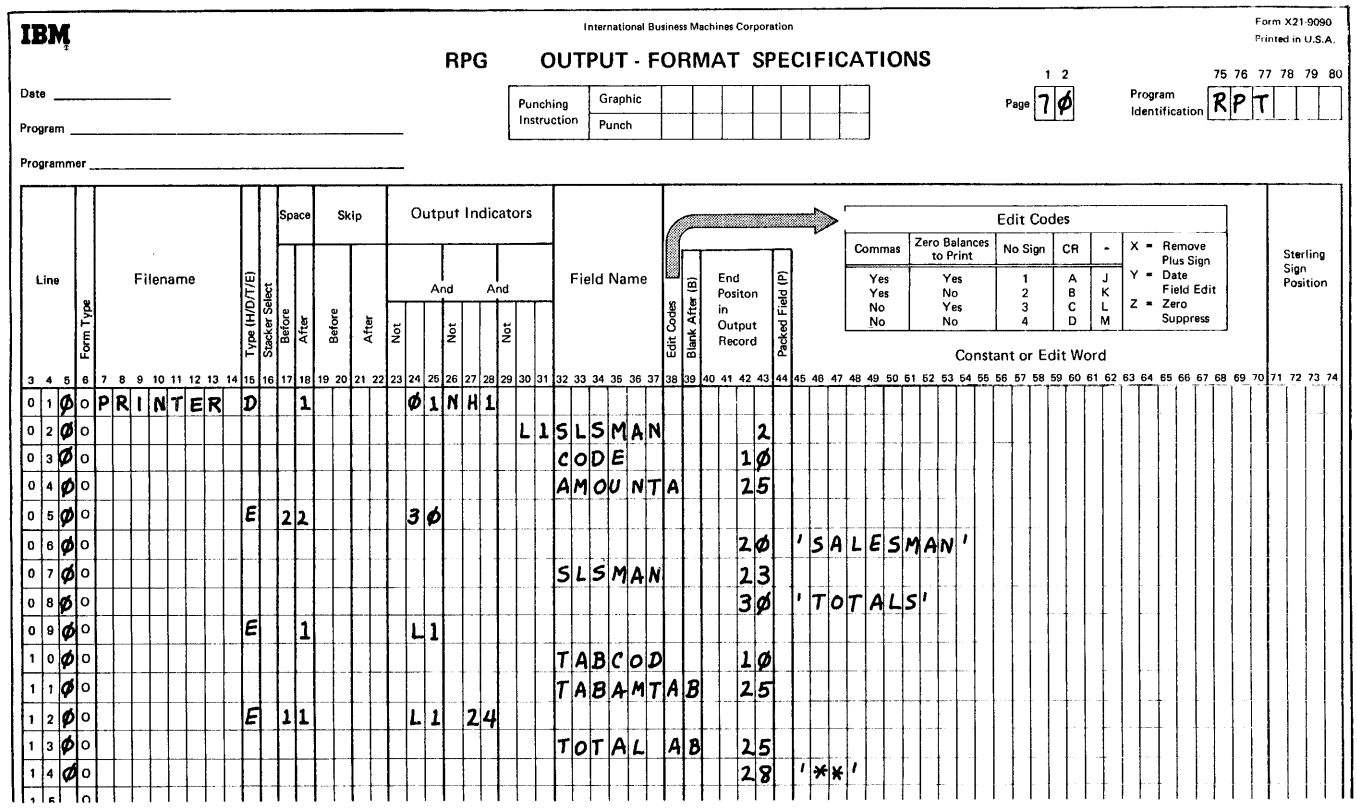


Figure 113. Retrieving an Updated Table (Part 2 of 3)

Beginning Values of Tables		
TABNUM	TABCOD	TABAMT
01	017	0000.00
02	018	0000.00
03	019	0000.00
04	041	0000.00
05	042	0000.00
06	060	0000.00
07	091	0000.00
08	092	0000.00
09	093	0000.00
10	094	0000.00
11	101	0000.00
12	102	0000.00
13	175	0000.00
14	176	0000.00
15	201	0000.00
16	202	0000.00
17	301	0000.00
18	302	0000.00
19	401	0000.00
20	472	0000.00

Figure 113. Retrieving an Updated Table (Part 3 of 3)

The *File Description* entries show two table files on disk and a card and printer file. The Extension Specifications shows TABL1 as being an alternating table containing TABCOD (the code numbers of the items sold) and TABAMT (initially loaded as zeros and updated during the program). TABL1 contains twenty entries, one for each of the items.

TABL2 also has twenty entries and is made up of consecutive numbers 01 through 20. This table will be used to assist in producing output.

The Input Specifications describe the item card and provide a level 1 control break on salesman number.

The Calculation Specifications show three detail calculations. Line 010 does a lookup of the field CODE of the item card against TABCOD to find the same item in the table. The result field of TABAMT denotes that the corresponding amount is also made available.

Indicator 21 is set on if the search produces an equal entry. Line 020 sets on H1 if the entry is not found. Line 030 adds the field AMOUNT from the input card to the table TABAMT if indicator 21 is on.

This process continues until all of the salesman cards have been read and a level – 1 control break occurs.

The total calculations start at line 060 where indicator 30 is set on. This will be used to produce a heading line prior to the output of the table. Line 070 sets up the field COUNT to be equal to 1 so it can be used to search for the first entry in TABCOD.

Line 090 does a lookup of TABNUM and produces the first entry of TABCOD. Line 100 moves the first entry in TABCOD to a saved area. Line 110 takes the saved area and looks for the same entry in TABCOD. The result field of TABAMT produces the corresponding entry.

The programmer now has the data needed to print the first summary line. Line 120 adds the amount in TABAMT to a total field. Line 130 determines if the field COUNT has reached 20. When it reaches 20, the last item in the table is ready to be printed.

Line 140 branches to exception output. Line 150 sets off indicator 30 to prevent repeated printing of the heading line.

Lines 160 and 170 are performed if the field COUNT has not yet reached 20. The branch to LOOP starts the process over again. Count equals two the second time through LOOP. This causes the second item number to be retrieved from the table and its amount to be printed.

The Output-Format Specifications show a detail output of the input card in lines 010-040. The SLSMAN field is printed on the first line following an L1 control break.

Lines 050-080 show the heading line which is to be printed just prior to printing the first item. Indicator 30 was set on to allow this, then set off after the first exception output was called for.

Lines 090-110 show the output required for each item. TABCOD and TABAMT are printed to provide the proper information. TABAMT is blanked-after (zeroed out) to allow the proper total to be accumulated for the next salesman.

Lines 120-140 provide for the total amount of each salesman to be printed and a double asterisk to denote it. The field TOTAL is also blanked-after to allow for the proper accumulation for the next salesman.

An RPG program may use subroutines to avoid repetitious coding of frequently used procedures. These subroutines are of two types:

Internal: is a subroutine written in the RPG language. This type of subroutine is generated and executed as a part of the RPG program. It is defined by the operations, BEGSR and ENDSR, and is invoked by the EXSR operation. Since the subroutine is a part of the RPG program, all field names defined in the main program are available to the subroutine and vice-versa.

External: is a subroutine written in the Assembler Language. This type of subroutine must be assembled and stored in the System Subroutine Library prior to its use. This subroutine is invoked from an RPG program by the EXIT operation, and RLABL operation.

INTERNAL SUBROUTINES

Internal subroutines are written in RPG. They are a part of the RPG program which invokes them. They are coded on the Calculation Specifications form using EXSR, BEGSR, and ENDSR operations.

An EXSR statement passes control to the RPG subroutine named in Factor 2 of that EXSR statement. EXSR statements may appear in detail calculations, total calculations, or within another subroutine. However, when EXSR is contained in a subroutine, it may not invoke the subroutine of which it is a part.

A BEGSR statement defines the beginning of an internal subroutine. Factor 1 of that statement contains the name (entry point) of the internal subroutine. This name is used in EXSR statements to pass control to the subroutine.

An ENDSR statement defines the end of the subroutine and the point of exit from the subroutine. At the completion of the subroutine execution, control is returned to the calculation specification immediately following the EXSR statement that invoked the subroutine.

ENDSR may have a name entered in Factor 1. This name serves as the destination for a GOTO statement within the subroutine. A TAG statement is not needed when the GOTO operation is used with the ENDSR statement.

Order of Specification: RPG subroutines follow all detail and total calculations on the Calculation Specifications form.

Order of Execution: Although the RPG subroutine specifications follow all other calculation specifications, they are executed according to the order and placement of the EXSR statements that invoke them. EXSR may be used anywhere in the calculation specifications. However, the user should consider the following:

1. When the EXSR operation is the first detail calculation of the program, control will be transferred to the subroutine upon completion of the input routine; i.e., as soon as the pertinent input data is available for processing.
2. When the EXSR operation is the last detail calculation of the program, control will be transferred to the designated subroutine immediately before heading and detail records are printed or punched.
3. When the EXSR operation is the first total calculation of the program, control will be transferred to the subroutine after the record type has been determined and the control field has been tested.
4. When the EXSR operation is the last total calculation, control will be transferred to the subroutine immediately before the totals are printed or punched.

Since the subroutine is internal to RPG, any data field may be referred to in the subroutine just as with any calculation specifications.

A GOTO operation may not be used to branch to the label of the BEGSR operation.

If a TAG operation is located within a subroutine, the corresponding GOTO must be within the same subroutine.

A GOTO statement within a subroutine may branch to a TAG statement outside of the subroutine. However, that TAG statement cannot be in another subroutine. *Example:* Figure 114 shows an updating routine using an RPG subroutine. This approach is typical of most "spread-card" type applications.

The three files involved are described on the File Description Specifications form.

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: SPREAD

Control Card Specifications

Line	Form Type	Core Size to Compile	Object Output	Listing Options	Core Size to Execute	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	Refer to the specific System Reference Library manual for actual entries.
01	H														

File Description Specifications

Line	Form Type	Filename	I/O/U/C	P/S/C/R/T	E	A/D	F/V	Block Length	Record Length	L/R	K/I	I/D/T or 1-9	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels (S, N, or E)	Name of Label Exit	Extent Exit for DAM	A	Number of Tracks for Cylinder Overflow	Number of Extents	N/U	Tape Rewind
02	F	CARD	I	P			F		80								READ	42								
03	F	INVFIL	I	C			F		100	R	3	K	1			1	DISK									
04	F	PRINTER	O				F		120								PRINTER									

IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date _____

Program _____

Programmer _____

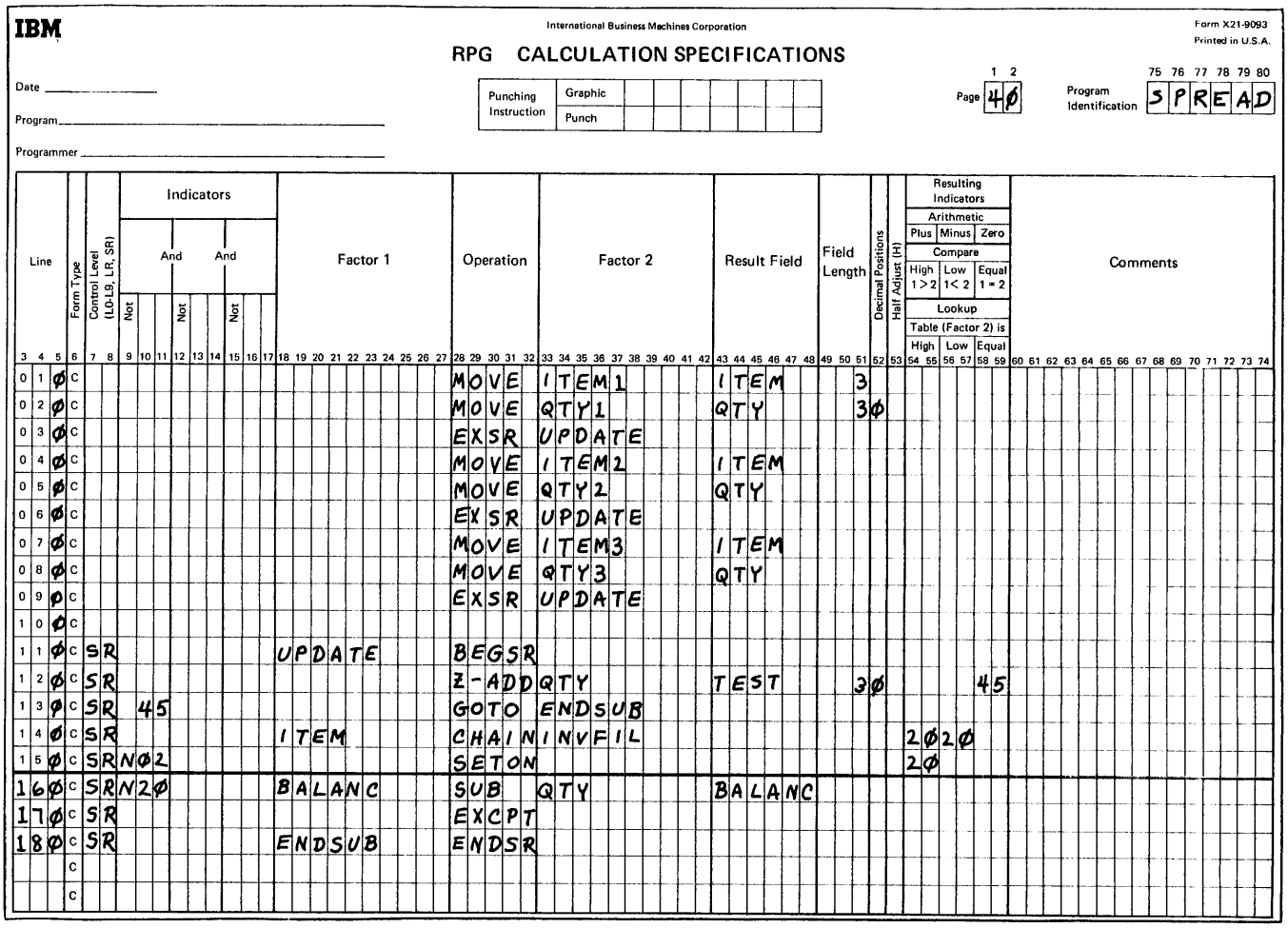
Punching Instruction: Graphic / Punch

Page: 10

Program Identification: SPREAD

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	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	CARD	AA			01																									
02	I	* HEADING PORTION																													
03	I																				1	5		CUST							
04	I	* SPREAD PORTION																													
05	I																				6	8		ITEM1							
06	I																				9	10		QTY1							
07	I																				11	14		ITEM2							
08	I																				15	17		QTY2							
09	I																				18	20		ITEM3							
10	I																				21	23		QTY3							
11	I																														
12	I																														
13	I	INVFIL	AB			02	100		C1																						
14	I																				1	3		KEY							
15	I																				4	7		0BALANC							
16	I		AC			03																									

Figure 114. Using an Internal (RPG) Subroutine (Part 1 of 2)



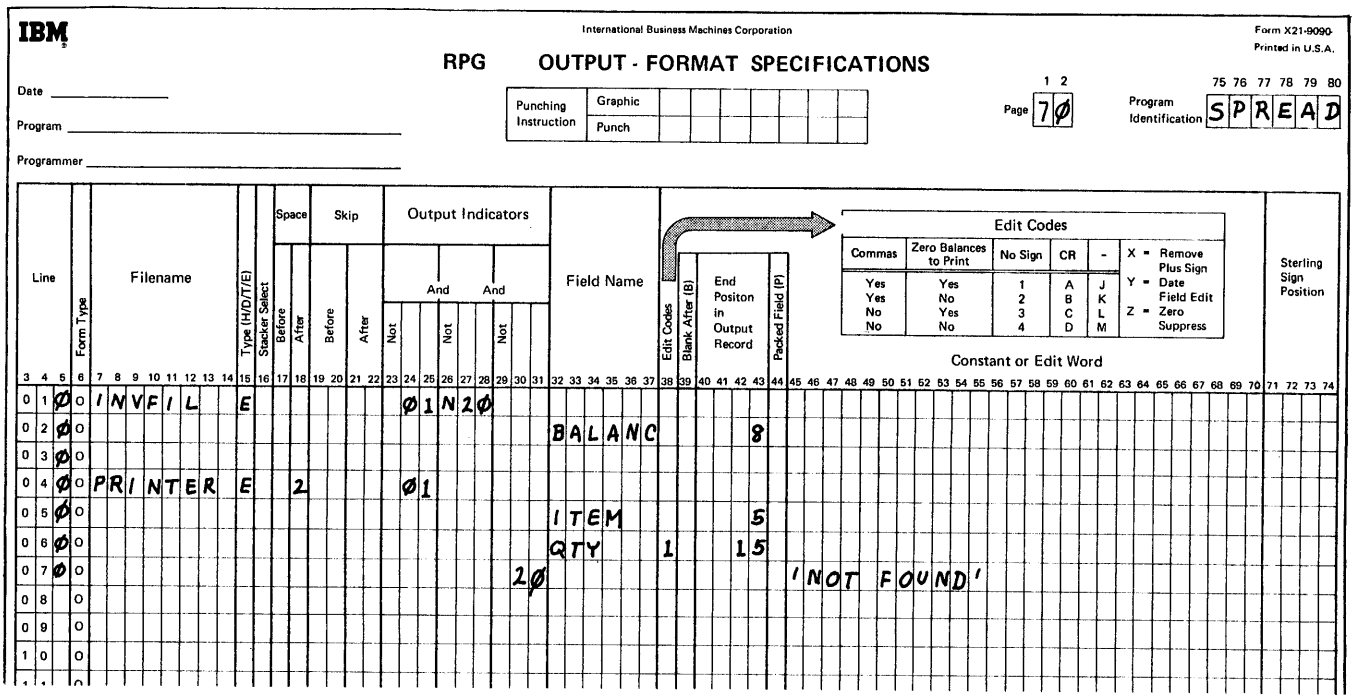


Figure 114. Using an Internal (RPG) Subroutine (Part 2 of 2)

The two input files are described on the Input Specifications form. The fields ITEM1, ITEM2, ITEM3 are defined for part numbers. The fields QTY1, QTY2, QTY3 are defined for the quantity of parts for the primary file CARD. The inventory file (INVFIL) key and balance fields are also defined.

The specifications necessary to use the subroutine are contained on the Calculation Specifications form. A description of the entries by line number follows:

Lines 010-030 — The fields ITEM and QTY are defined and the first data item is moved into these fields. The subroutine UPDATE is invoked.

Lines 040-090 — Information for the second and third data items is used to invoke the subroutine UPDATE.

Line 110 — The beginning of the subroutine UPDATE is identified by BEGSR.

Lines 120-130 — If no parts (QTY) have been issued, return control to the statement immediately following the EXSR statement.

Line 140 — The CHAIN operation is used with field ITEM and the file INVFIL to access the inventory file record for this item (See *Chaining*). Indicator 20 will be turned on if the appropriate record cannot be found.

Line 150 — The Inventory File contains both active and inactive items. The active items are coded as a 1 in position 100 of the record. This turns on indicator 02. If an inactive item is encountered, indicator 20 is turned on.

Line 160 — When indicator 20 is off the inventory file record is updated by the field QTY.

Line 170 — EXCPT causes any exception records defined on the Output-Format Specifications form to be produced at this time. (Exception records are identified by an E in column 15). In this example, the record produced is the inventory record containing the updated quantity. This record is written back into the file INVFIL, completing the update.

Line 180 — ENDSR defines the end of the subroutine and causes control to return to the statement immediately following the EXSR statement.

The output specification necessary for putting out the exception records is contained on the Output-Format Specifications form.

EXTERNAL SUBROUTINES

The EXIT operation on the Calculation Specifications form causes RPG to transfer control from the RPG object program to a subroutine that has been coded in the Assembler language. A subroutine might be a standard routine (such as the indexing subroutine shown in Appendix E) or it might be a routine that performs a function not easily accomplished using RPG (such as a square-root computation). The subroutine, written in Assembler language, is coded by the user. Entries made on the Calculation Specifications form enable the programmer to:

1. Exit from the RPG program to a subroutine.
2. Execute the subroutine.
3. Return to the main program after the subroutine has been executed.

The use of the EXIT operation and the understanding of this section requires thorough familiarity with the 1130 Assembler language as described in *IBM 1130 Assembler Language,* Form C26-5927.

A control-level entry and indicators (columns 7 through 17 of the Calculation Specifications form) can be used with the EXIT operation. When this is done, the EXIT operation is executed only if the conditions established by the control-level entry and/or indicators are satisfied. An EXIT operation whose execution depends on a control-level entry or on indicators is called a conditional EXIT operation. An EXIT can also be used within an internal RPG subroutine.

Columns 7 through 17 are used only with conditional EXIT operations. The mnemonic EXIT must be entered in columns 28-31. The name of the subroutine to which control is to be transferred must be specified in columns 33-37. The name of a subroutine is restricted to five characters. The first character must be alphabetic; the other characters may be alphabetic or numeric. Special characters or embedded blanks must not be used. The subroutine must be present in the System Subroutine Library prior to execution of the RPG program. *Position:* The

EXIT operation may be used anywhere in the Calculation Specifications. However, the user should consider the following:

1. When the EXIT operation is the first detail calculation of the program, control will be transferred to the subroutine upon completion of the input routine; i.e., as soon as the pertinent input data is available for processing.
2. When the EXIT operation is the last detail calculation of the program, control will be transferred to the subroutine immediately before heading and detail records are printed or punched.
3. When the EXIT operation is the first total calculation of the program, control will be transferred to the subroutine after the record type has been determined and the control field has been tested.
4. When the EXIT operation is the last total calculation, control will be transferred to the subroutine immediately before the totals are printed or punched.

Field Definition: Any fields that are used in both the subroutine and the RPG program must be defined in the RPG program. This is achieved by RLABL statements, as follows:

1. If the field is not used as a result field elsehwere in the program, or if it has not been defined in the Input Specifications form, it must be defined on the Calculation Specifications form. The symbol RLABL must be placed in columns 28-32. The name of the field is placed in the Result Field (columns 43-48). The length of a field name used in an RLABL statement must not exceed six characters. The first character must be alphabetic. Field Length and Decimal Positions must be provided.
2. If an RPG field that is to be used in a subroutine has been defined in the Input Specificaitons form or in the result field of another Calculation Specification form, RLABL must be placed in Operation and the name of the field in Result Field. Field Length and Decimal Positions should be left blank.

A field whose name has been defined by means of a TAG specification must not be used in a subroutine. Consequently, the name of such a field cannot be used in an RLABL specification. RLABL specifications must immediately follow the EXIT specification with which they are associated.

Indicator Definition: The user may want to test RPG indicators in a subroutine. In this case, each indicator required must be defined in an RLABL statement on the Calculation Specifications form.

The symbol RLABL must be in columns 28-32. The entry in columns 43-46 of the Result Field consists of the letters INxx, where "xx" is the symbol for the indicator. For example, the condition of the Matching Record indicator is to be tested in the subroutine, the entry in the Result Field portion of the appropriate RLABL statement is INMR. In the subroutine, the indicator may be referred to as the data located at INxx.

If the user sets on, sets off, or tests resulting indicators in the subroutine, he must observe the following rules when he codes his subroutine:

1. To set on a resulting indicator, set the data located in INxx to hexadecimal 0001.
2. To set off a resulting indicator, set the data located at INxx to hexadecimal 0000.
3. To test resulting indicators:

   - *If on,* the data at INxx will be hexadecimal 0001.
   - *If off,* the data at INxx will be hexadecimal 0000.

RLABL statements must be specified immediately following the EXIT statement which refers to the subroutine in which the desired fields are to be referenced.

If the same routine must be given control at various points in the RPG source program, the EXIT and RLABL statements should be placed in an RPG internal subroutine.

EXIT causes an assembler CALL statement to be generated. Each RLABL statement causes the compiler to generate the address of the referenced field or indicator.

The address of the field is actually the address of the control word of the field. This control word can be used to determine field length and to determine whether the field is numeric or alphameric. For additional information, refer to the section, *Data Records.*

Coding of Subroutines: All subroutines to be incorporated into RPG programs by means of the EXIT operation must be coded in Assembler language. The following must be observed:

1. RPG passes control to a subroutine via the assembler CALL statement. The Core Load Builder changes the CALL to a BSI indirect to the subroutines. Refer to *IBM 1130 Assembler Language,* Form C26-5927.
2. The CALL statement places the address following the CALL statement in the first word at the subroutine entry point. This address, plus 2, points to a list of RLABL referenced field addresses. The subroutine

can reference the desired field by using the supplied address.

3. The subroutine must return to the address following the CALL statement. The user's subroutine must return to the address contained in the first word of that subroutine.

4. Index register 3 is used by the system to pass control to subroutines. If the subroutine uses this register, it must save and restore its contents.

All subroutines to be incorporated in an RPG program must be assembled separately using the 1130 Assembler. The resulting object program deck must be stored in the System Subroutine Library with a // DUP function (refer to *IBM Disk Monitor System, Version 2, Programming and Operator's Guide*, Form C26-3717).

Since the subroutines and the RPG program are to be combined, the subroutines must be relocatable and all Assembler linking conventions must be observed.

Restrictions: The following restrictions must be observed when using subroutines written in Assembler language.

1. Control cannot be transferred from one subroutine to another.
2. Subroutines should not contain input/output operations. This may interfere with the RPG routines.
3. The control word of an RPG field must not be altered by an external subroutine. *Example:* Figure 115 illustrates the EXIT and RLABL operations. The RPG calculation specifications show an EXIT to the subroutine PROC which refers to indicator 01 and field AFLD.

IBM

IBM 1130 Assembler
Coding Form

Form X26 5994
Printed in U.S.A.

Program _____ Date _____

Programmed by _____ Page No. _____ of _____

Label (21–25)	Operation (27–30)	F (32)	T (33)	Operands & Remarks (35–70)	Identification (75–80)
	ENT			PROC1	
PROC1	DC			*-* TO CONTAIN RETURN ADDRESS	
	STX		1	SAVE+1 SAVE INDX REG 1	
	LD			PROC1 SAVE RETURN ADDRESS	
	STO			RETRN+1	
	MDX	L		PROC1,,+2 INCREMENT ADDR TO PARMS	
	LDX	1	1	PROC1 GET ADDRESS	
	STX		1	A+1 OF	
A	LDX	1	1	*-* INØ1 IN INDX REG 1	
	.				
	MDX	L		PROC1,,+1 STEP TO NEXT PARM	
	LDX	1	1	PROC1 GET ADDRESS	
	STX		1	B+1 OF	
B	LDX	1	1	PROC1 AFLD IN INDX REG 1	
	.				
SAVE	LDX	L	1	*-* RESTORE INDX REG 1	
RETRN	BSC	L		*-* RETURN TO RPG	
	.				
	END				

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page (1–2)

Program Identification (75–80)

Line (3–5)	Form Type (6)	Control Level (L0-L9, LR, SR) (7–8)	Indicators: And (9–12) Not	Indicators: And (13–16) Not	Indicators: Not (17)	Factor 1 (18–27)	Operation (28–32)	Factor 2 (33–42)	Result Field (43–48)	Field Length (49–51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Arithmetic Plus (54–55)	Arithmetic Minus (56–57)	Arithmetic Zero (58–59)	Compare High 1>2 (54–55)	Compare Low 1<2 (56–57)	Compare Equal 1=2 (58–59)	Lookup Table (Factor 2) is High (54–55)	Lookup Table (Factor 2) is Low (56–57)	Lookup Table (Factor 2) is Equal (58–59)	Comments (60–74)
010	C						•															
020	C		Ø1				EXIT	PROC1														
030	C						RLABL		INØ1													
040	C						RLABL		AFLD													
050	C						•															

Figure 115. External Subroutine

The RPG object program handles records that are fixed-length only (i.e., all the records of a file have the same length). One logical record is contained in one physical record.

UNBLOCKED RECORDS

All logical records of a file have the same length.

The length of a logical record is entered in Record Length on the File Description Specifications form.

BLOCKING RECORDS

RPG automatically blocks all disk files for maximum space utilization.

A disk pack consists of fixed-length sectors with a capacity of 640 characters each. A data record must be contained in one sector. However, a sector may be capable of containing more than one record. The process of grouping records on a sector is called blocking.

The blocking of disk records saves disk operations and increases the processing speed of the object program. When records are grouped in blocks of one sector, an unused area may occur at the end of the sector. For additional information, see the section, *Disk File Space Calculation.*

Object Time Format of Data Fields. All data at object time will be held in core storage in the following format:

1. LLdd $00Z_1D_1$ $00Z_2D_2$ $00Z_3D_3$ $00Z_4D_4$ . . . . . $00Z_nD_n$

   where:

   LL = 8 bits representing n-1 in binary
   dd = /FF for alphameric fields
   = 8 bits representing the number of digits to the right of the decimal point for numeric fields
   n = maximum of 14 for numeric fields, maximum of 256 for alphameric fields
   Z = any hexadecimal number from /0 to /F
   D = any hexadecimal number from /0 to /F

Operations on numeric fields will recognize only the D's and will consider Z_n as the sign of the number. Z_n=/D will be recognized as negative. Z_n= any other hexadecimal number will be considered positive. The result of an arithmetic operation (Z-ADD, Z-SUB, ADD, SUB, MULT, DIV) will force /F for all Z's except Z_n=/D for negative results.

Numeric literals in the source program will be stored with Z/s = /F. Fields in an I/O area will have the following format:

$Z_1D_1Z_2D_2 \ldots . Z_nD_n$

The Z_1D_1 may occupy either the first half or second half of an 1130 word.

An exception to the above occurs when P (packed) is specified for a disk file. The following is the packed format of a numeric field in the I/O area:

$D_1D_2D_3D_4 \ldots . . D_nZ$

Z is the sign of the number. The pair D_nZ may occupy either the first half or second half of a word. This format requires a leading /0 for padding for fields whose length is even. When this format is unpacked to the form in 1 above, Z's = /F should be inserted.

Comments, comment statements, and the Indicator Summary Form can be used to provide documentation that will assist in writing, testing, and maintaining RPG programs.

Comments are explanatory notes written in the comments sections of the Extension and Calculation Specifications forms (Figures 116 and 117). These comments may describe the function performed by the statements to the left of them. They have no effect on the processing of the program, and are merely printed on the source program listing.

Comment statements are explanatory sentences or notes that can be written on any specifications form. They are preceded by an asterisk (*) in column 7. Among other items, they can be used at the beginning of the program to describe functions of that program (see Figure 118), or they can be used to denote the beginning of major sections of a program (Figure 119).

The Indicator Summary Form (see Figure 120) can be used to keep track of the indicators used in a program and to describe their purpose. The form provides a convenient aid when writing a program and a good means of providing documentation. Entries on the form can be punched into cards. The cards are normally placed in the program after the File Description Specifications. Entries on the Indicator Summary Form are treated as comment statements; they have no effect upon program processing and are merely printed in the source program listing.

IBM International Business Machines Corporation — Form X21-9091 Printed in U.S.A.

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

Date _____ Program _____ Programmer _____

Punching Instruction: Graphic / Punch

Page 05 Program Identification COMENT

Extension Specifications

Line	Form Type	Record Sequence of the Chaining File / Number of the Chaining Field / From Filename	To Filename	Table Name	Number of Table Entries Per Record	Number of Entries Per Table	Length of Table Entry	Packed (P)	Decimal Positions	Table Sequence (A/D)	Table Name (Alternating Table)	Length of Table Entry	Packed (P)	Decimal Positions	Table Sequence (A/D)	Comments
3 4 5	6	7 8 9 10 11 12 13 14 15 16 17 18	19 20 21 22 23 24 25 26	27 28 29 30 31 32	33 34 35	36 37 38 39	40 41 42	43	44	45	46 47 48 49 50 51	52 53 54	55	56	57	58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
01 Ø	E	TABL1		TABMAN	1	75	3				TABRAT	4	2			MAN AND RATE
02 Ø	E	TABL2		TABCOD	1	25	2									DEPARTMENT CODES
03 Ø	E	TABL3		TABDAY	1	7	1				TABHRS	6	2			DAY AND HOURS
04	E															

Figure 116. Program Documentation

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic								
	Punch								

Page 40

Program Identification COMENT

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 / Minus / Zero); Compare (High 1>2 / Low 1<2 / Equal 1=2); Lookup (Table (Factor 2) is High / Low / Equal)	Comments
01	C			ONHAND	SUB	QTY	TEST		60		20	TEST QTY
02	C		20		GOTO	END						NOT ENOUGH
03	C				Z-ADD	TEST	ONHAND					
04	C			MINBAL	SUB	QTY	TEST				21	MIN BAL TEST
05	C		21		EXCPT							PUNCH NOTICE
06	C											

Figure 117. Program Documentation

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 PAY050

Control Card Specifications

Line	Form Type	Core Size to Compile	Object Output	Listing Options	Core Size to Execute	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	Refer to the specific System Reference Library manual for actual entries.
01	H														

File Description Specifications

Line	Form Type	Filename	I/O/U/C	P/S/C/R/T	E	A/D	F/V	Block Length	Record Length	L/R	K/I	I/D/T or 1-9	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels (S, N, or E)	Name of Label Exit	Extent Exit for DAM	File Addition: A	File Addition: Number of Tracks for Cylinder Overflow	File Addition: Number of Extents	File Addition: Tape Rewind N/U
02	F												PAY050										
03	F																						
04	F	PAYROLL DEDUCTION REGISTER																					
05	F																						
06	F	THIS PROGRAM TAKES THE OUTPUT OF PAY050 AND LISTS IT																					
07	F	AS THE DEDUCTION REGISTER. GROSS PAY SHOULD BALANCE																					
08	F	WITH PAY030. DEDUCTIONS SHOULD CROSSFOOT AND AN																					
09	F	ERROR MESSAGE IS PRINTED IF THEY DO NOT.																					
02	F																						
03	F																						
04	F	DISKIN	IP			F			100						DISK								
05	F	PRINTER	O			F			120				OF		PRINTER								

Figure 118. Comment Cards on a Specifications Form

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic							
	Punch							

Page 40 (1 2)

Program Identification COMENT (75 76 77 78 79 80)

Line (3 4 5)	Form Type (6)	Control Level (L0-L9, LR, SR) (7 8)	Indicators: And / Not (9 10 11)	Indicators: And / Not (12 13 14)	Indicators: And / Not (15 16 17)	Factor 1 (18 19 20 21 22 23 24 25 26 27)	Operation (28 29 30 31 32)	Factor 2 (33 34 35 36 37 38 39 40 41 42)	Result Field (43 44 45 46 47 48)	Field Length (49 50 51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Arithmetic Plus / Minus / Zero; Compare High 1>2 / Low 1<2 / Equal 1=2; Lookup Table (Factor 2) is High / Low / Equal (54 55 56 57 58 59)	Comments (60 61 62 63 64 65 66 67 68 69 70 71 72 73 74)
01	C	X				DETAIL CARD								
02	C													
03	C													
04	C													
05	C													
06	C													
07	C	X				FEDERAL WITHHOLDING TAX								
08	C													
09	C													
10	C													
11	C	X				FICA CALCULATION								
12	C													
13	C													
14	C													
15	C	X				NET PAY								
	C													
	C													
	C													
	C													
	C													

Figure 119. Program Documentation

IBM

International Business Machines Corporation

Form X21-9095
Printed in U.S.A.

RPG INDICATOR SUMMARY

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic							
	Punch							

Page 02

Program Identification PAY050

Circle Indicators Used:

Note: All indicators are not valid with all systems.

(01)	(02)	(03)	(04)	05	06	07	08	09	10	(11)	(12)	(13)	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	
(L1)	(L2)	L3	L4	L5	L6	L7	L8	L9		OA	OB	OC	OD	OE	OF	OG	OV		
(H1)	H2	H3	H4	H5	H6	H7	H8	H9		U1	U2	U3	U4	U5	U6	U7	U8		

Predefined Indicators: IP LO (LR) MR H0

M-C: M1 M2 M3 M4 M5 M6 M7 M8 M9 C1 C2 C3 C4 C5 C6 C7 C8 C9

Line	Form Type	Record Identifying	Input Field	Calculation Result	Matching and Chaining	Control Level, Overflow, Halt and User	
010	F	ID	F	C	M	L	FUNCTION OF INDICATORS
020	F	01					MASTER RECORD
030	F	02					CURRENT EARNINGS
040	F	03					YTD SUMMARY
050	F	04					DEDUCTIONS
060	F			11			CROSSFOOT OF MAN IS EQUAL
070	F			12			FICA CUTOFF REACHED
080	F			13			TOO MANY DEDUCTIONS
090	F					L1	MAN BREAK
100	F					L2	DEPARTMENT BREAK
110	F		H1				NO DATE CARD
120	F						
130	F						
140	F						
150	F						
	F						
	F						
	F						

Figure 120. Using the Indicator Summary Form

1130 RPG does not change the sign of any fields defined as numeric in the following situations:

- Removed from an Input Record
- Moved by a MOVE or MOVEL operation
- Moved by a Move Zone operation (the sign of the result field will be the same as the sign of Factor 2)
- Used as a Chaining Field

The result field of any arithmetic operation (ADD, SUB, Z-ADD, Z-SUB, MULT, DIV, MVR) always produces either the positive sign (F) or the negative sign (D).

If two numeric fields are compared and the digit portions are equal, the following occurs:

1. If the signs are identical, the compare is equal.
2. If one sign is a D (negative) and the other is not a D (positive), the compare is unequal.
3. If the two signs involved are not D's and are not the same (e.g., F and C), the compare is equal.

Numeric Match Fields and Control Level Hold Areas

An F sign is forced on all numeric fields regardless of what the sign was previously. In addition, all high-order zones are removed.

Editing

Edit codes and edit words treat all D signs as negative. All other signs are considered positive.

Chaining

A numeric key field used to create an indexed sequential file can have any sign. Normally, the sign is an F. If the user has a chaining field, it must contain the same sign as the record on disk. Otherwise, the record will not be found. Normally, the input field will also contain an F sign. But if the units position has a 12-punch, a C zone is created and the record will not be found.

This section contains a general discussion of file organization and file processing methods, and describes how to calculate the amount of storage needed for a file. For a more detailed discussion of the 1130 disk, refer to the publication *IBM 1130 Disk Monitor System, Version 2: Programming and Operator's Guide,* Form C26-3717.

The distinction between the terms file organization and file processing is important in understanding disk storage concepts.

File organization is the method of arranging *data records* on a direct access storage device; that is, building the file.

File processing is the method of *retrieving* data records from the file; that is, using the file.

FILE ORGANIZATION

The two types of file organization available for use with 1130 RPG are *sequential* and *indexed sequential.* The method best suited to a particular file of disk records depends upon many factors. For example: Are records to be added to a file after it is built? Records can be added to an ISAM file but a sequentially organized file must be rebuilt. What is the best way to process disk files? RPG provides four methods of file processing. In addition, records within a file might be processed at random during an updating run, and sequentially during a billing run. All these factors must be analyzed for each file in each particular application before the type of file organization is selected.

Sequential File Organization

A sequentially organized file is one in which records are placed on disk in the same order they are read in, one after another. Card files are always organized this way. That is, record six cannot be written until record five is written, record five until four, and so on.

Indexed Sequential File Organization

An indexed sequential file is one in which records are placed on the disk in ascending collating sequence by record key. This key may be a part number, a man number, or any other identifying information that is present in the records on the file. In addition, the indexed sequential file uses an index to locate desired records. Each index entry contains a cylinder address and the highest record key on that cylinder. All index entries are formed into an index table. For cylinders that have overflowed, the index entry also contains the overflow sector address and key of the first sector overflowed from that cylinder.

Index tables are analogous to the index card file in a library. If you know the name of a book (record key), you can look in the card file (index table) until you find the card (entry) for that book. On the card, you will find a number (cylinder address) where the book (record) is located. You go to the shelf (seek) and find the number (cylinder address) you are looking for. Now you can search for the particular book (record) by title (record key).

FILE PROCESSING

Four methods of file processing are available with 1130 RPG:

1. Sequential processing of sequentially organized files
2. Random processing of sequentially organized files
3. Sequential processing of indexed sequentially organized (ISAM) files
4. Random processing of indexed sequentially organized (ISAM) files

Sequential Processing (Sequential Files)

All records in the file are processed in order starting with the first record in the file.

Random Processing (Sequential Files)

In random processing, the sequence of record processing is not related to the sequence of records on the file. To find a record in a sequentially organized file, the record number must be supplied to the program. The record number indicates the relative position (sequential location) of the record in the file. The disk input/output routine calculates the sector address from the record number and reads the proper record. The program makes no comparison by record content.

When processing a sequential file randomly, it is possible to access any record location within the limits of the total number of sectors allocated, even if the entire file area was not filled during the sequential load.

Sequential Processing (Indexed Sequential Files)

All records in an indexed sequential file are available in a sequence determined by record key. Processing may start at the beginning of the file or at any point within the file.

The entire file or only a segment of it may be processed.

Processing segments of a file is called processing within limits. This is accomplished by reading a file that specifies the limits of the file to be processed. The file that specifies the limits is called a record address (RA) file.

For example, a payroll file is to be updated with new pay increases. The payroll file is in sequence by department number. Each week, pay raises for various departments become effective. Therefore, on each week's processing, only segments of the payroll file are updated. The updating is accomplished by reading a card file that specifies the limits of the file to be processed. One such record might indicate that the records for departments 26-41 are to be updated, another for departments 76-80, and so forth.

Random Processing (Indexed Sequential Files)

To find a random record in an indexed sequential file, the index is searched using the key of the record. The matching entry in the index points to the cylinder containing the record. That cylinder is then searched for the desired record. The match is again made by record key.

SEQUENTIALLY ORGANIZED DISK FILES

As mentioned previously, a sequentially organized file is one in which records are placed on disk in the same order they are read in, one after the other.

Creating a Sequential File

The space for a sequentially organized file is initially established on the disk by using a DUP STOREDATA function. STOREDATA sets aside a specified number of sectors for the file and records the file name. This file name must be used in all future references to this file. The DUP STOREDATA function is described in the publication *IBM 1130 Disk Monitor System, Version 2: Programming and Operator's Guide,* Form C26-3717.

Once the space for the file to be created is established, the file should be described to RPG by the following entries on the File Description Specifications sheet:

- A filename in columns 7-14.
- An O in column 15.
- An F in column 19.
- A record length in columns 24-27.
- A Device name in columns 40-46.

A sample file definition to create a sequentially organized file is shown in line 030 of Figure 121.

The RPG program then loads the records from the input device, one after another, beginning at the starting address of the disk file, and continues loading until the end address is reached or until all the records have been loaded.

If the file is defined as an update file, the records to be updated can be specified on the Output-Format Specifications form. Furthermore, only the fields of an update record that are to be changed need be defined on output specifications. The RPG program combines these fields with the unchanged fields and restores the updated records into the same locations on disk, before retrieving a new record for processing.

To extend a sequential file or add records, the entire file must be reloaded.

If a sequential disk file is processed sequentially, it can be updated only at detail time during the processing of this record. At any other processing time, updating would be performed on another record. If a sequential disk file is processed randomly, it can be updated at either detail or total time.

Figure 121 shows the coding on the File Description Specifications form required to process records of sequential input, update and output disk files.

Each file is defined to have records 80 characters in length. Since each sector can contain 640 characters each file will have 8 records on each sector defined for that file. Line 010 shows an input file as the primary file. Line 020 shows a file being processed as a sequential file and updating is allowed. Line 030 shows a file being created. Line

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 (1 2): Ø1

Program Identification (75 76 77 78 79 80): SEQUEN

Control Card Specifications

Line (3 4 5)	Form Type (6)	Core Size to Compile (7 8 9)	Object Output (10)	Listing Options (11)	Core Size to Execute (12 13 14)	MFCM Stacking Sequence (15 16)	Sterling: Input-Shillings (17)	Sterling: Input-Pence (18 19)	Sterling: Output-Shillings (20)	Sterling: Output-Pence (21)	Inverted Print (22)	360/20 2501 Buffer (22)	Number Of Print Positions (23 24 25)	Alternate Collating Sequence (26)	Refer to the specific System Reference Library manual for actual entries. (27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74)
0 1 Ø	H														

File Description Specifications

Line (3 4 5)	Form Type (6)	Filename (7 8 9 10 11 12 13 14)	I/O/U/C (15)	P/S/C/R/T (16)	E (17)	A/D (18)	F/V (19)	Block Length (20 21 22 23)	Record Length (24 25 26 27)	L/R (28)	K/I (29 30 31)	I/D/T or 1-9 (32)	Type of File Organization or Additional Area (33 34)	Overflow Indicator (35 36 37 38)	Key Field Starting Location (35 36 37 38)	Extension Code E/L (39)	Device (40 41 42 43 44 45 46)	Symbolic Device (47 48 49 50 51 52)	Labels (S, N, or E) (53)	Name of Label Exit (54 55 56 57 58 59)	Extent Exit for DAM (60 61 62 63 64 65)	File Addition: A (66)	Number of Tracks for Cylinder Overflow (67 68 69)	Number of Extents (70)	Tape Rewind: N/U (71)	(72 73 74)
0 2 Ø	F	INPUT	I	P		A	F		8Ø								DISK									
0 3 Ø	F	SEQUPDAT	U	S		A	F		8Ø								DISK									
0 4 Ø	F	OUTPUT	O				F		8Ø								DISK									
0 5 Ø	F	RANDOM	I				F		8ØR								DISK									
0 6 Ø	F	RANDOMUPU					F		8ØR								DISK									
0 7	F																									

Figure 121. Sample File Definitions for Sequential Organization

040 shows a sequential file being processed randomly and being retrieved by a relative record number. Line 050 shows a file being processed randomly and updated. Retrieval is by means of a relative record number.

Calculating Size of a Sequential File

The number of sectors needed for a file depends on record size and number of records. The records are fixed length and can be defined as any size between 1 and 640 characters. Records cannot extend across sector boundaries. Thus, a 640 character record (one sector) and a 322-character record would each require one sector of disk space. Careful planning is required in calculating optimum record size for a file.

When calculating file size, one record must always be added for the end of file record.

To change record sizes or add records to a sequential file, the file must be rebuilt. If the revised file requires additional sectors, it must be redefined and rebuilt.

Figure 122, at the end of this section, can be used to quickly calculate the space needed for a sequential file.

File Description Specifications

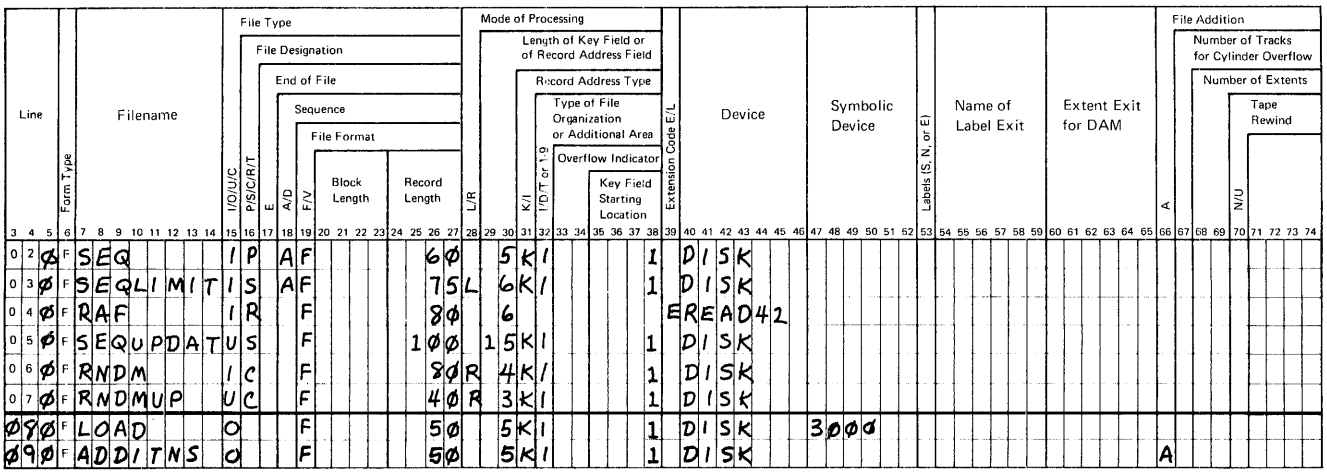


Figure 122. Sample File Definitions for Indexed Sequential Organization

INDEXED-SEQUENTIALLY ORGANIZED FILES

Indexed-sequential organization gives the programmer much flexibility in the operations he can perform on a file. He can read or write records whose keys are in ascending collating sequence. He can read and update random records. He can add new records.

Based on information supplied by the programmer on RPG specification forms, indexed-sequentially organized files are automatically handled. This is accomplished by the indexed sequential access method, or ISAM.

Figure 122 shows the coding on the File Description Specifications form required to process records of indexed sequential input, update and output disk files.

ISAM is a file management method specifically designed for direct access storage devices. It has the following advantages:

- It permits files to be processed in random or sequential order.
- It processes records directly in the input/output area.
- It establishes an index allowing ease of access to any record on the file.
- It uses an efficient chaining method to allow new records to be added to a file.
- It prevents records from being lost if a disk error occurs during an add operation.

ISAM has these restrictions:

- Records must be presorted in ascending collating key sequence before they are loaded on the file.
- Only one input/output area is permitted when a file is loaded or processed.
- All records must contain key areas starting in position one of the record, and all the key areas must be the same length.
- All records on a file must be the same length.
- Records cannot be both processed and added in the same program.
- The entire area for an indexed sequential file must be on one disk.

Contents of an Indexed Sequential File

An indexed sequential file comprises the following: file label; file index; prime data area; overflow area.

The relative position of these components within the ISAM file is as follows:

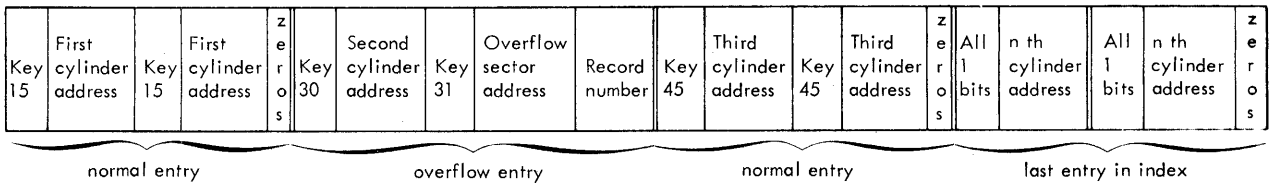
File Label	Index	Prime Data Area	Overflow Area

File Label: The first sector of any indexed sequential file contains the file label. This label contains information required for future processing of the file. All label operations are performed automatically by the ISAM routines within the Disk Monitor System. The user need only reserve one sector for the label when the file is initially defined. The first word of the label contains the key length supplied by the user.

File Index: The ability to read or write records anywhere in a file is provided by the file index. An entry in this index contains a cylinder address and the highest key that is associated with that cylinder. The ISAM routines locate a given record by searching the index for the key, then searching the specified cylinder for the desired record, again searching by key.

The key may be a part number or an employee name or any other identifying information that is contained in any record of the file. Its size cannot exceed 50 alphameric characters. The key entries in the index are the numbers of the highest key on each cylinder in ascending collating sequence. The end-of-file record key is the key with the highest possible value; i.e., all bits are ones.

A portion of an index or index table is shown in the following drawing. Note that each entry contains two sets of the same information. The second set is overlayed to show overflow data when the effected cylinder overflows.

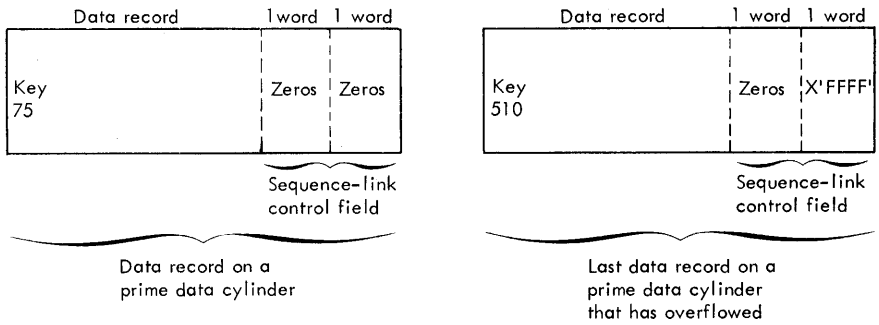


Overflow Area: When a new record is added to an indexed sequential file, it is placed according to key sequence. If records were to remain in precise physical order, the insertion of each new record would require all records with higher keys to be shifted up. However, since indexed sequential files have an overflow area, a new record can be entered into its proper position on a cylinder and only cause records with higher keys on that cylinder to be shifted. The record that is forced off the end of the cylinder by the addition of the new record is written in the overflow area.

The index entry of any cylinder that has overflowed points to the overflow sector address and record number of the overflowed record in the overflow area. If two or more records in key order are added, the overflowed records are chained together in the overflow area through the entries in their sequence-link control field. The entry in the first record points to the second, the second to the third, etc.

Prime Data Area: This area contains the data records placed in the file by the ISAM load routine. The records must all be the same length (maximum 636 characters). ISAM adds a two-word control field to each record. This control field, called the sequence-link control field, is used in the overflow area as a chaining indicator. It is used in the prime data area to indicate whether or not a cylinder has overflowed.

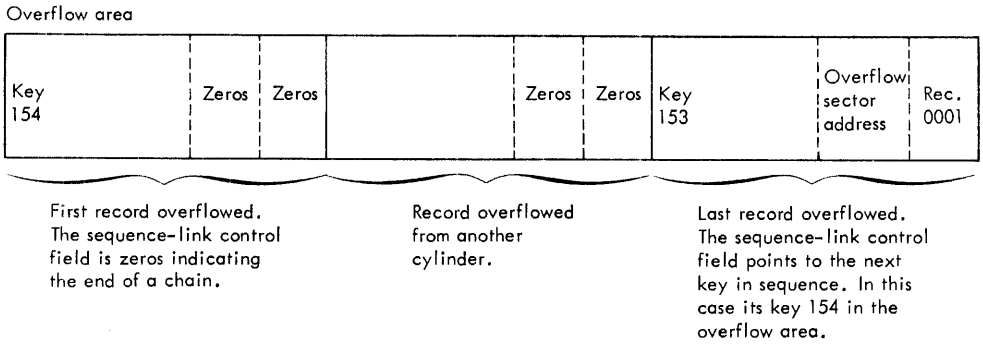
Prime data area records appear as follows:



The last overflow record in the chain has a sequence-link control field of all zeros.

The number of cylinders to be allotted to the overflow area must be determined by the programmer when the file is initially defined. Records are placed in the overflow area in the order they have overflowed, not in key sequence.

To illustrate the overflow area, assume that the last three entries on cylinder six of a defined file have keys 150, 152 and 154. Key 154 would identify cylinder six in the index. Now we add a record with key 153, a record on another cylinder, and a record with key 151. The overflow area would appear as shown below. Key 152 would identify cylinder six in the index. The overflow entry for cylinder six in the index would point to the overflow area.



Creating Indexed Sequential Files

An indexed sequential file is initially established on the disk in the same manner as a sequential file. STOREDATA is used to set aside a specified number of sectors for that file and to record the file name. This file name must be used in all future references to this file.

Estimating the Space Needed for an Indexed Sequential File

As a technique of good system design, it is desirable to quickly estimate possible file sizes for the methods of file organization. Since 1130 RPG automatically blocks records and the block size cannot exceed 636, there are certain optimal record sizes.

For example, an indexed sequential file of three thousand 125-character records requires about 140 cylinders. If the record size can be reduced to 120 characters, the file size is reduced to approximately 85 cylinders. Thus, with a 4% reduction in record size, a 40% reduction in file space is achieved. In addition, this significantly improves the retrieval speed of the file. Record sizes can often be reduced with system design considerations or with packing of numeric fields.

If, on the other hand, the record size cannot be reduced to 120 characters, the user should consider expanding the record size to 200 characters since the same amount of file space will be required. This may allow additional information to be stored on the record which may reduce the total number of operations in the system design or improve the capability of the system.

Because of the potential savings, the system designer should be familiar with Figure 123 at the end of this section. This chart shows the number of cylinders required per 1000 data records. From it, the user can quickly determine the amount of space required for an indexed sequential file. That is, he can determine the amount of space required for the records, for the index, and for the overflow area.

Record size in Characters	Cylinders Used Per 1,000 Records	
	Sequential	Index-Sequential: Prime Data Area
1	0.4	1.1
2	0.4	1.1
6	1.2	1.9
8	1.6	2.4
20	4.0	4.8
50	12.8	11.3
80	15.6	17.8
120	25.0	25.0
200	41.7	41.7
250	62.5	62.5
316	62.5	62.5
320	62.5	125.0
636	125.0	125.0
640	125.0	INVALID

Key Length in Characters	Index Sequential: Cylinder Index Chart
	Number of Entries On One Sector
2	45
3	35
5	24
8	16
10	13
14	10
18	8
22	6
25	6
30	5
40	3
50	3

Figure 123. Space Utilization for Various Size Records

The 1130 System requires that space be reserved for the file. This space must be expressed in terms of sectors required.

RPG automatically calculates the number of sectors required for the file for each program that will load an indexed sequential file. This number will be printed as part of the compilation listing.

The user must specify the total number of records he wishes to have in the file and enter this on the File Description Specifications form. This number should provide for a typical overflow area.

Examples: User specifies 2000 records to be loaded (100 characters long, key field=6 characters). The RPG compiler shows that 408 sectors (51 cylinders) are required for the disk area. (The user could have calculated this using Figure 123). The user must reserve a number of sectors on the disk in a DUP run for this file. He need not reserve the exact amount specified by RPG. The following depicts some examples and what factors should be considered.

1. If 408 sectors are specified, the user can initially load 1500 records and use the remaining space for overflow records. This will waste a small amount of disk space that has been reserved for the index to the records that were not loaded. (This is negligible in most cases. Here, it amounts to one sector.) If the user initially loads 2000 records, no records can be added to the file.
2. If 500 sectors are specified, the user could load 2000 records and use the remaining area for additions. He could not initially load 2200 records, because RPG has already set the maximum index size at 2000 records. (The load program halts when too many records are loaded.) The RPG load program would have to be recompiled with a larger maximum file size before reloading.
3. If 350 sectors are specified, the user obviously cannot load 2000 records. (The actual number would be 1730 records.) However, 1500 records could be loaded and the remaining area could be used for additions. (As in example 1, a small area would be lost due to the requirements for a larger index).

Summary of Indexed Sequential Organization

1. An indexed sequential file must be loaded in the ascending sequence of the key field.
2. Once loaded, records can be added to the file, but the size of the prime data area is fixed by the number of records actually loaded. (Rather than by number specified).
3. If a large number of records are added, the throughput performance in retrieving will be degraded due to the number of searches in the overflow area.
4. If the user attempts to add more records than he has allotted sectors for, the file must be copied sequentially, then reloaded into a larger area. Once this is accomplished, new records may be added.
5. Records must be deleted through record codes. When the file is reorganized, the user can drop the deletions by means of a record code. Deletions remain on the disk until the file is reorganized.
6. An indexed sequential file can only be retrieved randomly by specifying the appropriate key as a chaining field.
7. Only one of the following functions can be used per file per program: LOAD, ADD, retrieve sequentially (with or without updating), retrieve randomly (with or without updating), or retrieve sequentially within limits (with or without updating).
8. The last sector used from the cylinder index remains in core for any ISAM function (no access to the cylinder index is necessary and the access arm may stay in the prime data area). This can have a dramatic effect on system performance, especially when dealing with sorted transactions and random updating.
9. Prior to loading the file, an area on the disk containing the number of sectors for the file and overflow area must be reserved by a DUP run. This area should be given the same filename as that used for the file in the File Description Specifications.

DISK FILE SPACE CALCULATION

Figure 123 can be used to determine the amount of disk storage necessary for a particular type of file.

For example, to determine how much disk space is required for an indexed sequential file of three thousand, 60-character records, the figure can be used as follows:

1. Scan the Record Size in Characters column for a number equal to or greater than 60. This is 80 on the 7th line down.
2. Scan across under the column labeled Indexed Sequential Prime Data Area for the number of cylinders required to hold 1000 records. This is 18.0.
3. Since the file is to contain 3000 records, multiply the number of cylinders obtained in Step 2 by 3. This gives the amount of cylinders necessary for the file less the index: 54. Next, the size needed for the index must be calculated.
4. If the key length is seven characters, scan down the column Key Length in Characters for a number equal to or greater than 7. This is 8 on the 4th line.
5. Scan across for the number of entries on one section. This is 16.
6. Divide the number of cylinders required (obtained in step 3) by the number of entries on one sector. This is 54 divided by 16 which equals 4. Four sectors, or less than one cylinder, (8 sectors per cylinder) are required for the index. Thus, the indexed sequential file requires 55 cylinders (54+1).
7. Add 10% of the size calculated in Step 6 for an area to be used for overflow records. (The size needed varies with applications; 10% is typical of the amount needed). Ten percent of 55 is about 6. Thus, the file requires 61 cylinders. (This would be specified as 8 x 61, or 488 sectors in a DUP run.)

Determining the amount of disk storage needed for a sequential file of three thousand 60-character records requires the top portion of Figure 123. Merely scan down the Sequential column for the cylinders required for one thousand, 80-character records. This is 15.6. Multiply this by 3 which gives the number of cylinders required for the file: 46.8.

When more than one input file is used for an RPG program, processing can be controlled by using the matching records or the chaining technique. How the technique is employed depends on the organization of the file and the way that file is processed.

Remember, card files can only be organized sequentially, while disk files can be organized either sequentially or indexed sequentially. Card files can only be processed sequentially. Disk files can be processed sequentially or randomly.

The method of processing of a file should be chosen after the file organization for all input files has been determined. Figure 124 summarizes the possible choices.

Sequential Organization: All card files are sequentially organized. Disk files may also be sequentially organized. Such files require that every record be processed in sequence. When more than one sequentially organized file is used for input, three processing techniques can be used:

1. The records of the input files may be related in some way, so that processing alternates between the files. The records of the primary and secondary files are matched by special fields in these records. This type of processing is called *Matching Records.*
2. The records of the input file may be processed independently. All records from the primary file will be processed first. Then, each entire secondary file will be processed in the order in which it was specified on the File Description Specifications form.
3. The file can be processed randomly by use of the CHAIN operation on the Calculation Specifications form.

Indexed Sequential Organization: Only disk files may be organized as indexed sequential. While such files are arranged sequentially, each record contains a key which is indexed. This allows either sequential or random processing of the file, depending on the needs of the job:

1. The file may be sequentially processed, either the entire file, or between some limits (see *Record Address Files*). Matching Record techniques may be used.
2. The file may be processed randomly, by accessing records through their keys. These keys are specified by another file through a process called Chaining (see *Chaining*).

Combinations: The file organizations and processing methods might be combined in several ways. For example, an application might require three input files: two sequentially organized and one indexed sequentially organized file might be related to either or both of the sequential files through chaining.

File Organization	Possible Mode of Processing
All Card Files (Sequential)	Sequentially
Sequential Disk File	1. Sequentially (Entire File Only) 2. Randomly (Retrieval by Relative Record Number)
Indexed-Sequential Disk Only	1. Sequentially • Entire File • Limits of the File (Using a record address file to prescribe the limits) 2. Randomly (Using Chaining) • CHAIN Operation • Chaining by C1-C3 • Using a record address file to supply the Chaining fields (seldom used technique).

Figure 124. Comparison of File Organization and Mode of Processing

PRIMARY AND SECONDARY FILES

When sequentially processing multiple input files, one (and only one) file must be designated the primary file. It is specified by entering P in column 16 of the File Description specification for that file. Other input files are designated secondary files, and are specified by entering S in column 16 of their File Description specifications.

The primary file is generally the controlling file in a program. In the case of processing by matching records, the primary file is processed first when records match.

Matching Records

The technique of controlling processing by matching records is used only when processing two or more files *sequentially.* Examples of matching records are:

- A card file and a sequentially organized disk file (being processed sequentially).
- Three sequentially organized disk files.
- A card file and an indexed sequentially organized disk file being processed sequentially.

RPG processes only one record at a time. Matching record techniques determine which file supplies the next record to process.

Match fields: Matching fields are specified by entering one of the codes M1 through M9 in columns 61-62 of the Input Specification for the field. (See *Matching Fields or Chaining Fields* under Input Specifications Form.)

Figure 125 illustrates the specification of matching fields. In this example, there are two input files, MASTER

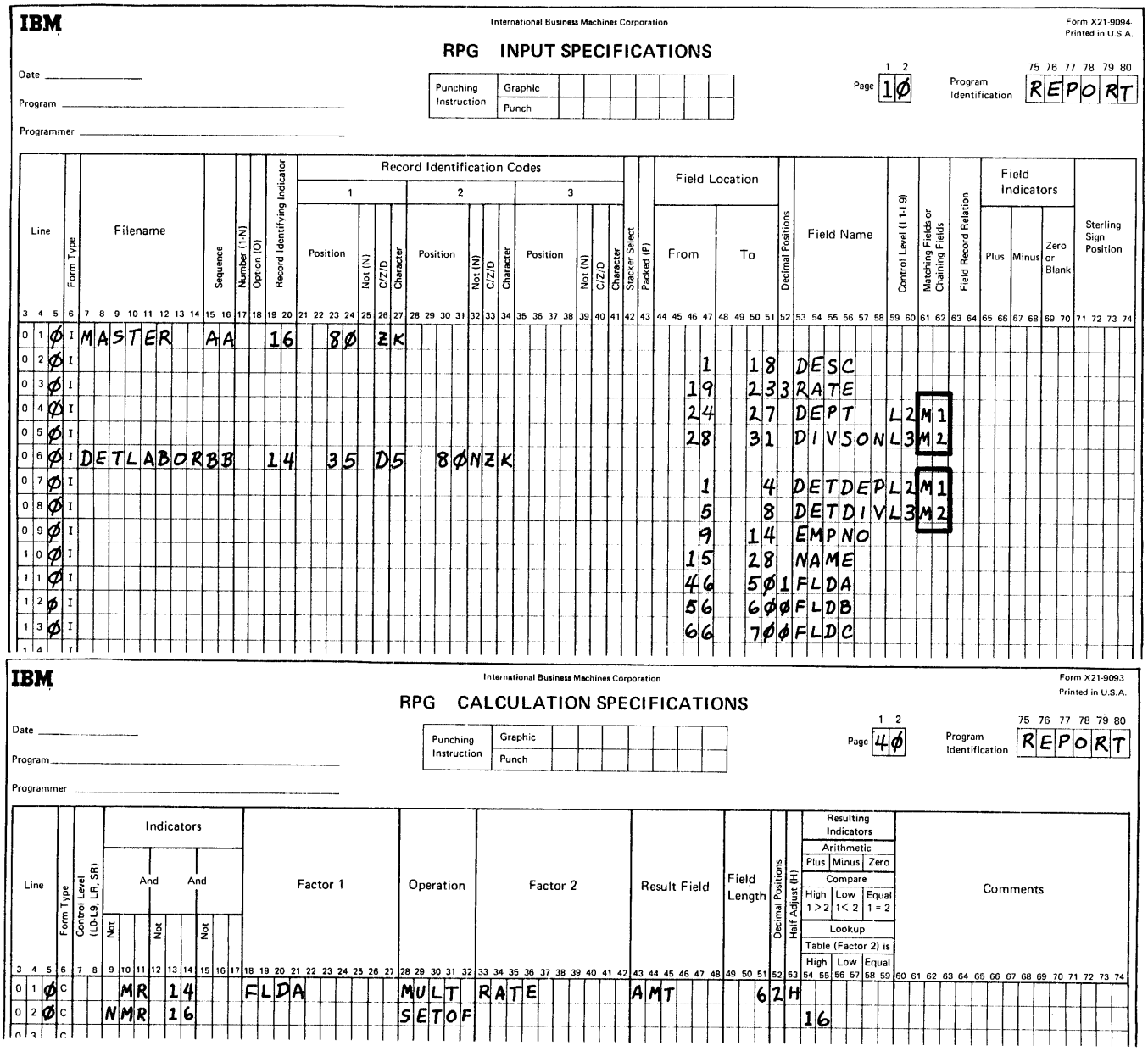


Figure 125. Specifying Matching Fields in Multiple Input Files

containing certain rate information, and DETLABOR, containing detail records. MASTER is designated as the primary file. Field DEPT from file MASTER is to be compared with field DETDEP from file DETLABOR (M1), and field DIVSON from file MASTER is to be compared with field DETDIV from file DETLABOR (M2). Whenever these fields match, the MR indicator is turned on. (In this example, the match fields have different names, but they may have the same name.)

Matching fields determine the manner in which records are matched. Matching records determines which input file supplies the next record for processing. The program begins by reading one record from each input file. The match fields are compared and one record is selected for processing. The program then selects succeeding records by:

1. Reading the next record from the file from which the record just processed came.
2. Comparing the new record with the record previously read but not selected for processing. One is selected for processing, based on the comparison.

Selection of the next record to process follows these rules:

1. The record with the lowest value in its match fields is processed first, if ascending order is used. If descending order is used, the highest record is selected.
2. If the primary record matches one or more secondary records, the primary record is processed before the secondary records which match it.
3. If two or more secondary records match, they are processed in the order they are specified on the File Description Specifications form.
4. Records that have no match fields are processed before records that have match fields.
5. If two or more records are without match fields, they are processed primary file first, followed by secondary files in the order specified on the File Description Specifications.

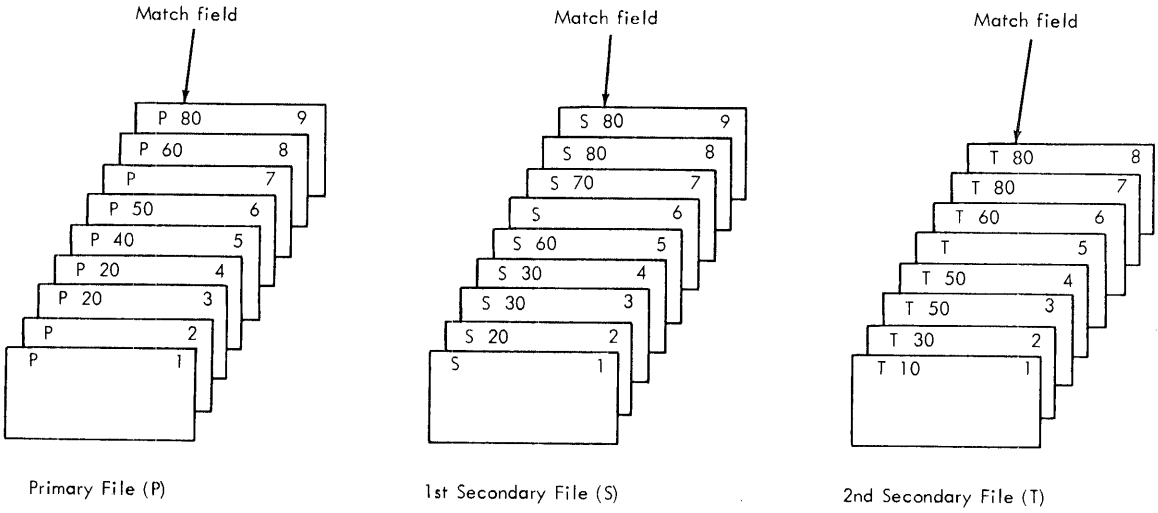
Figures 126 and 127 illustrate how records are selected for processing. Three files are shown in the example. Figure 126 shows the files as they appear and the sequence in which they are processed. Figure 127 shows the selection process step-by-step for the first ten records.

Matching Record Rules:

1. All files using matching fields must use the same matching fields. That is, if one file used matching fields M1 and M2, M1 and M2 must be defined in all files using matching fields.
2. All matching fields of the same level must be defined to be of the same length and type. That is, M1 of the primary file must have the same length and type as M1 of all secondary files, etc.
3. The order of processing when not under matching record control is primary file first, followed by the secondary files in the order they are specified on the File Description Specifications form.
4. If a sequentially-processed input file is defined without matching fields, the entire file is processed in order of priority as described in item 3.
5. If record types without matching fields are in files containing records with matching fields, the records without matching fields are processed as they are encountered, and according to the priority described in item 3.
6. The matching record indicator (MR) is set ON only when the record from the primary file matches one or more of the records from the secondary files. It is *not* set on when secondary file records match without matching the primary file record.
7. The matching record indicator is turned off after the last record of a matched group has been processed, including total calculations and output for that group. This matched group includes all primary and secondary field records which have identical match-field contents.
8. The sequence check of matching fields will operate correctly until a record without matching fields is encountered. At this point, sequence checking will resume with the last out-of-sequence matching field as the base.

Matching Record Indicator (MR)

The matching field entries of M1-M9 have an associated internal indicator MR (Matching Record). This indicator, which is similar to a resulting indicator, is used to condition functions specified on the Calculation and Output-Format Specifications forms.



The records in the three files shown are processed in this order. (A blank match field means the record has no match field.) A sequence number is contained in column 80. These records are on disk, and are not to be construed as cards.

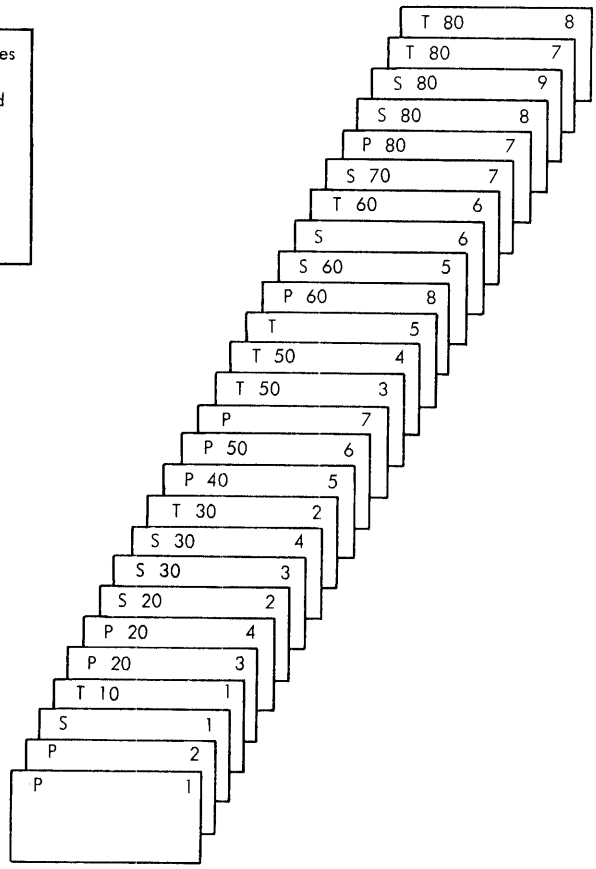


Figure 126. Record Selection (3 Disk Files)

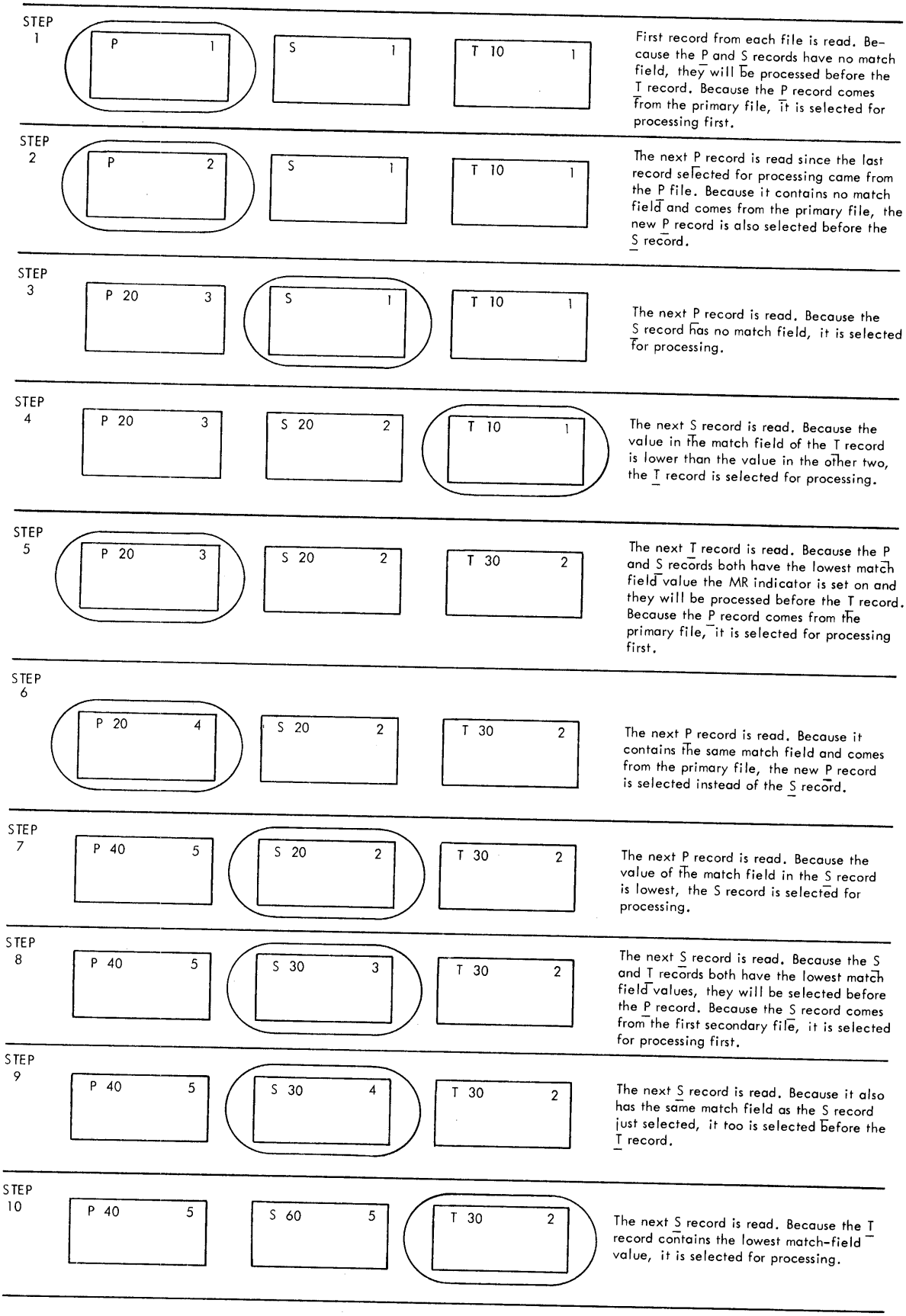


Figure 127. Step-by-Step Record Selection Process (3 Disk Files)

The MR indicator is turned on when the matching field of a record from a secondary file matches the matching field of a record from the primary file. The comparison is made on all fields (M1-M9) at the same time. If the M1 fields match, but the M2 fields do not, the MR indicator is turned off. The MR indicator is turned on before the first record of the matched primary file is processed. The MR indicator remains on during processing of all succeeding primary and secondary file records that contain the same matching field.

The MR indicator is turned off when all total calculations and output are completed for the last record of the matching field.

Figure 128 illustrates when the MR indicator is turned on and off. This example shows a simple block diagram of the object program logic to illustrate logic flow. Four cycles of the program are shown; subsequent cycles would be a repetition of this pattern. Two input files are used, and the record selected for processing in each program cycle is shown. The status of indicators is shown by vertical bars along the right side of each step of the figure. A solid bar is used to denote the indicator ON, and a dotted line indicates OFF. Three indicators, one for each file, are used to show which file is being processed.

The MR indicator must not be turned on or off by the operations SETON and SETOF.

All record types do not require processing by the matching record technique. When record types with no matching fields are specified on the Input Specifications form, the matching fields specification is left blank. This indicates to the program that these record types do not need to be checked for a matching field. They are processed immediately as they are read. The MR indicator is turned off.

Use of the MR Indicator: As specified on the calculations form in Figure 125, whenever the MR indicator is on, RATE from the master file record is multiplied by FLDA from the record in the detail labor file. If the records do not match, indicator 16 is set off. In this case, all subsequent processing conditioned by indicator 16 is suppressed. The MR indicator could be used to condition calculation specifications to prevent calculations upon an unmatched detail or master record. It could also be used in the output specifications to select unmatched detail cards or unmatched primary cards.

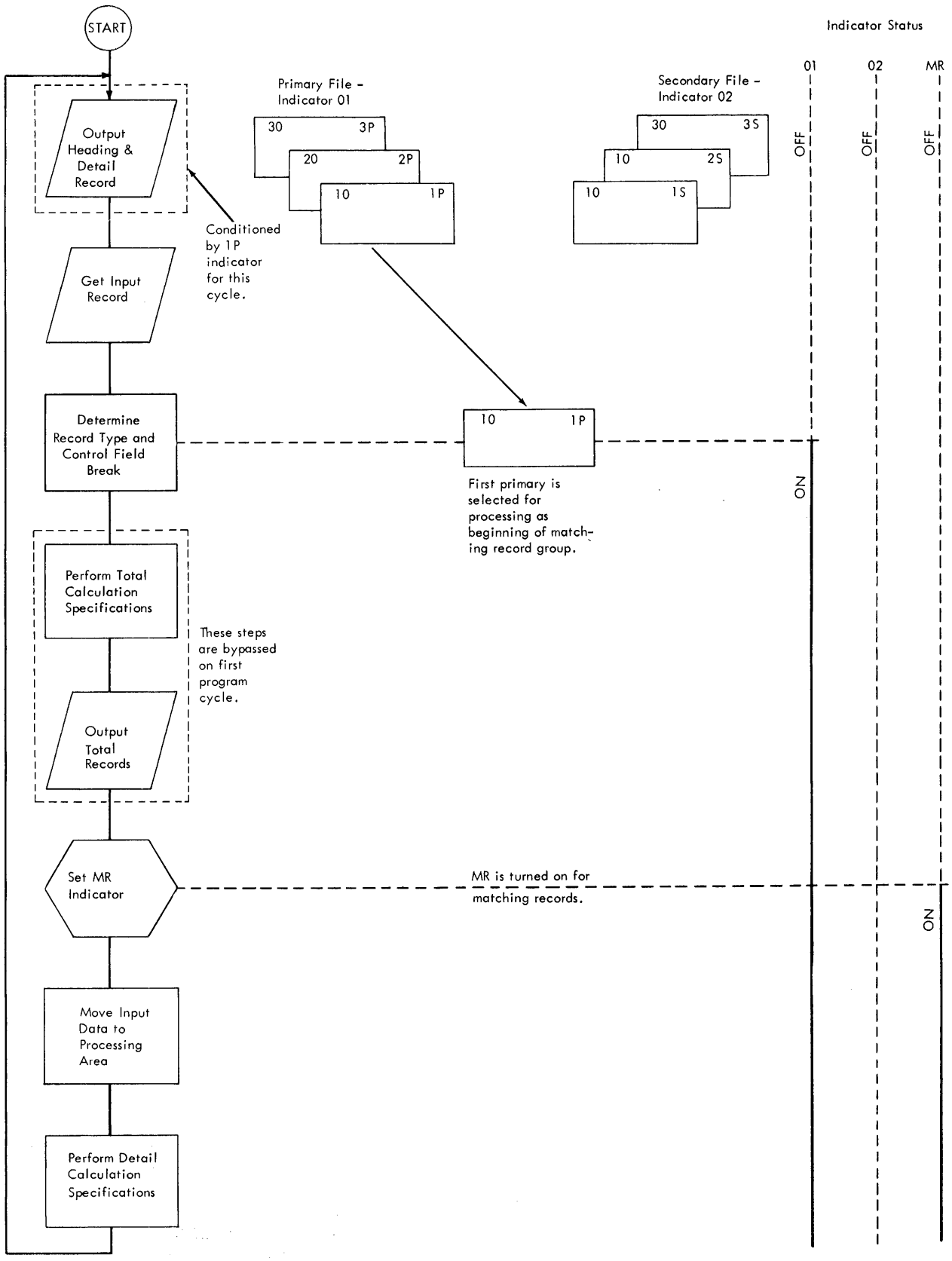


Figure 128. MR Indicator Setting (Part 1 of 4)

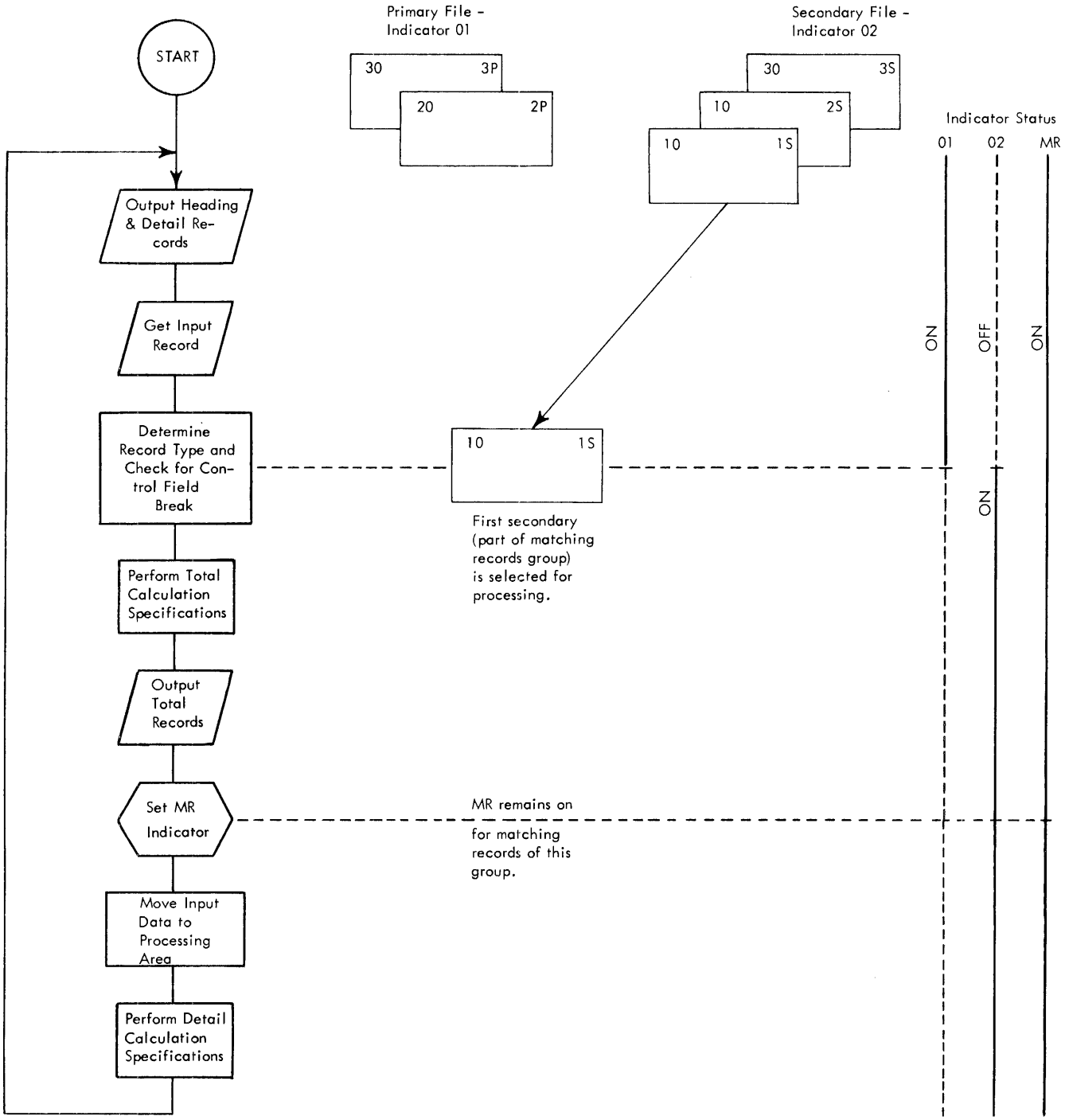


Figure 128. MR Indicator Setting (Part 2 of 4)

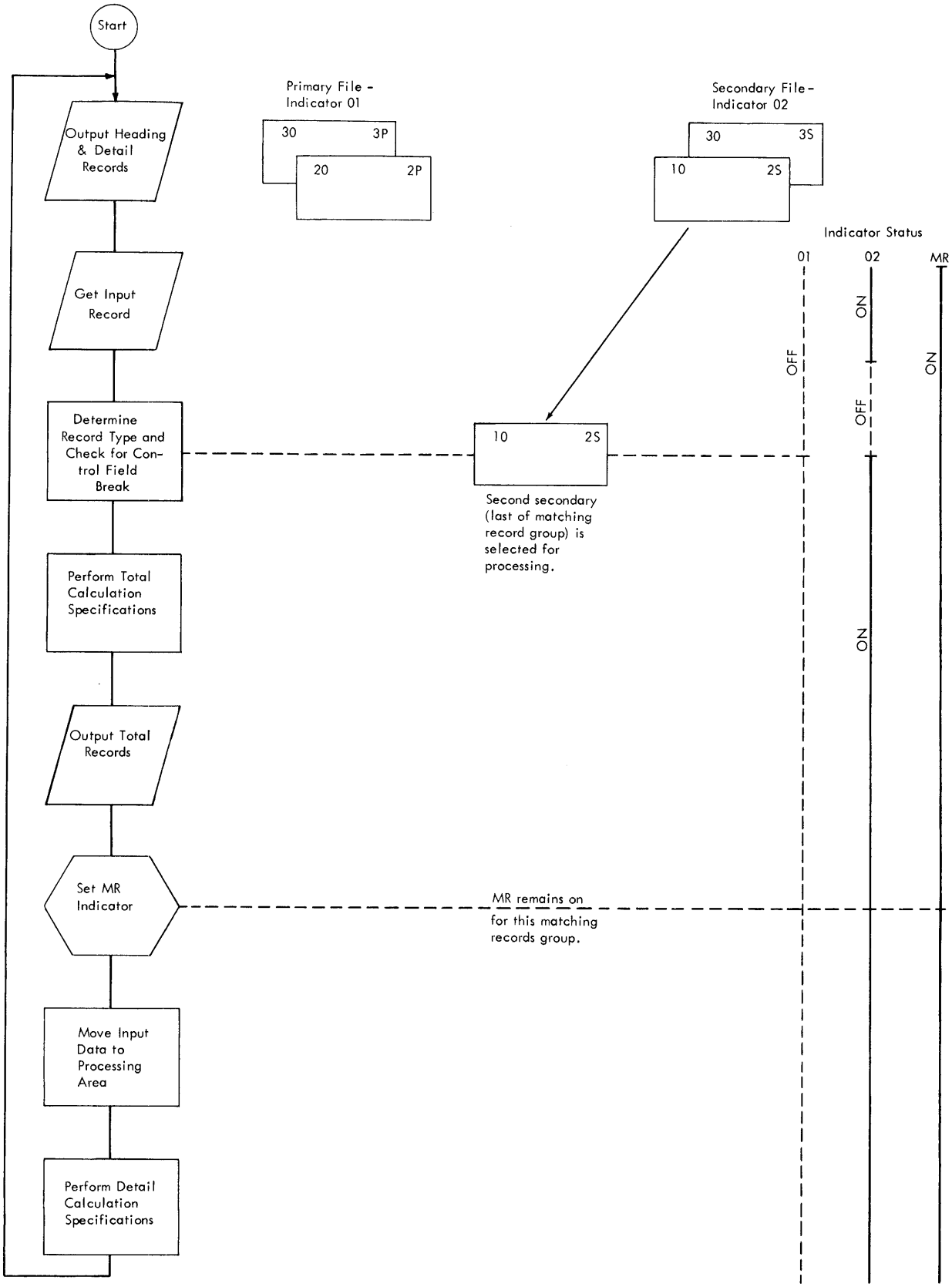


Figure 128. MR Indicator Setting (Part 3 of 4)

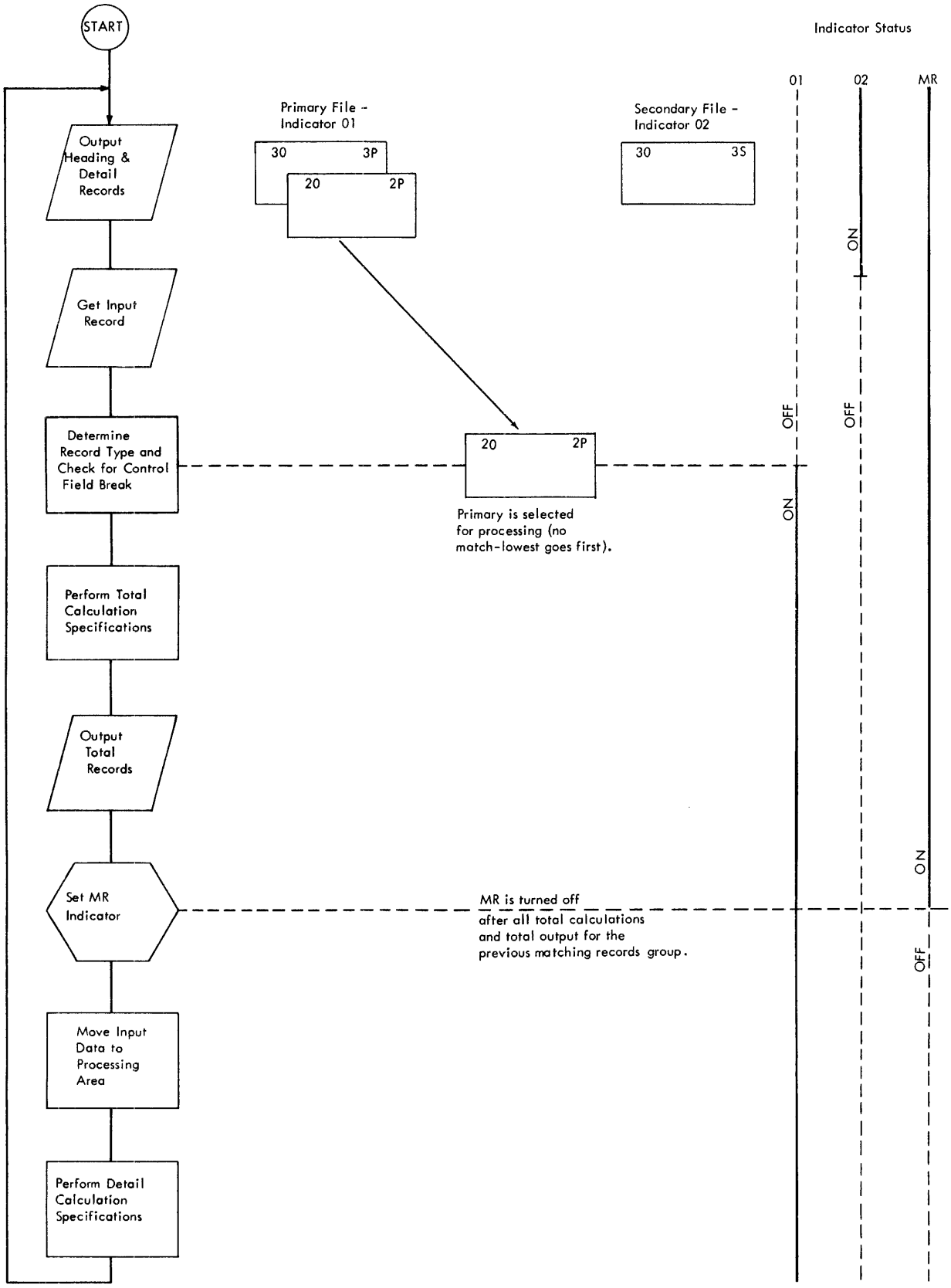


Figure 128. MR Indicator Setting (Part 4 of 4)

End-of-File Processing: By specifying an E in column 17 of the File Description Specifications form, the user indicates that the job is to be ended after all records in the file denoted by the E are processed. When matching records are used to control the processing of multiple files and an E is specified only for one of these files, the job will end after the last record in the file with the E designation is processed. Records remaining in the other files will not be processed. The records that will be processed in certain end-of-file situations are indicated in Figure 129.

1130 RPG multi-file logic will discontinue processing when all specified EOF conditions are met, and the next record type to be processed does not contain matching fields.

Therefore, any subsequent secondary files with matching fields that match the last primary matching field will not be processed. A method of processing these records is to code an E in column 17 of the File Description Specifications for the secondary files. Processing could be terminated by checking pertinent fields and setting a halt indicator.

Matching Fields and Control Fields

The matching record technique has no direct relationship with control level specification. These are independent functions in RPG. However, it is frequently desired to specify both match fields and control fields within the same record types.

The RPG object program uses the matching field specifications to determine the order of processing records. If a total or summary card is to be punched after each set of control groups has been processed (control break), the same control field must be specified for both the primary and secondary file.

If the same control fields are specified in both files, these files are processed in the same manner as a single-file program. A control break occurs only once for each control group (either in the primary or the secondary file).

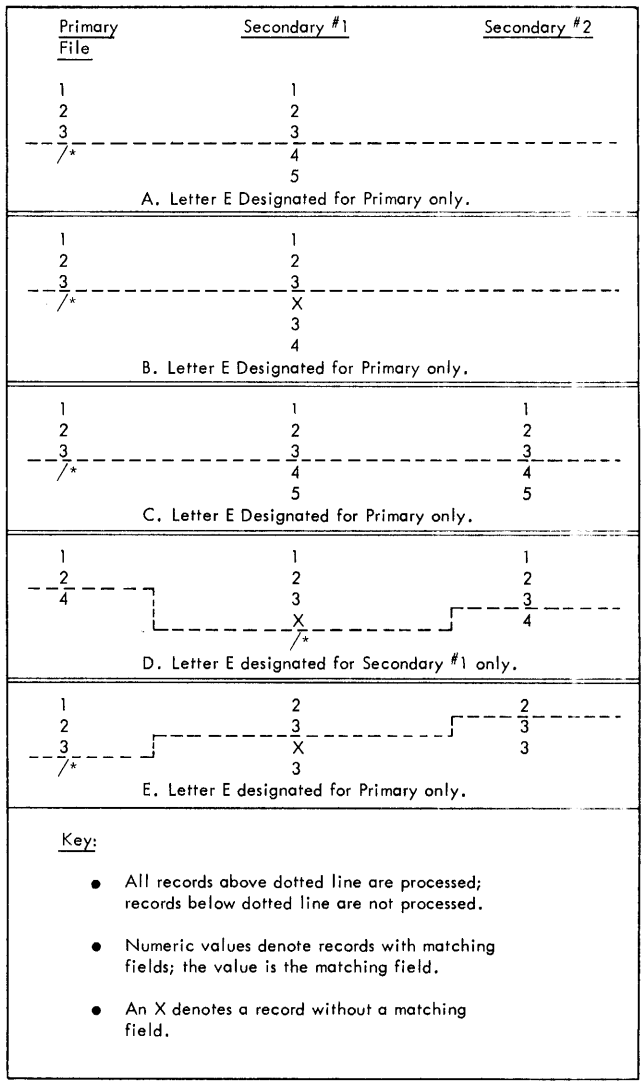


Figure 129. Status of Records in Multiple Files After End-of-File Condition

Examples

Figure 130 illustrates a primary and secondary file to be processed.

Indicator 01 is turned on whenever there is a primary record. Indicator 02 is turned on whenever there is a secondary record. Thus:

1. A matching primary record will be coded 01 and MR.
2. A matching secondary record will be coded 02 and MR.
3. An unmatched primary record will be coded 01 and NMR.
4. An unmatched secondary record will be coded 02 and NMR.

Figure 131 (Part 1) shows the Input Specification sheet for the example shown in Figure 130. Figure 131 (Part 2) shows what the output would look like for this coding. It also shows what the effect would be if the Control Level entry (Columns 59-60) would be removed for each of the files.

Normally, the correct coding technique is to include the control level indicator on each file. This will prevent erroneous totals that would occur when one of the files does not have a control level indicator. The output also shows the status of the MR indicator at detail time for the input records and at total time for the L1 break.

Using Matching Record Techniques Without Matching Fields

If records without matching fields are mixed with records with matching fields in a multi-file program, the following occurs:

1. Records without matching fields are processed as they are encountered. The object program processes only initial records without matching fields until the first record within each file with matching fields is encountered. The processing sequence is primary first, followed by secondary files in the order they are specified on the File Description Specifications. If a file contains no records with matching fields, the entire file is processed before any records of the subsequent file(s) are processed.
2. During the processing of the record without matching fields, MR is turned off.

If only one file contains records with matching fields, this specification is used for sequence checking only. All primary and secondary files are processed in the order they are specified on the File Description Specifications form.

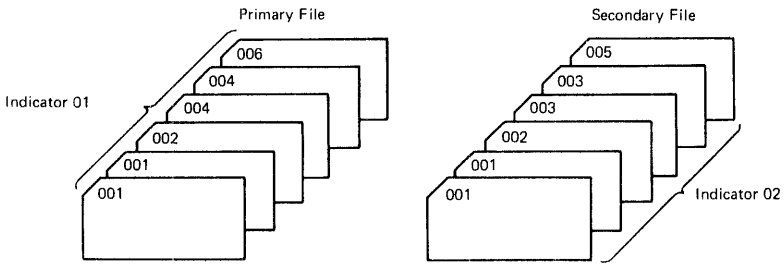


Figure 130. Example of an Input File

IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic							
	Punch							

Page 10

Program Identification MATCHG

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	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	PRIMARY	AA			01																								
02	I																				1	3	PRIMTCL1M1							
03	I	SECONDRYAB				02																								
04	I																				1	3	SECMTCL1M1							
05	I																													

Figure 131. Matching Fields and Control Levels (Part 1 of 2)

Control Levels on Both Files			Control Level on Primary Only			Control Level on Secondary Only		
001	P	MR	001	P	MR	001	P	MR
001	P	MR	001	P	MR	001	P	MR
001	S	MR	001	S	MR	001	S*	MR
001	S	MR	001	S	MR	001	S	MR
L1 Total		MR	L1 Total		MR	002	P	MR
002	P	MR	002	P	MR	L1 Total		MR
002	S	MR	002	S	MR	002	S	MR
L1 Total		MR	003	S	NMR	L1 Total		MR
003	S	NMR	003	S	NMR	003	S	NMR
003	S	NMR	L1 Total		NMR	003	S	NMR
L1 Total		NMR	004	P	NMR	004	P	NMR
004	P	NMR	004	P	NMR	004	P	NMR
004	P	NMR	005	S	NMR	L1 Total		NMR
L1 Total		NMR	L1 Total		NMR	005	S	NMR
005	S	NMR	006	P	NMR	006	P	NMR
L1 Total		NMR	L1 Total		NMR	L1 Total**		NMR
006	P	NMR						
L1 Total		NMR						

* This card would actually cause a control break. Since it is the first control break, total calculations and total output will be bypassed by the RPG cycle.

** On the last cycle, L1 is forced on by the LR indicator.

Figure 131. Matching Fields and Control Levels (Part 2 of 2)

Exit to a Translate Subroutine

If the sequence of the matching fields is not EBCDIC collating sequence, the RPG program can provide an automatic exit to an external user written subroutine that translates the sequence of the matching fields to the collating sequence desired.

To cause this, an entry in the RPG Control Card is required (an A in column 26). The automatic branch occurs after the input card is read and before the RPG program checks the sequence of the matching field.

The subroutine to translate the matching fields must use the predefined label ALTSE. The address of the RPG matching fields hold area containing the matching fields to be translated will be contained in index register 1. This address will be the control word of the field. If index registers 2 and 3 are used in the subroutine, they must be saved and restored. The subroutine must place the translated fields back into the matching-field holding area before it returns control to RPG.

The original data fields (to be used for controlling, printing, arithmetic calculations, etc.) are not translated by the subroutine.

Chaining is used to randomly retrieve selected records from disk. The file from which the records are retrieved is called a chained file.

Two or more input files are always involved in chaining. One file contains a field which can be used to point to the particular record in the chained file. This field is called a chaining field.

In 1130 RPG, chaining can be done by using the CHAIN operation on the Calculations Specifications form, or by using C1-C3 on the Input Specifications form. Use of the CHAIN operation offers the most flexibility.

Chaining to an Indexed Sequential File

With indexed sequential organization, every record of a file has a key through which the record can be accessed. These keys are used in chaining to link records from a chaining file to records in a chained file.

Assume that a sequential input file, as shown in Figure 132, contains information about transactions made with several customers. The card file contains the customer's number, but does not contain his name, address, or balance. Another file called the master customer file contains this additional information about each customer. The RPG program can use the second file when preparing a customer report.

The master customer file has indexed sequential organization, and it is to be processed randomly. The key in each disk record contains the customer's number. The field in the transaction-file records which contains the customer's number can be used to chain the files together. The object program takes the customer's number from the transaction file and locates the record with the same key in the master customer file. The additional information, such as the customer's name, address, balance, etc., is associated with each key and it is immediately available for processing.

Figure 133 illustrates the coding required for a typical chaining operation. This operation is similar to the one described in the preceding paragraphs. In this case, a card file CARDIN contains transactions which are to be updated against an indexed sequential file called MASTCUST.

The File Description Specifications form describes CARDIN as the primary file for the job, describes MASTCUST as the indexed sequential file to be updated, and describes PRINTER as the output file. CARDIN is the chaining file. MASTCUST is the chained file.

The Input Specifications form describes the records in CARDIN and MASTCUST. The transaction record in CARDIN has two fields: CUST which is used as the chaining field and AMTPD which is used to update the appropriate customer record.

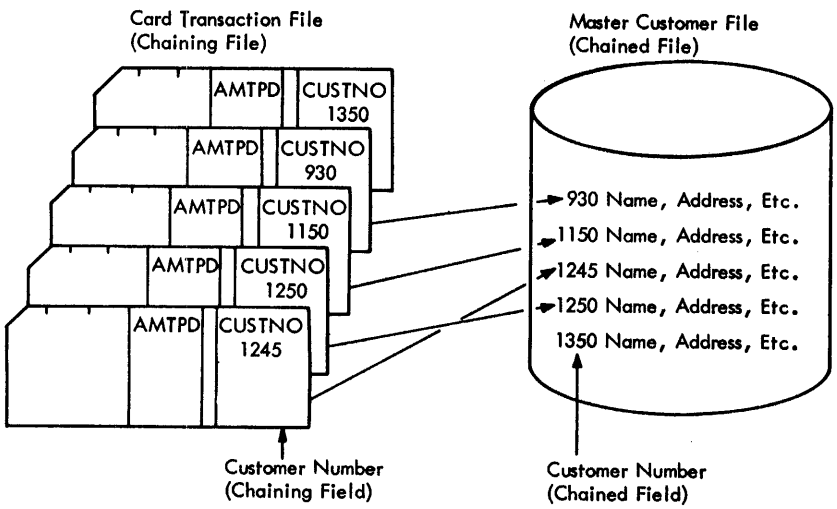
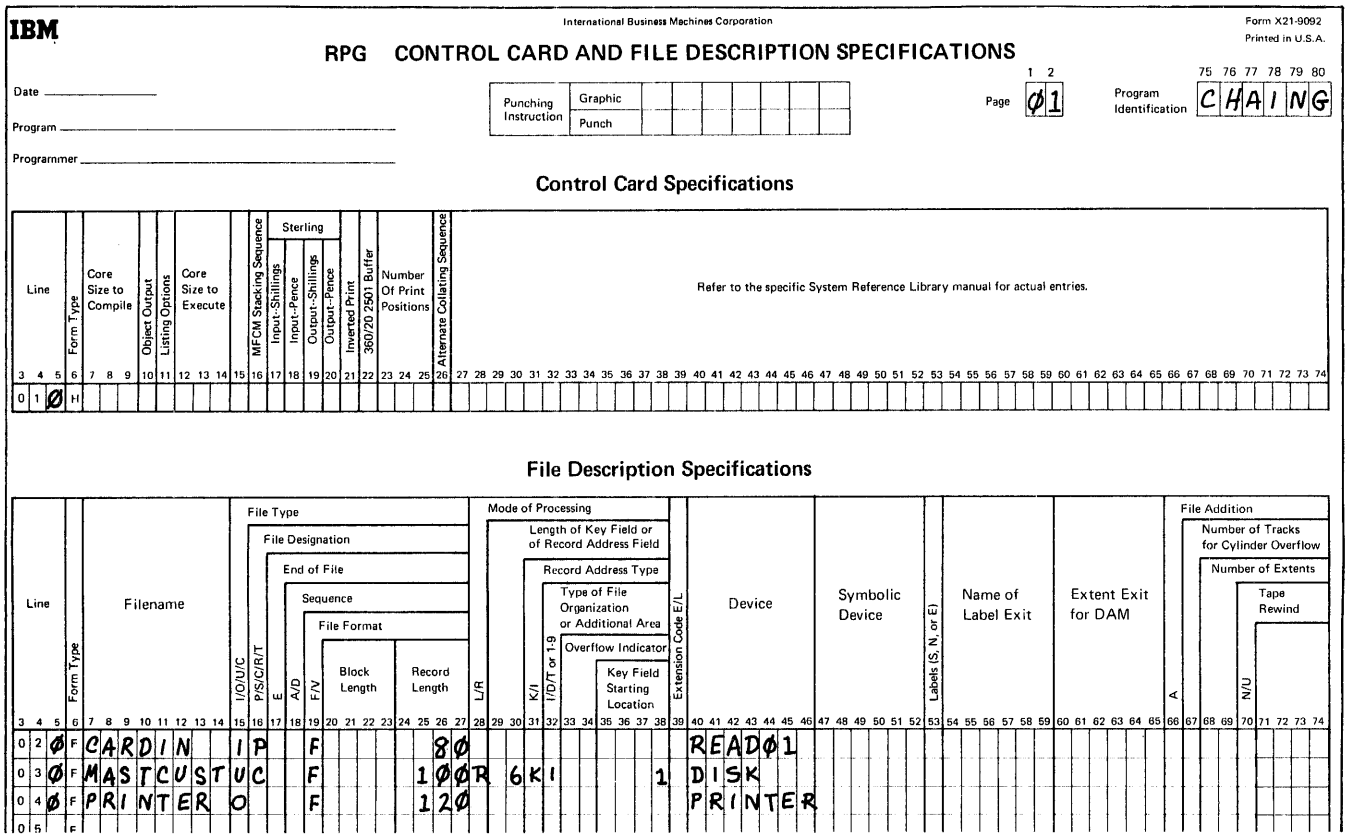


Figure 132. Chaining to an Indexed Sequential File



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

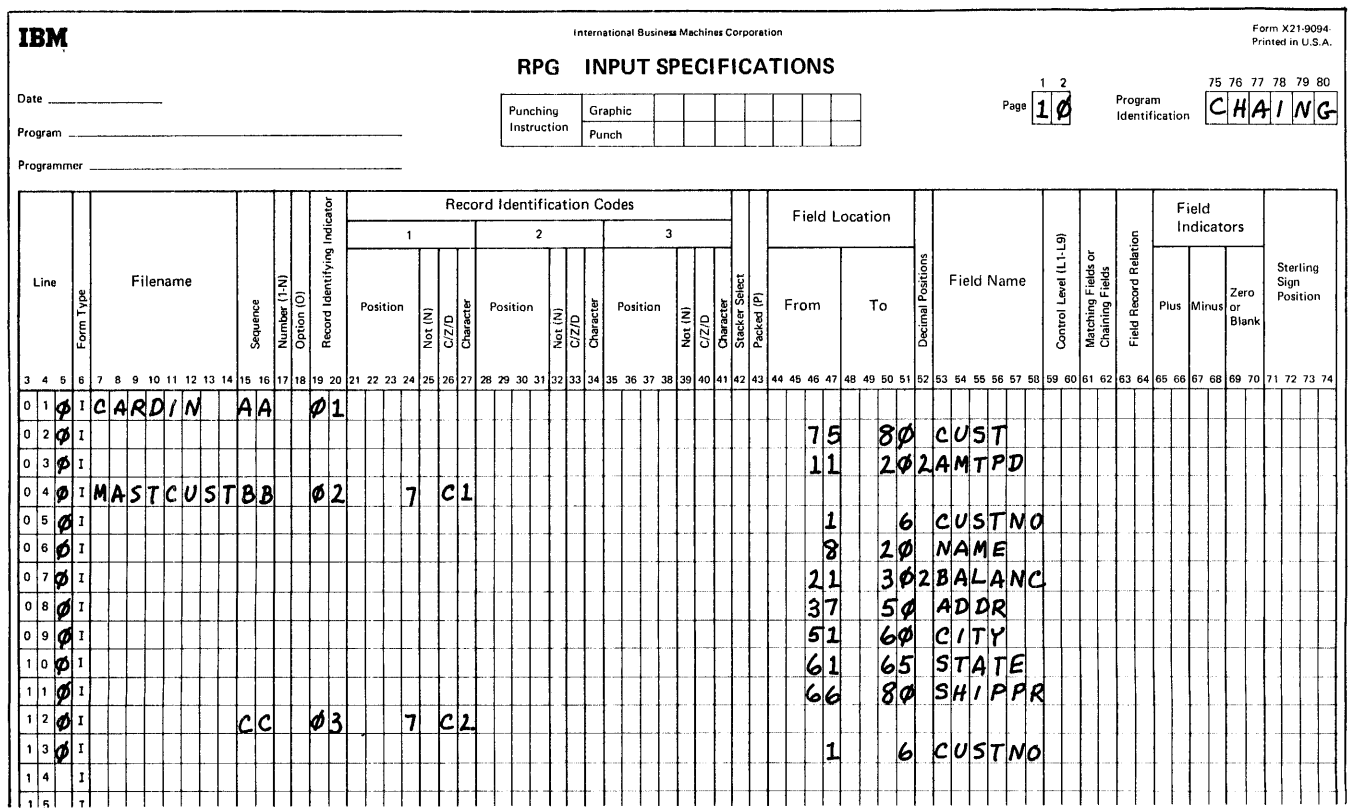
Program Identification: CHAING

Control Card Specifications

Line	Form Type	Core Size to Compile	Object Output	Listing Options	Core Size to Execute	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	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, or E)	Name of Label Exit	Extent Exit for DAM	File Addition: A	Number of Tracks for Cylinder Overflow	Number of Extents	N/U	Tape Rewind
			I/O/U/C	P/S/C/R/T	E	A/D	F/V			L/R		K/I	I/D/T or 1-9													
02	F	CARDIN	I	P			F		80								READ01									
03	F	MASTCUSTUC					F		100	R	6	K	I			1	DISK									
04	F	PRINTER	O				F		120								PRINTER									
05	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 10

Program Identification: CHAING

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	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	CARDIN	AA			01															75	80		CUST							
02	I																				11	20	2	AMTPD							
03	I																														
04	I	MASTCUSTBB				02		7		C1																					
05	I																				1	6		CUSTNO							
06	I																				8	20		NAME							
07	I																				21	30	2	BALANC							
08	I																				37	50		ADDR							
09	I																				51	60		CITY							
10	I																				61	65		STATE							
11	I																				66	80		SHIPPR							
12	I		CC			03		7		C2																					
13	I																				1	6		CUSTNO							
14	I																														
15	I																														

Figure 133. Specifying Chaining to an Indexed Sequential File (Part 1 of 2)

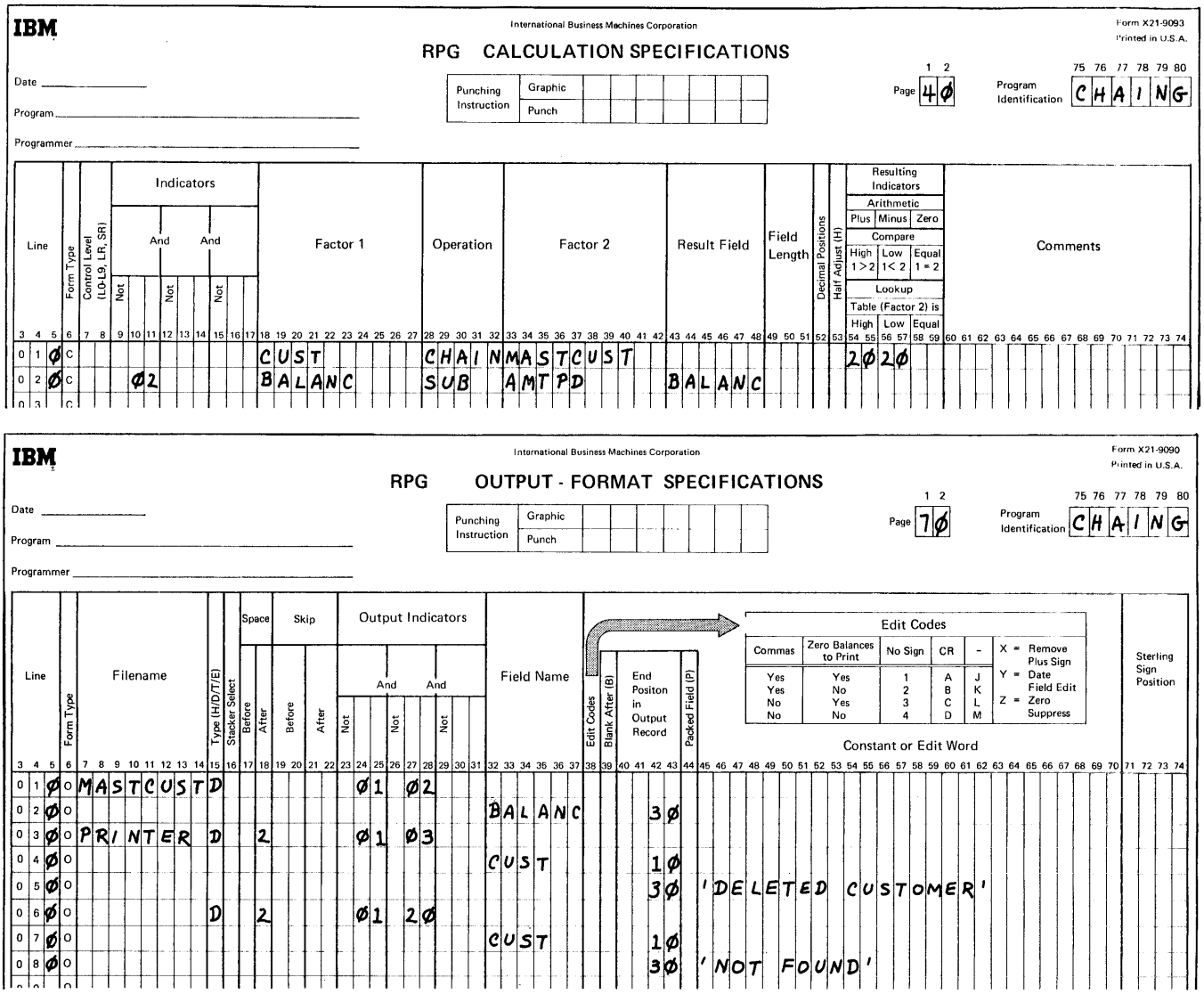


Figure 133. Specifying Chaining to an Indexed Sequential File (Part 2 of 2)

The MASTCUST file has two types of records and is typical of indexed sequential files. Records with a 1 in position 7 are active records and can be updated. Records with a 2 in position 7 are customer records which are no longer active. Through some error, a transaction may be processed for a customer who has been deleted from the file. The programmer normally includes coding to handle this condition as is done in lines 120 and 130.

The field BALANC in the MASTCUST record is the field to be updated.

The Calculation Specifications form shows the CHAIN operation using the field CUST as the chaining field. The CHAIN operation brings the customer record into core storage for processing. MASTCUST is defined as the chained file. Indicator 20 will be turned on if no record is found in MASTCUST for the customer number in the transaction file. Assuming the record is found, the input fields described for the MASTCUST file are removed and the program returns to the next calculation statement.

Line 020 shows the subtraction to get the new balance field. This is only done when indicator 02 is on, denoting that an active record was found in MASTCUST.

The Output-Format Specifications form shows the coding necessary to update MASTCUST. The file is updated only when an active record was found. Only the fields which have been changed need to be put out. Thus, only BALANC is mentioned. The other fields for this record remain unchanged.

The printer file is used for error conditions. The first error condition shown is the case of a transaction affecting a deleted customer record. This was noted by indicator 03. Lines 030-050 provide a message for this condition.

The second error condition possible is when the transaction record contains a customer number for which no corresponding customer record exists on MASTCUST. This error is detected by indicator 20 being set on by the CHAIN operation. An appropriate message is then printed.

If desired, statements need not be included to handle error conditions. If the deleted customer record was not coded on lines 120 and 130 of the Input Specifications form and a deleted record was found, the RPG program halts with a message to the operator describing the undefined-record condition. The operator may then bypass the record or end the job.

If a record cannot be found on MASTCUST and there are no indicators in columns 54-57 of the CHAIN operation, the RPG program halts with a message to the operator describing the record-not-found condition. The operator may then bypass the processing of the transaction or end of the job.

Figure 134 depicts chaining to two files from the same transaction.

Figure 135 shows the coding for the operation.

The File Description entries are similar to Figure 133 except two indexed sequential files are being processed randomly.

The Input Specifications show the transaction record with the two fields that will be used in chaining. The two indexed sequential files both have active and inactive records as described in the previous example.

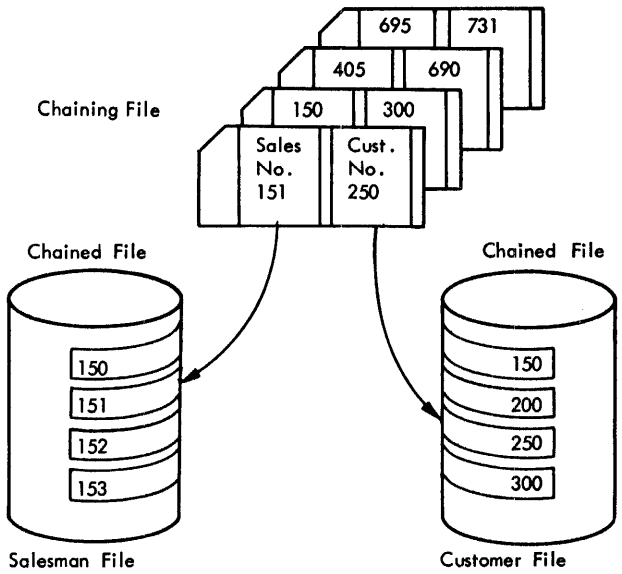


Figure 134. Chaining to Two Files

The Calculation Specifications show the two chaining operations. Indicators 20 and 21 are used to denote the NOT FOUND condition described in the previous example.

The Output-Format Specifications show the valid and invalid conditions. The valid condition exists when indicators 02 and 04 are on to denote that active records have been found. Lines 010-050 print this condition.

An invalid condition is noted by the lack of either indicator 02 or 04 being on. Therefore, an OR line has been coded. A description of the error condition is printed similar to that described in the previous example.

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: CHNTWO

Control Card Specifications

Line	Form Type	Core Size to Compile	Object Output	Listing Options	Core Size to Execute	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	
01	H														Refer to the specific System Reference Library manual for actual entries.

File Description Specifications

Line	Form Type	Filename	I/O/U/C	P/S/C/R/T	E	A/D	F/V	Block Length	Record Length	L/R	K/I	I/D/T or 1-9	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels (S, N, or E)	Name of Label Exit	Extent Exit for DAM	A	Number of Tracks for Cylinder Overflow	Number of Extents	N/U	Tape Rewind
02	F	CARD	I	P			F		80							READ42									
03	F	SLSMAN	I	C			F		60	R	3	K	I		1	DISK									
04	F	CUSTOMER	I	C			F		100	R	8	K	I		1	DISK									
05	F	PRINTER	O				F		120							PRINTER									
06	F																								

IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date _____

Program _____

Programmer _____

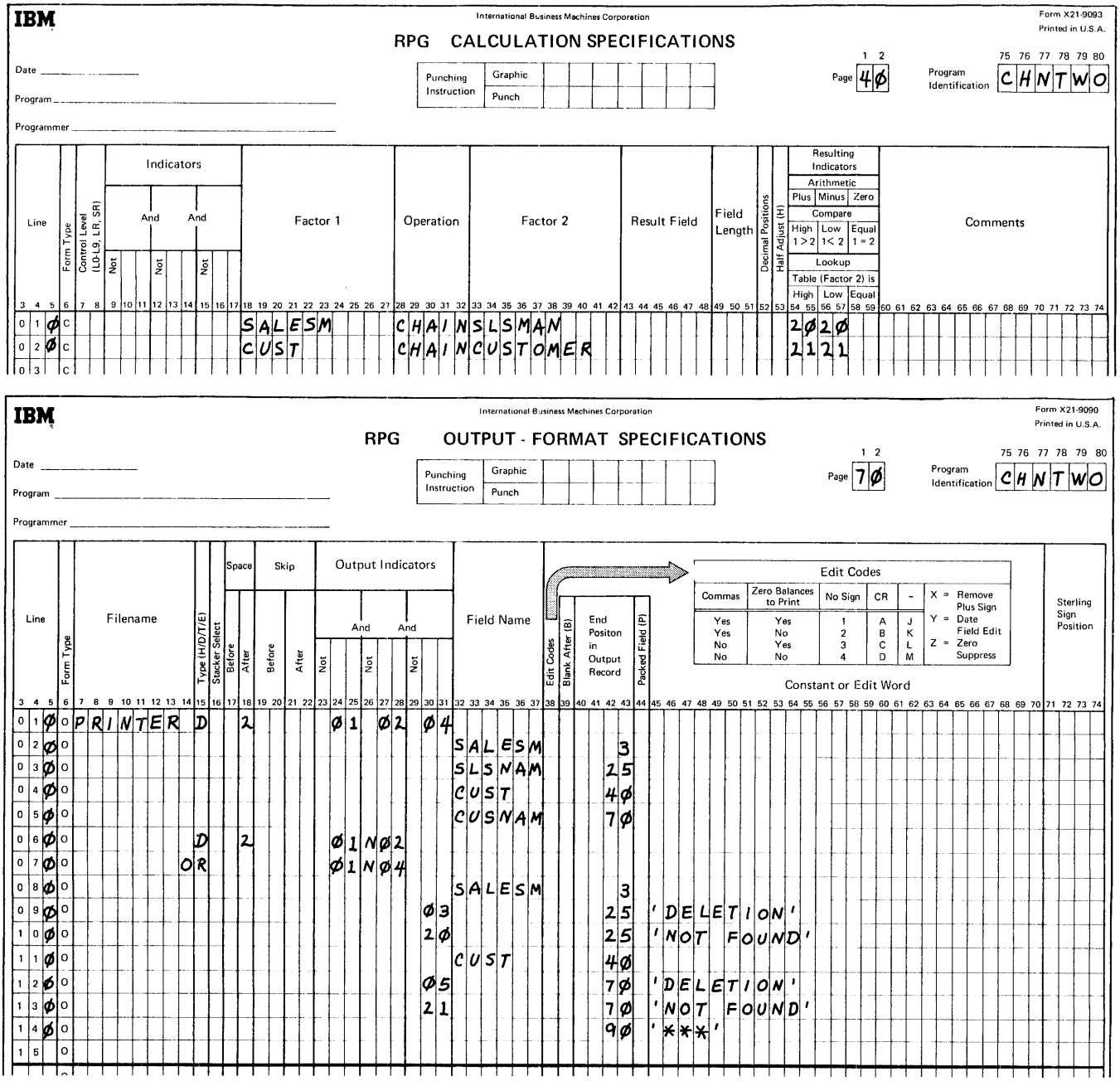
Punching Instruction: Graphic / Punch

Page: 10

Program Identification: CHNTWO

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	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	CARD	A			01																									
02	I																				15	17		SALESM							
03	I																				31	38		CUST							
04	I	SLSMAN	B			02		4	C	1																					
05	I		O	R		03		4	C	2																					
06	I																				5	20		SLSNAM							
07	I	CUSTOMER	C			04		9	C	1																					
08	I		O	R		05		9	C	2																					
09	I																				21	40		CUSNAM							
10	I																														

Figure 135. Chaining to Two Files, Coding for (Part 1 of 2)



IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page 40

Program Identification: CHNTWO

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And / Not	And / Not	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					SALESM	CHAINSLSMAN						2020	
02	C					CUST	CHAINCUSTOMER						2121	
03	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 70

Program Identification: CHNTWO

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

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select	Space: Before	Space: After	Skip: Before	Skip: After	Output Indicators: And / Not	And / Not	And / Not	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
01	O	PRINTER	D		2				01	02	04							
02	O											SALESM			3			
03	O											SLSNAM			25			
04	O											CUST			40			
05	O											CUSNAM			70			
06	O		D		2				01	N02								
07	O		OR						01	N04								
08	O											SALESM			3			
09	O										03				25		'DELETION'	
10	O										20				25		'NOT FOUND'	
11	O											CUST			40			
12	O										05				70		'DELETION'	
13	O										21				70		'NOT FOUND'	
14	O														90		'***'	
15	O																	

Figure 135. Chaining to Two Files, Coding for (Part 2 of 2)

In Figure 136, the card file is chained to the customer file. Within each customer record is a field which contains an account number. The account number is in turn used to chain to the account file.

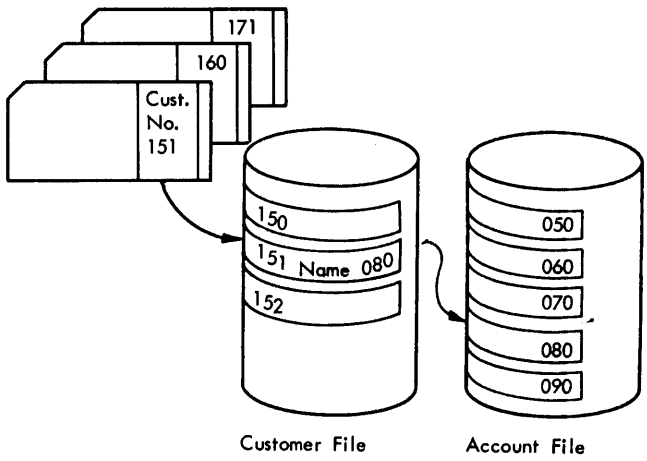


Figure 136. Using a Chained File as a Chaining File

Chaining to a Sequential Disk File

Chaining can be used to select a record from a sequential disk file. It differs from retrieving from an indexed sequential file in that a relative record number must be used instead of a key field. A relative record number means that the user must know which record he wants to process by describing it in terms of its position within the file. A request for number 151 will produce the 151st record in the file. The relative record number used in Factor 1 of the CHAIN operation must be defined as a numeric field of zero decimal positions.

The following describes some of the normal uses for this approach:

1. The actual code number (e.g., customer number) is used as the relative record number. (This depends greatly on the coding structure used.)

   A typical example of this is usually found in a file of department numbers. Many organizations have chosen a 2 or 3-digit number as the code structure for their departments. Assuming that there are 25 departments, the code structure might allow numbers in the range of 01 to 47. The user could then define a file to contain at least 47 records (probably a few more) and directly access the department records using the actual code number as the relative record number. An example of this is shown in Figure 137. The File Description Specifications show three files. The file DEPTFILE is a sequential file which is being processed randomly.

IBM International Business Machines Corporation

RPG CONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS

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

Date _____ Program _____ Programmer _____

Punching Instruction: Graphic / Punch

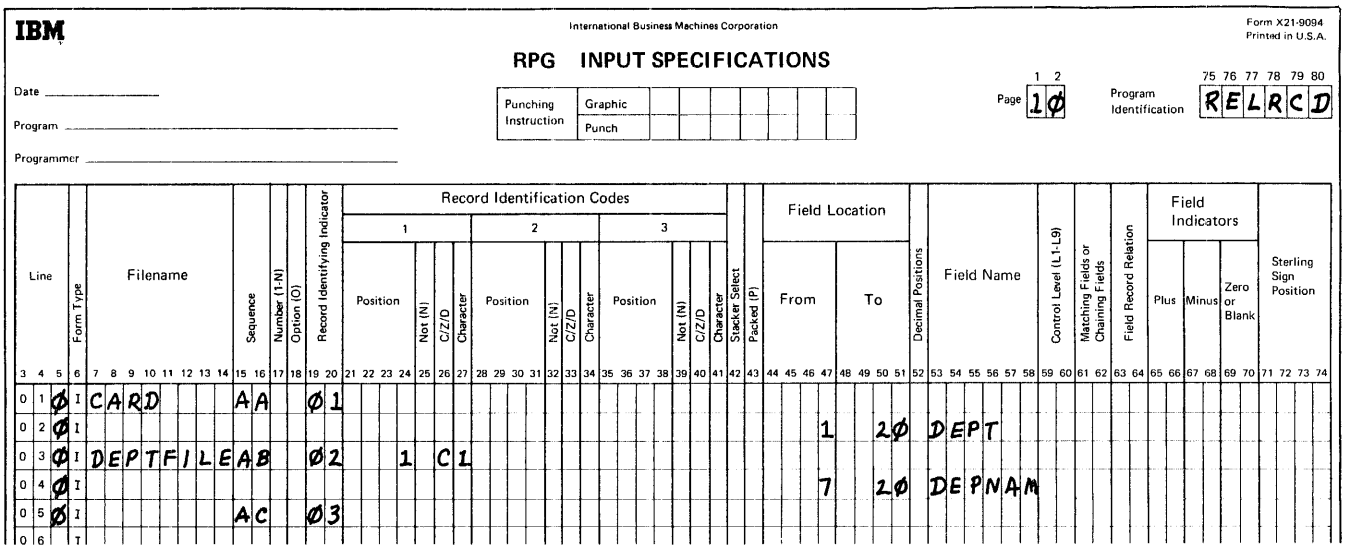
Page 01

Program Identification: RELRCD

File Description Specifications

Line	Form Type	Filename	I/O/U/C	P/S/C/R/T	E	A/D	F/V	Block Length	Record Length	L/R	K/I	I/D/T or 1-9	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels (S, N, or E)	Name of Label Exit	Extent Exit for DAM	A	N/U
02	F	CARD	I	P			F	80							READ42						
03	F	DEPTFILE	I	C			F	60		R					DISK						
04	F	PRINTER	O				F	120							PRINTER						

Figure 137. Chaining to a Sequential Disk File (Part 1 of 2)



IBM — International Business Machines Corporation — Form X21-9094 Printed in U.S.A.

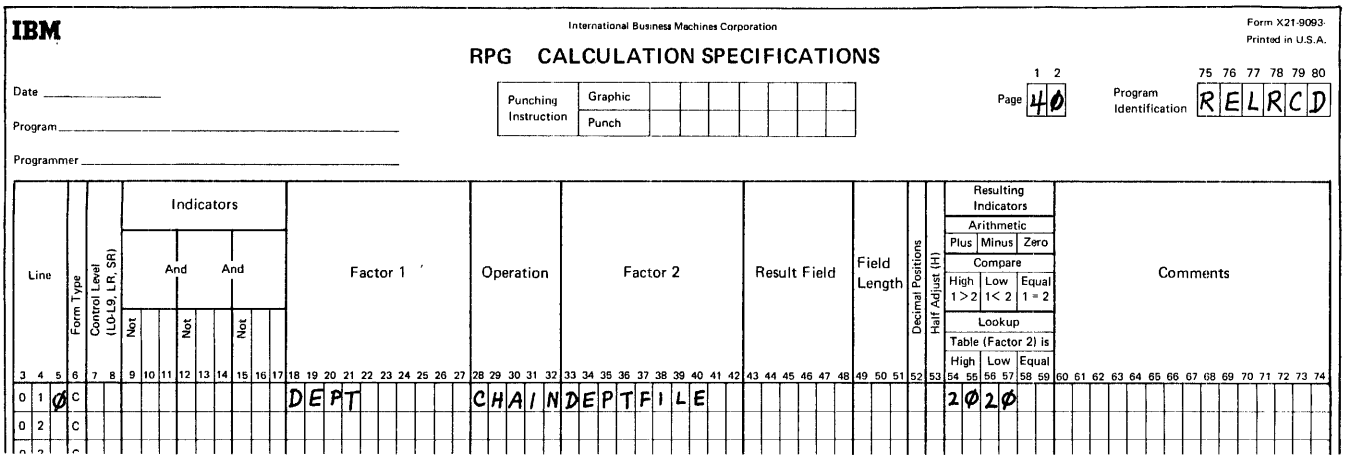
RPG INPUT SPECIFICATIONS

Date _____ Program _____ Programmer _____

Punching Instruction: Graphic / Punch

Page 10 — Program Identification: RELRCD

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	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	CARD	AA			01																									
02	I																				1	20		DEPT							
03	I	DEPTFILE	AB			02	1		C	1																					
04	I																				7	20		DEPNAM							
05	I		AC			03																									
06	T																														



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

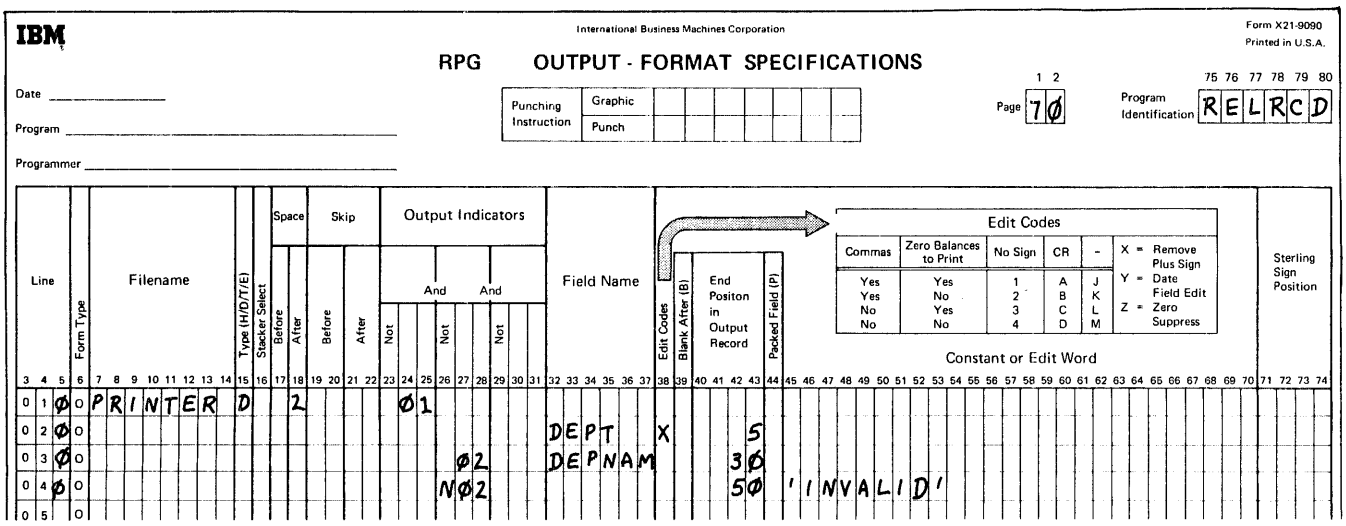
RPG CALCULATION SPECIFICATIONS

Date _____ Program _____ Programmer _____

Punching Instruction: Graphic / Punch

Page 40 — Program Identification: RELRCD

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And Not	Indicators: And Not	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 Low	Lookup Equal	Comments
01	C					DEPT	CHAIN	DEPTFILE											20	20		
02	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 70 — Program Identification: RELRCD

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select	Space: Before	Space: After	Skip: Before	Skip: After	Output Indicators: And Not	Output Indicators: And Not	Output Indicators: And Not	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
01	O	PRINTER	D		2				01									
02	O											DEPT	X		5			
03	O									02		DEPNAM			30			
04	O									N02					50		'INVALID'	
05	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 137. Chaining to a Sequential Disk File (Part 2 of 2)

The Input Specifications show the file CARD which contains the relative record number DEPT. The DEPTFILE requires record identification as the programmer must protect against retrieving an invalid department number. (This would be caused by a coding or keypunching error in the DEPT field. Indicator 02 is used for the valid departments.

The Calculation Specifications show the chaining operation. Indicator 20 would be on if the department number was higher than the total number of records in the file. It will also be on if DEPT is a negative number or zero. The Output Specifications show a line which is always printed. An invalid description is printed if it occurs. This type of file can also be processed sequentially using the record identification code to describe the valid records. The chief advantage of this technique is that relative record number processing is a fast retrieval method and the core storage requirements of the retrieval routine are less than for indexed sequential retrieval methods.

2. A table look-up approach may be used to locate the assigned relative record number. This method may require a large table in core.

3. A mathematical formula is used to convert the code number to a relative record number. (This method requires some randomizing technique and the user must be able to account for two or more code numbers that would randomize to the same number.)

Figure 138 shows the coding required for converting an item number to a relative record number by means of a randomizing technique.

The File Description Specifications describe the card file CARDIN, which contains the item number; MASTITEM file, which is a sequentially organized (blank in column 32) chained file (C in column 16) to be processed randomly (R in column 28); and a printer file.

The Input Specifications describe the field ITEM within the CARDIN file. The MASTITEM file contains an item number description and the relative record number (NEXT) of the next record in a series of records whose item numbers randomize to the same number. The field NEXT is being tested for a zero condition by indicator 22.

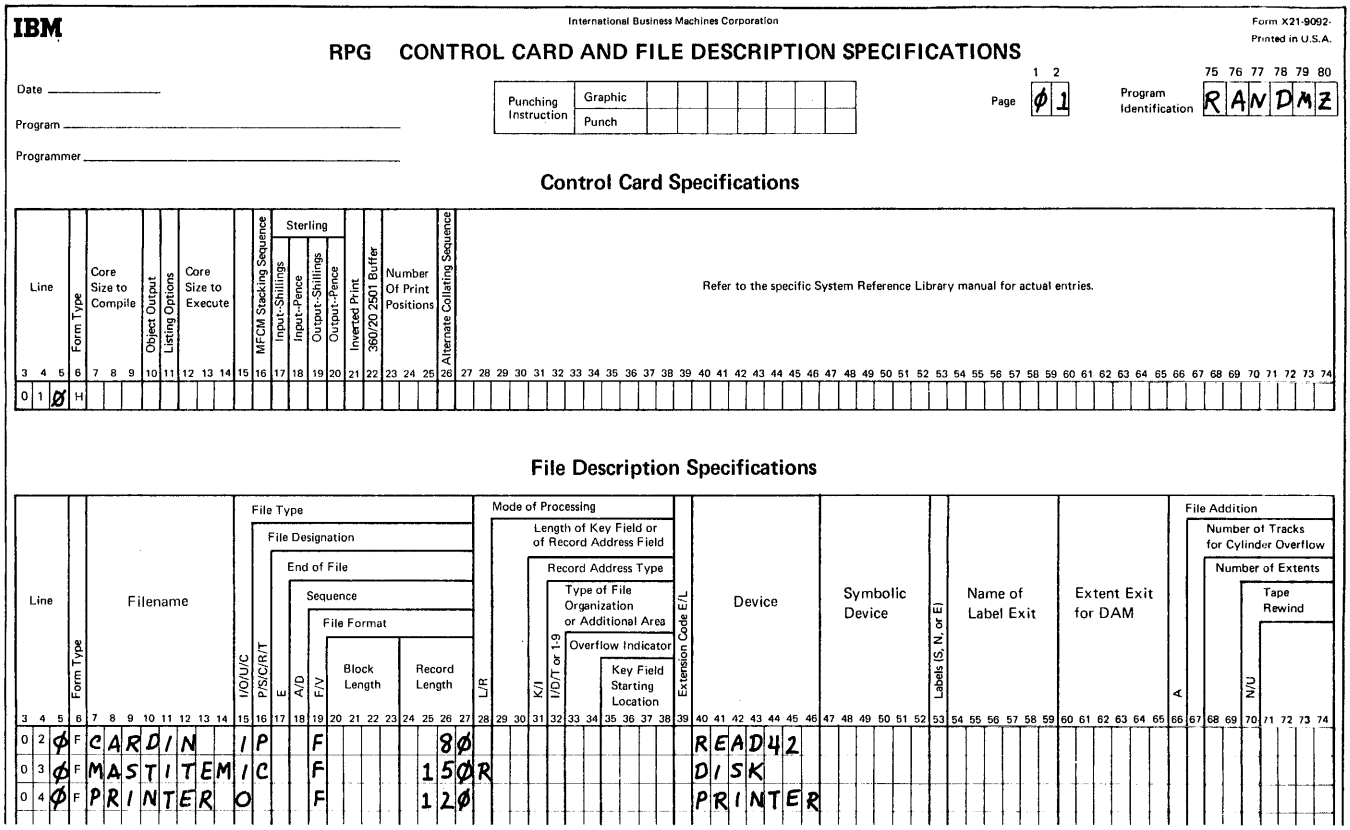


Figure 138. Randomizing Method of Obtaining a Relative Record Number (Part 1 of 2)

IBM International Business Machines Corporation Form X21-9094 Printed in U.S.A.

RPG INPUT SPECIFICATIONS

Date _____ Program _____ Programmer _____

Punching Instruction: Graphic / Punch

Page 10 Program Identification RANDMZ

Line	Form Type	Filename	Sequence	Number (1-N) Option (O)	Record Identifying Indicator	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	Packed (P)	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank	Sterling Sign Position
01	I	CARDIM	AA		01																								
02	I																			1	60		ITEMA						
03	I	MASTITEMBB			02																								
04	I																			1	60		ITEMB						
05	I																			7	20		DESCRP						
06	I																			101	1040		NEXT				22		

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

RPG CALCULATION SPECIFICATIONS

Date _____ Program _____ Programmer _____

Punching Instruction: Graphic / Punch

Page 40 Program Identification RANDMZ

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And Not	And Not	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 Low	Lookup Equal	Comments
01	C					ITEMA	MULT	ITEMA	SAVE12	12	0											
02	C						MOVE	SAVE12	SAVE8	8	0											
03	C						MOVEL	SAVE8	SAVE4	4	0											
04	C					SAVE4	COMP	5000									31					
05	C		31			SAVE4	SUB	5000	SAVE4													
06	C					LOOP	TAG															
07	C					SAVE4	CHAIN	MASTITEM									2222					
08	C					ITEMA	COMP	ITEMB												32		
09	C		N32	N22			MOVE	NEXT	SAVE4													
10	C		N32	N22			GOTO	LOOP														

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 70 Program Identification RANDMZ

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators: And Not	And Not	And Not	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
01	O	PRINTER	D			2			01									
02	O											ITEMA	X		10			
03	O									32		DESCRP			30			
04	O									22					50		'NOT FOUND'	

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 138. Randomizing Method of Obtaining a Relative Record Number (Part 2 of 2)

The Calculation Specifications show the randomizing formula and the chaining technique. The item number is squared (line 010) and the middle four digits of the result are extracted. This is accomplished by a MOVE and MOVEL operation (lines 020-030). The result (SAVE4) is then tested to see if it exceeds 5000. If so, 5000 is subtracted from it. These steps limit the size of the file to 5000 records.

Lines 070-100 show the chaining technique. SAVE4 is chained to MASTITEM (line 070). Line 080 turns on indicator 32 if the item from the card file is equal to the item in the disk file. Line 090 moves the next relative record number to be used to SAVE4 if indicators 32 and 22 are off. Line 100 branches to LOOP to repeat the chaining operation if required.

Either the proper record is found (indicator 32 is on) or an invalid item number exists.

The Output-Format Specifications show a line printed with the ITEMA field. The description is printed if the correct record was found. The words NOT FOUND are printed if the chaining technique cannot produce a record.

Files Spanning More Than One Disk Cartridge

If a logical file spans more than one disk cartridge, it cannot be called one file in 1130 RPG. However, it is possible to process such a file by splitting it into two files and chaining to each file separately.

Figure 139 shows a simple calculation test for this type of chaining. Customer #5000 is the last customer on the first half of the file. Both files are online during the job.

It is also possible to process this logical file sequentially by describing the first half as the primary file and the second half as the secondary file.

Use of CHAIN Operation Code

The chaining field may be specified by use of the CHAIN operation in Calculations Specifications. This enables the chaining function to be:

1. Performed with a field whose value is calculated or read through an input file.
2. Performed at either detail or total time in the program cycle.
3. Conditioned on the results of another calculation or test.
4. Performed more than once on one chaining field.
5. Performed more than once on one or more chained files.
6. Performed on nine or fewer chained files in one program.

Refer to *CHAIN* under Calculations Specification Form for further information.

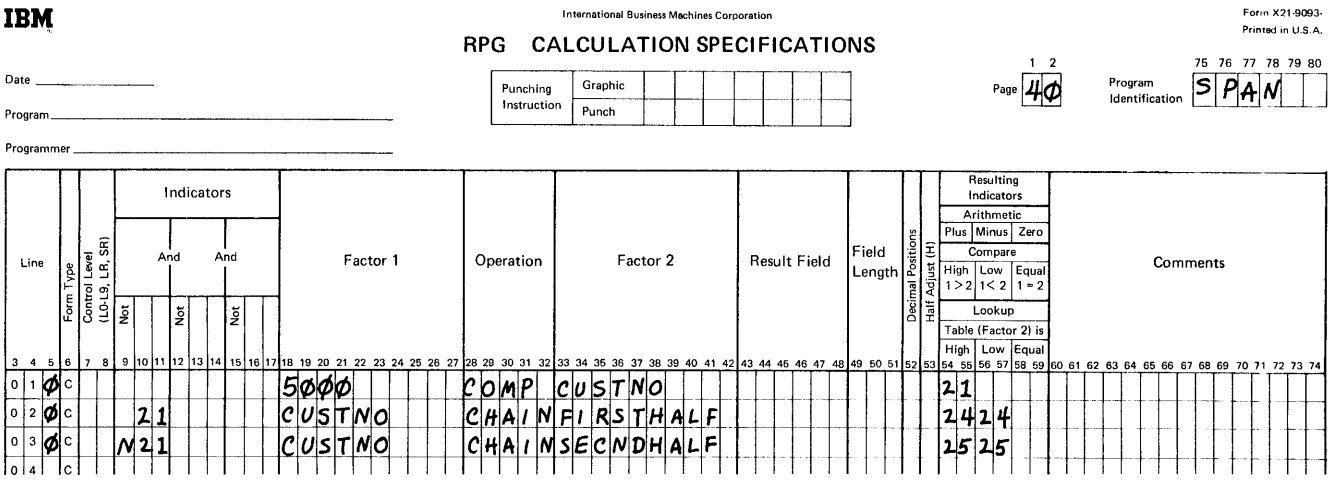


Figure 139. Chaining for a Spanned File

Application Uses of the CHAIN Operation

The uses of the CHAIN operation are varied. It allows processing basic jobs where a master file is retrieved randomly, or complex jobs where the same file is accessed more than once for the same transaction.

The following are some of the functions which may be performed.

1. Simple chaining to randomly processed transactions. (See Figure 133).
2. Chaining to two different files from the same transaction (See Figure 134).
3. Chaining to one indexed sequential file, then using a field in that file to chain to a second indexed sequential file (See Figure 135).
4. Chaining to the same file more than once from the same input record.
5. Handling of cards which contain multiple transactions commonly called spread cards.
6. Handling inventory problems which involve substituting items when the original requested item is out of stock.
7. Handling Bill of Material problems where one input record causes several chaining requests.
8. Accessing Bill of Material items and printing a list of them from random input.

Use of C1-C3 Codes

This is an alternate method of chaining that can only be used on an indexed sequential file.

Input fields may be defined as chaining fields by entering C1 through C3 on the Input Specifications line (columns 61-62) for that field.

The C1-C3 codes differ from the CHAIN operation in that they apply only to input fields, and cause chaining to occur only before detail time in the program cycle.

There is no specific relationship between levels C1 to C3, other than specifying three possibilities for chaining fields and the order in which chaining will occur.

The same input field may chain, with C1-C3 codes, to more than one file if it is defined more than once on the Input Specifications form with a different chaining code every time.

Two chaining files may chain to the same file, using C1-C3, if the same chaining code is specified for each file on the Input Specifications form.

Figure 140 is an example of chaining using C1-C3. The File Description entry for the chaining file CARDIN has an E in the Extension code (column 39), denoting that the file will be further described on the Extension Specifications. The other File Description entries are similar to those of previous examples of chaining to an indexed sequential file.

The Extension Specifications have an entry of AA in Record Sequence of the chaining file (columns 7-8). This corresponds with columns 15-16 of the chaining file CARDIN on the Input Specifications form. C1 is shown in the Number of the chaining field (columns 9-10). This corresponds with the entry in columns 61-62 of the Input Specifications form for the chaining field.

The Input Specifications show the chaining field CUST in the file CARDIN. MASTCUST has both active and inactive records.

No Calculation Specifications are required for this job.

The Output Specifications show the valid conditions in lines 010-030 and the invalid conditions in lines 040-070. The valid condition is denoted by indicator 02 being on. The invalid condition has indicator 02 off. If indicator 03 is on, a deleted master record was found.

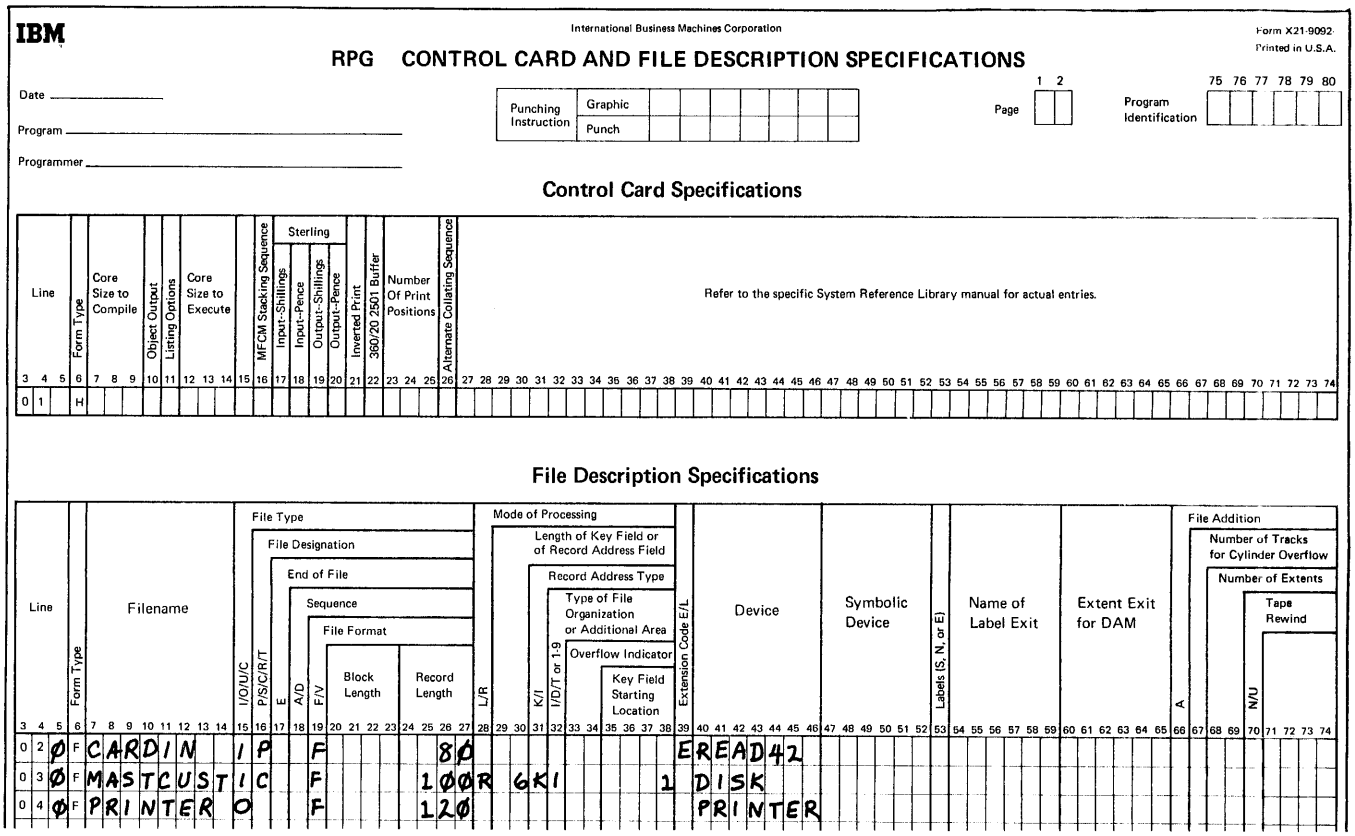
Using C1-C3, the programmer cannot prevent a system halt when a no-record-found condition exists.

Chaining with Record Address (RA) File

This is a seldom-used function and is supported as a modification of processing between limits. Only an indexed sequential file can be processed in this manner. The coding is similar to processing within limits except that the indexed sequential file is processed randomly (R in column 28). The record address fields may be single or multiple entries per record. If there are multiple entries, they must be adjacent. The next record is read from an RA file when a blank is found. The RA file may be on cards or disk.

Split Chaining Fields

C1-C3 Codes: Several fields that are not in adjoining positions in an input record can be specified as one chaining field. The fields are specified with the same chaining code (C1, for example) on the Input Specifications form. The first field defined on the input form is placed in the high-order position, and the last field is placed in the low-order position. The fields are placed in the same sequence as they are defined on the Input Specifications form.



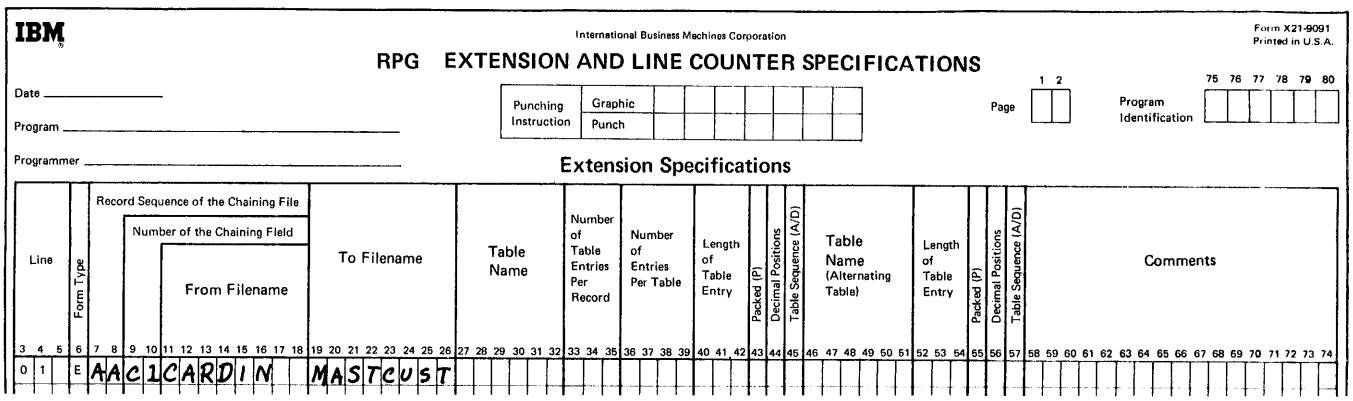


Figure 140. Chaining Using C1-C3 (Part 1 of 2)

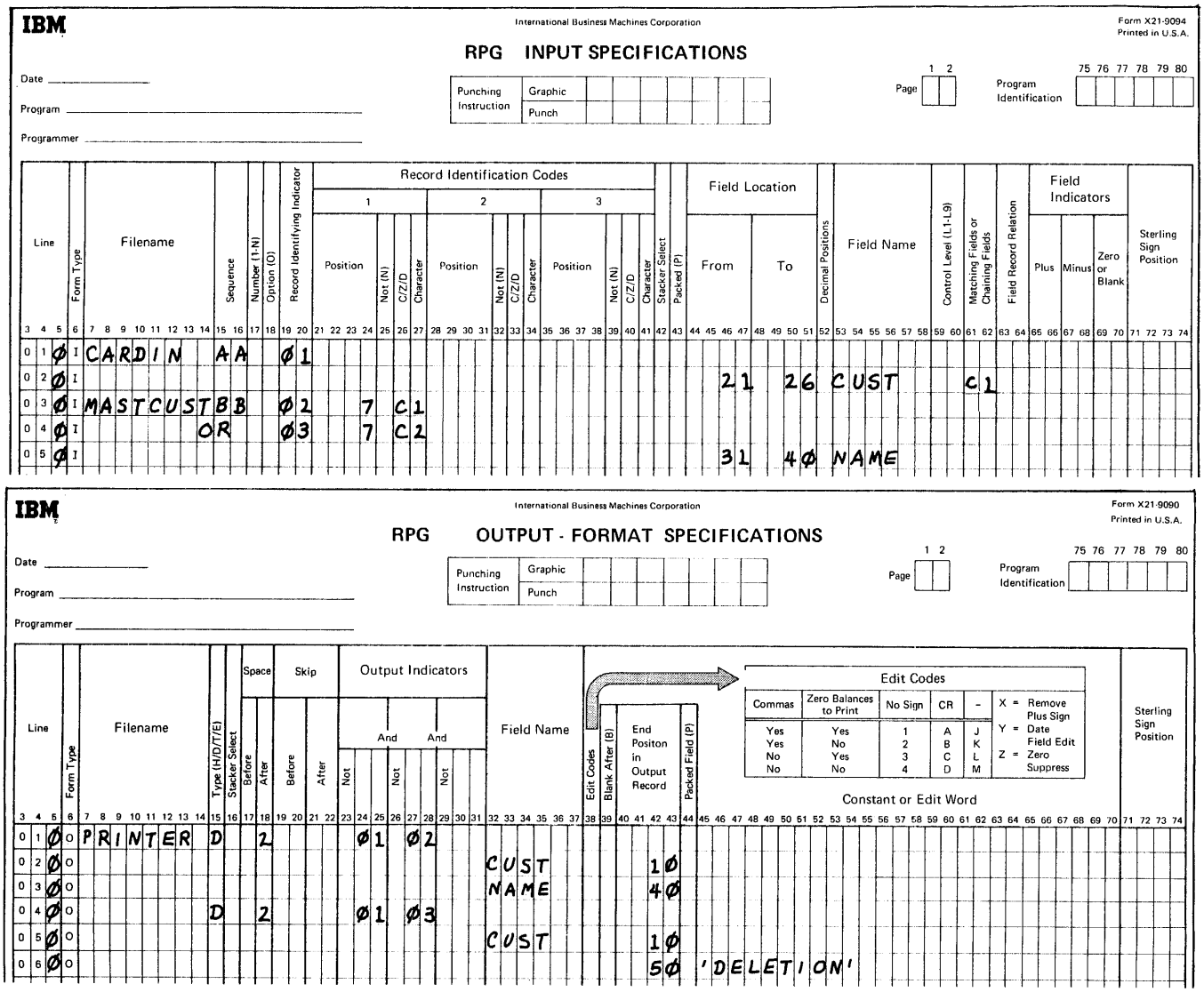


Figure 140. Chaining Using C1-C3 (Part 2 of 2)

CHAIN: Several fields can be collected into one chaining field through the use of move operations, and the resulting field used with the CHAIN operation. Thus, the order in which the several fields are collected is completely under the control of the programmer.

An example of this is shown in Figure 141. In this case the chaining field is made up of two fields from the same card, but they are in different locations. The Input Specifications show their locations. The Calculation Specifications show how they are placed together to form one field.

The field SAVE will be the chaining field. The field HIGH is moved to the high order (leftmost) positions of the field by a MOVEL operation. The field LOW is moved to the low order (rightmost) positions of the field by a MOVE operation. Then the CHAIN operation is used to retrieve the appropriate record.

Processing Between Limits

Only an indexed sequentially organized file can be processed between limits. This is accomplished by using a record address file to specify the limits. The record address file is never defined on the Input Specifications.

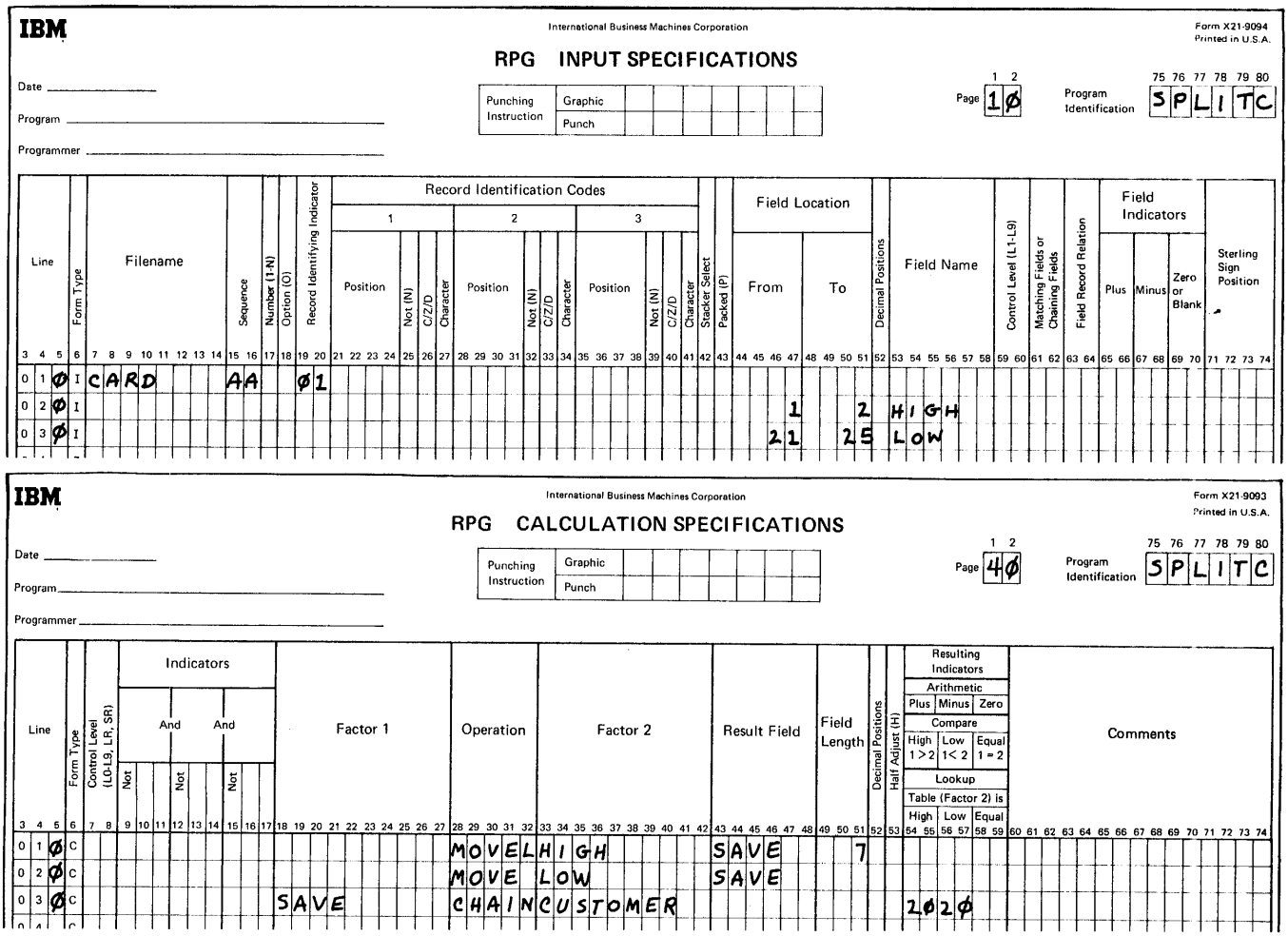


Figure 141. Collected Chaining Field

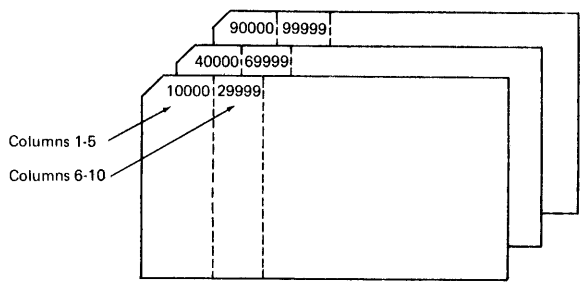
The file being limited is processed sequentially. It can be used in a single input program or with multiple inputs and matching records.

The limits provided do not have to be actual key numbers to be found on the indexed sequential file. They can be any numbers and can be lower or higher than the lowest or highest key fields on the file.

It is possible to process limits which overlap as long as the matching records are not used.

For example, a disk file contains customer records. The file is sequenced on customer number (5 digits). The first digit of customer number denotes the type of account (e.g., all customer numbers between 10000 and 19999 are a type 1 account).

A report is desired which prints customer types 1, 2, 4, 5, 6 and 9. This is accomplished by describing a record-address file of the appropriate limits, as follows:



This is shown in Figure 142. The beginning and ending customer number of each record must be specified in columns 1-5 and 6-10 respectively.

The File Description Specifications describe RAFLIMIT as the record address file, MASTCUST as an indexed sequential file on disk, and PRINTER as the output file.

The Extension Specifications indicate that RAFLIMIT is being used to define the records to be processed in the file named MASTCUST.

The Input Specifications describe the fields needed in the customer record. Record identifying indicator 02 is used to avoid deleted records in the customer file that should not be processed.

The Calculation and Output-Format Specifications describe the needed results.

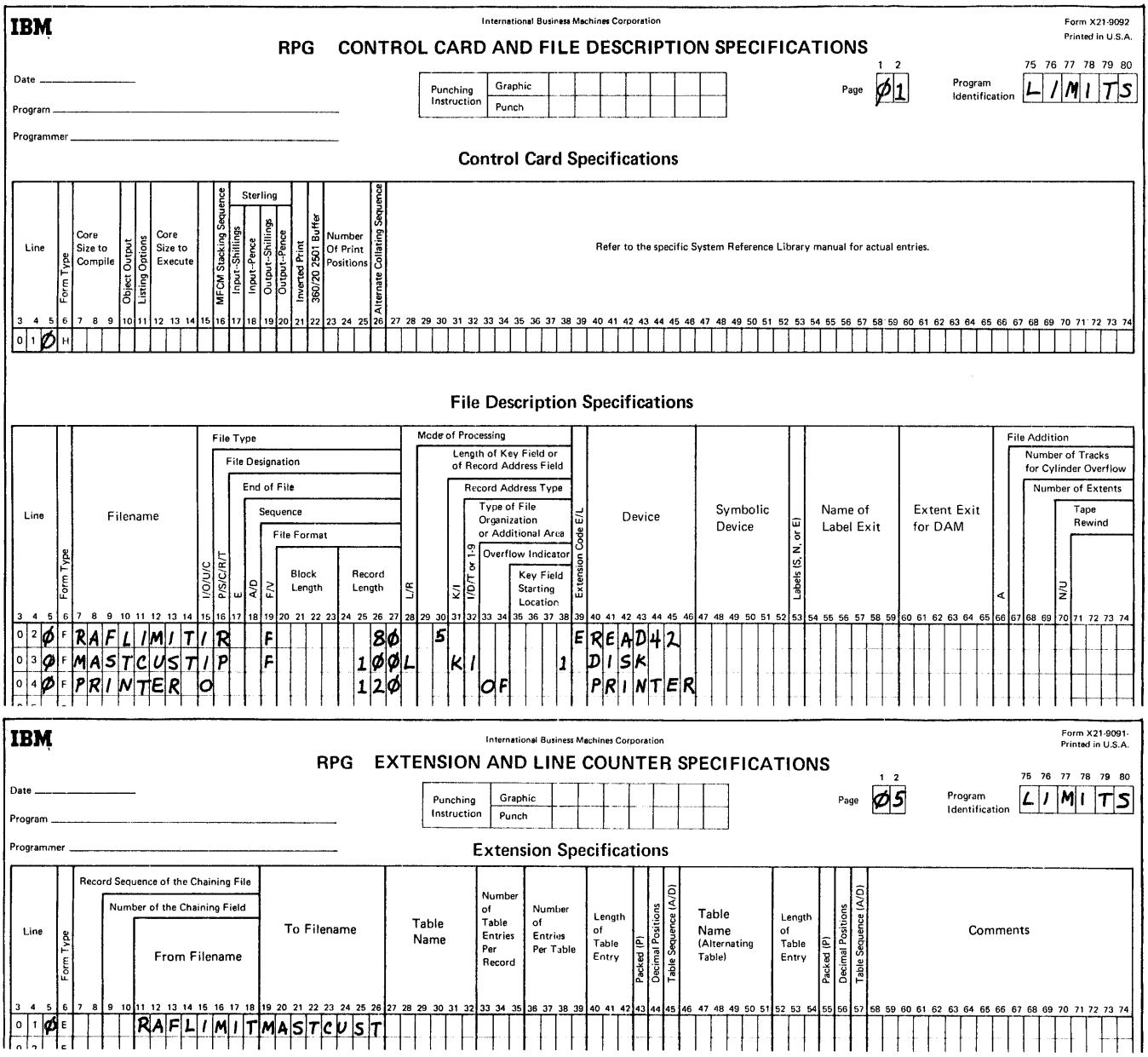
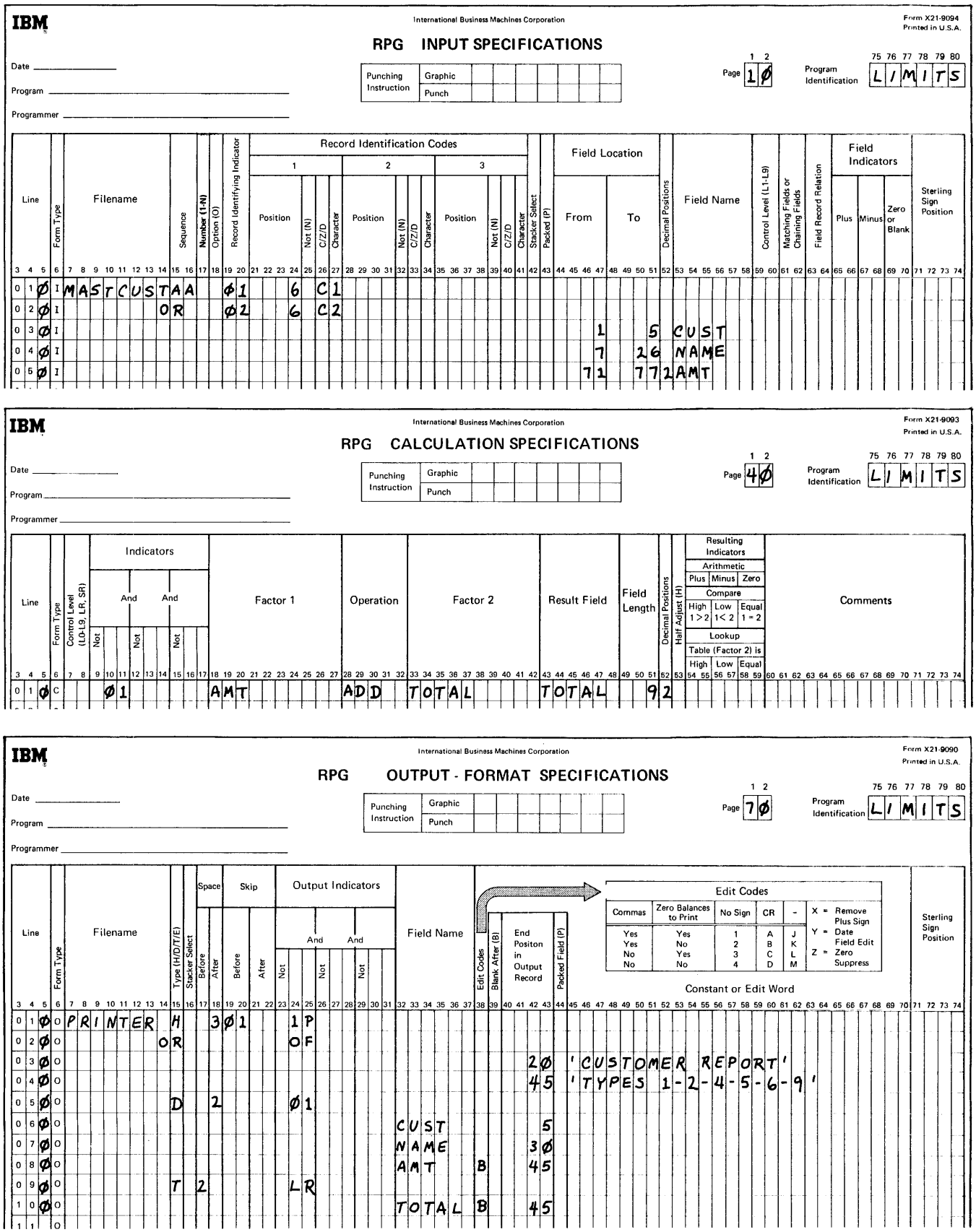


Figure 142. Processing Between Limits (Part 1 of 2)



RPG INPUT SPECIFICATIONS

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	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	MASTCUSTAA				01		6	C	1																					
02	I	OR				02		6	C	2																					
03	I																				1	5		CUST							
04	I																				7	26		NAME							
05	I																				71	771		AMT							

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And Not	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		01		AMT	ADD	TOTAL	TOTAL	92												

RPG OUTPUT - FORMAT SPECIFICATIONS

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators And Not	Output Indicators And Not	Output Indicators And Not	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
01	O	PRINTER	H		3		01		1P									
02	O	OR							OF									
03	O														20		'CUSTOMER REPORT'	
04	O														45		'TYPES 1-2-4-5-6-9'	
05	O		D		2				01									
06	O											CUST			5			
07	O											NAME			30			
08	O											AMT		B	45			
09	O		T		2				LR									
10	O											TOTAL		B	45			

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 142. Processing Between Limits (Part 2 of 2)

The following methods are available to RPG for processing one file against another:

1. Sequentially, using the matching record technique.
2. Sequentially between limits, using a record-address file.
3. Randomly, using the chaining technique.

Figure 143 shows these processing possibilities.

A sequential file is processed sequentially and controls the processing of records in:

1. Another sequential file. Both files are processed sequentially, using matching records to govern processing.
2. An indexed sequential file. If the file is processed sequentially, the matching-record technique is used to control processing of the indexed sequential file. If the file is processed randomly, chaining fields in the sequential file specify which records in the indexed sequential file are to be processed.

An indexed sequential file may be processed sequentially or randomly and controls processing records in:

1. A sequential file. The records in both files will be processed sequentially, using matching records to control processing.
2. Another indexed sequential file. If the file is processed sequentially, matching records are used to control processing. (Both files are processed sequentially.) If the file is processed randomly, chaining fields are used to control processing.

	Type of File Organization		
Mode of Processing	Card Files (Sequential)	Disk Files (Sequential)	Disk Files (Indexed Sequential)
Sequential - - As a single file or multiple files using matching records.			
• Entire file	Yes	Yes	Yes
• Limits of the file as described in a record address file	No	No	Yes
Random or Chaining			
• CHAIN Operation	No	Yes- - Using a relative record number	Yes- - Using a key field
• C1-C3 Indicators	No	No	Yes- - Using a key field
• Record address file	No	Yes- - Using a relative record number	Yes- - Using a key field

Figure 143. Processing Mutiple Input Files

This section describes entries that may be made on the File Description Specification for various jobs. The following numbers and descriptions correspond to the circled numbers in Figure 144.

1. Card to print: A card file is read in and a printed report is made.
2. Card to disk: A card file is read in and a sequentially organized disk file called SALES is created.
3. Disk to print: The same sequentially organized disk file (SALES) that was created by 2 is now used to print a report.
4. Card and disk to print: A card file is matched to a sequentially organized disk file (SALES) and a report is prepared. The card file is the primary file (P in column 16). Both files are in ascending sequence.
5. Disk update sequentially and print: The sequentially organized file (SALES) is matched against a sequentially organized file (YTDSALES). The SALES file is the primary file (P in column 16). The YTDSALES file is the secondary file (S in column 16) and is also to be updated (U in column 15). RPG allows updating only at detail time when processing sequentially. Therefore, the transaction file (SALES) is processed first for matched records and the summarized data is posted to the YTDSALES file. Both files must be in the same sequence.
6. Disk update randomly and print: This is the same job as in 5 except that the YTDSALES file is being updated randomly. It is called a chained file (C in column 16) and the R in column 28 determines that it is processed randomly. Since YTDSALES is a sequentially organized file it can only be processed randomly by means of a relative record number. The SALES file must contain that number (or allow a randomizing formula).

   Since the YTDSALES file is being processed randomly it does not have to be in the same sequence as the SALES file. However, if it is, a faster job time will be achieved since there will be less disk access arm movement, and multiple transactions for the same master will cause only one updating cycle.
7. Create ISAM file: A card file is read in and an indexed sequentially organized file is created. This file has a record length of 115 (columns 24-27). The key field is 5 characters long (columns 29-30) and begins in position 1 (column 35-38). The K and I entries (columns 31-32) are necessary to describe an ISAM file with keys. The entry of 4000 (columns 47-52) describes the number of records to be loaded into the file.
8. Additions to an ISAM file: A card file contains additions to the file CUSTOMER which was created in 7. The CUSTOMER file is described identically except for two entries. The number of records to be loaded (columns 47-52) is left blank and an A for File addition (column 66) has been added. Additions to an ISAM file must be a separate program. No other function can occur on the CUSTOMER file in this program.
9. Sequentially updating an ISAM file: A disk file of transactions called DETAIL is matched against the ISAM file CUSTOMER. The DETAIL file is the primary file (P in column 16). When processed sequentially, a disk file can only be updated at detail time. Therefore, all transactions for a particular customer should be processed before updating occurs. Calling the file DETAIL, the primary file takes care of this. The CUSTOMER file is defined as an Update file (U in column 15). Both files must be in the same sequence.
10. Randomly updating an ISAM file: This is the same job as 9 except that the DETAIL file does not have to be in the same sequence as the CUSTOMER file. Processing is done randomly on the CUSTOMER file as described by the R (column 28). It is a chained file as described by the C (column 16) and it is to be updated (U in column 15).

    While the DETAIL file does not have to be in the same sequence as the CUSTOMER file, faster throughput will be achieved if it is. The reason for this is that 1130 RPG keeps a sector of the cylinder index to the ISAM file in core, which minimizes the disk access arm movement if the records are in sequence. In addition, multiple transactions will not cause multiple updating of the CUSTOMER file.
11. ISAM file to print: This job shows the CUSTOMER file being processed sequentially and a report being prepared.
12. Limiting an ISAM file: This is the same as 11 except that only specified limits of the CUSTOMER file are being processed. The limits are supplied by a card file called CARD. This file is described as an RA file (P in column 16). The extension sheet is called for (E in column 39) to describe the fact that the card file is being used to limit the CUSTOMER file. The CUSTOMER file is described as being processed with limits (L in column 28).

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: SAMPLE

File Description Specifications

Line	Form Type	Filename	File Type	File Designation	End of File	Sequence	File Format	Block Length	Record Length	L/R	Length of Key Field or of Record Address Field	K/I	Record Address Type	I/O/T or 1-9	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels (S, N, or E)	Name of Label Exit	Extent Exit for DAM	File Addition	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind
			I/O/U/C	P/S/C/R/T	E	A/D	F/V																	A		N/U	
020	F	CARD	I	P			F		80										READ42								
030	F	PRINTER	O				F		120						OF				PRINTER								
040	F																										
050	F	CARD	I	P			F		80										READ42								
060	F	SALES	O				F		60										DISK								
070	F																										
080	F	SALES	I	P			F		60										DISK								
090	F	PRINTER	O				F		120						OF				PRINTER								
100	F																										
110	F	CARD	I	P		A	F		80										READ42								
120	F	SALES	I	S		A	F		60										DISK								
130	F	PRINTER	O				F		120						OF				PRINTER								
140	F																										
150	F	SALES	I	P		A	F		60										DISK								
160	F	YTDSALES	U	S		A	F		100										DISK								
170	F	PRINTER	O				F		120						OF				PRINTER								
180	F																										
190	F	SALES	I	P			F		60										DISK								
200	F	YTDSALES	U	C			F		100	R									DISK								
210	F	PRINTER	O				F		120						OF				PRINTER								
220	F																										
230	F	CARD	I	P			F		80										READ01								
240	F	CUSTOMER	O				F		115		5	K	I					1	DISK	4000							
250	F																										
260	F	CARD	I	P			F		80										READ01								
270	F	CUSTOMER	O				F		115		5	K	I					1	DISK					A			
280	F																										
290	F	DETAIL	I	P		A	F		60										DISK								
300	F	CUSTOMER	U	S		A	F		115	R	5	K	I					1	DISK								
310	F																										
320	F	DETAIL	I	P			F		60										DISK								
330	F	CUSTOMER	U	C			F		115		5	K	I					1	DISK								
340	F	CUSTOMER	I	P			F		115		5	K	I					1	DISK								
350	F	PRINTER	O				F		120						OF				PRINTER								
360	F																										
370	F	CARD	I	R			F		80									E	READ01								
380	F	CUSTOMER	I	P			F		115	L	5	K	I					1	DISK								
390	F	PRINTER	O				F		120						OF				PRINTER								

Figure 144. Various File Description Entries

SAMPLE PROGRAM 1. SALES COMMISSION CALCULATION

A commission report is to be prepared using invoice summary cards (Figure 145). The cards are in sequence by salesman number. They are coded with a 5 in column 1; other cards maintained as part of the invoice file that are to be processed are coded as having no 5 in column 1.

The commission amount is calculated using the invoice amount, as follows:

- 10% commission for invoice amounts up to $10,000.
- 12% commission for invoice amounts above $10,000.

The invoice summary card for each salesman is calculated and detail printed. A total commission amount for each salesman and a final commission total amount are accumulated.

The specifications necessary to produce this report are shown in Figure 146. The report is shown in Figure 147.

RPG Control Card

This card must precede any input to 1130 RPG.

File Description Specifications Form

Line 050 describes the invoice summary cards that will be read into the system from an IBM 1442 Card Read/Punch. Line 060 describes the output file PRINT. The length of the print line is 120 characters. Comments are included to help document the program.

Indicator Summary Form

This form has been included to assist in documenting the program.

Input Specifications Form

The input specifications identify and describe the invoice summary card and those cards in the invoice file that are not to be processed (line 060 of the form). Resulting indicator 12 identifies the cards not used by the program. These cards are selected into stacker 2.

Calculation Specifications Form

Line 010 compares the invoice amount with $10,000.00. If the invoice amount is greater, resulting indicator 22 is turned on.

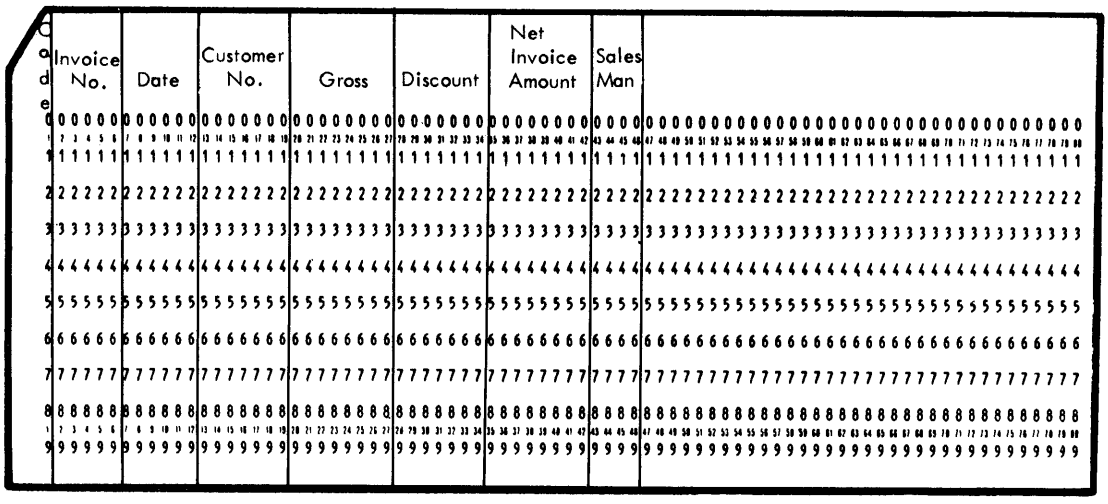


Figure 145. Input Card for Sample Sales Commission Report

IBM

International Business Machines Corporation

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

RPG CONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS

Date _____

Program SALES COMMISSION REPORT

Programmer _____

Punching Instruction: Graphic / Punch

Page 01

Program Identification SLSCOM

Control Card Specifications

Line	Form Type	Core Size to Compile	Object Output	Listing Options	Core Size to Execute	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	Refer to the specific System Reference Library manual for actual entries.
3 4 5	6	7 8 9	10	11	12 13 14	15	16	17	18	19	20	21	22 23 24 25	26	27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
0 1 Ø	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, or E)	Name of Label Exit	Extent Exit for DAM	File Addition	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind
			I/O/U/C	P/S/C/R/T	E	A/D	F/V			L/R		K/I I/D/T or 1-9										A		N/U	
3 4 5	6	7 8 9 10 11 12 13 14	15	16	17	18	19	20 21 22 23	24 25 26 27	28	29 30	31	32	33 34	35 36 37 38	39	40 41 42 43 44 45 46	47 48 49 50 51 52	53	54 55 56 57 58 59	60 61 62 63 64 65	66	67 68 69	70	71 72 73 74
0 2 Ø	F	*																							
0 3 Ø	F	* SALES COMMISSION REPORT																							
0 4 Ø	F	*																							
0 5 Ø	F	INPUT	I	P					8Ø								READ42								
0 6 Ø	F	PRINT	O						12Ø					OF			PRINTER								

IBM

International Business Machines Corporation

Form X21-9095
Printed in U.S.A.

RPG INDICATOR SUMMARY

Date _____

Program SALES COMMISSION REPORT

Programmer _____

Punching Instruction: Graphic / Punch

Page 05

Program Identification SLSCOM

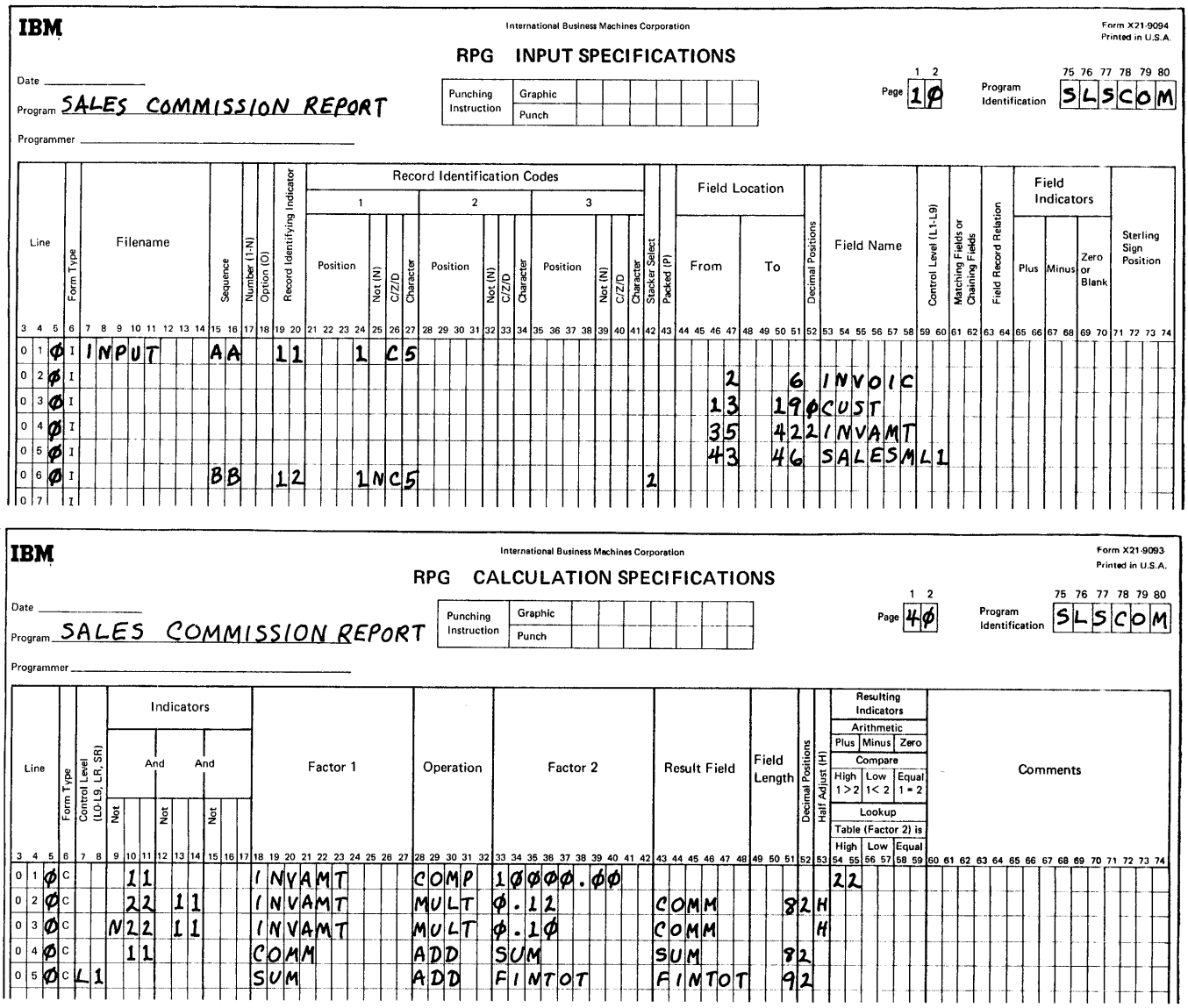
Circle Indicators Used:

Note: All indicators are not valid with all systems.

01	02	03	04	05	06	07	08	09	10	(11)	(12)	13	14	15	16	17	18	19	20
21	(22)	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	
(L1)	L2	L3	L4	L5	L6	L7	L8	L9		OA	OB	OC	OD	OE	(OF)	OG	OV		
H1	H2	H3	H4	H5	H6	H7	H8	H9		U1	U2	U3	U4	U5	U6	U7	U8		
Predefined Indicators:				IP	LO	LR	MR	H0											
M-C:	M1	M2	M3	M4	M5	M6	M7	M8	M9	C1	C2	C3	C4	C5	C6	C7	C8	C9	

Line	Form Type	Indicators: Record Identifying	Input Field	Calculation Result	Matching and Chaining	Control Level, Overflow, Halt and User	Comments
3 4 5	6	7 8 9 10 11 12 13	14 15 16	17 18 19	20 21 22		23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
0 1 Ø	F	* ID F C M L					FUNCTION OF INDICATORS
0 2 Ø	F	* 11					INVOICE SUMMARY CARDS
0 3 Ø	F	* 12					ERROR CARDS-STACK IN 2
0 4 Ø	F	*		22			INVOICE TOTAL EXCEEDS 10,000.00
0 5 Ø	F	*				L1	SALESMAN TOTAL
0 6 Ø	F	*			OF		OVERFLOW

Figure 146. Sales Commission Sample Program Coding (Part 1 of 3)



IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date

Program SALES COMMISSION REPORT

Programmer

Punching Instruction: Graphic / Punch

Page 1Ø

Program Identification SLSCOM

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	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
01Ø	I	INPUT	AA			11	1		C	5																					
02Ø	I																				2	6		INVOIC							
03Ø	I																				13	19	Ø	CUST							
04Ø	I																				35	42	2	INVAMT							
05Ø	I																				43	46		SALESMN	1						
06Ø	I		BB			12		1	N	C	5								2												
07	I																														

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date

Program SALES COMMISSION REPORT

Programmer

Punching Instruction: Graphic / Punch

Page 4Ø

Program Identification SLSCOM

Line	Form Type	Control Level (LO-L9, LR, SR)	Indicators: And Not	Indicators: And Not	Indicators: And Not	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus	Resulting Indicators: Arithmetic Minus	Resulting Indicators: Arithmetic Zero	Resulting Indicators: Compare High 1>2	Resulting Indicators: Compare Low 1<2	Resulting Indicators: Compare Equal 1=2	Resulting Indicators: Lookup Table (Factor 2) is High	Resulting Indicators: Lookup Table (Factor 2) is Low	Resulting Indicators: Lookup Table (Factor 2) is Equal	Comments
01Ø	C		11			INVAMT	COMP	1ØØØØ.ØØ											22			
02Ø	C		22	11		INVAMT	MULT	Ø.12	COMM		82	H										
03Ø	C		N22	11		INVAMT	MULT	Ø.1Ø	COMM			H										
04Ø	C		11			COMM	ADD	SUM	SUM		82											
05Ø	C	L1				SUM	ADD	FINTOT	FINTOT		92											

Figure 146. Sales Commission Sample Program Coding (Part 2 of 3)

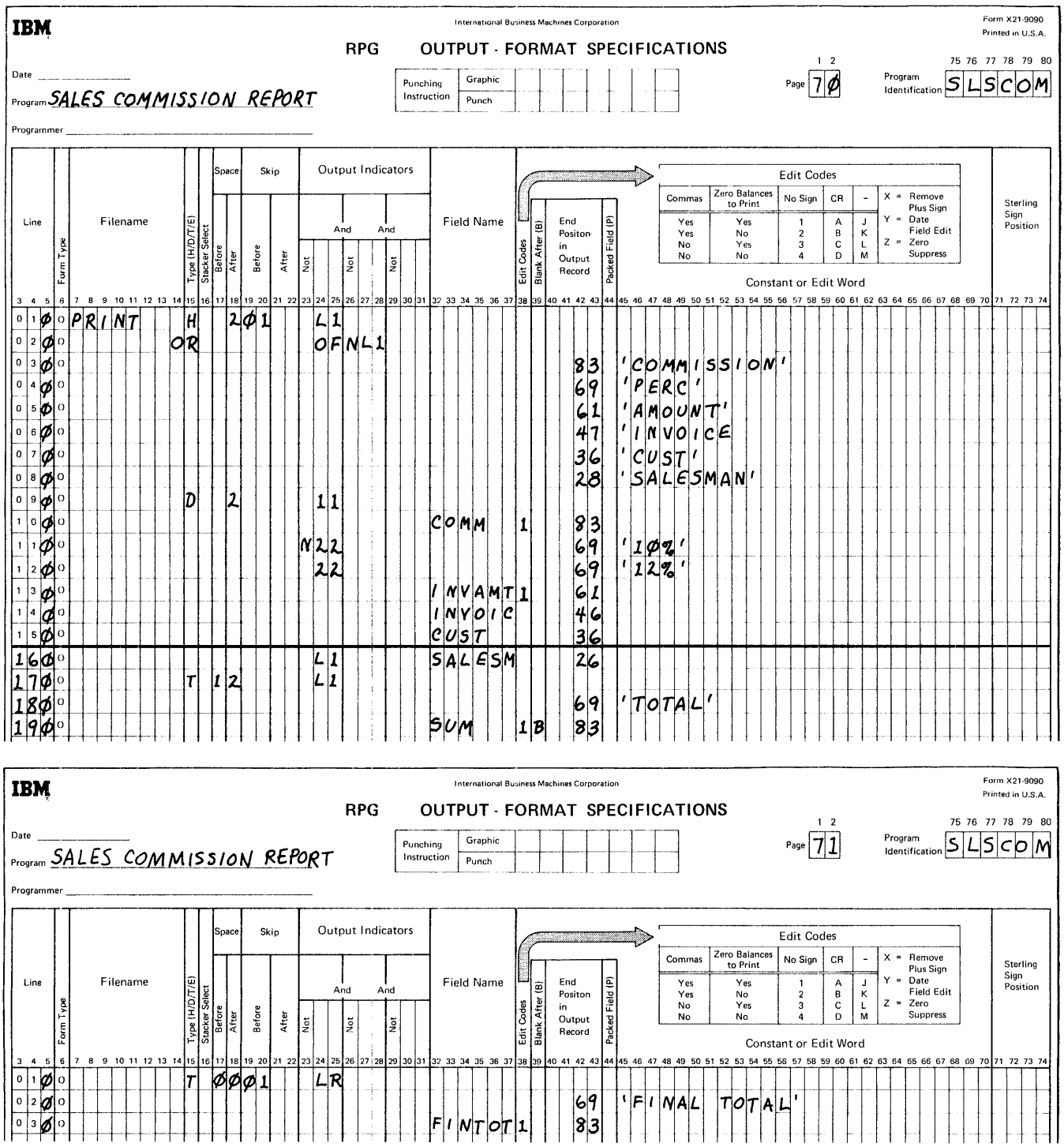
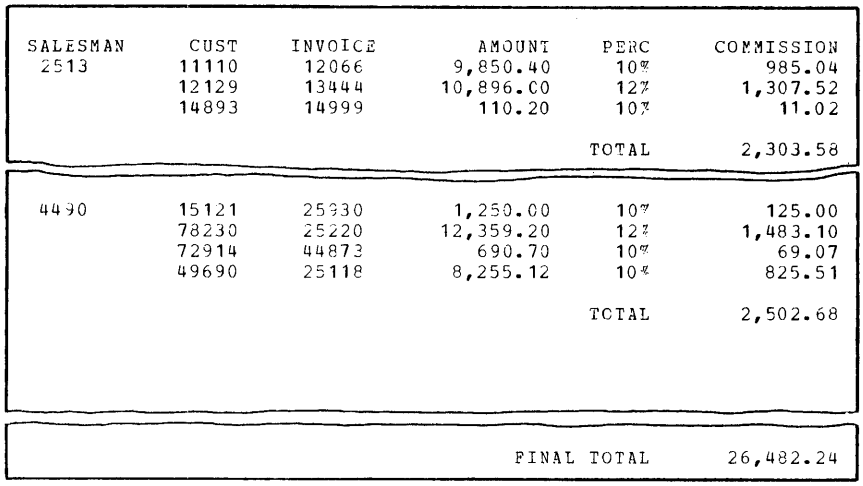


Figure 146. Sales Commission Sample Program Coding (Part 3 of 3)



SALESMAN	CUST	INVOICE	AMOUNT	PERC	COMMISSION
2513	11110	12066	9,850.40	10%	985.04
	12129	13444	10,896.00	12%	1,307.52
	14893	14999	110.20	10%	11.02
				TOTAL	2,303.58
4490	15121	25330	1,250.00	10%	125.00
	78230	25220	12,359.20	12%	1,483.10
	72914	44873	690.70	10%	69.07
	49690	25118	8,255.12	10%	825.51
				TOTAL	2,502.68
				FINAL TOTAL	26,482.24

Figure 147. Sample Sales Commission Report

Line 020 multiplies the invoice amount by 12 percent only if resulting indicator 22 is on. Therefore, this operation takes place only if the invoice amount is greater than $10,000.00.

Line 030 multiplies invoice amount by 10 percent if resulting indicator 22 is off. Therefore, this operation takes place only if the invoice amount is less than or equal to $10,000.00.

Line 040 adds the calculated amount COMM from each detail card to the field SUM. The field SUM is used to accumulate the commission amount for each salesman.

Line 050 is operated upon only when a level-1 control break occurs; i.e., when there is a change in salesman number. This line adds the accumulated commission amount for each group of salesmen cards (SUM) to the field FINTOT. The field FINTOT is used to accumulate a final total of all salesmen's commissions.

Output-Format Specifications Form

Lines 010 through 080 provide the specifications necessary for printing field headings. The heading is printed and the form is skipped to channel 1 each time one of the following conditions exists:

1. On control break (L2 in columns 24-25 on Line 010).
2. On overflow condition (OF in columns 24-25 on Line 020).

The indicators OFNL1 are required to omit a double printing of the heading when conditions 1 and 2 occur together.

Lines 090 through 160 cause the printing of the detail card information.

The remaining specification lines cause:

1. The printing of the level-1 total SUM, including the constant TOTAL (lines 170-190).
2. The printing of the final total, including the constant FINAL TOTAL (lines 010-030 on the lower half of the figure).

SAMPLE PROGRAM 2. CUSTOMER TRANSACTIONS

Two input files are used. The transaction file is a card file with fields as shown in Figure 148. Another input file (master customer file), which is on disk, contains information about the firm's customers (Figure 149).

The program processes the master customer file, using records from the transaction file to produce printed receipts. The master file is updated by producing a new master customer file. The coding required for this program is shown in Figure 150.

RPG Control Card

This card must precede any input to 1130 RPG.

Field	Label	Card Columns
Code	Minus (-) Zone, or Plus (+)	1
Customer name	NAME	8-29
Invoice date	MONTH	32-33
Invoice number	INVNO	34-38
Customer number	CUSTNO	39-43
State	STATE	44-45
City	CITY	46-48
Invoice amount	INVAMT	74-80

Figure 148. Transaction File for Sample Program 2

Field	Label	Location
Customer number	MASNUM	1-5
Customer name	MASNAM	6-27
Street	MASTRT	28-46
City	MASCTY	47-57
State	MASTAT	58-62
Customer balance	MASBAL	63-70
Date of last payment	PAYDAT	71-76
Date of last purchase	PAYPUR	77-82

Figure 149. Master Customer File for Sample Program 2

IBM International Business Machines Corporation — Form X21-9092 Printed in U.S.A.

RPG CONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS

Date _____ Program CUSTOMER TRANSACTIONS Programmer _____ Punching Instruction: Graphic / Punch Page 01 Program Identification 75 76 77 78 79 80: REPORT

Control Card Specifications

Line	Form Type	Core Size to Compile	Object Output	Listing Options	Core Size to Execute	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	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	L/R	Mode of Processing	Length of Key Field or of Record Address Field	Record Address Type	K/I	I/O/T or I/P	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code	E/L	Device	Symbolic Device	Labels (S, N, or E)	Name of Label Exit	Extent Exit for DAM	File Addition	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind
			I/O/U/C	P/S/C/R/T	E	A/D	F/V																			A		N/U	
02	F	*																											
03	F	* DAILY CUSTOMER TRANSACTION REPORT																											
04	F	*																											
05	F	MASTI	I	P	E	A	F		100												DISK								
06	F	TRANSIN	I	S		A	F		80												READ42								
07	F	MASTO	O				F		100												DISK								
080	F	MASTLISTO					F		120							OF					PRINTER								
	F																												

Figure 150. Coding for Sample Program 2 (Part 1 of 4)

IBM

International Business Machines Corporation

Form X21-9095
Printed in U.S.A.

RPG INDICATOR SUMMARY

Date

Program CUSTOMER TRANSACTIONS

Programmer

Punching Instruction	Graphic							
	Punch							

Page 05

Program Identification REPORT

Circle Indicators Used:

Note: All indicators are not valid with all systems.

Line	Form Type	Indicators: Record Identifying	Input Field	Calculation Result	Matching and Chaining	Control Level, Overflow, Halt and User
01	F	ID	F	C	M	L
02	F	01				
03	F	02				
04	F	03				
05	F	04				
06	F					L1
07	F				M1	

Line	Function of Indicators
01	FUNCTION OF INDICATORS
02	MASTER RECORD
03	POSITIVE TRANSACTION
04	NEGATIVE TRANSACTION
05	DATE CARD
06	CUSTOMER NUMBER
07	CUSTOMER NUMBER

IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date

Program CUSTOMER TRANSACTIONS

Programmer

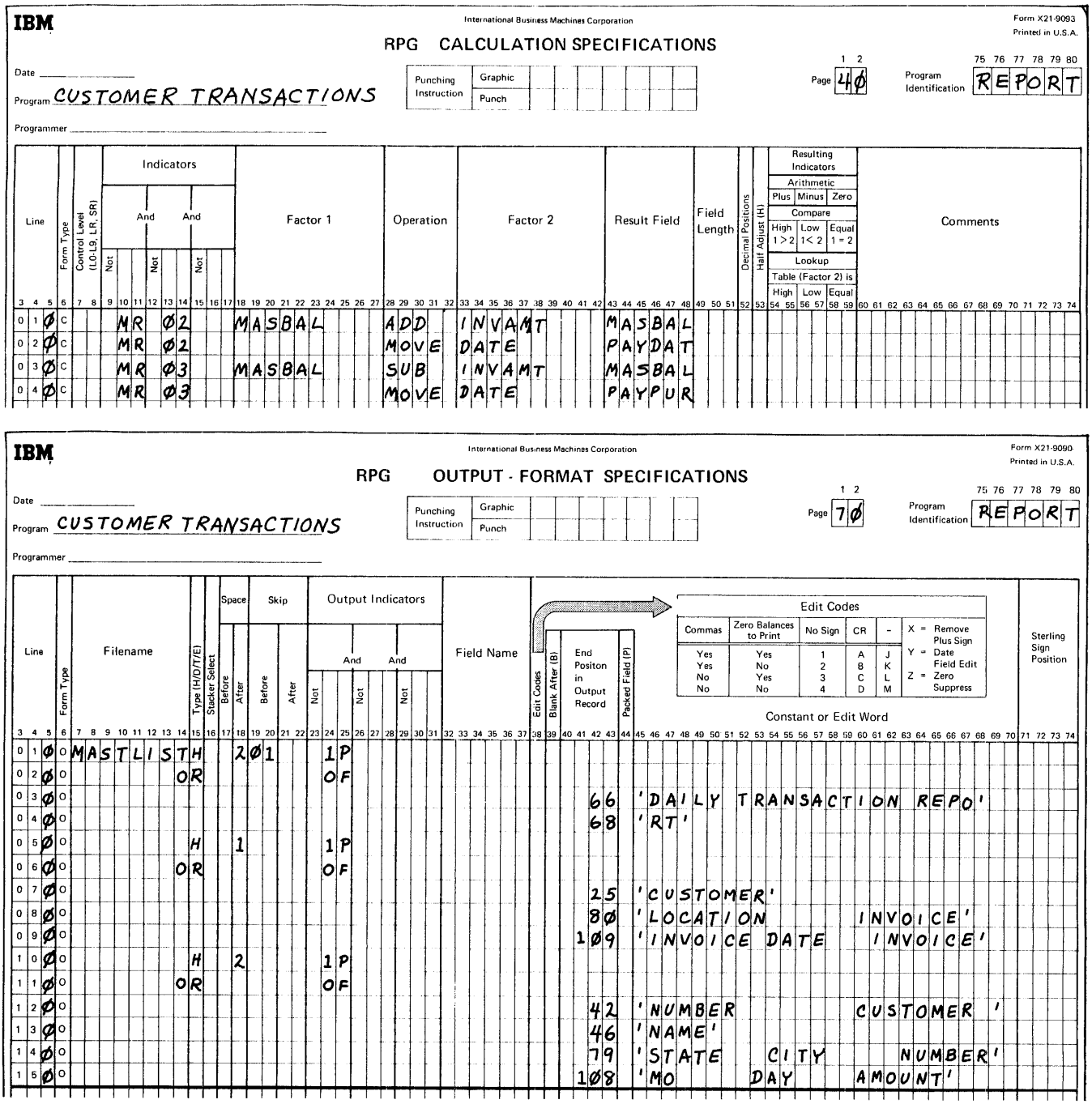
Punching Instruction	Graphic							
	Punch							

Page 10

Program Identification REPORT

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	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	MASTI	AA			01															1	10		RECORD							
02	I																				1	5		MASNUML1M1							
03	I																				63	70	2	MASBAL							
04	I																				71	76		PAYDAT							
05	I																				77	82		PAYPUR							
06	I	TRANSIN	AA			02		1	Z	-									2												
07	I			OR		03		1	Z	B																					
08	I																				8	29		NAME							
09	I																				30	31		MONTH							
10	I																				32	33		DAY							
11	I																				34	38		INVNO							
12	I																				39	43		CUSTNO	L1	M1					
13	I																				44	45		STATE							
14	I																				46	48		CITY							
15	I																														
16	I		BB			04		1	C	S											74	80	2	INVAMT							
17	I																														
18	I																				2	5		DATE							
	I																														
	I																														

Figure 150. Coding for Sample Program 2 (Part 2 of 4)



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

RPG CALCULATION SPECIFICATIONS

Date _____ Program CUSTOMER TRANSACTIONS Programmer _____

Punching Instruction: Graphic / Punch — Page 40 — Program Identification: REPORT

Line	Form Type	Control Level (LO,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
010	C		MR 02	MASBAL	ADD	INVAMT	MASBAL					
020	C		MR 02		MOVE	DATE	PAYDAT					
030	C		MR 03	MASBAL	SUB	INVAMT	MASBAL					
040	C		MR 03		MOVE	DATE	PAYPUR					

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

RPG OUTPUT - FORMAT SPECIFICATIONS

Date _____ Program CUSTOMER TRANSACTIONS Programmer _____

Punching Instruction: Graphic / Punch — Page 70 — Program Identification: REPORT

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

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select	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	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
010	O	MASTLIST	H			2	01		1P							
020	O		OR						OF							
030	O												66		'DAILY TRANSACTION REPO'	
040	O												68		'RT'	
050	O		H			1			1P							
060	O		OR						OF							
070	O												25		'CUSTOMER'	
080	O												80		'LOCATION INVOICE'	
090	O												109		'INVOICE DATE INVOICE'	
100	O		H			2			1P							
110	O		OR						OF							
120	O												42		'NUMBER CUSTOMER '	
130	O												46		'NAME'	
140	O												79		'STATE CITY NUMBER'	
150	O												108		'MO DAY AMOUNT'	

Figure 150. Coding for Sample Program 2 (Part 3 of 4)

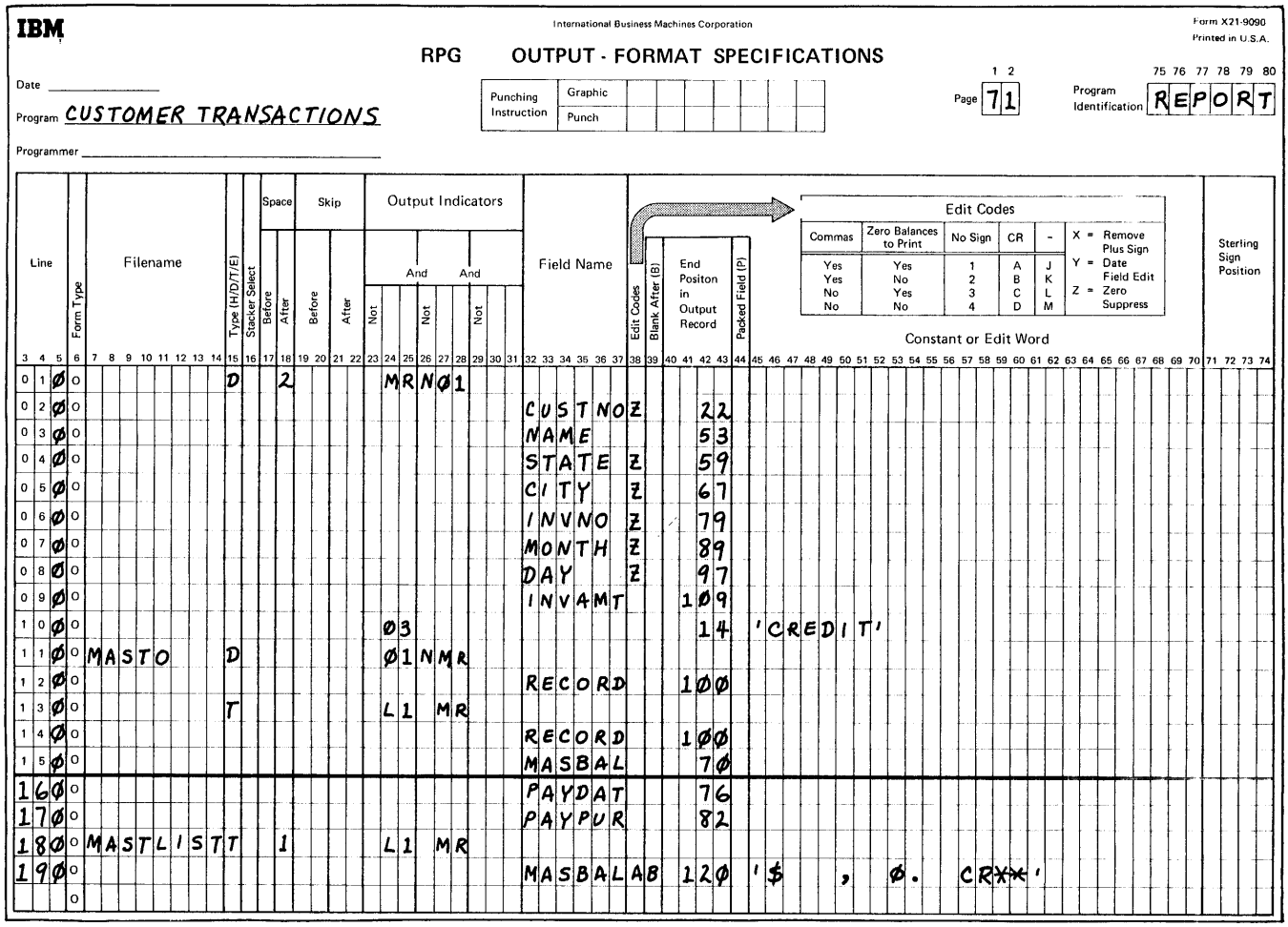


Figure 150. Coding for Sample Program 2 (Part 4 of 4)

File Description Specifications Form

The comments cards provide the name of the program. The two input files MASTI and TRANSIN, the output file MASTO (the updated master file), and MASTLIST (the printed report) are defined.

MASTI is the sequential primary file. Processing continues until all the records in that file have been processed (indicated by the E in column 17). The input records are in ascending order. Each record is 100 characters long.

TRANSIN is the secondary file; it may not contain transactions for all the customers on the master customer file. TRANSIN must be in ascending order. Its records are 80 characters long.

MASTO is a sequential disk file, containing the updated master customer records. The output records are 100 characters long.

MASTLIST is the printed report. The records can be 120 characters or less. The overflow indicator OF is assigned.

Indicator Summary Form

This form has been included to assist in documenting the program.

Input Specifications Form

The two input files MASTI and TRANSIN are defined.

MASTI: The disk input file that contains information about the firm's customers is assigned sequence AA. The first entry under field name defines the entire record. This entry (RECORD) is made to enable the entire record to be referred to on the Output-Format Specifications form. MASNUM corresponds to CUSTNO in the transaction file. Whenever a new master number is read, L1 is set on. M1 indicates that the master number will be matched with the customer number in the transaction file.

TRANSIN: The input records may be obtained from three types of cards. Sequence AB has been assigned to two types. If card column 1 contains an 11-punch, record identifying indicator 01 is turned on. If card column 1 contains a 12-punch, indicator 03 is turned on. The cards that have an 11-punch in column 1 are selected into stacker 2. The locations of the input records and their labels are defined in columns 44-58.

The field CUSTNO (customer number) has entries in columns 59-60 (Control Level) and columns 61-62 (Matching Fields) of the Input Specifications form. Whenever a new customer number is read, control level 1 (L1) is turned on. This condition is tested on the Output-Format Specifications form to govern printing of total lines and to produce the updated customer file. The entry in Matching Fields specifies that CUSTNO will be used to match another field (MASNUM) in the MASTI file.

The first card in the transaction file is a date card. It is assigned sequence BB. Whenever column 1 contains an S, indicator 04 is turned on. The date is contained in columns 2-5 of the card.

Calculations Specifications Form

Whenever the matching record indicator MR and indicator 02 are on, the contents of the field INVAMT are added to MASBAL. The result is stored in MASBAL. The date is moved to the field PAYDAT.

Whenever the matching record indicator MR and indicator 03 are on, INVAMT is subtracted from MASBAL, and the result is stored in MASBAL. The date is moved to PAYPUR.

Output-Format Specifications Form

The specifications for the printed report are listed under the name of the output file MASTLIST.

The output file MASTO is the updated disk file. The entries in Output Indicators allow for the following:

Whenever conditions 01 and NMR are satisfied (record identifying indicator 01 is on and no matching record is present), the entire record from MASTI is written on disk at detail time. This condition results when there is no corresponding customer number in the transaction file for the master customer number. (To keep the master customer file complete, the old input record is written on the updated disk file when no information is present in the transaction file.) If, however, L1 and MR are on, indicating that there was a matching record on the transaction file and calculations were performed, the input record from MASTI is written on disk. The fields MASBAL, PAYDAT, and PAYPUR contain the new entries based on the calculations. By coding the entries in this way, the new information for MASBAL, PAYDAT, and PAYPUR is entered on the updated master customer file MASTO, and the customer's name and address from MASTERIN are retained.

SAMPLE PROGRAM 3

This example shows the steps required to load or reorganize an indexed sequential file. One reason for reorganization might be that no space is left in the originally defined disk area to add records. Both the old file and the new file may be located on the same disk.

RPG Control Card

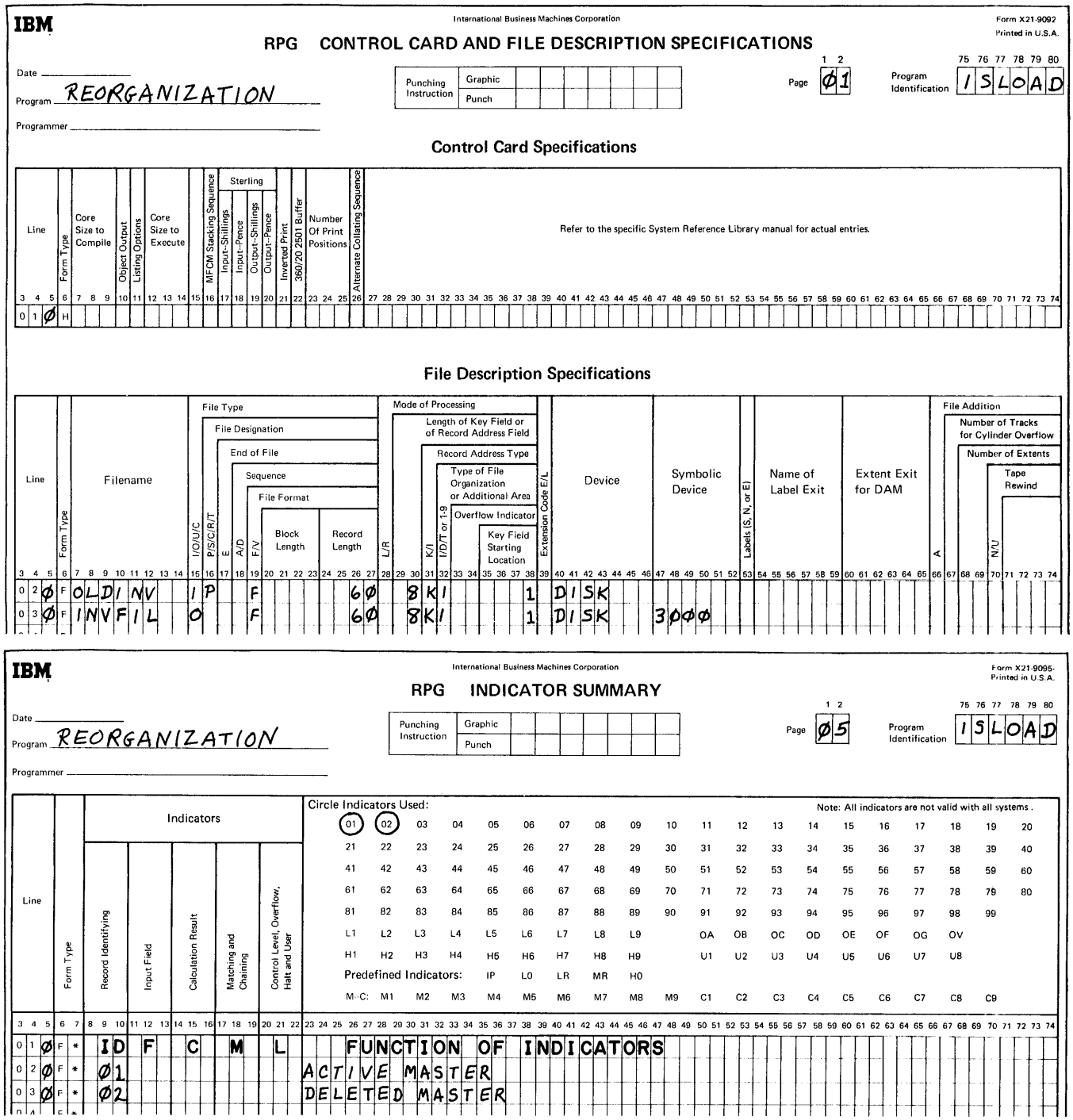
This card must precede any input to 1130 RPG.

File Description Specifications Form

Comments cards have been added to provide a program description. OLDINV, which is to be reorganized, is defined as an indexed sequential input file. The records are arranged in ascending sequence by means of a key field beginning in position 1 in each record. The length of the key field is entered in columns 29-30.

INVFIL is defined as the indexed sequential file to be created. Since the data arrangement of the new file should correspond to that of the old file, all entries concerning file organization must be equal. The entry 3000 in columns 47-52 is the number of records that can be placed in INVFIL.

To execute a load operation from card only, the first file description card must be replaced by the specifications shown in Figure 151 (Part 3).



IBM

International Business Machines Corporation

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

RPG CONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS

Date

Program REORGANIZATION

Programmer

Punching Instruction: Graphic / Punch

Page 01

Program Identification 75 76 77 78 79 80: ISLOAD

Control Card Specifications

Line	Form Type	Core Size to Compile	Object Output	Listing Options	Core Size to Execute	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	Refer to the specific System Reference Library manual for actual entries.
01	H														

File Description Specifications

Line	Form Type	Filename	I/O/U/C	P/S/C/R/T	E	A/D	F/V	Block Length	Record Length	L/R	K/I	I/D/T or 1-9	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels (S, N, or E)	Name of Label Exit	Extent Exit for DAM	A	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	N/U
02	F	OLDINV	I	P			F		60		8	K	I		1	DISK									
03	F	INVFIL	O				F		60		8	K	I		1	DISK	3000								

IBM

International Business Machines Corporation

Form X21-9095
Printed in U.S.A.

RPG INDICATOR SUMMARY

Date

Program REORGANIZATION

Programmer

Punching Instruction: Graphic / Punch

Page 05

Program Identification 75 76 77 78 79 80: ISLOAD

Line	Form Type	Indicators: Record Identifying	Input Field	Calculation Result	Matching and Chaining	Control Level, Overflow, Halt and User	Function of Indicators
01	F *	ID	F	C	M	L	FUNCTION OF INDICATORS
02	F *	01					ACTIVE MASTER
03	F *	02					DELETED MASTER

Circle Indicators Used:

Note: All indicators are not valid with all systems.

(01)	(02)	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	
L1	L2	L3	L4	L5	L6	L7	L8	L9		OA	OB	OC	OD	OE	OF	OG	OV		
H1	H2	H3	H4	H5	H6	H7	H8	H9		U1	U2	U3	U4	U5	U6	U7	U8		
Predefined Indicators:		IP	LO	LR	MR	H0													
M-C:	M1	M2	M3	M4	M5	M6	M7	M8	M9	C1	C2	C3	C4	C5	C6	C7	C8	C9	

Figure 151. Sample Program 3: Reorganization and Load (Part 1 of 3)

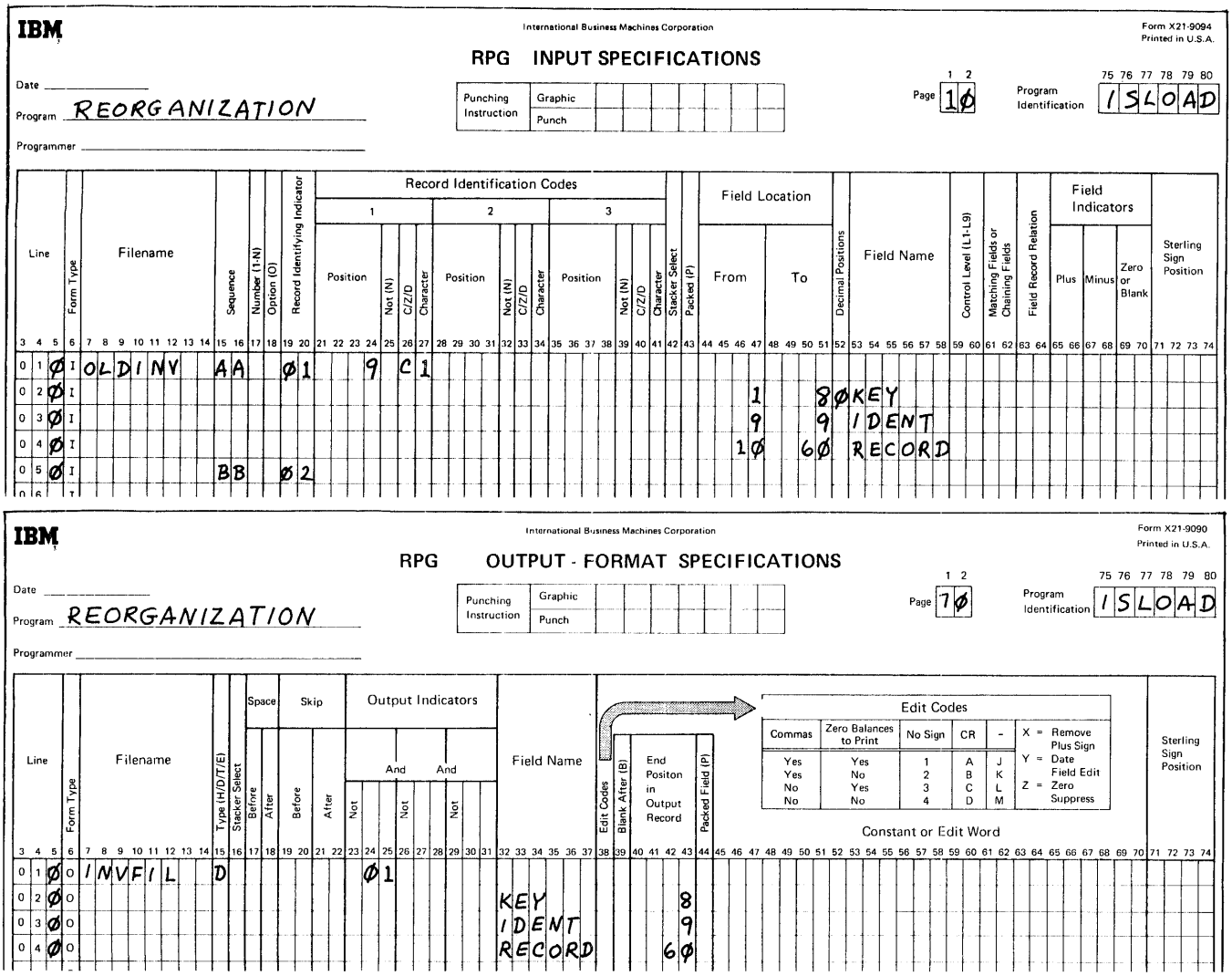


Figure 151. Sample Program 3: Reorganization and Load (Part 2 of 3)

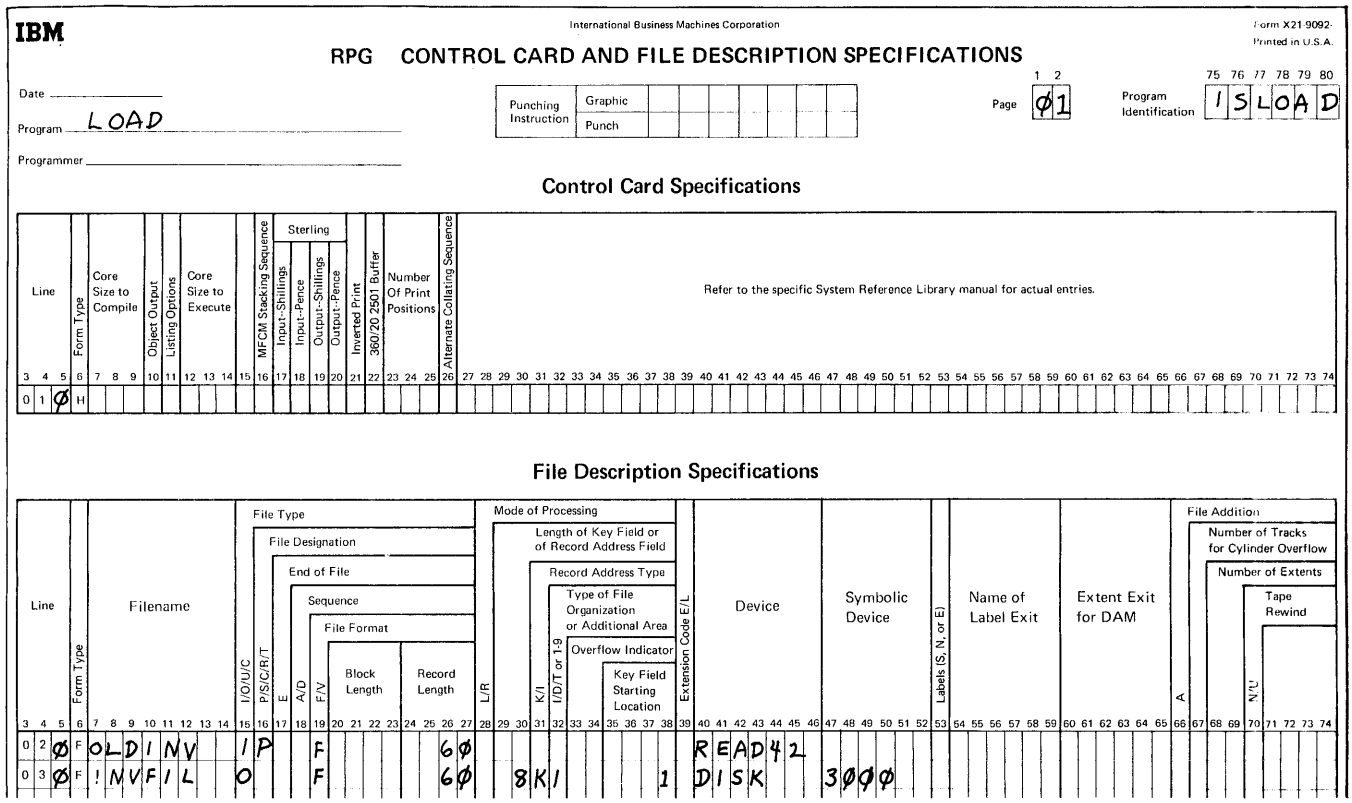


Figure 151. Sample Program 3: Reorganization and Load (Part 3 of 3)

Indicator Summary Form

This form has been included to assist in documenting the program.

Input Specifications Form

The records of the indexed sequential file OLDINV are to be automatically retrieved from the disk in the sequence of their key fields. However, only data from the records with character 1 in column 9 is actually transferred to the input fields. Indicator 01 is to be turned on for these records. All other record types, if any, are to be ignored. This is achieved by placing an alphabetic sequence entry in columns 15-16 and record identifying indicator 02. These record types are considered as deleted and are dropped from the new file.

Output-Format Specifications Form

All records for which indicator 01 is on are to be included in the new file INVFIL.

SAMPLE PROGRAM 4

This program retrieves and updates records in an indexed sequential file by means of the CHAIN operation. Format of the records is shown in Figure 152. Coding for the program is shown in Figure 153.

RPG Control Card

This card must precede any input to 1130 RPG.

File Description Specifications Form

A program description has been provided via comments cards. CARDIN is specified as a primary input file and read from the 2501 Card Reader. The E in column 17 instructs the program to terminate the job when records are fixed length and 80 bytes long.

INVFIL is a chained file located on disk. The U in column 15 describes the file as an update file (used for both input and output). The file is organized indexed-sequentially

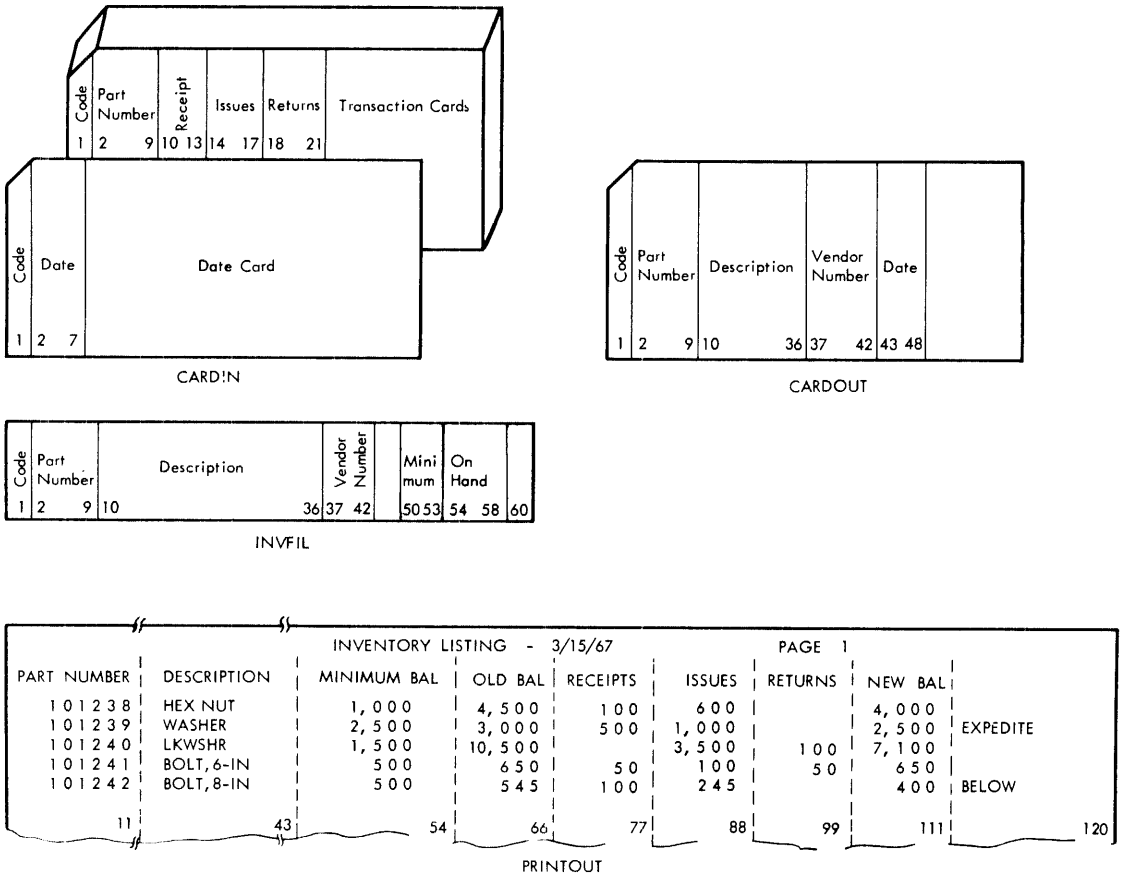


Figure 152. Input and Output Records for Sample Program 4

(I in column 32) and is to be processed randomly as specified by the R in column 28. The key field begins in position 1 in each record, and is 8 bytes long. The records are 60-byte, fixed-length records.

Two output files are required: CARDOUT, to punch the exception cards (located on a 1442 Card Punch), and PRINTOUT, to list the transaction report.

Indicator Summary Form

This form has been included to assist in documenting the program.

Input Specifications Form

The data fields of the input file CARDIN and the update file INVFIL are described.

Two kinds of data cards may appear in CARDIN. The DATE card (sequence entry DD) is identified by an asterisk (*) in column 1. Indicator 03 is turned on when this record type is read.

Cards identified by a 1 in column 1 (sequence entry CC) supply the fields used for calculations and output with actual data. Indicator 03 is turned on when this card type is read.

The indexed sequential file INVFIL is also defined. Active records are denoted by a 1 in position 9. All other record types turn on indicator 99. The fields from the active record that will be used are defined.

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: ISUPDT

Control Card Specifications

Line	Form Type	Core Size to Compile	Object Output / Listing Options	Core Size to Execute	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	Refer to the specific System Reference Library manual for actual entries.
01Ø	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, or E)	Name of Label Exit	Extent Exit for DAM	File Addition	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind
			I/O/U/C	P/S/C/R/T	E	A/D	F/V			L/R		K/I I/D/T or 1-9										A		N/U	
02Ø	F	*																							
03Ø	F	* INVENTORY UPDATE																							
04Ø	F	*																							
05Ø	F	CARDIN	I	P	E		F		8Ø								READØ1								
06Ø	F	INVFIL	U	C			F		6ØR		8	K	I			1	DISK								
07Ø	F	CARDOUT	O				F		8Ø								PUNCH42								
Ø8Ø	F	PRINTOUT	O				F		12Ø					OF			PRINTER								

IBM

International Business Machines Corporation

Form X21-9095
Printed in U.S.A.

RPG INDICATOR SUMMARY

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page: 05

Program Identification: ISUPDT

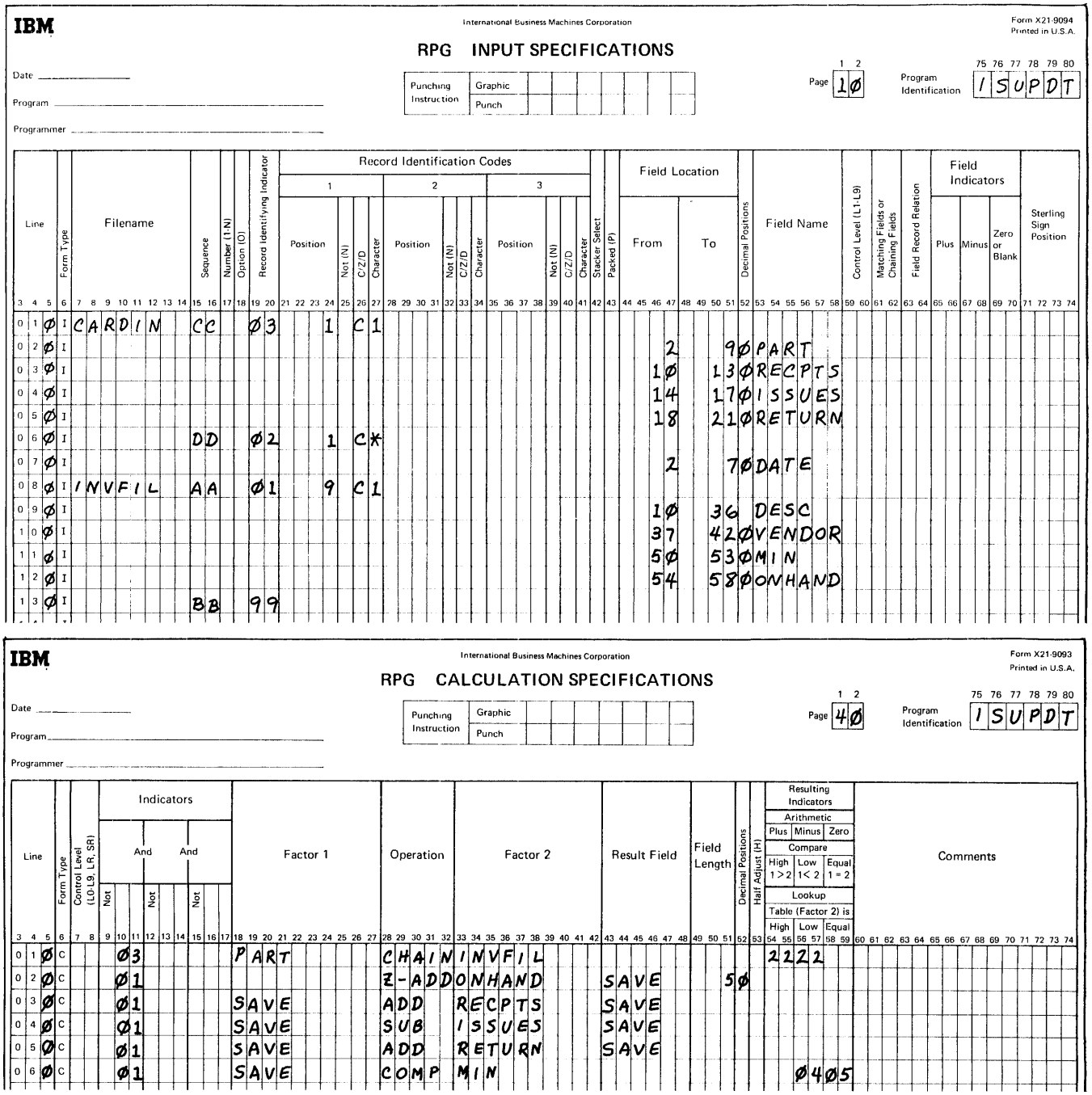
Circle Indicators Used:

Note: All indicators are not valid with all systems.

01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	
L1	L2	L3	L4	L5	L6	L7	L8	L9		OA	OB	OC	OD	OE	OF	OG	OV		
H1	H2	H3	H4	H5	H6	H7	H8	H9		U1	U2	U3	U4	U5	U6	U7	U8		
Predefined Indicators:				IP	LO	LR	MR	H0											
M-C:	M1	M2	M3	M4	M5	M6	M7	M8	M9	C1	C2	C3	C4	C5	C6	C7	C8	C9	

Line	Form Type	Indicators: Record Identifying	Input Field	Calculation Result	Matching and Chaining	Control Level, Overflow, Halt and User	
01Ø	F *	ID	F	C	M	L	FUNCTION OF INDICATORS
02Ø	F *	Ø3					PARTS CARD
03Ø	F *	Ø2					DATE CARD
04Ø	F *	Ø1					ACTIVE MASTER ITEM
05Ø	F *	99					DELETED MASTER ITEM
06Ø	F *			Ø4			BELOW MINIMUM
07Ø	F *			Ø5			AT MINIMUM

Figure 153. Coding for Sample Program 4 (Part 1 of 3)



IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date

Program

Programmer

Punching Instruction: Graphic / Punch

Page 10

Program Identification 75 76 77 78 79 80: ISUPDT

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	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	CARDIN	CC			03	1		C	1																					
02	I																				2	9		PART							
03	I																				10	13		RECPTS							
04	I																				14	17		ISSUES							
05	I																				18	21		RETURN							
06	I		DD			02		1		C*																					
07	I																				2	7		DATE							
08	I	INVFIL	AA			01		9		C	1																				
09	I																				10	36		DESC							
10	I																				37	42		VENDOR							
11	I																				50	53		MIN							
12	I																				54	58		ONHAND							
13	I		BB			99																									

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date

Program

Programmer

Punching Instruction: Graphic / Punch

Page 40

Program Identification 75 76 77 78 79 80: ISUPDT

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And	Indicators: And	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		03		PART	CHAIN	INVFIL					2	2	2							
02	C		01			Z-ADD	ONHAND	SAVE		5											
03	C		01		SAVE	ADD	RECPTS	SAVE													
04	C		01		SAVE	SUB	ISSUES	SAVE													
05	C		01		SAVE	ADD	RETURN	SAVE													
06	C		01		SAVE	COMP	MIN											04		05	

Figure 153. Coding for Sample Program 4 (Part 2 of 3)

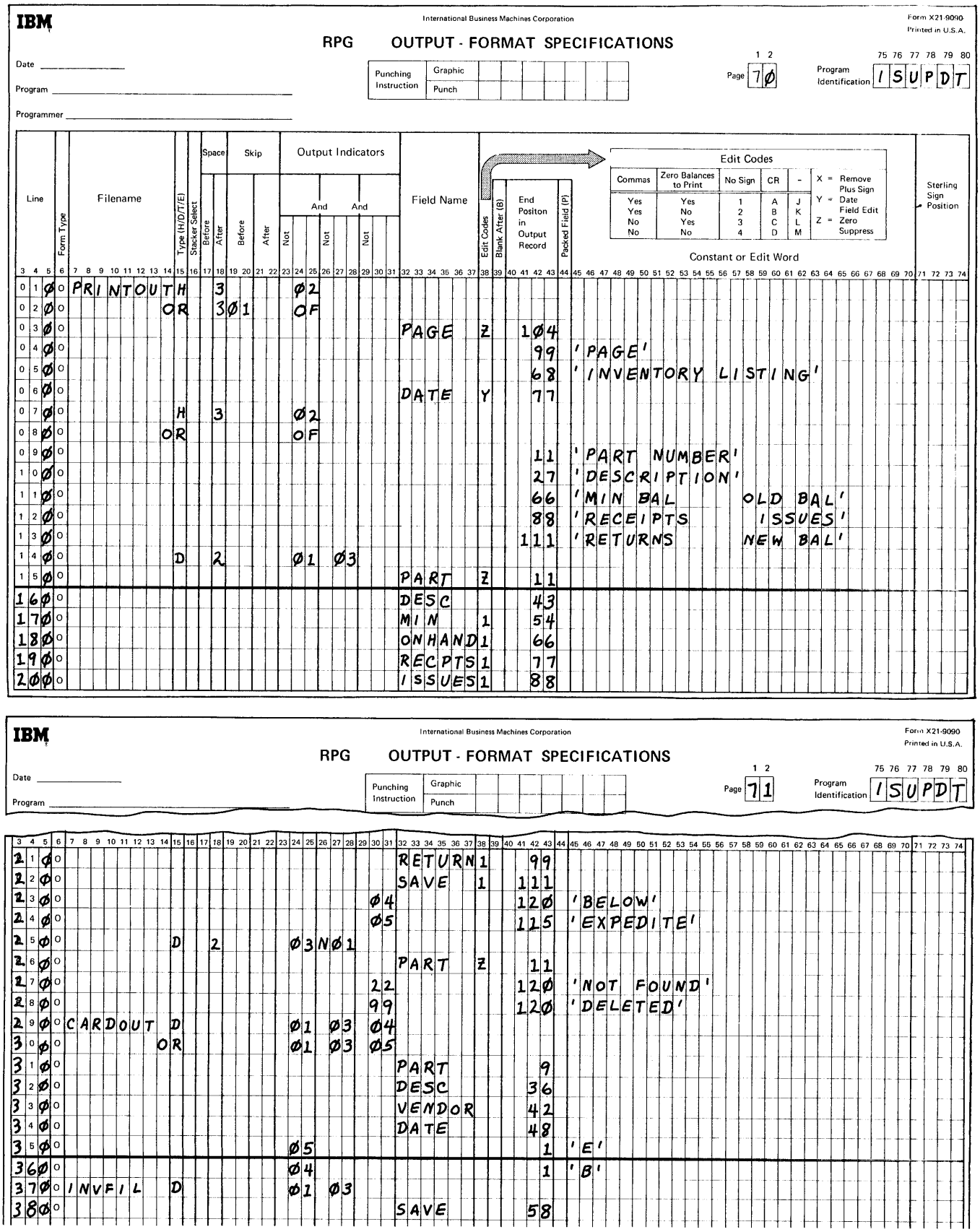


Figure 153. Coding for Sample Program 4 (Part 3 of 3)

Calculation Specifications Form

Line 010 specified that the field PART is chained to the file INVFIL to provide the inventory file record which corresponds to number in PART. If a record of the transaction file CARDIN has an active record in the inventory file INVFIL (i.e., key field and chaining field are of equal contents), both record identifying indicators 01 and 03 are on at the same time. Calculations are executed only if this condition is satisfied. The ONHAND field from the INVFIL record is reset and added to a work area (SAVE). Data fields RECPTS and RETURN from CARDIN records are added to SAVE; ISSUES are subtracted. When each input card for an INVFIL record is processed, the resulting new ONHAND field in SAVE is compared to the minimum balance. If ONHAND equals the minimum, resulting indicator 05 is set on. If ONHAND is below the minimum, resulting indicator 04 is set on.

Output-Format Specifications Form

The PRINTOUT file's heading information can be printed under control of either record identifying indicator 02, which is set for the date card (the first card in the CARDIN file), or overflow (OF). OF will govern all heading printing after the first page. The entry PAGE in line 030 causes the page number to be updated automatically for each new page.

The detail line described by entries 140-240 requires the presence of both the CARDIN and INVFIL records (record identifying indicators 01 and 03 on). If resulting indicator 04 is on, the words BELOW indicate the stock condition. If indicator 05 is on, the message EXPEDITE is added.

The output line described in PRINTOUT, entries 250-280, is the message printed when the error condition of a CARDIN record with no corresponding INVFIL record occurs. This condition is identified by record identifying indicator 01 being off when indicator 03 is on.

The exception file CARDOUT will have a card punched for each transaction that results in a below-minimum or at-minimum stock level. These cards will contain the part number and description, vendor number, date, and the E or B code for EXPEDITE or BELOW. Below-minimum cards, identified by the simultaneous settings of indicators 01, 03, and 05 on, will be selected into stacker 2.

Lines 370 and 380 provide for the updating of the INVFIL records. If indicators 01 and 03 are both on, indicating that the INVFIL record has been updated by a CARDIN transaction, the new ONHAND is moved into its INVFIL location on disk from the work area SAVE.

SAMPLE PROGRAM 5

This program (Figure 154) shows how new item records are added to an indexed sequential file. The new records are in card form. The indexed sequential routines for the 1130 do not allow retrieving a file (either randomly or sequentially) in the same program that additions are being made.

RPG Control Card

This card must precede any input to 1130 RPG.

File Description Specifications Form

ADDITNS is a primary file containing the new records. MASTITEM is the indexed sequential file to which the records will be added. Each record is 50 characters long. The key field is eight characters long. PRINTER is an output file for the printer.

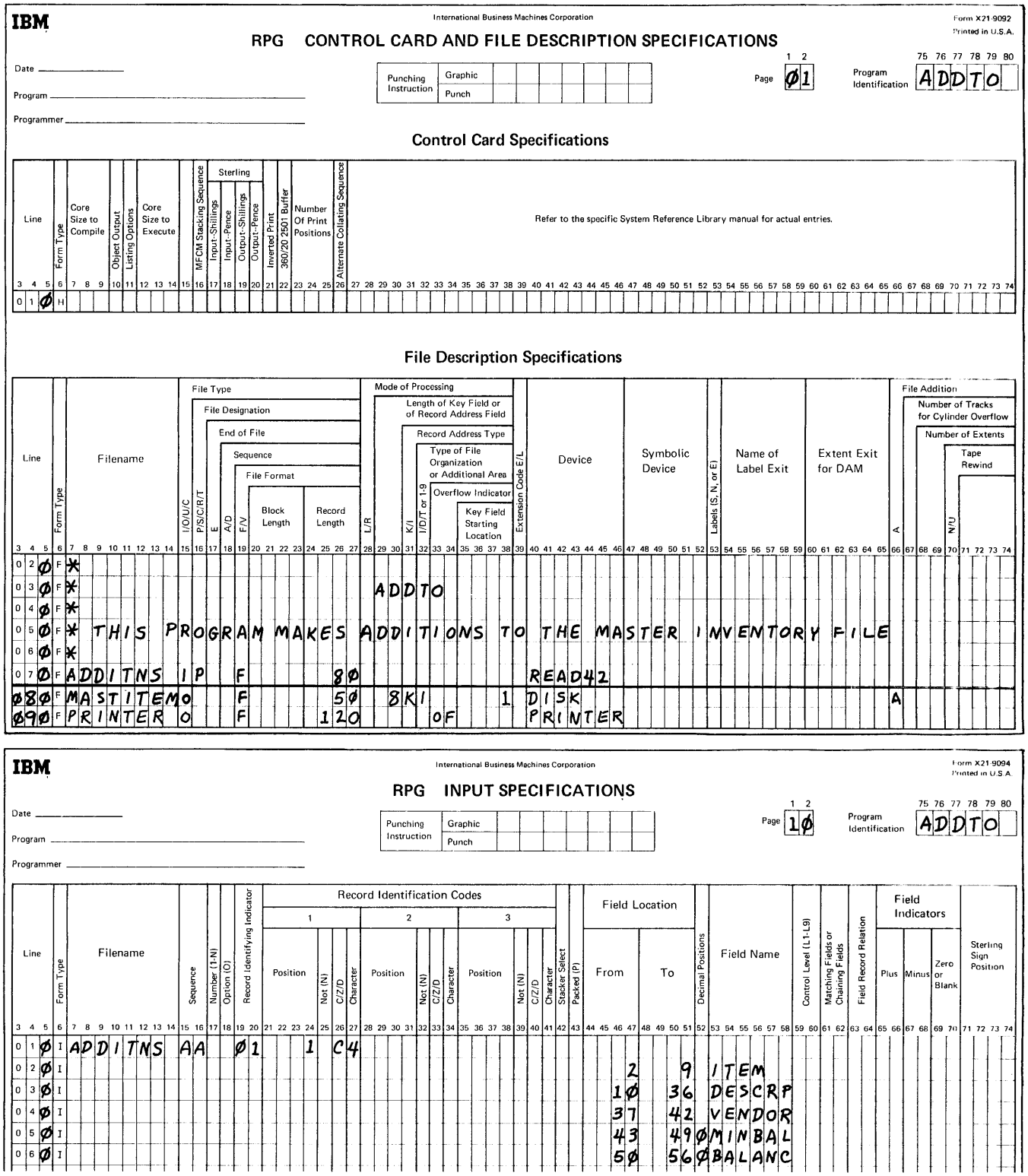
Input Specifications Form

The fields within the ADDITNS file are defined. All records must have a 4 in column 1.

Output-Format Specifications Form

The heading and detail lines and the fields of the record being added are described. MASTITEM is being added to at detail time on indicator 01. A record code of 1 is placed in position 9 of the record to denote that it is now an active record. The fields DESCRP and VENDOR are inserted into the record in a normal fashion. The fields MINBAL and BALANC are put out in a packed format to conserve disk space. Each of these fields was defined as 7 positions long and as numeric fields. Packing allows two digits (or one digit and a sign) to be placed in one position on the disk. Each field takes 7 + 1 (1 for the sign) ÷ 2 or 4 positions on the disk. (If the fields were 8 positions long, they would require 5 positions each on disk.)

Packing is an effective means of reducing disk file space required. If this file requires 5000 records, almost 7 cylinders of disk space can be saved by packing these two fields.



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 75 76 77 78 79 80: ADDTO

Control Card Specifications

Line	Form Type	Core Size to Compile	Object Output	Listing Options	Core Size to Execute	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	Refer to the specific System Reference Library manual for actual entries.
01	H														

File Description Specifications

Line	Form Type	Filename	I/O/U/C	P/S/C/R/T	E	A/D	F/V	Block Length	Record Length	L/R	K/I	I/D/T or 1-9	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels (S, N, or E)	Name of Label Exit	Extent Exit for DAM	File Addition A	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind N/U
02	F	*																						
03	F	*									ADDTO													
04	F	*																						
05	F	* THIS PROGRAM MAKES ADDITIONS TO THE MASTER INVENTORY FILE																						
06	F	*																						
07	F	ADDITNS	I	P			F		80							READ42								
08	F	MASTITEMO					F		50		8	K	I		1	DISK					A			
09	F	PRINTER	O				F		120				OF			PRINTER								

IBM — International Business Machines Corporation — Form X21-9094 Printed in U.S.A.

RPG INPUT SPECIFICATIONS

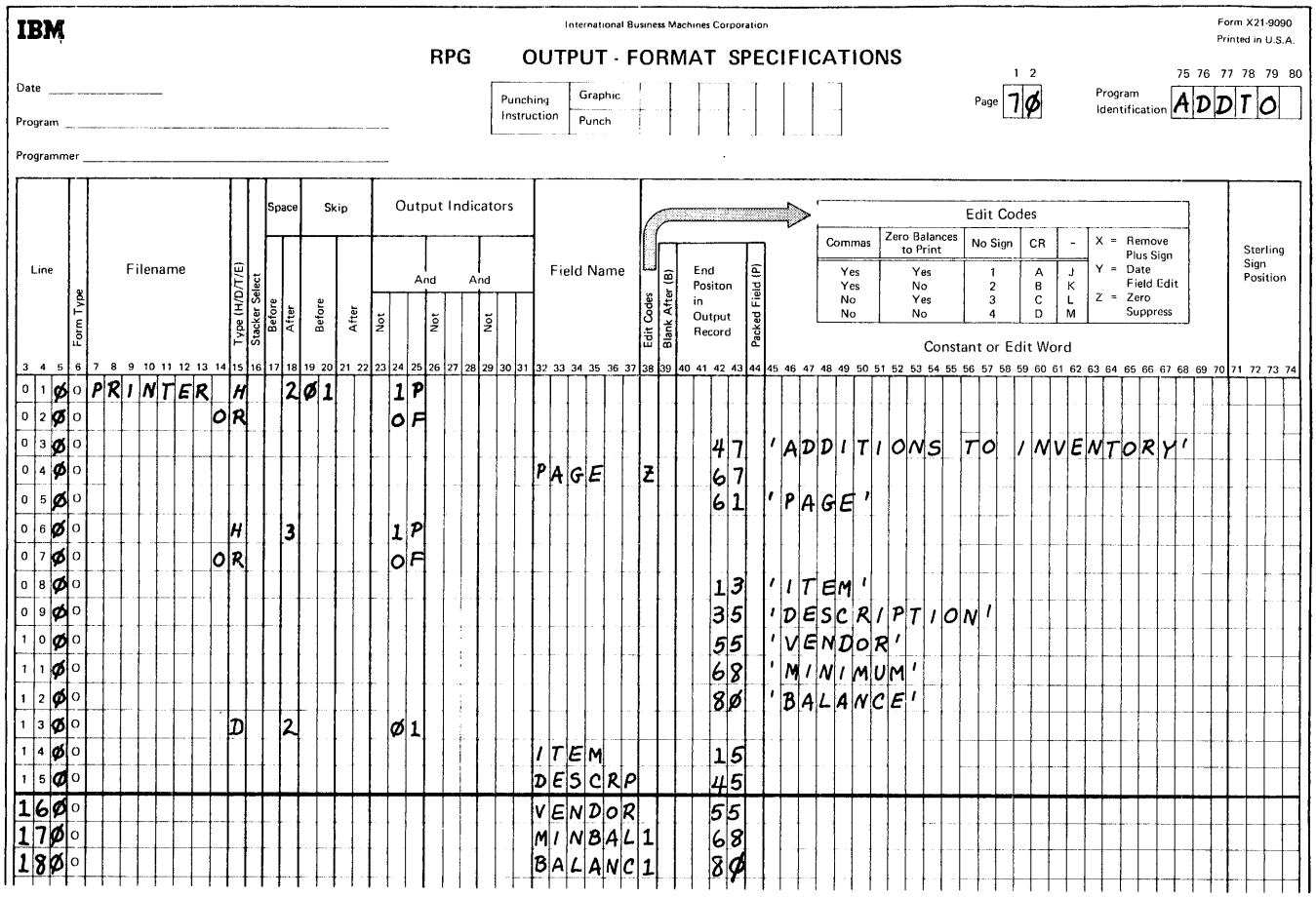
Date _____ Program _____ Programmer _____

Punching Instruction: Graphic / Punch

Page 10 — Program Identification 75 76 77 78 79 80: ADDTO

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	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	ADDITNS	AA			01		1	C	4																					
02	I																				2	9		ITEM							
03	I																				10	36		DESCRP							
04	I																				37	42		VENDOR							
05	I																				43	49	0	MINBAL							
06	I																				50	56	0	BALANC							

Figure 154. Coding for Sample Program 5 (Part 1 of 2)



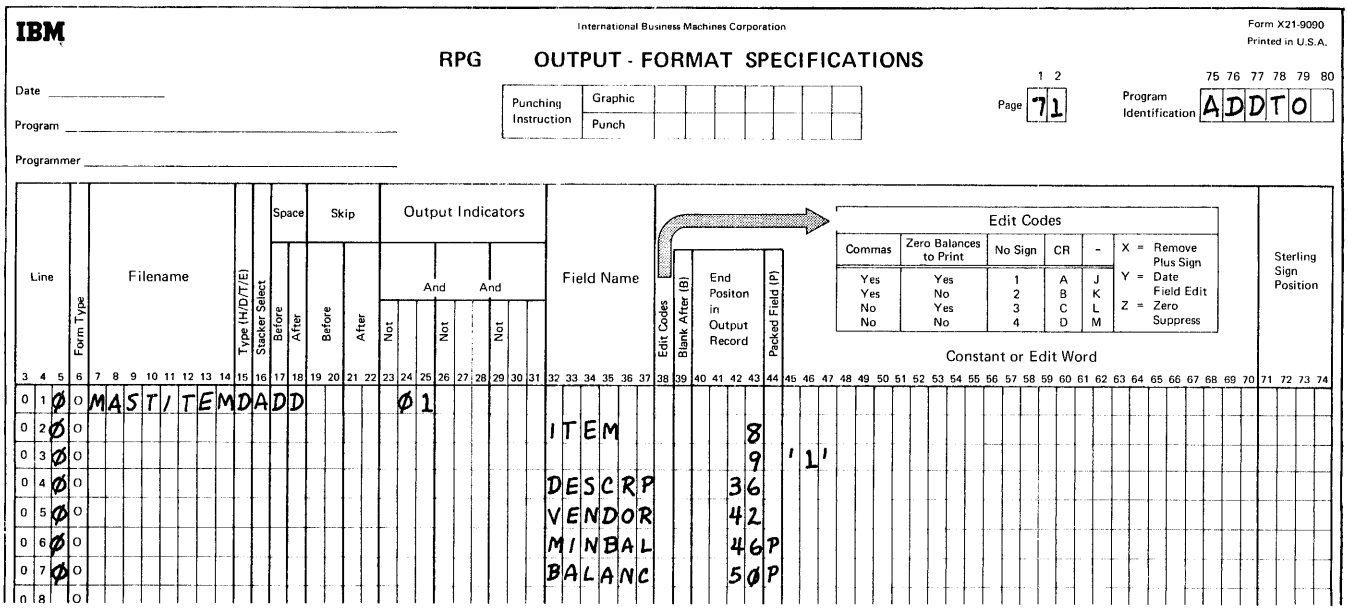


Figure 154. Coding for Sample Program 5 (Part 2 of 2)

Duplicate Records

Sample program 5 allows additions to be made to MASTITEM based on the key field which is in positions 1-8 of the output record. When a record is read from ADDITNS, MASTITEM is searched for the proper location to add the record. If a record of the same key as in the new record exists on MASTITEM, the program halts and an appropriate message is provided. The operator may bypass the record or may end the job.

SAMPLE PROGRAM 6

This program (Figure 155) shows how a typical file maintenance program is written. The file which was added to in sample program 5 is now being maintained.

The changes can take the form of:

- Changes to active records.
- Deletion of active records.
- Reactivation of deleted records.

The programmer should print an error message for an invalid condition such as trying to delete a record which has already been deleted.

RPG Control Card

This card must precede any input to 1130 RPG.

IBM International Business 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 75 76 77 78 79 80: MAINTN

Control Card Specifications

Line	Form Type	Core Size to Compile	Object Output	Listing Options	Core Size to Execute	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	Refer to the specific System Reference Library manual for actual entries.
3 4 5	6	7 8 9	10	11	12 13 14	15	16	17	18	19	20	21	22 23 24 25	26	27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
01	Ø	H													

File Description Specifications

Line	Form Type	Filename	File Type	File Designation	End of File	Sequence	File Format	Block Length	Record Length	L/R	Mode of Processing	Length of Key Field or of Record Address Field	Record Address Type	Type of File Organization or Additional Area	I/O/T or 1-9	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels (S, N, or E)	Name of Label Exit	Extent Exit for DAM	File Addition	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind
			I/O/U/C	P/S/C/R/T	E	A/D	F/V				K/I													A	N/U		
02	Ø	F *																									
03	Ø	F *									MAINTN																
04	Ø	F *																									
05	Ø	F * THIS PROGRAM MAINTAINS THE MASTER INVENTORY FILE																									
06	Ø	F *																									
07	Ø	F CHANGES IP				F			80										READ42								
Ø8	Ø	F MASTITEMUC				F			50	R	8	K	I				1		DISK								
Ø9	Ø	F PRINTER O				F			120					OF					PRINTER								

Figure 155. Coding for Sample Program 6 (Part 1 of 4)

IBM

International Business Machines Corporation

Form X21-9095
Printed in U.S.A.

RPG INDICATOR SUMMARY

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic							
	Punch							

Page 05

Program Identification MAINTN

Circle Indicators Used:

Note: All indicators are not valid with all systems.

(01)	(02)	(03)	(04)	(05)	06	07	08	09	10	11	12	13	14	15	16	17	18	19	(20)
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	
L1	L2	L3	L4	L5	L6	L7	L8	L9		OA	OB	OC	OD	OE	(OF)	OG	OV		
H1	H2	H3	H4	H5	H6	H7	H8	H9		U1	U2	U3	U4	U5	U6	U7	U8		
Predefined Indicators:				IP	LO	LR	MR	H0											
M-C	M1	M2	M3	M4	M5	M6	M7	M8	M9	C1	C2	C3	C4	C5	C6	C7	C8	C9	

Line	Form Type	Indicators: Record Identifying	Input Field	Calculation Result	Matching and Chaining	Control Level, Overflow, Halt and User	Comments
010	F	ID	F	C	M	L	FUNCTION OF INDICATORS
020	F	01					CHANGE TO AN ACTIVE RECORD
030	F	02					DELETION OF AN ACTIVE RECORD
040	F	03					REACTIVATE A DELETED RECORD
050	F	04					ACTIVE MASTER RECORD
060	F	05					DELETED MASTER RECORD
070	F			20			NOT FOUND CONDITION

IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic							
	Punch							

Page 10

Program Identification MAINTN

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	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	Packed (P)	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
010	I	CHANGES	AA			01	1		C	1																				
020	I		OR			02	1		C	2																				
030	I		OR			03	1		C	3																				
040	I																				1	1		CODE						
050	I																				2	9		ITEM						
060	I																				10	36		NEWDES						
070	I																				37	42		NEWVEN						
080	I																				43	49		NEWMIN						
090	I																				50	56		NEWBAL						
100	I	MASTITEM	BB			04	9		C	1																				
110	I		OR			05	9		C	2																				
120	I																				10	36		DESCRP						
130	I																				37	42		VENDOR						
140	I																			P	43	46	0	MINBAL						
150	I																			P	47	50	0	BALANC						

Figure 155. Coding for Sample Program 6 (Part 2 of 4)

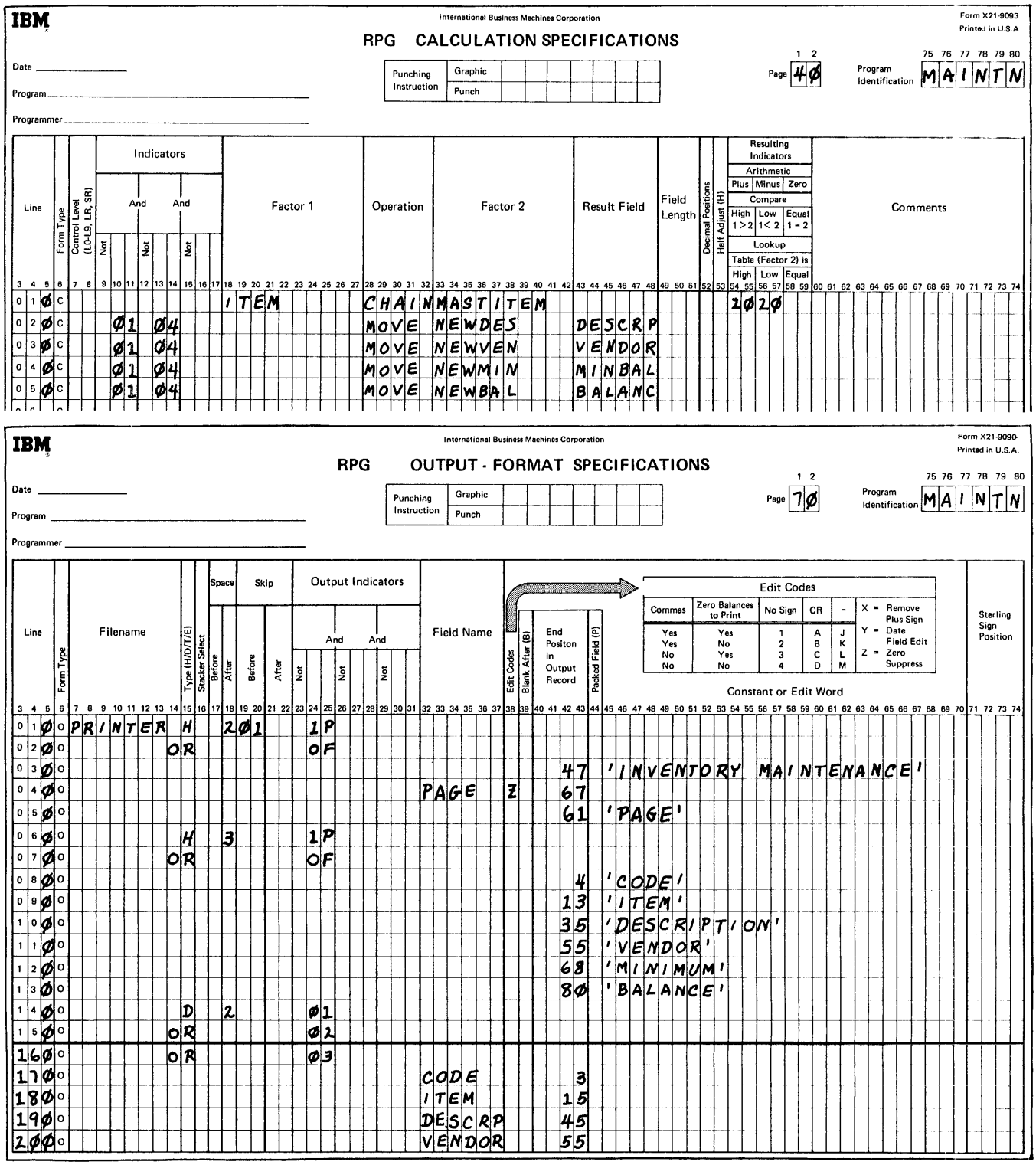


Figure 155. Coding for Sample Program 6 (Part 3 of 4)

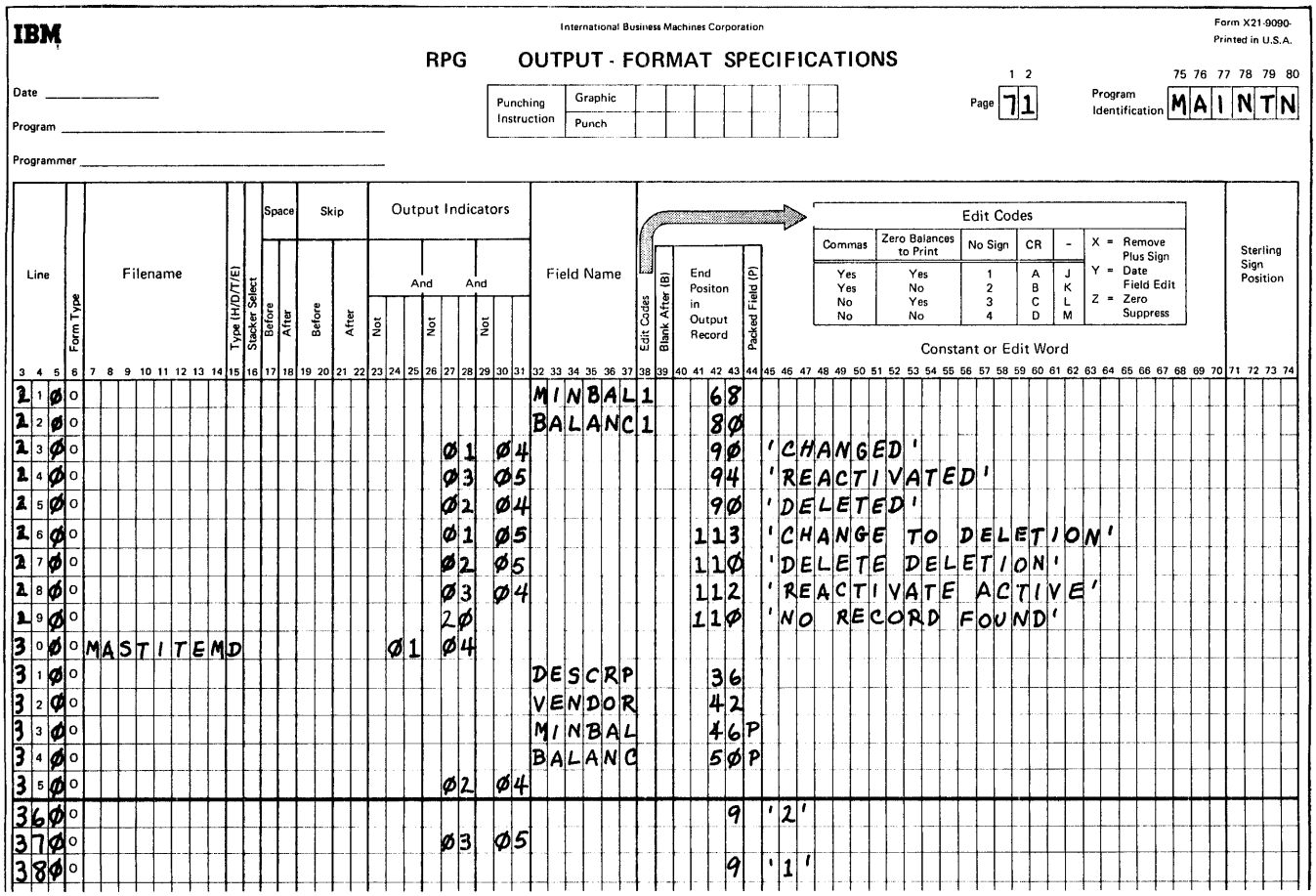


Figure 155. Coding for Sample Program 6 (Part 4 of 4)

File Description Specifications Form

The changes are in a card file called CHANGES. The indexed sequential file to be changed is called MASTITEM. It is defined as an update file (U in column 15) and a chained file (C in column 16). The file will be processed randomly (R in column 28). The other entries are the same as the previous example. A printer file is also defined.

Indicator Summary Form

The Indicator Summary form describes the indicators and their use. It has no effect upon processing.

Input Specifications Form

The file CHANGES and the three possible types of records are described. The fields are defined for all record types even though the fields will only be present when a record is read with a 1 in position 1.

MASTITEM is the file to be updated. It has both active records (1 in position 9) and deleted records (2 in position 9). The fields MINBAL and BALANC are in packed form on disk as described in the previous example.

Calculation Specifications Form

The calculations show all input records chaining to the MASTITEM file using ITEM as the chaining filed. Indicator 20 is used for the Not-Found condition.

The fields are moved only when indicators 01 and 04 are on. Indicator 01 being on denotes a change to an active record. Indicator 04 being on denotes that an active master was found.

Output-Format Specifications Form

The output specifications show two heading lines and the fields from the input card to be printed. The detail line also contains messages which show the status of the line. All of the valid conditions are printed in positions 84 to 94. The invalid conditions are printed in positions 96 to 113.

The MASTITEM file is updated for the three valid conditions. A deletion and reactivation require only a change to the record code (position 9).

This appendix contains a brief column-by-column description of each of the RPG specification forms, except the Control Card Specifications form. For a detailed description of the entries, refer to the applicable section of this manual. In the discussion that follows, the column location of the entry appears in parentheses after the name of the entry.

INFORMATION COMMON TO ALL FORMS

RPG source program cards should be in ascending numeric sequence by columns 1 through 5. If not, cards out of sequence will be flagged. Duplicate sequence numbers on adjacent cards will not be flagged.

Page (1-2)

Enter the page number of the specifications forms in the following sequence:

1. Control Card Specifications
2. File Description Specifications
3. Extension Specifications
4. Input Specifications
5. Calculation Specifications
6. Output-Format Specifications

Line (3-5)

First two digits of line number are preprinted. Use column 5 to identify additional lines to be inserted between two preprinted lines.

Form Type (6)

Contains preprinted code H, F, E, I, C, or O which must be punched into all RPG specification cards.

Comments (7)

Enter an asterisk in each line to be used as a comments line.

Program Identification (Columns 75-80)

Insert any characters to identify certain cards or portions of the program.

FILE DESCRIPTION SPECIFICATIONS (F)

Filename (7-14)

Enter a name for each file. The name may be eight characters or less. The first five characters must be unique. Names must be left-justified. First character must be alphabetic. Special characters or embedded blanks must not be used.

File Type (15)

I = Input File
O = Output File
U = Update File (only disk files)
C = Combined File (only card files on the 1442 Card Reader/Punch)

File Designation (16)

P = Primary File
S = Secondary File
C = Chained File
T = Table File
R = RA File

Leave blank for output files, including output table files.

End of File (17)

An E is required whenever processing of other input files is to be discontinued after the last record of a given input file has been read and processed. Specify an E for each input file that must come to end of file before processing is complete. If all input files are specified with blanks or if all are specified with an E, all are processed to EOF. An E can only be specified here if column 16 contains a P or an S.

Sequence (18)

If matching fields are specified for an input, combined, or update file, enter A if the file is in ascending sequence; D if the file is in descending sequence.

File Format (19)

F = Fixed-length records

Block Length and Record Length (20-27)

Enter, right-justified, the length of one record in Record Length. Leave Block Length blank.

Minimum and maximum record lengths are as follows:

Device	Minimum	Maximum
Card	1	80
1403 Printer	1	120
1132 Printer	1	120
1130 Disk		
Sequential	1	640
Indexed Seq.	1	636

Mode of Processing (28) (Disk Only)

Enter the mode by which the file is processed:

R = random processing of a sequential or indexed sequential file.
L = sequential processing of an indexed sequential file file between limits.
Blank = sequential processing.

Length of Key Field or of Record Address Field (29-30)

Enter number of positions that each entry in the record address file occupies.

Enter the length of the key, if the file is an indexed sequential. Maximum key length is 50 characters.

Record Address Type (31) (Disk Only)

k = Indexed-sequential file

Type of File Organization (32)

blank = sequential organization
I = indexed sequential organization
2 = two input/output areas assigned to a 1132 or 1403 printer or reader with no stacker select specified.

Overflow Indicators (33-34)

If overflow indicators are used, enter the overflow indicator associated with the file. A maximum of two overflow indicators is allowed: OF and OV. Do not specify an overflow indicator for the console printer.

Key Field Starting Location (35-38)

If this record is for an indexed-sequential file, enter a 1, right-justified. The key field must begin in position 1 of 1130 indexed sequential records.

Extension Code (39)

E = The file specified on this line is a table file, a chaining file (specified by codes C1-C3), or a record-address file.

Device (40-46)

Enter the device code for the input/output unit used by the file specified in columns 7-14, as follows:

Input/Output Unit	Device Code
IBM 2501 Card Reader	READ01
IBM 1442 Card Reader/Punch	READ42
IBM 1442 Card Punch	PUNCH42
IBM 1403 Printer	PRINT03
IBM 1132 Printer	PRINTER
IBM 1130 Console Keyboard	CONSOLE
IBM 2310 Disk	DISK
IBM 1131 Internal Disk	DISK

Symbolic Device (47-52)

If the file is indexed sequential and is to be loaded, enter the maximum number of records in the file. Number must be left justified with no commas.

Columns 53-65

Leave blank.

File Addition (66) (Disk Only)

Enter an A if the file defined on this line is indexed sequential and new records are to be added. The record to be added is described on the Output-Format Specifications form with ADD in columns 16-18. If this column contains an A, column 15 must contain an O (output). It is not possible to add and retrieve from the same file in the same program. Column 28 must be blank.

Columns 67-74

Leave blank.

EXTENSION SPECIFICATIONS (E)

Columns 7-8

If the file defined on this line is a chaining file (using the C1-C3 method), enter the sequence of that file from columns 15-16 of the Input Specifications form. Leave blank if the file described is a record-address file.

Number of the Chaining Field (9-10) (Disk Only)

If the file is a record-address file or a table file, leave these columns blank; otherwise, enter the number of the chaining field that has also been entered in Chaining Fields (Columns 61-62) of the Input Specifications form.

From Filename (11-18)

Enter, left-justified, the name of the table input file, chaining file, or record-address file defined on the File Description Specifications form.

To Filename (19-26)

If the From file is a chaining file (using the C1-C3 method), enter the name of the chained file. If the From file is a record-address file, enter the name of the file that contains the data records to be processed. If the From file is a table file, enter the name of the output file to be used after the table has been updated. Leave blank if the table is not to be put out.

Table Name (27-32)

Enter name of table as follows:

- For alternating input format, enter name of table to which the first entry in the input record belongs.
- For non-alternating input format, enter name of single table.

The name must be left-justified and must consist of 4, 5, or 6 characters of the form TABnnn, where nnn is any combination of alphameric characters.

Number of Table Entries per Record (33-35)

Enter, right-justified, the maximum number of table entries in each input record.

Number of Entries per Table (36-39)

Enter, right-justified, the number of table entries.

Length of Table Entry (40-42)

Enter, right-justified, the length of each table entry. Maximum lengths are:

- 248 characters for alphameric entries
- 14 digits for numeric entries

For a packed field, enter the number of digits of the field.

Packed (43)

P = Table entries are in packed-decimal format (disk only)
blank = Table entries are in unpacked-decimal format.

Decimal Position (44)

Digits 0 through 9 = Number of decimal positions in numeric table entries cannot exceed number specified in columns 40-42.
Blank = Alphameric table entries.

Sequence (45)

A = Table entries in ascending sequence.
D = Table entries in descending sequence.
blank = Table entries not in sequence.

A sequence must be specified if a LOKUP is used with indicators in the High or Low columns (54-57).

Table Name (46-51)

For alternating input formats only, enter name of table to which the second entry in each input record belongs. Name must consist of 4, 5, or 6 characters: TAB and one to three alphameric characters.

Length of Table Entry (52-54)

For alternating input formats only. Enter, right-justified, the length of each table entry. Maximum lengths are:

- 248 characters for alphameric entries.
- 14 digits for numeric entries.

For a packed field, enter the number of digits of the field.

Packed (55)

P = Table entries are in packed-decimal format (disk only)
blank = Table entries are in unpacked-decimal format.

Decimal Positions (56)

Digits 0 through 9 = Number of decimal positions in numeric tables cannot exceed number specified in columns 40-42.
Blank = Alphameric table entries.

Sequence (57)

A = Table entries in ascending sequence.
D = Table entries in descending sequence.
blank = Table entries not in sequence.

A sequence must be specified if a LOKUP is used with indicators in the High or Low columns (54-57).

Comments (58-74)

Enter any desired comments.

INPUT SPECIFICATIONS (1)

File Name (7-14)

Enter a name for each file. Maximum length is 8 characters. Name must be left-justified. First character must be alphabetic. Special characters or embedded blanks must not be used. The first five characters must be unique.

AND/OR-Relationship (14-16)

AND = The AND-relationship of record identification codes in the preceding line is to be continued.
OR = The entries in record identification codes of this line are to be in an OR-relationship to the entries of the preceding line.

Sequence (15-16)

Enter a number, beginning with 01 for each file and continuing in consecutive sequence to 99, to specify sequence-checking of record types. Enter leading zeros. Enter any two alphabetic characters to indicate that sequence checking is not required. Lines with alphabetic entries must precede lines with numeric entries. Any numeric entry in Sequence requires an entry in column 17.

Number (17) --- Used Only With Numeric Entry in Columns 15-16

1 = Only one record of a specific record type may be present before reading another record type.
N = One or more records of a specific record type may be present before reading another record type.

Option (18) --- Used Only With Numeric Entry in Columns 15-16

O = This record type is optional.
blank = Record must be present.

Record Identifying Indicator (19-20)

01-99 = indicator for the input record type defined in columns 21-41.

Record Identification Codes (21-41)

This field is divided into three identical subfields:

Columns 21-27
Columns 28-34
Columns 35-41

An AND-relationship exists between these three fields.

Position (21-24, 28-31, 35-38)

Enter the number of the input record position containing the identifying code. The number must be right-justified.

Not (25, 32, 39)

N = Code described must not be contained in the record position.

C/Z/D (26, 32, 40)

C = Entire character of the specified position is to be checked.
Z = Zone portion of the specified position is to be checked.
D = Digit portion of the specified position is to be checked.

Character (27, 34, 41)

If C/Z/D contains C or D, enter any one of the 256 EBCDIC characters. If C/Z/D contains Z, enter one of the following:

Card Zones:
- 12-zone – check for &, A through I, or any of the other 7 EBCDIC characters that have the same hexadecimal zone as an A.
- 11-zone – check for a minus sign, J through R, or any of the other 7 EBCDIC characters that have the same hexadecimal zone as J.
- no zone – absence of 11 or 12 zone – check for blank or any one of the 16 characters that have the same hexadecimal zone as 1.

Core Zones (all others): Any specification in these columns other than those used in testing for card zones can be used to identify record codes with the same zone in core storage as the character specified here.

Stacker Select (42)

Number of stacker to which input cards are to be selected. Leave blank for normal stacker of multi-stacker devices or for single-stacker devices.

Note: Stacker-select entries for combined-file cards that are punched must not be made on the Input Specifications form. Stacker select will be ignored if two input/output areas are specified.

Packed (43)

P = input data in packed-decimal format (disk only).
blank = input data in unpacked format.

Field Location (44-51)

Location of fields in input record.

From (44-47): Leftmost location of the field specified in Fieldname

To (48-51): Rightmost location of the field defined in Fieldname

Numbers must be right-justified; leading zeros may be omitted.

Decimal Positions (52)

For numeric fields only. Enter one of the digits 0 through 9 to indicate number of decimal positions in input field (cannot exceed the field length).

Fieldname (53-58)

Name of each field defined in Field Location. Maximum length is 6 characters. First character must be alphabetic. Special characters or embedded blanks may not be used. Fieldname must be left-justified.

Control Level (59-60)

Any one of the control-level indicators L1 (lowest) through L9 (highest) to identify control fields.

Matching Fields or Chaining Fields (61-62)

M1-M9 = record/matching for nine or less input fields or sequence checking for the fields of a single input file.
C1-C3 = alternate method of chaining is specified; codes identify the chaining field.

Field-Record Relation (63-64)

Enter any indicator specified to provide field-record relation for identical fields contained in different locations (OR-relationship). Normally, these are the record identifying indicators specified in columns 19-20 of the Input Specifications form. Only record identifying indicators may be used when the field is a control field or a match field.

Field Indicators (65-70)

If a field is alphameric, only zero or blank specifications may be used. Enter any one of the indicators 01-99, H1-H9, as required. If the field specified in columns 46-58 contains any positive value, except +0, the indicator in Plus (columns 65-66) is turned on. If the field contains any negative value except -0, the indicator in Minus (columns 67-68) is turned on. If the field contains only zero or blank, the indicator in Zero or Blank (columns 69-70) is turned on. It is also turned on if the field is numeric and contains only +0 or -0. The indicators are off at the beginning of the program.

Sterling Sign Position (71-74)

Enter in these columns the position in the record that contains the sign of the sterling field. If the sign is in the normal position, enter S in column 74.

CALCULATION SPECIFICATIONS (C)

Control Level (7-8)

The sequence of entries must be as follows:

blank = operation is to occur during detail time.
L0 = calculation is to be performed at total time.
L1-19
LR
SR = statement is part of an RPG subroutine.

Indicators (9-17)

Enter one to three indicators, right-justified. Any of the indicators may be used.

Columns 9, 12, and 15 may contain blank or N only. If there is more than one indicator in a line, RPG assumes an AND-relationship between the individual indicators.

Factor 1 (18-27)

Enter a field name, a label, or a literal.

Field Name – Left-justified, maximum 6 characters. First character must be alphabetic. Special characters and embedded blanks cannot be used. Must be defined on the input form or as a result field on the calculation form.
Literal – *Numeric:* Left-justified, maximum length 10 characters. One decimal point and/or one sign (plus or minus) may be used.

Alphameric: Left-justified, must be enclosed in apostrophes ('), maximum length 8 characters. Any one of the 256 EBCDIC characters may be used.
Label – Name for use in a TAG or ENDSR statement. First character must be alphabetic. Special characters and blanks may not be used. Factor 1 is only used with the operations ADD, SUB, MULT, DIV, COMP, LOKUP, TAG, CHAIN, BEGSR, and ENDSR.

Operation (28-32)

Operation code left-justified.

Factor 2 (33-42)

Field name, label, or literal to be used in the specified operation. (See Factor 1 for definition of field name, label, and literal.)

Factor 2 is not used with the operations MVR, TESTZ, RLABL, TAG, SETOF, SETON, BEGSR, ENDSR, and EXCPT.

Result Field (Columns 43-48)

Result field name. First character must be alphabetic. Special characters or embedded blanks cannot be used. Field name must be left-justified. It is not used with the operations COMP, GOTO, EXIT, TAG, SETOF, SETON, CHAIN, BEGSR, ENDSR, EXSR, and EXCPT.

Field Length (49-51)

Number of storage positions required for result field. Maximum length of numeric fields is 14 digits. Maximum length of alphameric fields is 256 characters. Must be right-justified. May be left blank if already defined in another line of the calculation form or in the input form.

Decimal Position (52)

Decimal positions of the result field (0 through 9). Must be blank if the result field is alphameric. The number of decimal positions cannot exceed the field length.

Half Adjust

H = Half-adjustment of result field.
blank = No half-adjustment of result field.

Resulting Indicators (54-59)

Any indicator 01-99, H1-H9, L1-L9, LR, OF, or OV may be used.

Arithmetic Operations:

Enter up to three indicators to be turned on whenever the result is positive (indicator in columns 54-55), negative (indicator in columns 56-57), or zero (indicator in columns 58-59).

Compare Operations:

Enter up to three indicators to be turned on whenever Factor 1> Factor 2 (indicator in columns 54-55), or Factor 1< Factor 2 (indicator in columns 56-57), or Factor 1 = Factor 2 (indicator in columns 58-59).

LOKUP Operation: Enter one or two indicators in High, Low, or Equal; or High and Equal; or Low and Equal.

An indicator in the high column causes a search for the first table argument (Factor 2) higher than the search argument (Factor 1). An indicator in the low column works in the opposite manner. If there is an indicator in the high or low columns, the table name in factor 2 must be specified as ascending or descending on the extension specifications.

TESTZ Operation: Enter up to three indicators to be turned on whenever a 12-zone (indicator in columns 54-55), or an 11-zone (indicator in columns 56-57), or any other zone (indicator in columns 58-59) is detected in the field specified in Result Field in this line.

SETOF, SETON Operations: Enter up to three indicators to be turned on (SETON) or off (SETOF).

CHAIN Operation: An optional entry of an indicator (the same indicator) in columns 54-57 to be turned on in the case of a not-found record.

Comments (60-74)

Enter any desired comments.

OUTPUT-FORMAT SPECIFICATIONS

Filename (7-14)

Enter name for each file. Maximum length is 8 characters. First character must be alphabetic. Special characters or embedded blanks cannot be used. Name must be left-justified. If several lines of the same file are specified in sequence, the name may be entered in the first line only. The first five characters must be unique.

AND/OR-Relationship (14-16)

AND = Records in an AND-relationship.
OR = Records in an OR-relationship.

Type (H/D/T/E) (15)

H = Heading line.
D = Detail line.
T = Total line.
E = Exception line.

The sequence of entries must be as described above within each file or each file may be defined within each type.

Adding Records to Indexed-Sequential File (16-18)

Enter ADD in these columns if output records are to be added to an indexed sequential file. The file description for this file must have an A in column 66.

For card and/or printer files, use these columns as described in the paragraphs that follow.

Stacker Select (16)

Number of stacker to which cards are to be selected. Leave blank for normal stacker (multi-stacker device) or for single-stacker device. Entry is allowed only if the file is described as combined or output in the File Description Specifications.

Space (17-18)

Enter at least one entry in columns 17-22 if line is to be printed.

BEFORE (17): Enter 0, 1, 2, or 3 to specify line spacing before printing.

AFTER (18): Enter 0, 1, 2, or 3 to specify line spacing after printing.

A 0 cannot be specified for the console printer.

Skip (19-22)

Enter at least one entry in columns 17-22 if a line is to be printed.

BEFORE (19-20): Enter any number from 01 through 12 to specify skipping before printing. Leave blank for no skip before printing.

A skip may not be specified for the console printer, or for channel 7, 8, 10, and 11 with 1132 printer.

AFTER (21-22): Enter any number from 01 through 12 to specify skipping after printing. Leave blank for no skip after printing.

Output Indicators (23-31)

Enter, right-justified, three or four indicators to identify or describe fields. Columns 23, 26, 29 may contain blank or the letter N (not) only.

Fieldname (32-37)

Enter any name defined in either the input or the calculation form. Leave blank for constants specified in columns 45-70. Use field name PAGE to cause automatic page numbering on normal printer. Use any field name beginning with PAGE (e.g. PAGE1) for additional page counters.

Edit Codes (38)

The entry in this column controls editing of numeric fields.

1 Print with commas, print zero balance, suppress sign.
2 Print with commas, suppress zero balance and suppress sign.
3 Print without commas, print zero balance, suppress sign.
4 Print without commas, suppress zero balance and suppress sign.
A Print with commas, print zero balance, print sign as CR.
B Print with commas, suppress zero balance, print sign as CR.
C Print without commas, print zero balance, print sign as CR.
D Print without commas, suppress zero balance, print sign as CR.
J Print with commas, print zero balance, print sign as –.
K Print with commas, suppress zero balance, print sign as –.
L Print without commas, print zero balance, print sign as –.
M Print without commas, suppress zero balance, print sign as –.
X This code is accepted by 1130 RPG, but no function is performed.
Y Edit date field:
    3 digit – (n) n/n
    4 digit – (n) n/nn
    5 digit – (n) n/nn/n
    6 digit – (n) n/nn/nn
  The first n will be zero suppressed.
Z High order zero suppression and remove sign.

An edit word may be used with a numeric field by leaving this column blank.

Note: If the inverted print option is chosen (I in column 21 of the RPG Control Card), a comma must be used instead of a decimal point in all numeric literals. Edit codes 1-4, A-D, J-M will invert commas for decimals and vice versa. The Y edit code will invert periods for slashes.

Blank After (39)

B = Reset alphameric output fields to blanks or numeric output fields to zeros.

Reset occurs after execution of the specified output operation.

End Position in Output Record (40-43)

Enter number of output-record position to contain rightmost character of output field.

Packed (44)

P = Output data in packed decimal format (disk only).
blank = Output data in unpacked decimal format.

Constant or Edit Word (45-70)

Constant: Enter any desired constant enclosed in apostrophes. May consist of any of the 256 EBCDIC characters. Maximum length is 24 characters.

Apostrophe required within constants must be represented as two consecutive apostrophes.

Edit Word: Enter any edit word to specify editing with respect to punctuation, printing of dollar sign status, zero suppression, etc. Must be enclosed in apostrophes (for numeric fields only).

With edit codes 1-4, A-D, and J-M two entries are possible:
'*' in columns 45-47 to denote asterisks.
'$' in columns 45-47 to denote a floating dollar sign.

Sterling Sign Position (71-74)

Enter in these columns the position in the record that contains the sign of the sterling field. If the sign is in the normal position, enter S in column 74.

The RPG Sterling routines furnish users with a convenient and time-saving means of handling sterling amounts. The presence of sterling fields is indicated to the RPG program by additional entries in the Input and Output-Format specifications forms and in the control card.

Sterling input information can be represented in two formats, IBM and BSI, as described in the control card. The RPG sterling routines convert the input fields into a pence-format field. A pence-format field is a sterling amount represented in pence. If the output is to be printed, the fields are converted with shillings and pence printed in two positions each with zero suppression in effect in the tens position of each field. If the output is not printed, the output is converted to either BSI or IBM formats.

Note 1: On both input and output, the pounds field must consist of at least one, and no more than eight positions.

Note 2: BSI or IBM input fields of one program must use the same code combination throughout.

INPUT SPECIFICATIONS

The position of the sign must be specified in columns 71-74. Enter an S in column 74 if the sign is in the normal position. If the pence field has decimal positions, the normal position of the sign is in the rightmost decimal position of the pence field. If the pence field has no decimal positions, the normal positions of the sign is in the units position of the pounds field.

Note 1: One of the digits 0, 1, 2, or 3 must be entered in column 52, to indicate the number of required decimal positions.

Note 2: The same name cannot be used for both a sterling field and a decimal field.

Note 3: The sign of the field must contain a numeric underpunch.

OUTPUT – FORMAT SPECIFICATIONS

The positions of the sign for sterling output fields must be specified in columns 71-74 in the same manner as for sterling input fields. The sterling sign will always appear on output whether the field is plus, minus, or zero.

Output Not Printed: The field may be specified as any combination of IBM or BSI shillings and pence formats. The sign may appear anywhere within the record. When outside the field, the sign will be supplied with a zero underpunch.

Printed Output: The normal sign position must be used. Insert the letter S in column 74 of the Output Specification Sheet.

Note 1: Shillings and pence are printed in two positions each. When no edit word is specified, zero suppression is in effect in the tens positions of each field. It is necessary to add two additional positions for printed output when input is in the BSI format and one extra print position for input in the IBM format.

Note 2: The pounds field consists of at least one and no more than 8 positions. Zero suppression on the pounds field may be obtained by placing Z in column 38 of the specification sheet.

Note 3: If a field is defined as a sterling field in the input but not in the output specification, the output will be in pence format.

Note 4: Editing is allowed only on printed output files. The rules governing the use of edit control words are the same as those for decimal fields. The features available are:

1. Zero suppression in the pounds field.
2. Zero suppression in the shillings field, if both pound and shilling values are zero.
3. Zero suppression in the pence field, if pound, shilling, and pence values are zero.
4. Suppression of zeros preceding signs, and suppression of separation marks between pounds and shillings, shillings and pence, and pence and decimals.
5. The high-order shilling and pence positions will be zero suppressed when a sterling field is edited.

CONTROL CARD

To select the required sterling routines, the RPG program needs information regarding the input and output formats. This information is entered in four columns of the RPG processor control card. The entries are: 1 for IBM code or 2 for BSI code.

CALCULATION SPECIFICATIONS

While no additional entries are required in this form, the user should keep in mind that all calculations are done in pence format. This must be considered when defining the length of result fields or when using Factors 1 or 2.

Lengths of Pence-Format Fields

If a pence-format result field is to be reconverted into a sterling output field, the highest amount it is permitted to contain is 23,999,999,999.999. This converts to a field containing eight pound positions which is the maximum allowed.

Note: if the two high-order digits of a pence field are greater than 23, a high-order digit will be lost when the field is converted to pound-shillings-pence format for output.

Pound Sterling Formats

In addition to the printed output format, RPG will support, on the input and output fields, two standards for pence and shilling portions of the sterling fields: IBM or BSI. Columns 17-20 of the RPG Processor Control Card indicate either the IBM or BSI formats. The formats for IBM and BSI are listed here.

Column 17 (Sterling Shilling Field on Input)

IBM Format: Two positions are allowed for the shilling option in the input fields: 00-19 for 0 to 19 shillings.

BSI Format: The shilling option in the input fields is indicated as listed here:
0-9 shillings by a 0-9 punch,
10 shillings by a 12-punch,
11-19 shillings by an A-I Punch.

Note: If the sterling shilling field on input is in the BSI format, one position is added to the field by RPG.

Column 18 (Sterling Pence Field on Input)

IBM Format: The pence option on input field is as listed here:
0-9 pence by a 0-9 punch,
10 pence by an 11-punch,
11 pence by a 12-punch.

BSI Format: The pence option on the input field is as listed here.
0-9 pence by a 0-9 punch,
10 pence by a 12-punch,
11 pence by an 11-punch.

Column 19 (Sterling Shilling Field on Output)

IBM Format: Two positions are allowed for the shilling option on the output field:
00-19 for 0-19 shillings

BSI Format: The shilling option on the output field is as listed here:
0-9 shillings by a 0-9 punch,
10 shillings by a 12-punch,
11-19 shillings by an A-I punch.

Column 20 (Sterling Pence Field on Output)

IBM Format: The pence option on the output field is as listed here:
0-9 pence by a 0-9 punch,
10 pence by an 11-punch,
11 pence by a 12-punch.

BSI Format: The pence option on the output field is as listed here:
0-9 pence by a 0-9 punch,
10 pence by a 12-punch,
11 pence by an 11-punch.

Indicator	Where Located	Where Used	Turned On	Turned Off	Notes
Field Indicators 01-99 Zero and Blank Plus Minus	Input form	Indicator (calc.), Output Indicators	by Blank or Zero in specified field by Plus in specified field by Minus in specified field	before this field status is to be tested the next time	Note 1
H1 through H9	Input form Calculation form	Indicator (calc.), Output	whenever the specified field status or record identification condition is satisifed	internal, at the end of the detail cycle (see Sign. of Program Logic)	Note 1
LR	Internal	Control Level (calc.), Output Indicators	after processing the last record of the last file (see column 17 of File Descr.)	at the beginning of processing	Note 1 Note 2
L0 (Level Zero)	Internal	Control Level (calc.) Output Indicators	at the end of every processing cycle	is never turned off by RPG	
Control Level Indicators L1 through L9	Input form Columns 59-60	Control Level (calc.), Indicators (calc.), Output Indicators	when the value in a control field changes. All indicators of the lower levels are also turned on	at end of following detail cycle	Note 1
MR (Matching)	Internal	Indicators (calc.), Output Indicators	if the matching-field contents of the record of a secondary file match the matching-field contents of a record in the primary file	when all total calculations and output are completed for the last record of the matching group.	
OF/OV Overflow	Internal	Indicators (calc.), Output Indicators	(see Significance of Program Logic)	after the following heading and detail lines are completed	Note 3

Indicator	Where Located	Where Used	Turned On	Turned Off	Notes
Record Identifying Indicator 01-99	Input form Columns 19-20	Indicators (calc.), Output Indicators Field-Record Relation	when specified record has been read and before total calculations are executed	before the next record is read during the next processing cycle	Note 1
Resulting Indicators 01-99 Plus Minus Zero Compare operation High Low Equal Lokup operation High Low Equal Testz operation High Low Equal Chain operation	Calculation form	Indicators (calc.), Output Indicators	by a positive balance in field, by a negative balance in field, by Zero balance in field if Factor 1 > Factor 2 if Factor 1 < Factor 2 if Factor 1 = Factor 2 if table > Factor 1 if table < Factor 1 if table=Factor 1 if 12 zone is present if 11 zone is present if no zone is present by a no record found condition	the next time a calculation is performed for which the program specifies the indicator as a resulting indicator and the specified condition is not satisifed	Note 1
1P (First Page)	Internal	Output Indicators	at beginning of processing before any input records are read	before the first detail card is read	Note 4

Note 1. Turning indicators on or off can also be accomplished by using SETON and SETOF operation codes.
Note 2. All control level indicators (L1-9) are also turned on when LR is turned on.
Note 3. The OF indicator remains on during the following detail calculations and output cycles.
Note 4. This indicator is used to condition printing of the first page of the report.

This appendix contains a detailed description of the flow of an RPG object program. Knowledge of this flow is helpful in preparing RPG programs with complex operations.

A detailed flowchart of an RPG object program is shown in Figure 156. The item numbers in the following text refer to the numbers in the figure. A program cycle begins with item 1 and continues through item 23.

1. Performs all specified heading and detail output operations whose conditions are satisfied. This does not include output line specifications that are conditioned by the OF or OV indicator. If a line is conditioned to output on NOF or NOV, that operation is done here.
2. Determines if a punch in channel 12 of the carriage-control tape was encountered at detail time. If so, the OF/OV indicator is set on. Otherwise, the indicator is set off.
3. Tests the halt indicators. If the halt indicators are off, the program proceeds to step 4. If they are on, program execution is interrupted and a message is displayed in the accumulator. The operator may cancel or continue the program.
4. Sets off record identifying indicators and the indicators 1P, LR, L1 through L9, and H1 through H9.
5. Determines if the record is an end-of-file record. If an end-of-file condition has occurred, the program branches to step 24.
6. Reads the next input records from the file just processed. At the beginning of processing, if it is a multifile job, one input record from each file is read into core storage.
7. Determines if the input records are in the sequence specified for them on the Input Specifications form. If the sequence is incorrect, the program branches to step 34. The program also branches to step 34 if nonsequential input records are specified and the record cannot be identified.
8. Branches to step 27 if matching fields are specified.
9. Sets on the record identifying indicator specified for the current record type.
10. Determines if a control break has occurred (i.e., the contents of this control field are not equal to the contents of a previously stored control field). If a control break has *not* occurred, the program branches to step 12. (The control fields are initially set to hexadecimal zeros.)
11. If a control break has occurred, the appropriate control-level indicators are set on and the control fields are moved into a save area.
12. If this is the first program cycle, RPG bypasses the total calculations and the total output specifications and branches to step 15.
13. Performs total calculations conditioned on the appropriate level indicators (L1-L9), and sets resulting indicators on or off as specified.

    If the user specified a closed subroutine, the program branches to the specific subroutine mentioned, then returns to the next total calculation.

    If the user specifies the EXCPT operation code, the program puts out the exception output, then returns to the next total calculation.
14. Performs all total output that is not conditioned by an overflow indicator.
15. Determines if an overflow condition has occurred (i.e., a punch in channel 12 of the carriage-control tape has been reached or passed in a spacing operation.) If an overflow condition has occurred at any time during this cycle, the indicator (OF or OV) associated with the file that overflowed is set on.
16. Determines if one or both indicators, OF and OV, are on. If no overflow indicator is on, RPG branches to step 18.
17. If overflow has occurred in either step 1 or step 14, total lines, heading lines, and detail lines (in that order) conditioned by overflow are printed. If no overflow output is specified, RPG skips until a punch is sensed in channel 1 of the carriage-control tape.
18. Determines if the last-record indicator (LR) is on. If the indicator is on, the job has been completed; therefore, the program ends.
19. The MR indicator is set on if this is a multifile job and the record to be processed is a matching record. Otherwise, the MR indicator is set off. If the user has coded a loop from Detail to Total Calculations, the MR indicator reflects only the correct status the first time Detail Calculations are performed.
20. Moves the input data to the process area (i.e., the fields that are specified on the Input Specifications form). No sign checking is performed on numeric fields.
21. Sets field indicators on or off as specified.
22. If chaining fields are specified, they are moved into the appropriate fields, C1 to C3, and internal chaining switches are set on by the program. Once all the fields of a chaining record have been moved to the process area, chaining begins with the lowest chaining field (C1). A chained file record may also contain chaining fields. If no record is found, the system halts

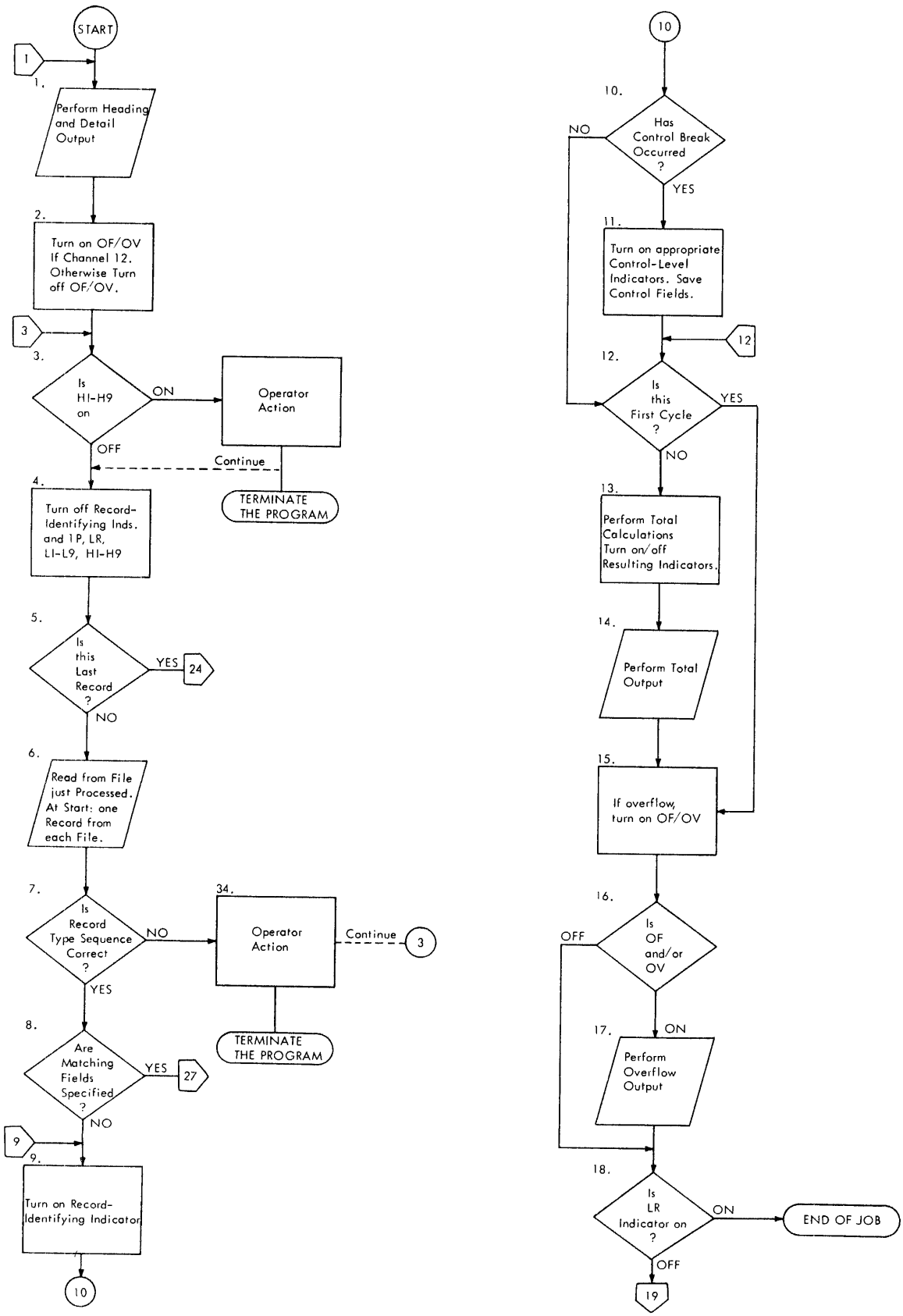


Figure 156. Detailed Logic Flow of a Program Generated by RPG (Part 1 of 2)

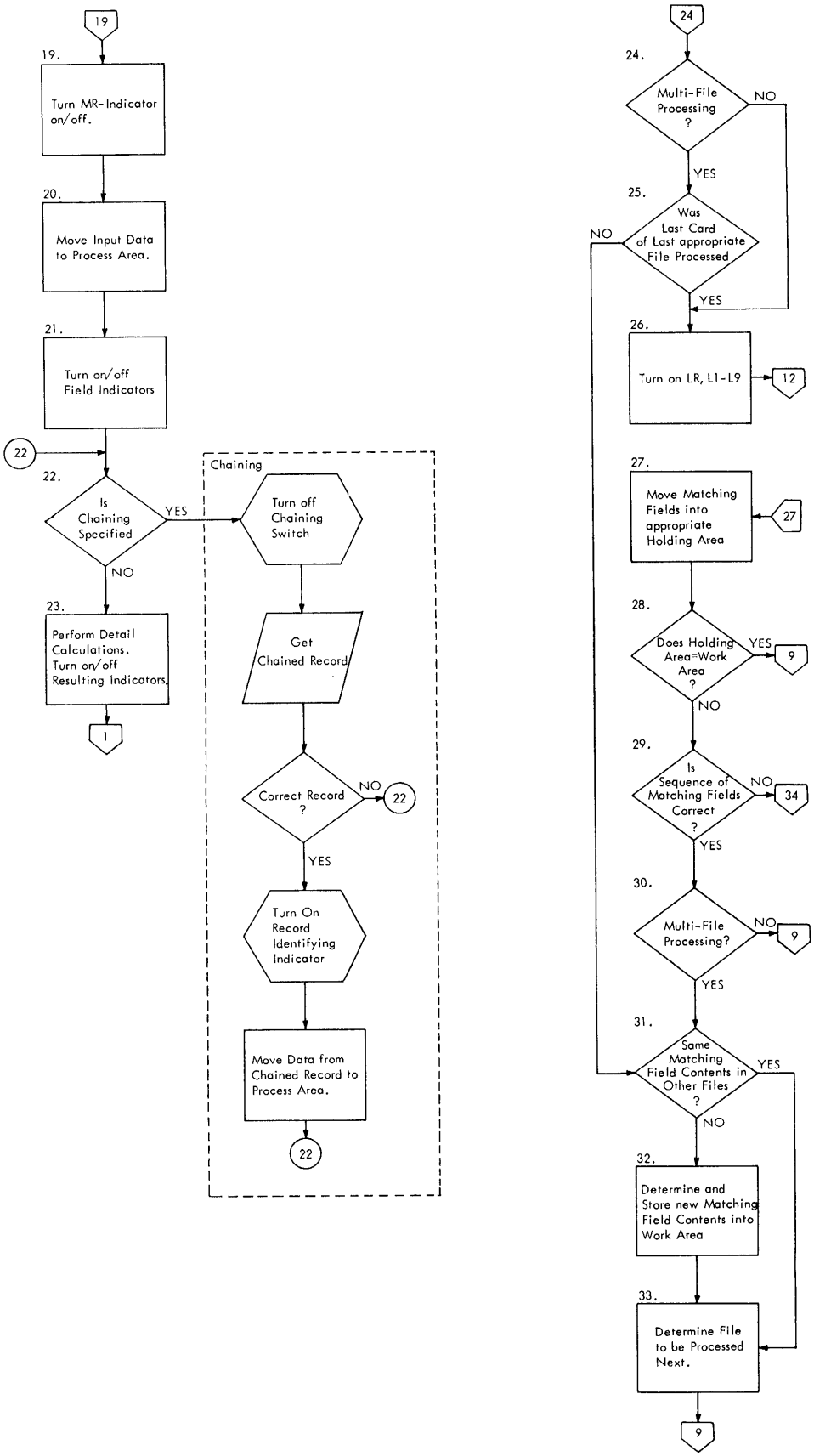


Figure 156. Detailed Logic Flow of a Program Generated by RPG (Part 2 of 2)

and provides an appropriate message to the operator. The operator has the option of either ending the job or bypassing the record and continuing processing at step 3.

23. Performs any specified detail calculations and sets resulting indicators on or off as specified. Processing continues with step 1.

    If the user specifies a closed subroutine, the program branches to the specified subroutine, then returns to the next detail calculation.

    If the user specifies the EXCPT operation code, the program puts out the exception output, then returns to the next detail calculation.

24. If only one input file is to be processed, RPG branches to step 26.

25. Determines if the processing of all the files specified with an E in column 17 of the File Description Specifications form has been completed. If not, the program branches to step 31.

    If the primary file is the only file with an E specified, but a matching secondary is present, all records which match the primary will be processed. The LR indicator will then be set on. Refer to *End of File Processing* for a more detailed discussion of end-of-file condition.

26. All control-level indicators (L1-L9) and the last-record indicator (LR) are set on and processing continues with step 12.

27. The specified matching fields for the record just read are moved into the appropriate holding area for that file. This holding area contains the control information for the file just processed.

28. The contents of the holding area for the record just read are compared to the contents of the matching-field work area. If the contents are equal, the record just read is processed next with the program branching to step 9.

29. Determines if the sequence of matching fields is correct. If the sequence is incorrect, the program branches to step 34.

30. Determines if more than one file is to be processed. If only one input file is to be processed, the program branches to step 9.

31. The contents of the holding area of the other input files are compared to the contents of the matching field work area. If the contents are equal, the program branches to step 33.

32. As no matching field has been found, each file's matching field holding area is checked to determine the new matching field. The selected field is moved into the matching field work area. Selection will depend upon whether ascending or descending sequencing was specified.

33. Determines the next file to be processed according to matching - records techniques discussed in the section *Processing Multiple Input Files*, and branches to step 9.

34. Program execution is halted and an appropriate message is made available to the operator. The operator has the option of either ending the job or bypassing the record and continuing processing with the next record.

The INDX (Indexing) subroutine significantly increases 1130 RPG capability. This standardized exit allows any one or combinations of the following functions to be performed in one program.

- Consecutively extracting fields from within a data area.
- Extracting the Nth field from within a data area.
- Consecutively moving into fields within a data area.
- Moving into the Nth field in a data area.
- Extraction of a variable length field beginning at the Nth byte of a data area.
- Moving into a variable length field at the Nth byte of a data area.

Once compiled this subroutine should work in any 1130 RPG program. The subroutine requires 128 words of core storage.

The user must accurately follow one or the other of the examples shown in this section and use the appropriate field names and indicators.

1. The concept of INDX is as follows:
   a. Data area with fixed-length fields. It is often desirable to create or process a data area consisting of fixed-length fields (much like a table) in core and treat it as *one* record or field. If this data area is made up of several fields, the RPG user must provide duplicate sets of coding to handle each field, or complex coding to move these fields to a work area for common processing.

      The table lookup routine does a *sequential search* through a table. The INDX routine allows *immediate access* to any field within the data area.

   b. Data area with variable-length fields. It is sometimes desirable to be able to access a character or characters within a data area (The user could request four characters beginning at character number 134 of a data area). This is especially helpful in programs which perform maintenance type activity on master records. INDX provides the ability to extract, or move into a field, a variable length of data, beginning at a variable location.

2. Requirements
The user must properly use the following named fields and indicators. They must be specified in this order, and immediately following the EXIT statement.

- IBIG – Must be used to describe the large data area. It must be defined as alphabetic. (RPG requires that a field be 256 characters or less, in length).
- IWRK – Must be used to describe the work area that is used to move to, or from, IBIG. It can be either alphabetic or numeric and can be of any length. (The move is performed within the subroutine).
- ILTH – Must be used to describe the length of the move to be made. It must be described as a three-position numeric field with zero decimals. The value that is placed into ILTH should be less than or equal to the length of IWRK.
- ICNT – Must be used to describe either:
  a) Which field in IBIG is to be used (all fields of equal lengths).
  b) At which byte in IBIG to begin (variable length fields). ICNT must be described as a 3-position numeric field with zero decimals.
- IN85 – (Indicator 85) – Must be used to cause INDX to either:
  1) If "On" – Move into IBIG (change value).
  2) If "Off" – Move out of IBIG (extract a field from IBIG).
- IN86 – (Indicator 86) – Must be used to cause INDX to either:
  1) If "On" – Begin move with value in ICNT (specific byte location within IBIG).
  2) If "Off" – Multiply ILTH x ICNT and use the result as the beginning byte (Index to the Nth field).

The user must provide RLABL statements which provide for communication between the RPG program and the BAL subroutine. They should be written immediately following the exit statement. If the field lengths have not been previously defined in RPG statements they should be described in the RLABL card.

3. Compiling with the Subroutine
Prior to compiling the RPG program, the INDX subroutine must be assembled and cataloged. The subroutine will require 128 words of storage plus the RPG linkage. During compilation, a warning will occur due to indicators 85 and 86 being unreferenced.

4. HALT – FF00 within INDX
A halt can occur in INDX which will be displayed in the accumulator lights as FF00. This is a *non-recoverable halt*. Push start to end the job. The program branches to a point within RPG where all files are closed. FF00 will usually occur because of a user error. It is the result of not correctly specifying the required name fields, or requesting a character which is outside of IBIG. This could occur through a multiplication of ILTH and ICNT which would produce a number greater than the length of IBIG.

If the user wished to specify that IBIG consisted of four, 10-character fields, he would state the following:

```
Z-ADD 10        ILTH 3/0
```

If through an error in logic, an ICNT=5 occurred, the subroutine would be working with a field that was outside of IBIG. Rather than allow this to occur, the subroutine will cause a FF00 HALT. Not all possible programmer, or data, errors are checked for. The following describes the checking provided.

ICNT = 000
ILTH = 000

If 86 is off
(ICNT -1) x ILTH must be equal to or less than the length of IBIG.

If 86 is on
(ICNT + ILTH -1) must be equal to or less than the length of IBIG.

If the user's data contains possible errors which will cause an undesired halt, he should make the above tests with RPG statements and not enter the subroutine if they fail. Errors could then be handled by stacker selection rather than by halting.

5. Other User Responsibilities
Since the INDX routine requires that values be specified for various fields, it is the user's responsibility to determine when the values should be placed in the appropriate field names. It is also important to control a count in several operations. Positioning the ADD and COMP statements relative to each other is important.

6. Performing Arithmetic in IBIG
It is possible, and may be desirable, to use IBIG as a series of counters. IBIG is limited to 256 characters, but several large fields could be moved into IBIG. It is possible to perform this same function by updating tables. However, the lookup can be slow because of its sequential nature.

7. Examples of Uses of INDX
The following is a discussion of some typical uses of INDX and some examples which are shown in Figures 157-161.

a. Example 1 – Extracting fields sequentially from IBIG (Figure 157).

   This example shows INDX being used to extract fields from an input record. This could be used in the case of a spread card or record. Both indicators (85 and 86) should be off for this function.

   IBIG is 30 bytes long and is made up of six fields of five bytes each. Line 05 of the Calculation specifications shows a COMP to six being made (Number of fields).

   Indicator 82 is used to start the extraction again. Looping and exception output is used to allow printing of each field extracted.

b. Example 2 – Moving a Variable Length Field to a Variable Location in IBIG (Figure 159).

   This example shows INDX being used to modify any part of or all of IBIG. Indicators 85 and 86 should be on for this function.

   Both IWRK and IBIG are in the same input record, but normally they would be in different records.

   Variable values are shown for ICNT and ILTH.

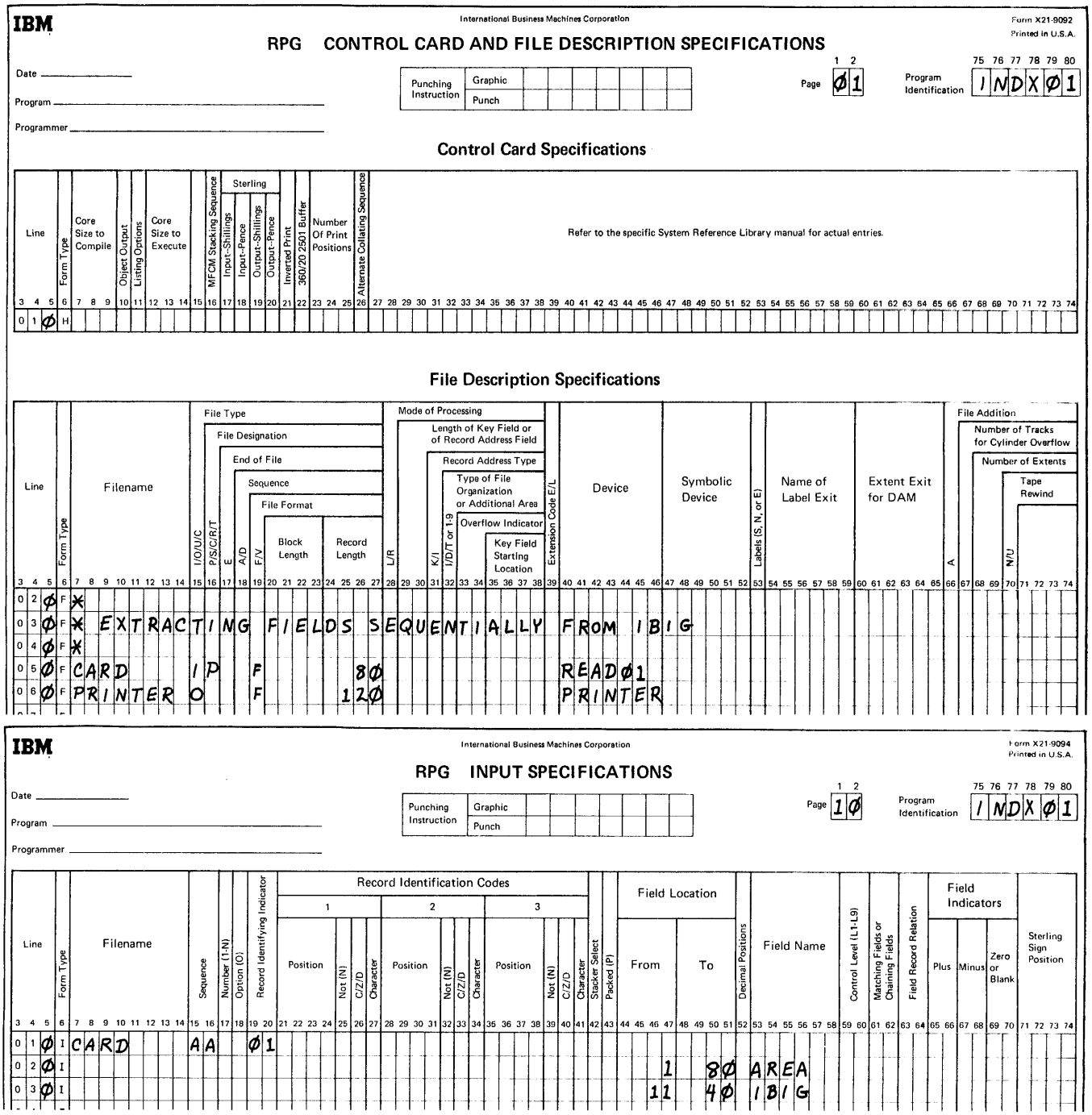
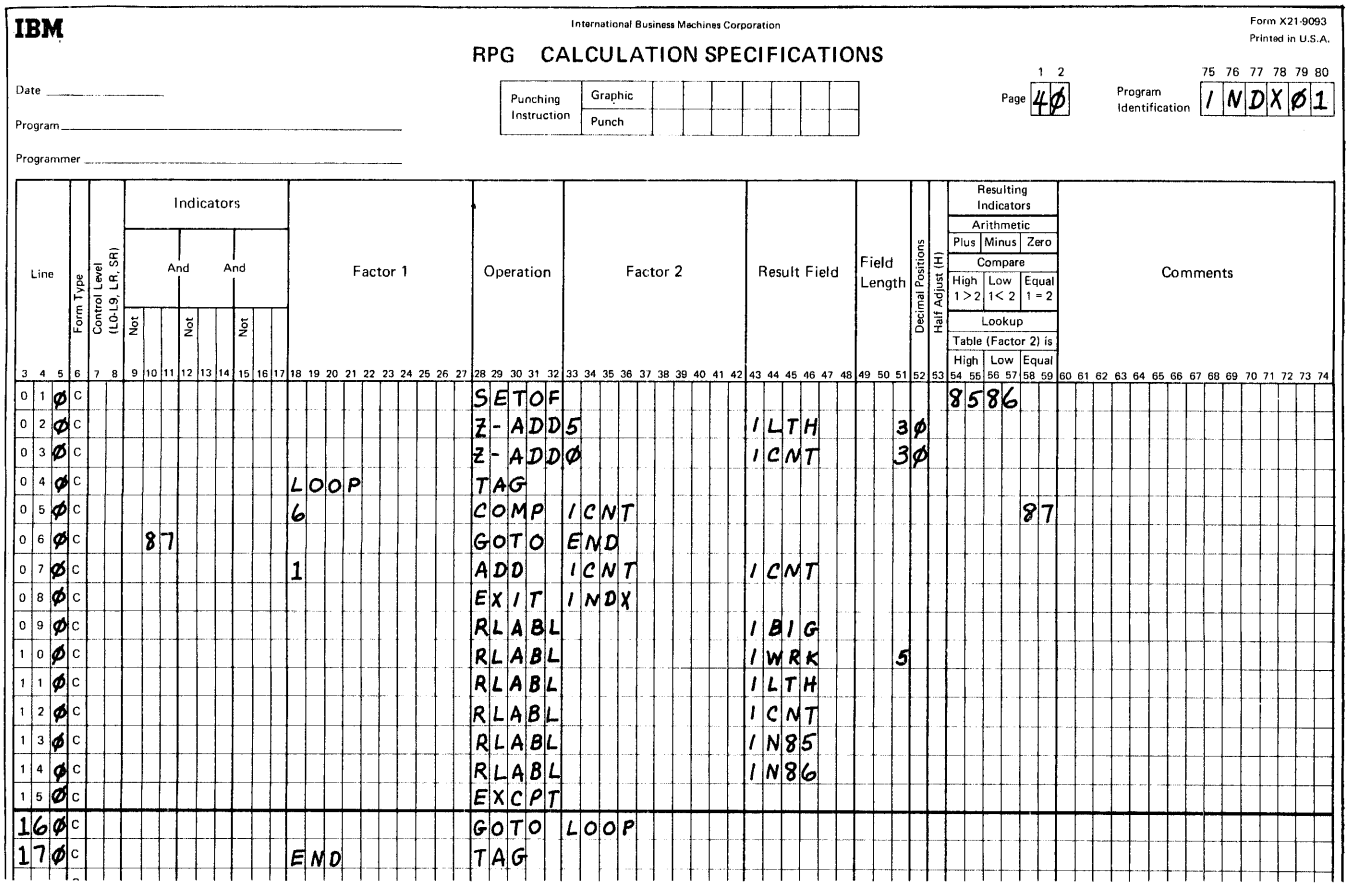


Figure 157. Extracting Fields Sequentially from IBIG (Part 1 of 2)



IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date

Program

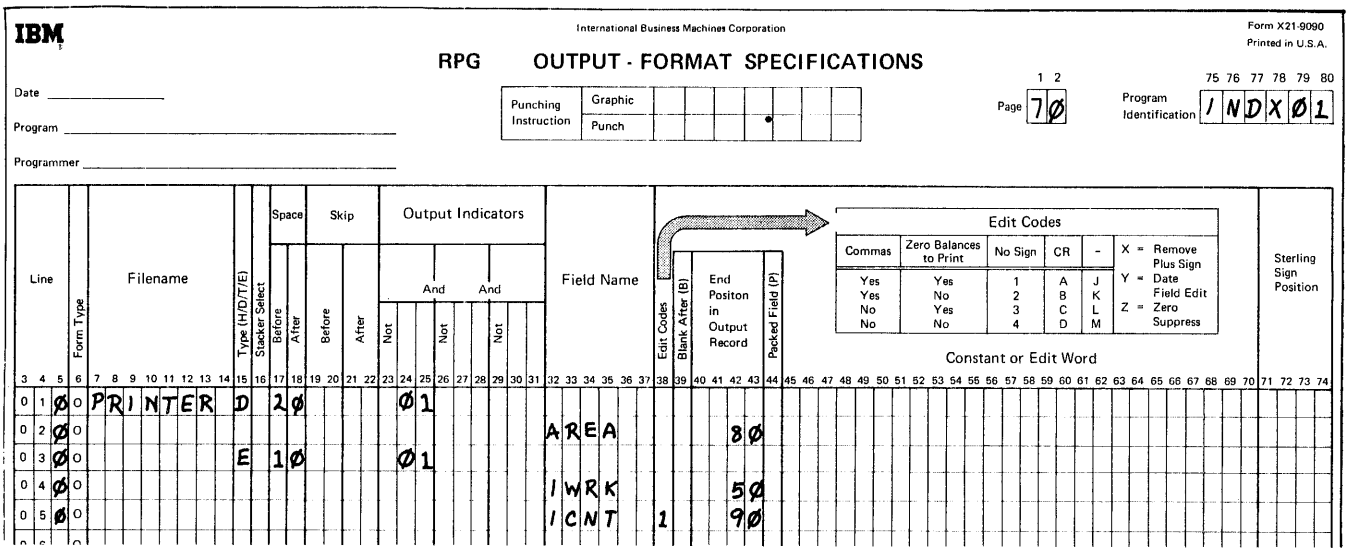
Programmer

Punching Instruction: Graphic / Punch

Page 40

Program Identification: INDX01

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 / Minus / Zero); Compare (High 1>2 / Low 1<2 / Equal 1=2); Lookup Table (Factor 2) is (High / Low / Equal)	Comments
010	C				SETOF						85 86	
020	C				Z-ADD 5		ILTH		30			
030	C				Z-ADD Ø		ICNT		30			
040	C			LOOP	TAG							
050	C			6	COMP	ICNT					87	
060	C		87		GOTO	END						
070	C			1	ADD	ICNT	ICNT					
080	C				EXIT	INDX						
090	C				RLABL		IBIG					
100	C				RLABL		IWRK		5			
110	C				RLABL		ILTH					
120	C				RLABL		ICNT					
130	C				RLABL		IN85					
140	C				RLABL		IN86					
150	C				EXCPT							
160	C				GOTO	LOOP						
170	C			END	TAG							



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 70

Program Identification: INDX01

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select	Space: Before / After	Skip: Before / After	Output Indicators: And / Not	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	Packed Field (P)	Sterling Sign Position
010	O	PRINTER	D		2		Ø1						
020	O							AREA			80		
030	O		E		1		Ø1						
040	O							IWRK			50		
050	O							ICNT	1		90		

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

Constant or Edit Word

Figure 157. Extracting Fields Sequentially from IBIG (part 2 of 2)

```
                                    11111                              1
                                    22222                              2
                                    33333                              3
                                    44444                              4
                                    55555                              5
                                    66666                              6

1111122222333334444455555666666

                                    AAAAA                              1
                                    BBBBB                              2
                                    CCCCC                              3
                                    DDDDD                              4
                                    EEEEE                              5
                                    FFFFF                              6

AAAAABBBBBCCCCCDDDDDEEEEEFFFFF
```

Figure 158. Sample Output of INDX01

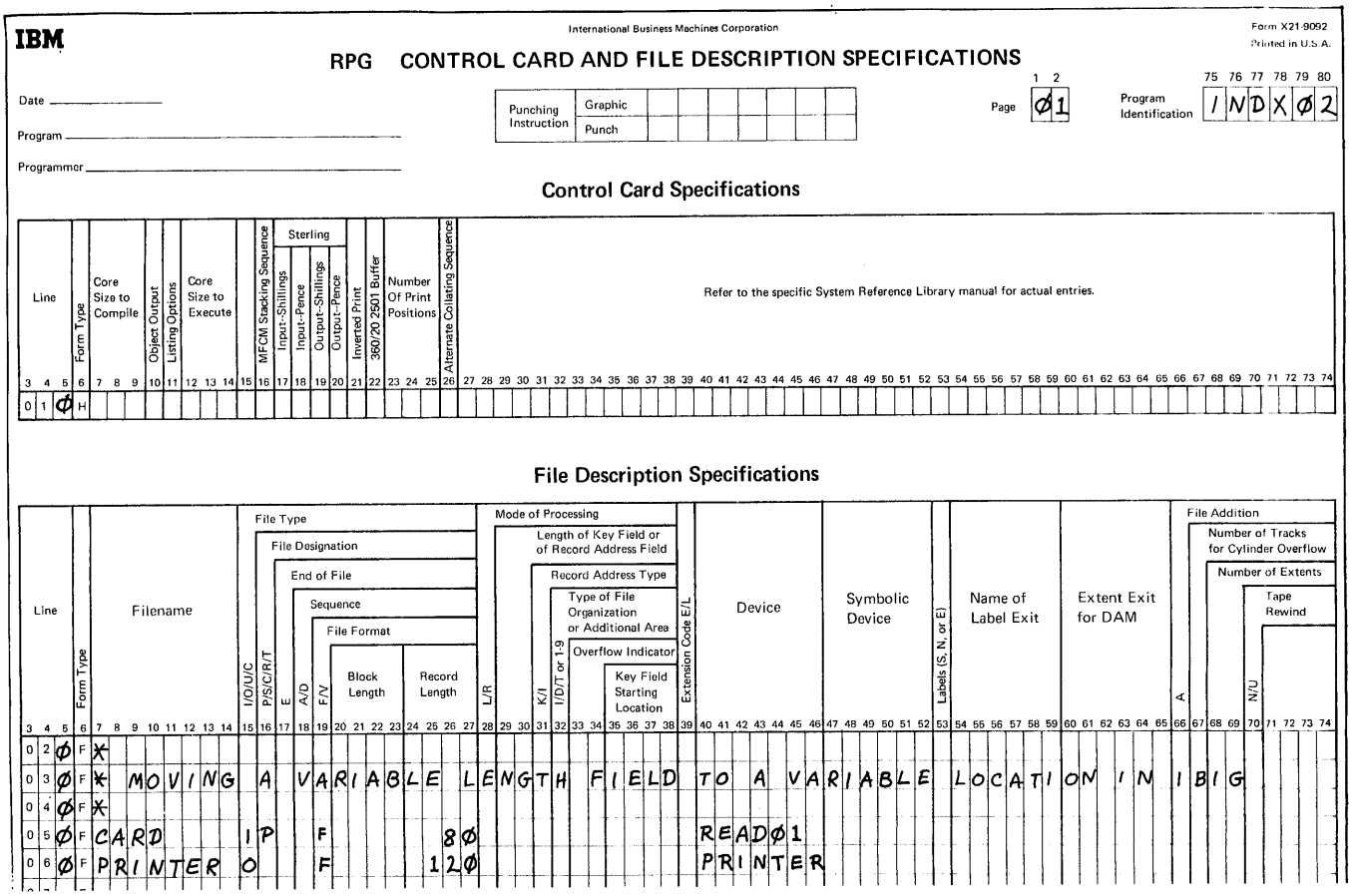


Figure 159. Moving a Variable Field to a Variable Location in IBIG (Part 1 of 2)

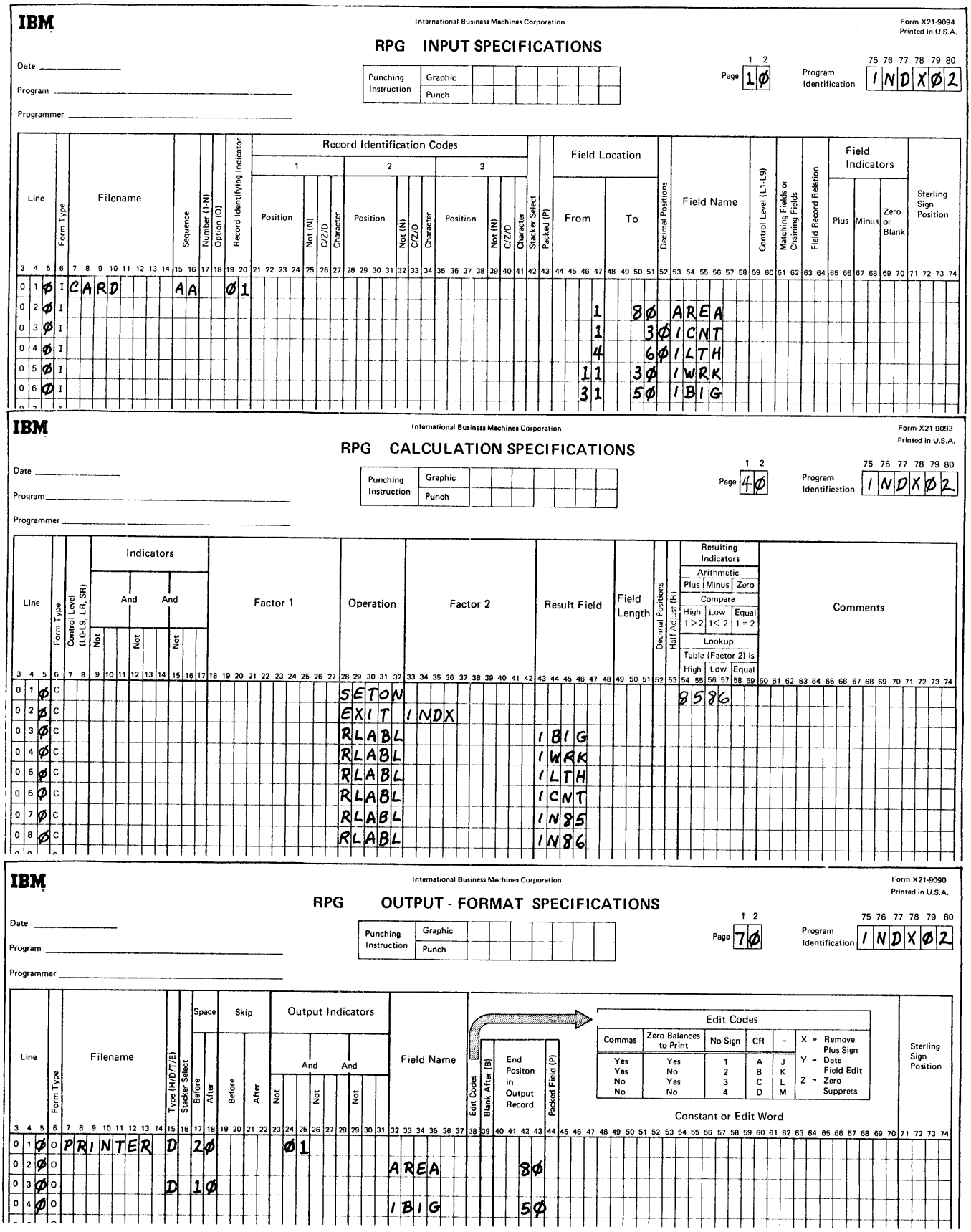


Figure 159. Moving a Variable Field to a Variable Location in IBIG (Part 2 of 2)

```
001001      X                         12345678901234567890
                                      X2345678901234567890

001020      XX                        12345678901234567890
                                      XX

001020      XXXXXXXXXXXXXXXXXXXX      12345678901234567890
                                      XXXXXXXXXXXXXXXXXXXX

005002      XX                        12345678901234567890
                                      1234XX78901234567890

009003      XXX                       12345678901234567890
                                      12345678XXX234567890

015005      XXXXX                     12345678901234567890
                                      12345678901234XXXXX0
```

Figure 160. Sample Output of INDX02

```
        ENT       INDX
INDX    DC        *-*           RETURN ADDRESS
        STX    1  RSTRX+Q1      SAVE CONTENTS OF REG 1
        STX    2  RSTRX+Q3      SAVE CONTENTS OF REG 2
        STX    3  RSTRX+Q5      SAVE CONTENTS OF REG 3
        LDX    I1 INDX          LOAD RLABL ADDR MINUS 2
        LD     1  Q2            RETRIEVE THE ADDRESS
        STO       IBIGA         OF FIELD AREA
        LD     1  Q3            RETRIEVE THE ADDRESS
        STO       IWRKA         OF WORK AREA
        LD     I1 Q6            RETRIEVE THE INDICATOR
        STO       IN85S         STATUS OF IN85
        LD     I1 Q7            RETRIEVE THE INDICATOR
        STO       IN86S         STATUS OF IN86
        MDX    1  Q4            POINT TO ADDR OF ILTH
        STX    1  *+Q1          LOAD REG 3 WITH THE ADDR
        LDX    I3 *-*           OF FIRST WORD OF ILTH
        LDX    L2 ILTHL         LOAD WORK AREA ADDR
        BSI       CVB           GO TO CONVERT TO BINARY
        MDX    1  Q1            POINT TO ADDR OF ICNT
        STX    1  *+Q1          LOAD REG 3 WITH THE ADDR
        LDX    I3 *-*           OF FIRST WORD OF ICNT
        LDX    L2 ICNTL         LOAD WORK AREA ADDR
        BSI       CVB           GO TO CONVERT TO BINARY
        S         ONE           SUBTRACT ONE FROM ICNT
        SRT       Q16           SAVE RESULT IN THE EXT
        LD        IN86S         LOAD INDICATOR IN86
        BNZ       CNTU1         BRANCH IF INDICATOR IS ON
        SLT       Q16           RESTORE ICNT
        M         ILTHL         MULTIPLY ILTH TO ICNT
CNTU1   SLT       Q16           MOVE RESULT TO THE ACC
        STO       ICNTL         SAVE THE RESULTS
        LDX    I1 IBIGA         LOAD ADDR OF FIELD AREA
        LD     1  Q0            LOAD FIELD AREA CONTROL WD
        SRA       Q8            RIGHT JUSTIFY THE LENGTH
        A         ONE           CORRECT THE FIELD LENGTH
        S         ICNTL         SUBTRACT THE DISPLACMENT
        BNP       INDXE         GO TO ERR IF DISP BAD
        SRT       Q16           SAVE RESULT IN EXT
        LD        IN86S         LOAD INDICATOR IN86
        BZ        CNTU2         BRANCH IF INDICATOR IS OFF
        SLT       Q16           RESTORE SAVED RESULTS
        S         ILTHL         SUBTRACT LENGTH OF MOVE
        BN        INDXE         GO TO ERR IF RESULT NEG.
CNTU2   LDX    I2 IWRKA         LOAD ADDR OF WORK AREA
        MDX    I1 ICNTL         ADD DISP TO FIELD ADDR
        LD        IN85S         LOAD INDICATOR IN85
        BZ        MOVLP         IF INDICATOR OF GO TO MOVE
        LD     L  Q1            IF IND OFF EXCHANGE THE
        LDX    I1 Q2            CONTENT OF REGISTER 1 AND
        STO    L  Q2            2 TO REVERSE THE MOVE
```

Figure 161. Indexing Subroutine, Listing of (Part 1 of 3)

```
MOVLP LD     1 Q1          MOVE THE CONTENTS OF ONE
      STO    2 Q1          FIELD TO THE OTHER FIELD
      MDX    1 Q1          INCREMENT REG 1
      MDX    2 Q1          INCREMENT REG 2
      MDX  L   ILTHL,-Q1   IS THE MOVE COMPLETE
      B        MOVLP       IF NOT CONTINUE MOVE
RSTRX LDX  L1 *-*          IF YES RESTORE THE
      LDX  L2 *-*          INDEX REGISTERS
      LDX  L3 *-*          1,2 AND 3
      BSC  I   INDX        RETURN TO RPG
*
*              CONSTANTS
*
X000F DC       /000F
XFF00 DC       /FF00
ONE   DC       1
LPCT  DC       2
TEN   DC       10
*
*              WORK AREA'S
*
IBIGA DC       *-*         ADDR OF FIELD AREA
IWRKA DC       *-*         ADDR OF WORK AREA
IN85S DC       *-*         SETTING OF INDICATOR 85
IN86S DC       *-*         SETTING OF INDICATOR 86
ILTHL DC       *-*         LENGTH OF MOVE
ICNTL DC       *-*         DISPLACEMENT OT FIELD
*
CVB   DC       *-*         CONTAINS RETURN ADDRESS
      LD     3 Q1          LOAD HIGH-ORDER DIGIT
      AND      X000F       STRIP OFF THE ZONE
SECTM M        TEN         MULTIPLY BY A HEX A
      LD     3 Q2          NEXT DIGIT
      AND      X000F       STRIP OFF THE ZONE
      STO    2 Q0          STORE IN WORK AREA
      SLT      Q16         SHIFT RESULT TO THE ACC
      A      2 Q0          ADD THE WORD OF IN WORK
      MDX    3 Q1          INCREMENT REG 3 BY 1
      MDX  L   LPCT,-Q1    FIRST TIME THROUGH LOOP
      B        SECTM       IF YES LOOP AGAIN
      MDX  L   LPCT,+Q2    INIALIZE LOOP CONTROL
      BNP      INDXE       IF NOT POS GO TO ERROR RTN
      STO    2 Q0          STORE RESULT AT WORK
      BSC  I   CVB         RETURN TO CALLER
INDXE MDX  L   $IOCT,Q0    I/O DONE
      B        INDXE       IF NOT LODP
      LD       XFF00       LOAD ERROR INDICATOR
      BSI  L   $PRET       GO TO WAIT
      LDX  L1  CL1         LOAD DISPLACMENT TO CLOSE
      MDX  I1  LD7B        ADD RELOCATION ADDRESS
      B    L1  Q0          EXIT TO CLOSE
*
*              EQUATES
*
```

Figure 161. Indexing Subroutine, Listing of (Part 2 of 3)

```
CL1     EQU     /0130       DISPLACMENT TO CLOSE
LD7B    EQU     /007B       RELOCATION ADDRESS
$PRET   EQU     /28         WAIT
$IOCT   EQU     /32         I/O IN PROCESS
Q0      EQU     0
Q1      EQU     1
Q2      EQU     2
Q3      EQU     3
Q4      EQU     4
Q5      EQU     5
Q6      EQU     6
Q7      EQU     7
Q8      EQU     8
Q16     EQU     16
        END
```

Figure 161. Indexing Subroutine, Listing of (Part 3 of 3)

READER'S COMMENT FORM

IBM 1130 RPG Language

Form C21-5002-1

- Your comments, accompanied by answers to the following questions, help us produce better publications for your use. If your answer to a question is "No" or requires qualification, please explain in the space provided below. Comments and suggestions become the property of IBM.

	Yes	No
• Does this publication meet your needs?	☐	☐
• Did you find the material:		
Easy to read and understand?	☐	☐
Organized for convenient use?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Written for your technical level?	☐	☐

- What is your occupation? _____
- How do you use this publication?

As an introduction to the subject?	☐	As an instructor in a class?	☐
For advanced knowledge of the subject?	☐	As a student in a class?	☐
For information about operating procedures?	☐	As a reference manual?	☐

Other _____

- Please give specific page and line references with your comments when appropriate.

COMMENTS:

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

YOUR COMMENTS PLEASE . . .

This SRL bulletin is one of a series which serves as reference sources for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

fold fold

FIRST CLASS
PERMIT NO. 387
ROCHESTER, MINN.

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
Systems Development Division
Development Laboratory
Rochester, Minnesota 55901

Attention: Programming Publications, Dept. 425

fold fold

IBM 1130 Printed in U.S.A. C21-5002-1

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]