File No. S360–28
Order No. GC24-3337-6

IBM Systems Reference Library

IBM System/360
Operating System
RPG Language Specifications

Program Number 360S-RG-038 (OS)

This reference publication contains fundamentals of RPG programming and language specifications for the IBM System/360 Operating System RPG. Also included is the job setup information necessary for executing RPG.

Preface

System/360 Operating System 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 reports.
4. Use table lookup.
5. Exit to a user's subroutine written in a language other than RPG.
6. Branch within the calculations.
7. Sequence check input records.
8. Maintain files.

RPG uses a set of specification 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.

Seventh Edition (July 1973)

This is a reprint of GC24-3337-5 incorporating changes issued in Technical Newsletter GN21-5204, dated September 20, 1972.

Information in this publication is subject to change. Before using this publication, be sure you have the latest edition and any Technical Newsletters.

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

Some illustrations in this manual have a code number in the lower corner. This is a publishing control number and is not related to the subject matter.

Address comments concerning the contents of this publication to IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901. Comments become the property of IBM.

Contents

RPG operates under control of the System/360 Operating System (referred to as the operating system). The operating system provides the RPG compiler with input and output services. Object programs generated by the RPG compiler also operate under operating system control and depend on it for similar services.

RPG supports the minimum configuration required by the operating system. The decimal arithmetic feature is required for RPG.

If you choose to run the RPG compiler under control of OS/VS1, you must apply changes to the compiler. See *Appendix F. Running the OS RPG Compiler Under OS/VS1* for the specific changes that must be made.

COMPATIBILITY OF SYSTEM/360 OPERATING SYSTEM

The RPG source language is upward compatible. The operating system Report Program Generator can compile a Model 20, Basic Programming Support, Tape Operating System, Disk Operating System, or Basic Operating System source program that adheres to its language specifications.

The File Description Specifications form of the operating system RPG may require additional information (record length and overflow indicators).

The control card of a Model 20 or Basic Programming Support program must be modified before the object program can be generated by the operating system RPG.

When a BOS, TOS, or DOS RPG program is used to retrieve records from a file with direct organization, it may be necessary to alter the method of obtaining the addresses of these records.

The operating system RPG does not use the address output option of the Basic or Disk Operating System Disk Sort/Merge to retrieve records.

FUNCTION OF RPG

When RPG is used, the IBM System/360 actually performs separate functions:

1. Program generating
2. Data processing.

In the first function, program specifications defined by the user produce machine-language instructions. Storage areas are automatically assigned, constants or other reference factors are included, and linkages to routines for computations for input/output operations and for other functions are produced.

In the second function, the machine-language instructions (created in the first function) are combined with the user input-data files, and both are processed through the system to produce the desired reports or output files.

USING RPG

The preparation of a report by means of RPG consists of the general operations shown in Figure 1 and described as follows:

1. The programmer must evaluate the report requirements to determine the format of the input files and the appearance of the finished report.

   For example, he must know what fields in the input files are to be used, what kind of calculations are to take place, the location of the data in the output records, and the number and the kind of totals that must be accumulated. More specific information regarding the evaluation of report requirements is described in *Program Logic, Problem Definition.*

2. After the programmer has evaluated the requirements of the report, he provides the same information to the RPG program.

   He must describe his input (record layout, fields used, etc.) by making entries on an Input Specifications form.

   He must state what processing is to be done (add, subtract, multiply) by entries on a Calculation Specification form.

   He must state how the finished report is to look (printing position, carriage control, etc.) by making entries on the Output-Format Specifications form.

   He must describe all files used by the object program (input files, output files, table files, etc.) by making entries on the File Description and Extension Specifications forms.

3. After the specifications have been written on the appropriate forms, cards are punched with the data from the forms.

4. These punched cards (called a source deck) are combined with the processor control card and the operating system job control statements. The source deck and the control cards are supplied to an input device and are processed under control of the operating system. At the end of this step (known as a compilation step), a program capable of preparing the report specified by the programmer has been produced. This program (known as an object module) is processed by the link edit step to create a loadable program. This loadable program (known as the load module) contains all of the computer instructions and linkages to the control system necessary to prepare the desired report.

5. The input files can then be read into the system, and processing of the program will begin. This is known as the execution step.

   At the end of the execution step, the report has been prepared and any other functions, such as file updating, are completed. Through facilities provided by the operating system, the object module can be retained for later runs without recompilation.

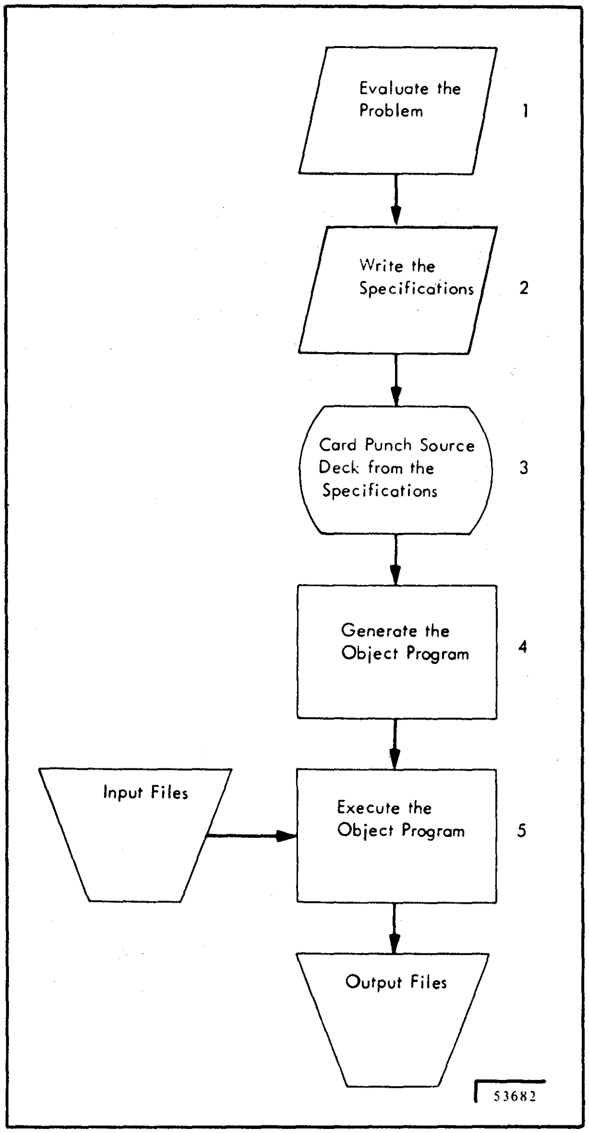

Evaluate the Problem
1
Write the Specifications
2
Card Punch Source Deck from the Specifications
3
Generate the Object Program
4
Input Files
Execute the Object Program
5
Output Files
53682

Figure 1. Producing Reports Using the RPG Program

MACHINE FEATURES REQUIRED

Source Program Compilation

1. 18,432 main storage bytes (minimum partition size).
2. One of the following for input:
   a. IBM 2540 Card Read-Punch
   b. IBM 2501 Card Reader
   c. IBM 2520 Card Read-Punch
   d. IBM 1442 Card Read-Punch
   e. IBM 2400 Series Magnetic Tape Unit
3. One of the following for output of object program:
   a. IBM 2540 Card Read-Punch
   b. IBM 2520 Card Read-Punch
   c. IBM 1442 Card Read-Punch
   d. IBM 2400 Series Magnetic Tape Unit
   e. IBM 2311 Disk Storage Drive
   f. IBM 2314 Direct Access Storage Facility
   g. IBM 2302 Disk Storage
   h. IBM 2321 Data Cell Drive
   i. IBM 2301 Drum Storage
   j. IBM 2303 Drum Storage
4. One direct access storage device for system residence
5. One of the following for system utility files:
   a. IBM 2400 Series Magnetic Tape Unit (three)
   b. Data convert feature (required only if 7-track tape is used)
   c. IBM Direct Access Storage Device with three files
6. Standard and decimal instruction sets
7. Minimum machine configuration required by the operating system

Object Module Execution

1. Partition size sufficient to execute the generated object module.
2. I/O units as requested by the specifications
3. Standard and decimal instruction sets
4. One direct access storage device for system residence
5. Minimum machine configuration required by the operating system

ADDITIONAL MACHINE FEATURES SUPPORTED

Source Program Compilation

1. 131,072; 262,144; or 524,288 main storage bytes
2. One of the following for listings:
   a. IBM 1443 Printer
   b. IBM 1403 Printer
   c. IBM 1404 Printer (continuous forms operation only)
   d. IBM 2400 Series Magnetic Tape Unit (9 track)
3. Two additional direct access storage devices for utility files

Object Module Execution

1. 131,072; 262,144; or 524,288 main storage bytes

This section introduces some of the basic RPG functions. These functions are considered basic because they are probably used in the most typical reports produced by RPG and because they depend on a single sequential input file only. They are also related to the functions commonly performed with conventional punched card equipment.

Although RPG programs can process files contained on magnetic tape or direct access storage devices, only card input files are used for the introduction to RPG. Programmers familiar with the basic concepts of RPG may omit this section and concentrate on the detailed specification descriptions contained in the manual under *RPG Specification forms.*

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

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

Fifteen of the most basic RPG functions are described in this section:

1. Describing a record and its fields to the system
2. Addition and subtraction
3. Detail printing
4. Control fields
5. Total calculations
6. Detail and total printing
7. Group printing
8. Group indication
9. Overflow printing
10. Summary punching
11. Testing for zero, plus, and minus balance
12. Using resulting indicators
13. Comparison of two fields
14. Multiplication and division
15. Sequence checking

DESCRIBING A RECORD AND ITS FIELDS TO THE SYSTEM

Entries on the Input Specifications form describe the data records and the fields to be read into the system.

Problem

A detail labor file contains card records, as shown in Figure 2. Three fields (A, B, and C) are to be read from each card into main storage. Field A is contained in positions 46-50, Field B is contained in positions 56-60, and Field C is contained in positions 66-70 of each record. The file that contains the records and the fields within each record are described on the Input Specifications form.

Each card that contains fields A, B, and C is identified by two distinct attributes: they must contain a digit 5 in column 35 and no 11-zone in column 80.

Specifications

Figure 3 shows the input specifications required. The numbers in the following list refer to items circled in Figure 3.

1. Each file of cards must be given a name. In this example the input deck of cards represents a detail labor file. A filename can be used to describe only one file. This filename is the same as the name on the DD card that describes the file.

2. If several types of cards exist in a file, each card must be identified by its particular card code. In this example, each card read by the system must contain a 5-punch in column 35 and no 11-zone (minus) punch in column 80. These identifying codes are placed in the fields called Record Identification Codes (positions 21-41 of the form). These fields provide for three identifying codes; however, more codes can be indicated, as illustrated later in the manual.

   In this example the card codes are specified by writing a 5 in the Character entry and a 35 in the Position entry for the first code and by writing an 80 in the Position entry, a minus in the Character entry, and an N in the Not entry for the second code. The field on the form called C/Z/D indicates how the card column containing the card code is to be compared: D = digit portion only, Z = zone portion only, or C = all portions of the card column.

   After all the identification codes are established, the programmer assigns a two-digit number (from 01 to 99) to the card type identified. This code, known as a Record Identifying Indicator (positions 19-20 of the form), is used to refer to this specific card type on other specifications forms. This code reduces the number of entries required on other specifications forms.

3. If the input card is to be selected into a stacker (other than the one into which it would normally be selected), the stacker number is written in Stacker Select (position 42).

4. Each field of the card to be read must be defined as shown on the lines following the record identification line. The specifications shown in this example locate the data from the input record and place it in fields A, B, and C (FLDA, FLDB, and FLDC) in three separate locations in core storage.

Once the card columns of the field have been specified and the appropriate field name has been defined for the field, other references to this field are made by using its field name rather than writing down the specific card columns each time.

Although the input illustrated in Figure 3 is simplified, the entries shown cause the contents of fields A, B, andC from the card to be read into the system during the processing of the object module. Calculations to be made from the input data are written on the Calculation Specifications form.

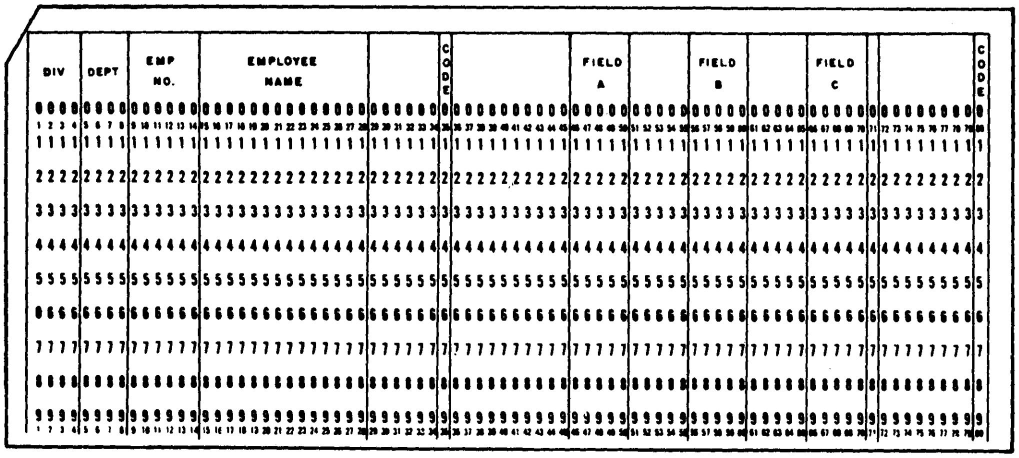
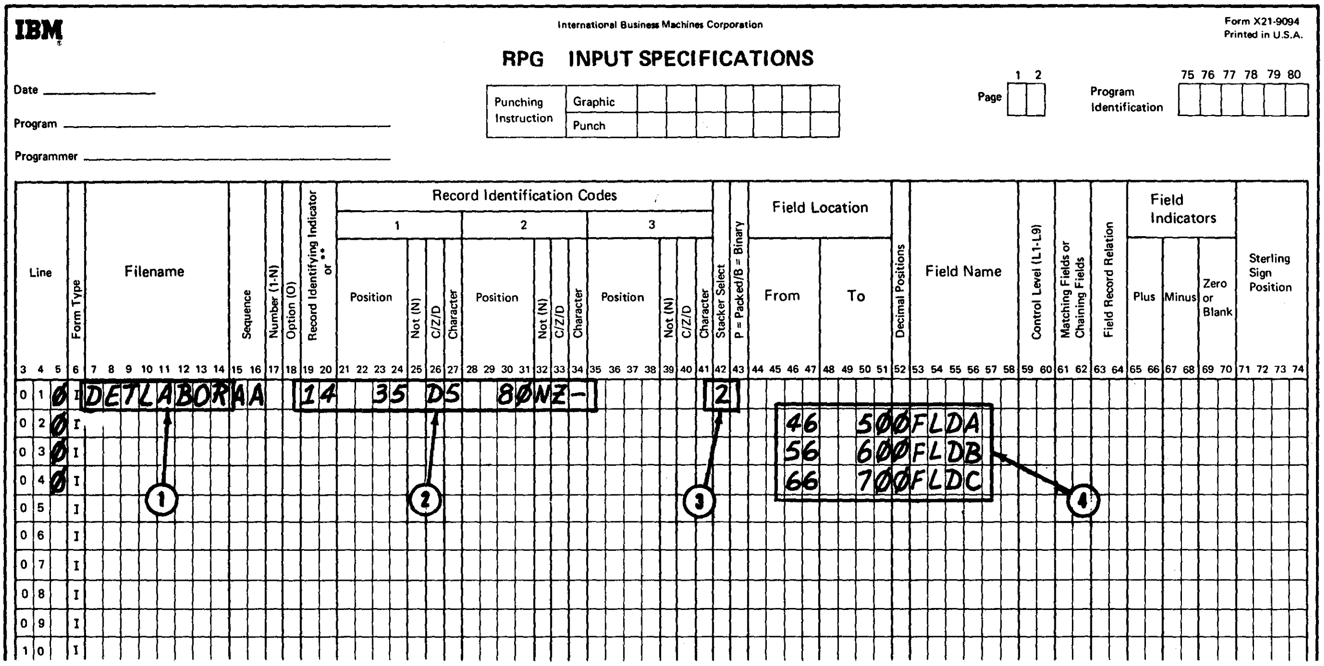


Figure 2. Input Card for Detail Labor File

53683



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

Program Identification 75 76 77 78 79 80

Line (3-5)	Form Type (6)	Filename (7-14)	Sequence (15-16)	Number (1-N) (17)	Option (O) (18)	Record Identifying Indicator or ** (19-20)	Record Identification Codes 1: Position (21-24)	Not (N) (25)	C/Z/D (26)	Character (27)	2: Position (28-31)	Not (N) (32)	C/Z/D (33)	Character (34)	3: Position (35-38)	Not (N) (39)	C/Z/D (40)	Character (41)	Stacker Select (42)	P = Packed/B = Binary (43)	Field Location From (44-47)	To (48-51)	Decimal Positions (52)	Field Name (53-58)	Control Level (L1-L9) (59-60)	Matching Fields or Chaining Fields (61-62)	Field Record Relation (63-64)	Field Indicators Plus (65-66)	Minus (67-68)	Zero or Blank (69-70)	Sterling Sign Position (71-74)
01Ø	I	DETLABOR	AA			14	35		D	5	80	N	Z	-					2												
02Ø	I																				46	50	Ø	FLDA							
03Ø	I																				56	60	Ø	FLDB							
04Ø	I																				66	70	Ø	FLDC							
05	I																														
06	I																														
07	I																														
08	I																														
09	I																														
10	I																														

Figure 3. Input Specifications Form

ADDITION AND SUBTRACTION

Problem

In this problem, fields A, B, and C from the previous example are used to calculate a new field (field D). The calculation A + B - C = D is to be performed.

Specifications

Figure 4 shows the required entries, using the fields from the previous example. The form required for this problem is the Calculation Specifications form.

The first line of the form tells the object program to add the contents of FLDB to the contents of FLDA and place the result in FLDD.

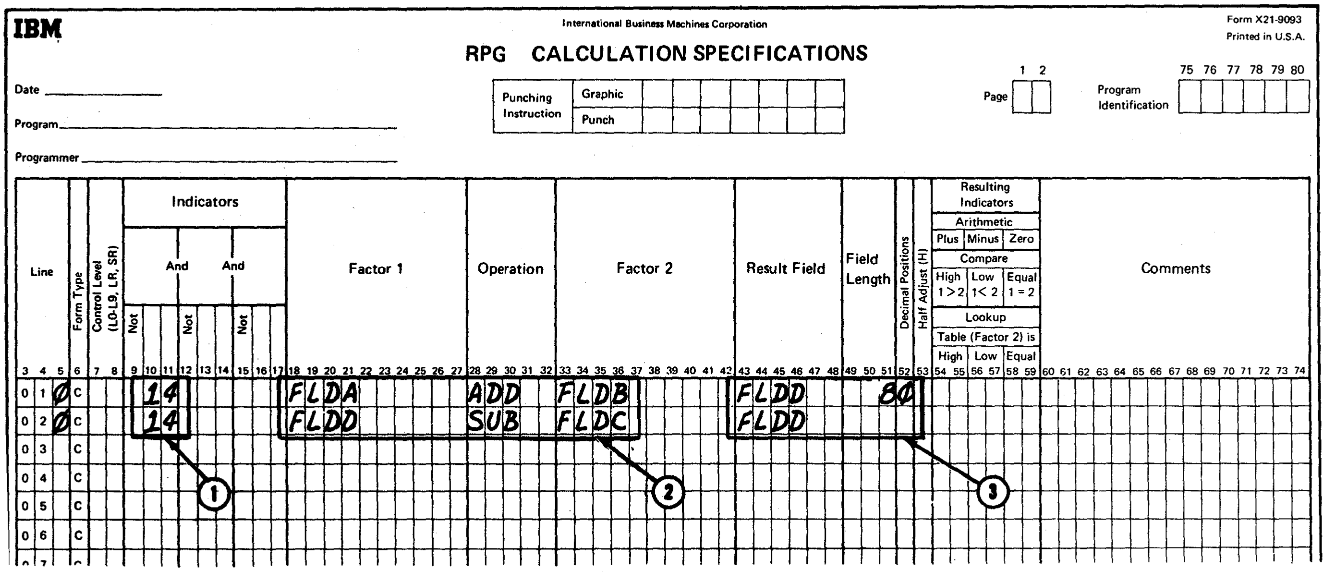
1. The 14 in Indicators defines the card type for which the calculations are to be performed as assigned by the Input Specifications form.

2. The card columns for FLDA and FLDB were defined by entries on the input forms and need not be specified again.

3. The result field FLDD is defined for the program by merely writing the name FLDD in Result Field (positions 43-48) and indicating in Field Length (positions 49-51) the number of positions that must be set aside for this field in the object program. 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 used in defining output specifications.

   Note: The result of A + B is not placed back into FLDA. This would destroy the original contents of FLDA. In this example it is necessary to save the original value of FLDA so that it can be printed on an output report.

The second line in Figure 4 causes the object module to subtract the contents of FLDC from the result just previously obtained in FLDD and to store the new result back into FLDD.



IBM

International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

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

Date ____________

Program ____________

Programmer ____________

Punching Instruction	Graphic								
	Punch								

Page (1 2) □□ Program Identification (75 76 77 78 79 80) □□□□□□

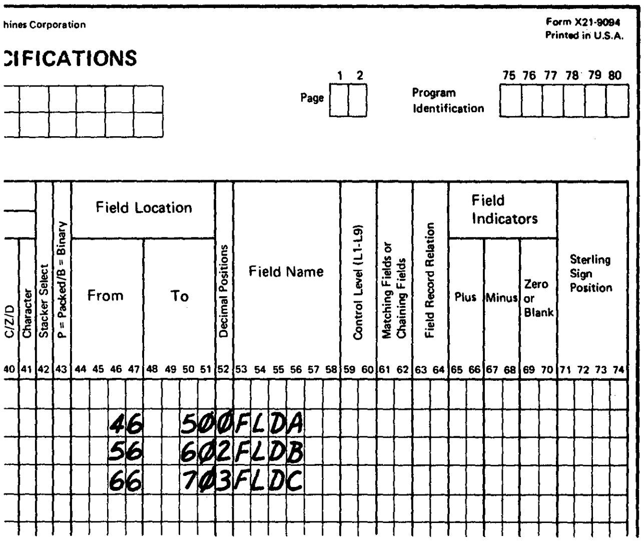
Line (3-5)	Form Type (6)	Control Level (L0-L9, LR, SR) (7-8)	Indicators (9-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field (43-48)	Field Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators (54-59)	Comments (60-74)
0 1 Ø	C		14	FLDA	ADD	FLDB	FLDD	8	Ø			
0 2 Ø	C		14	FLDD	SUB	FLDC	FLDD					
0 3	C											
0 4	C											
0 5	C											
0 6	C											

Figure 4. Calculation Specifications Form

Decimal Symbol Location

In Figure 4 there were no decimal positions in any of the three fields. In operations involving numbers containing decimal positions, the number of 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 in an arithmetic operation, shifting the factors to align the decimal point is usually required.

RPG automatically shifts the factors. The programmer must indicate the number of decimal positions contained in each factor used in calculations and the number of decimal positions required in the result. If a field is to have arithmetic operations performed upon it, the decimal positions must be specified. A zero is coded for no decimal positions (Figure 4). Assume that the fields in Figure 3 have decimal positions as indicated under Decimal Positions (position 52) in the following example:



hines Corporation

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

CIFICATIONS

Page 1 2

Program Identification 75 76 77 78 79 80

C/Z/D	Character	Stacker Select	P = Packed/B = Binary	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Field Indicators Minus	Field Indicators Zero or Blank	Sterling Sign Position
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
				46	5Ø	Ø	FLDA							
				56	6Ø	2	FLDB							
				66	7Ø	3	FLDC							

A set of values for these fields might appear to the program as:

Field	*Decimal Position*	*Contained in Cards*	*Actual Value*
A	0	00126	126.
B	2	01123	11.23
C	3	04264	4.264

The programmer must indicate the number of decimal positions required in the result.

The calculation A + B - C = D results in the values being shifted like this:

```
 126.000  =  A
+11.230   =  B
 137.230
- 4.264   =  C
 132.966  =  D
```

(Result Field has three decimal positions.)

The result 132.966 is stored as the value of the field called D. Entering the decimal positions on the Input Specifications form and the Calculations Specifications form enables the factors to be shifted automatically.

DETAIL PRINTING

Detail printing is the printing of information obtained from each record as it is read.

Problem

This problem illustrates how input fields A, B, and C and the calculated result, field D, can be specified for listing.

Specifications

Figure 5 shows the output specifications required. The numbers in the following list refer to items circled in Figure 5.

1. The output listing must be assigned its own specific filename. This filename is the same as the name on the DD card that describes the file. In this example it has been called DAILYRPT.

   The D in Type H/D/T/E indicates that the line being printed is a detail line. That is, it contains information from the record just read. (The other valid entries are described later.)

   The 2 in Space After provides a double-spaced listing. (There is one blank line after each printed line.)

2. The 14 in Output Indicators identifies the input record type present when detail printing occurs.

3. Each field to be printed must be specified under the Field Name entry. The Z in *Edit Codes* means that zeros to the left of significant digits are not printed. For example, the value stored in FLDA (00126) is printed as 126.

   The printing positions for data to be printed on the output report are specified in *End Position in Output Record* (positions 40-43). The programmer has to indicate only the last printing position of each field (low-order or rightmost position). The length of each field has already been established for the program as part of the input specification or established in the Result Field of the calculations specifications.

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

More elaborate reports are laid out on a printer form in advance of the program writing function. 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 names of data fields developed in the program). This is normally a function of job definition, a subject that is discussed later under *Program Logic, Problem Definition.*

Figure 3, 4, and 5 illustrate 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 necessary to prepare a report. A significant item not yet described is how to specify a control field.

CONTROL FIELDS

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

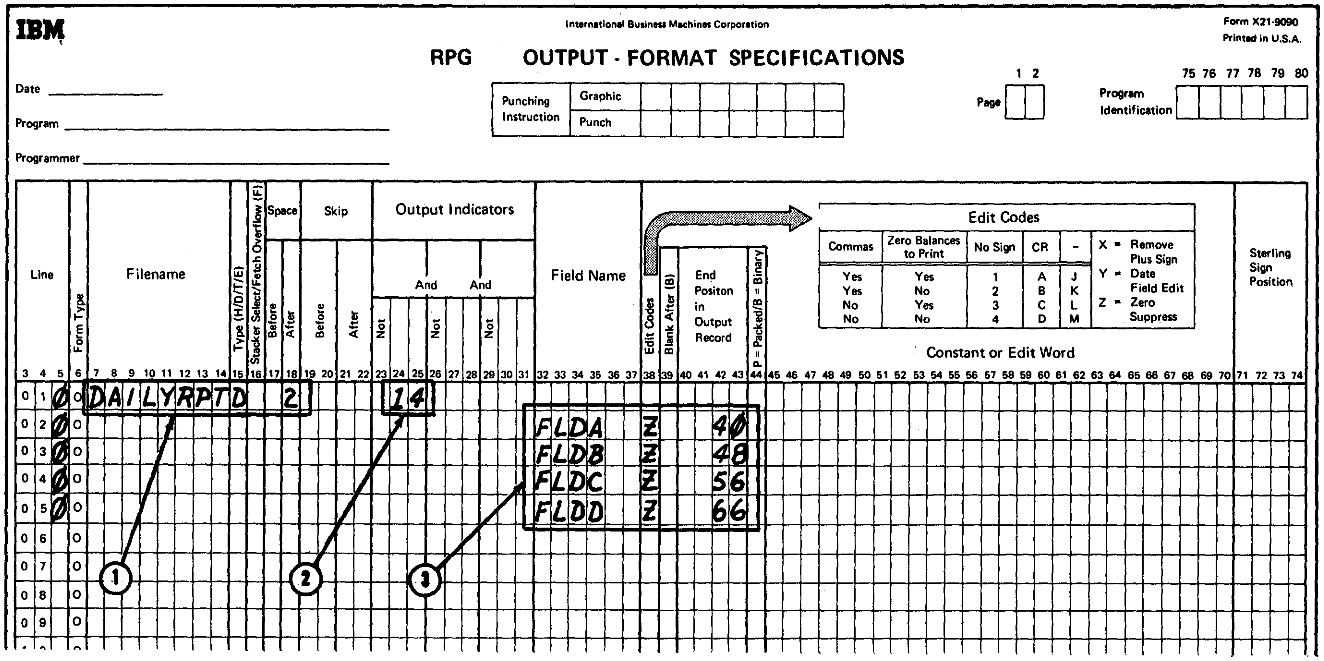
Problem

This example shows how to specify three levels of control.

Specifications

Figure 6 shows the input data from Figure 3 with the addition of the appropriate control data and employee name (the items inside the circle). In this example there are three levels of totals: the lowest level is employee number (EMPNO), the next is department (DEPT), and the highest level is division (DIVSON). These levels are designated L1, L2, and L3, respectively. (Because as many as nine levels of totals are possible in RPG, the common terms of minor, intermediate, and major totals are inadequate.)

To designate a control field and to establish a level of control, enter the card columns of the field in Field Location, enter an appropriate name in Field Name, and enter the correct control level (L1, L2, ... L9) in Control Level.



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 1 2

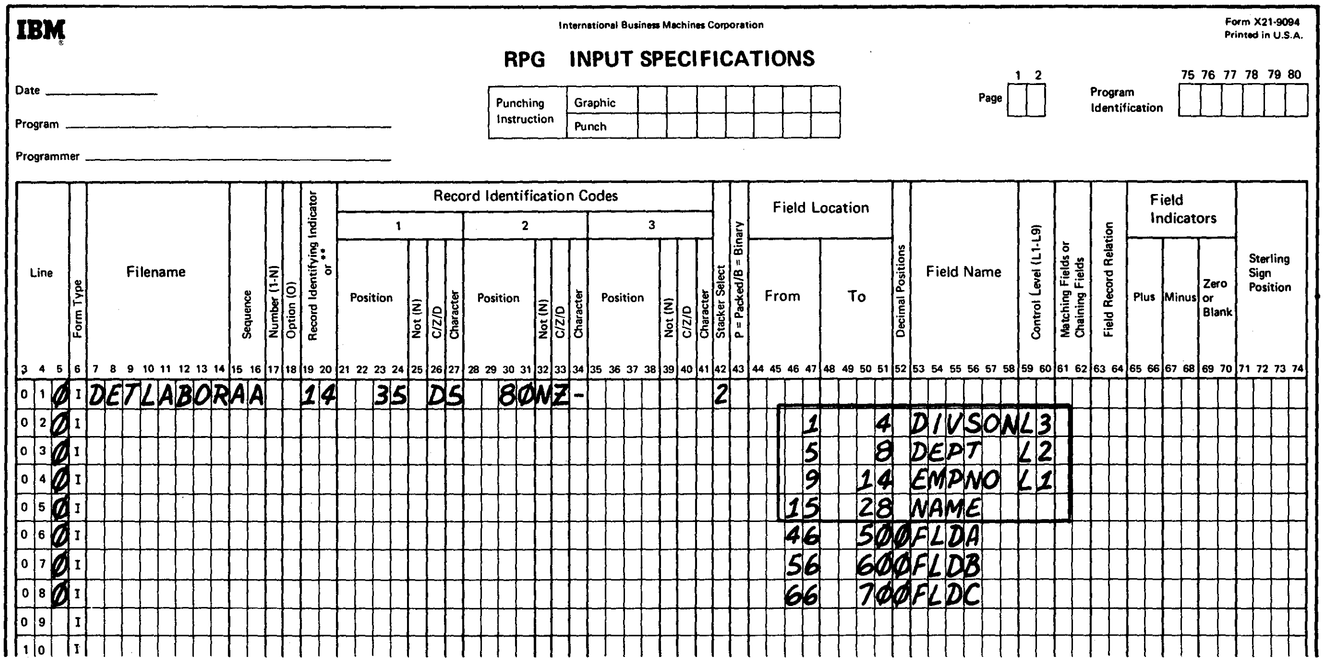
Program Identification 75 76 77 78 79 80

Edit Codes

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

Line	Form Type	Filename	Type (H/D/T/E)	Space After	Output Indicators	Field Name	Edit Codes	End Position in Output Record
0 1 Ø	O	DAILYRPT	D	2	14			
0 2 Ø	O					FLDA	Z	4Ø
0 3 Ø	O					FLDB	Z	48
0 4 Ø	O					FLDC	Z	56
0 5 Ø	O					FLDD	Z	66
0 6	O							
0 7	O							
0 8	O							
0 9	O							

Figure 5. Output-Format Specifications Form



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

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Record Identifying Indicator	1 Position	1 Not (N)	1 C/Z/D	1 Character	2 Position	2 Not (N)	2 C/Z/D	2 Character	Stacker Select	From	To	Decimal Positions	Field Name	Control Level (L1-L9)
0 1 Ø	I	DETLABOR	AA	14	35		D	5	8Ø	N	Z	-	2					
0 2 Ø	I													1	4		DIVSON	L3
0 3 Ø	I													5	8		DEPT	L2
0 4 Ø	I													9	14		EMPNO	L1
0 5 Ø	I													15	28		NAME	
0 6 Ø	I													46	5Ø	Ø	FLDA	
0 7 Ø	I													56	6Ø	Ø	FLDB	
0 8 Ø	I													66	7Ø	Ø	FLDC	
0 9	I																	
1 0	I																	

Figure 6. Specifying Control Levels on the Input Form

In the example (Figure 2 and 6) the sequence of the control fields (left to right) in the card is the same as the sequence of the control levels. The control fields, however, could have been in any location on the card, and the specifications for them could have been in any sequence on the form (Figure 7).

The reading of the employee-name field (fifth line of the form in Figure 6) is also included as part of this example, but it is not related to controls. It is shown here to indicate another aspect of the decimal-positions position of the Input Specifications form. When a numeric field is specified in Field Location, any digit (0 to 9) placed in Decimal Positions causes zone punches in all positions of the field (except the units position) to be removed.

Therefore, when an alphabetic field is specified, Decimal Positions should be left blank to permit the zone punches of the alphabetic field to be read into the system.

TOTAL CALCULATIONS

Total calculations are performed after a specified control break has occurred. When a control break occurs, special operations are normally performed before processing the record that caused the control break. These operations are called total operations.

Problem

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

Specifications

Figure 8 contains the data shown in Figure 4. The additional information for this example is circled. During processing of the object module, lines 1 and 2 cause the operation A + B - C = D. The third line adds the result, contained in FLDD, from each detail card to a field called TOTE. Therefore, TOTE is the accumulated amount for each employee group. The first three specification lines are performed in the object module each time a detail card is read (see Figure 8, Indicator 14).

When the level-1 (L1) control break occurs (a card from the next employee group has been sensed), specification line 4 adds the total of TOTE accumulated from each detail card into a field called TOTF. TOTF is used to accumulate the total for each employee group for each department.

When the level-2 control break occurs, TOTF is added into TOTG.

When the level-3 control break occurs, TOTG (which represents the accumulated amounts for each division) is added into FINTOT.

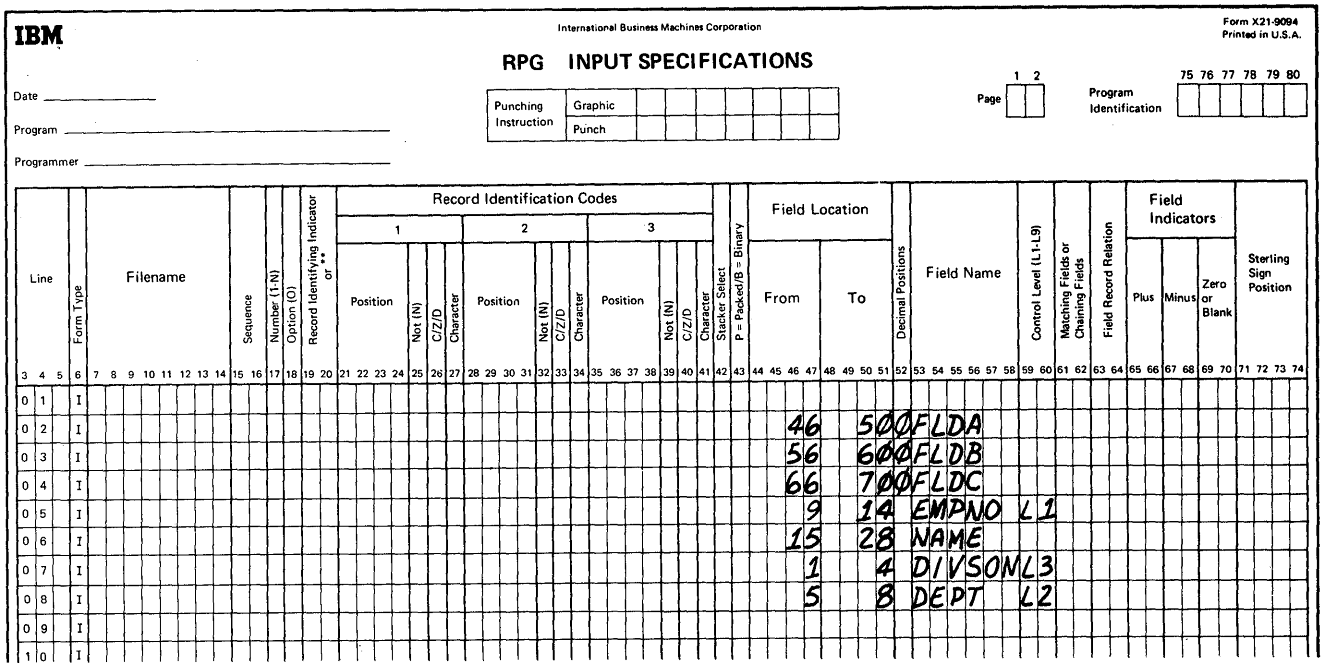
Accumulating totals at each control break level is normally done when the corresponding totals are printed on an output report.

Note: In System/360 RPG, as in IBM punched card equipment, a control break at one level forces control breaks for all lower levels.

DETAIL AND TOTAL PRINTING

Problem

This example shows the specifications necessary to print the three controlling fields, the name, the accumulated amount in field D, the three accumulated totals, and the final total at the end of the object module report.

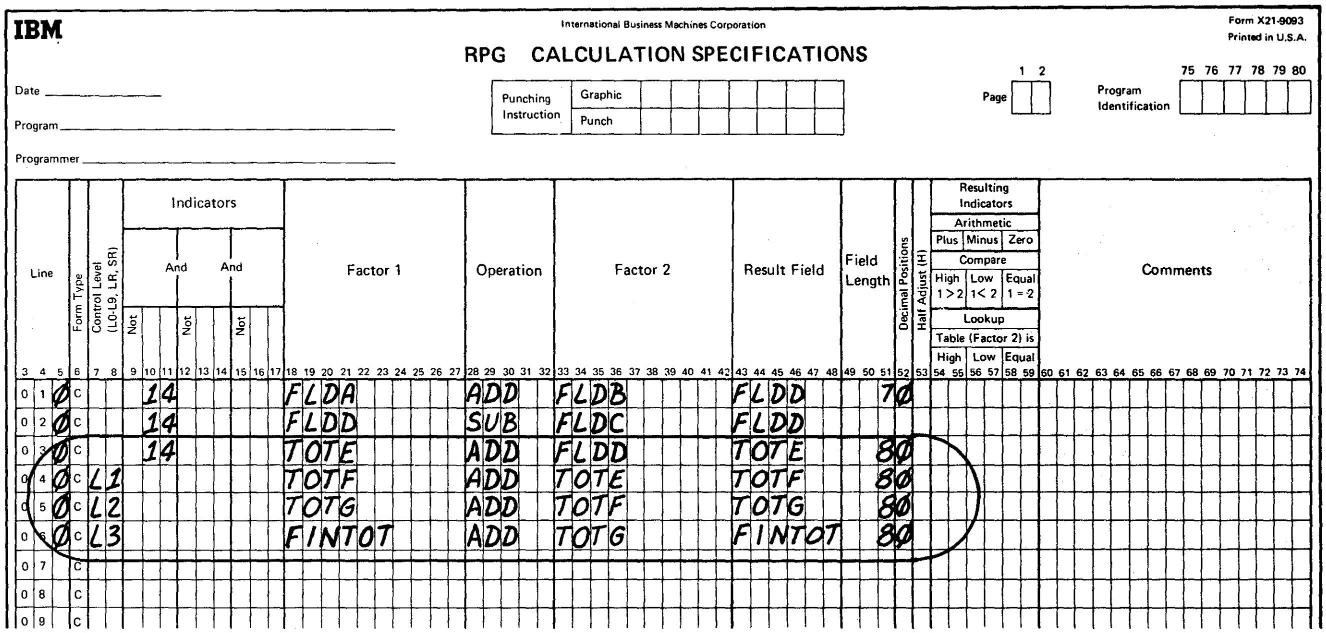


IBM International Business Machines Corporation Form X21-9094 Printed in U.S.A.

RPG INPUT SPECIFICATIONS

Line	Form Type	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)
01	I					
02	I	46	50	Ø	FLDA	
03	I	56	60	Ø	FLDB	
04	I	66	70	Ø	FLDC	
05	I	9	14		EMPNO	L1
06	I	15	28		NAME	
07	I	1	4		DIVSON	L3
08	I	5	8		DEPT	L2
09	I					
10	I					

Figure 7. Example of Control Field Sequence



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

RPG CALCULATION SPECIFICATIONS

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions
01Ø	C		14	FLDA	ADD	FLDB	FLDD	7	Ø
02Ø	C		14	FLDD	SUB	FLDC	FLDD		
03Ø	C		14	TOTE	ADD	FLDD	TOTE	8	Ø
04Ø	C	L1		TOTF	ADD	TOTE	TOTF	8	Ø
05Ø	C	L2		TOTG	ADD	TOTF	TOTG	8	Ø
06Ø	C	L3		FINTOT	ADD	TOTG	FINTOT	8	Ø
07	C								
08	C								
09	C								

Figure 8. Specifying Control Levels on the Calculation Form

Specifications

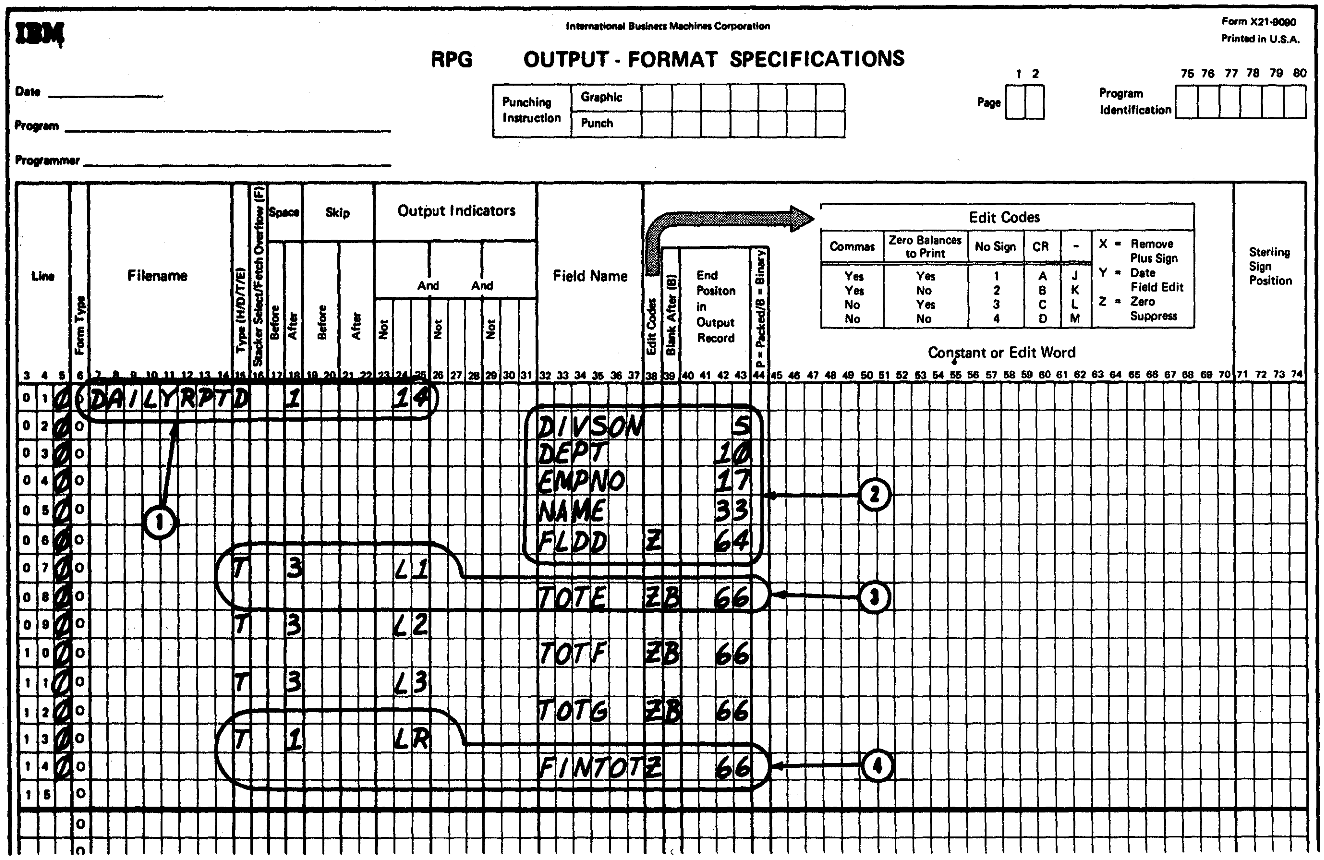
Figure 9 illustrates the specifications required for the report shown in Figure 10. The following numbers refer to the items circled in Figure 9:

1. This information is similar to the specifications in the previous example in Figure 5.

2. The data fields 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 7 and 8 are:

    a. The T in H/D/T/E (position 15) indicates that the line is a total line. A total line is an operation caused by a control break. The input record that causes the control break cannot contribute data to the accumulated totals or to the total line. Totals accumulated before a control break are printed.



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 (1 2) □□

Program Identification (75 76 77 78 79 80) □□□□□□

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 (3-5)	Form Type (6)	Filename (7-14)	Type (H/D/T/E) (15)	Stacker Select/Fetch Overflow (F) (16)	Space Before (17)	Space After (18)	Skip Before (19-20)	Skip After (21-22)	Output Indicators (23-31)	Field Name (32-37)	Edit Codes (38)	Blank After (B) (39)	End Position in Output Record (40-43)	P = Packed/B = Binary (44)	Constant or Edit Word (45-70)	Sterling Sign Position (71-74)
01Ø	O	DAILYRPT	D			1			14							
02Ø	O									DIVSON			5			
03Ø	O									DEPT			1Ø			
04Ø	O									EMPNO			17			
05Ø	O									NAME			33			
06Ø	O									FLDD	Z		64			
07Ø	O		T			3			L1							
08Ø	O									TOTE	Z	B	66			
09Ø	O		T			3			L2							
10Ø	O									TOTF	Z	B	66			
11Ø	O		T			3			L3							
12Ø	O									TOTG	Z	B	66			
13Ø	O		T			1			LR							
14Ø	O									FINTOT	Z		66			
15	O															
	O															
	O															

Figure 9. Specifications on the Output-Format Form

b. A 3 in Space After (position 18) provides two blank lines after each printed line to make the report easier to read.

c. The L1 in Output Indicators (positions 24-25) indicates that the line is to be printed only on a level-1 control break.

d. The field to be printed (TOTE) is indicated under Field Name (positions 32-37).

e. The Z in Edit Codes (position 38) indicates that zeros to the left of significant digits are not to be printed.

f. The B in Blank After (position 39) causes the core storage positions containing field TOTE to be set to zeros after the total is printed. This is done so the total for one group is not added to the total of the previous group.

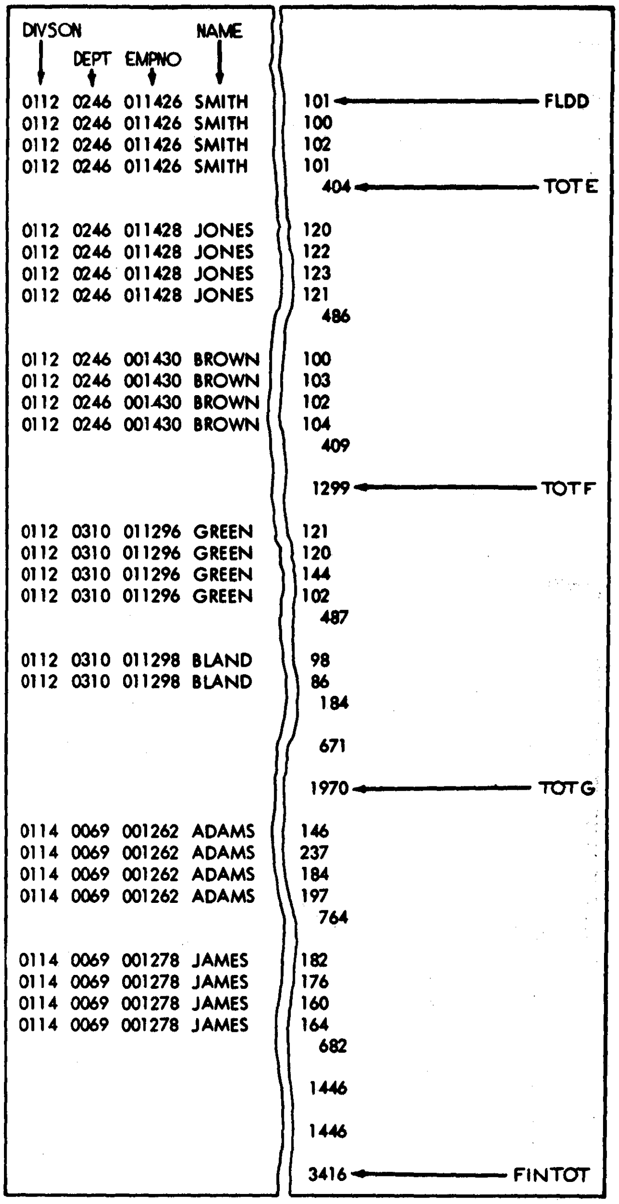
The specifications to print the second and third control levels are essentially the same as those for the first level.

4. The specifications for printing the final total contain the code LR (Last Record) in Output Indicators (positions 24-25). The indicator LR is set on automatically by the program when the last card of a file has been sensed. This indicator is used to cause the final total to be printed.

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.

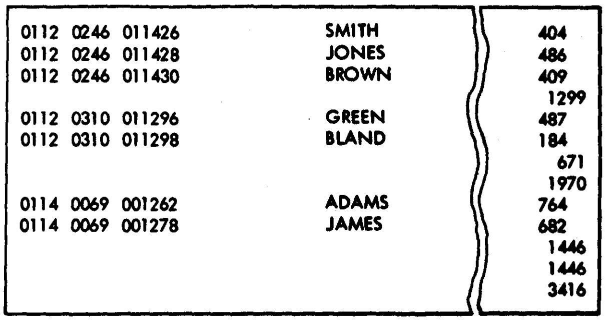
An example of a group printed report is shown in Figure 11.



```
DIVSON              NAME
     DEPT  EMPNO
0112 0246 011426 SMITH     101          FLDD
0112 0246 011426 SMITH     100
0112 0246 011426 SMITH     102
0112 0246 011426 SMITH     101
                             404        TOTE

0112 0246 011428 JONES     120
0112 0246 011428 JONES     122
0112 0246 011428 JONES     123
0112 0246 011428 JONES     121
                             486

0112 0246 001430 BROWN     100
0112 0246 001430 BROWN     103
0112 0246 001430 BROWN     102
0112 0246 001430 BROWN     104
                             409

                            1299        TOTF

0112 0310 011296 GREEN     121
0112 0310 011296 GREEN     120
0112 0310 011296 GREEN     144
0112 0310 011296 GREEN     102
                             487

0112 0310 011298 BLAND      98
0112 0310 011298 BLAND      86
                             184

                             671

                            1970        TOTG

0114 0069 001262 ADAMS     146
0114 0069 001262 ADAMS     237
0114 0069 001262 ADAMS     184
0114 0069 001262 ADAMS     197
                             764

0114 0069 001278 JAMES     182
0114 0069 001278 JAMES     176
0114 0069 001278 JAMES     160
0114 0069 001278 JAMES     164
                             682

                            1446

                            1446

                            3416        FINTOT
```

Figure 10. Detail-Printed Report

53684



```
0112 0246 011426        SMITH      404
0112 0246 011428        JONES      486
0112 0246 011430        BROWN      409
                                     1299
0112 0310 011296        GREEN      487
0112 0310 011298        BLAND      184
                                      671
                                     1970
0114 0069 001262        ADAMS      764
0114 0069 001278        JAMES      682
                                     1446
                                     1446
                                     3416
```

Figure 11. Group-Printed Report

53685

The detail printed report specifications in Figure 9 could be altered to provide a group printed report as illustrated by the specifications in Figure 12.

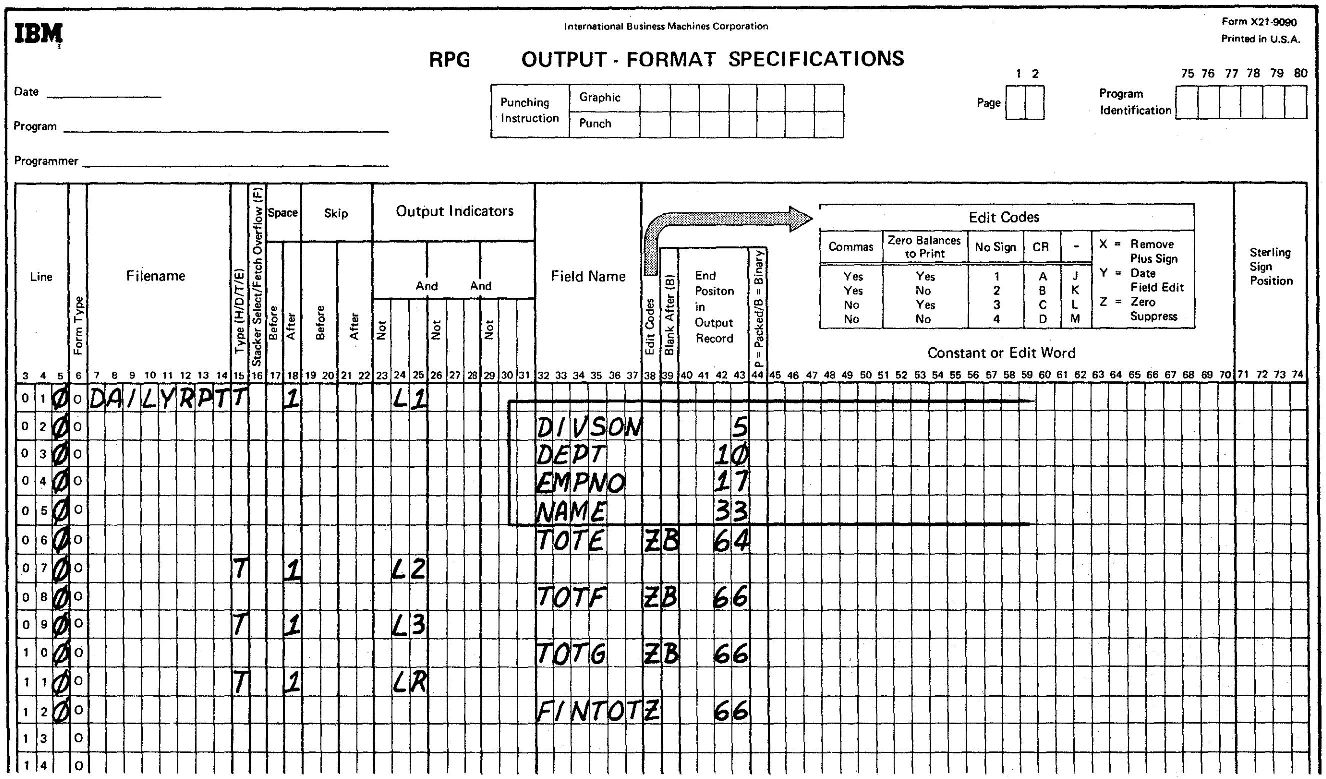
The differences between Figures 9 and 12 that provide for the two types of reports are:

1. The detail line specified in Figure 9 (line 01) has been changed to a total line conditioned on an L1 break (Figure 12, line 01).
2. The total line in Figure 9 (line 07) has been combined with the first total line 01 of Figure 12.
3. The spacing on lines 09 and 11 of Figure 9 have been changed from 3 to 1 space-after in Figure 12.

GROUP INDICATION

In group indication operations, each detail card is processed; however, only the control fields that identify the specific detail card are printed. An example of a group indication report is illustrated in Figure 13. In this example the employee name and number are printed for each control change. The division and department fields are printed only when there is a control change for the division and department fields, respectively.

Figure 14 illustrates how the detail printed report specifications in Figure 9 could be altered to specify a group indicated report.



IBM

International Business Machines Corporation

Form X21-9090
Printed in U.S.A.

RPG OUTPUT - FORMAT SPECIFICATIONS

Date ____________

Program ____________

Programmer ____________

	Graphic								
Punching Instruction	Punch								

Page 1 2 □□

Program Identification 75 76 77 78 79 80 □□□□□□

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

Line (3-5)	Form Type (6)	Filename (7-14)	Type (H/D/T/E) (15)	Stacker Select/Fetch Overflow (F) (16)	Space Before (17)	Space After (18)	Skip Before (19-20)	Skip After (21-22)	Output Indicators Not (23)	(24-25)	And Not (26)	(27-28)	And Not (29)	(30-31)	Field Name (32-37)	Edit Codes (38)	Blank After (B) (39)	End Positon in Output Record (40-43)	P = Packed/B = Binary (44)	Constant or Edit Word (45-70)	Sterling Sign Position (71-74)
01Ø	O	DAILYRPT	T			1				L1											
02Ø	O														DIVSON			5			
03Ø	O														DEPT			1Ø			
04Ø	O														EMPNO			17			
05Ø	O														NAME			33			
06Ø	O														TOTE	Z	B	64			
07Ø	O		T			1				L2											
08Ø	O														TOTF	Z	B	66			
09Ø	O		T			1				L3											
10Ø	O														TOTG	Z	B	66			
11Ø	O		T			1				LR											
12Ø	O														FINTOT	Z		66			
13	O																				
14	O																				

Figure 12. Specifying a Group-Printed Report

OVERFLOW PRINTING

Overflow printing, another function used in preparing reports, can be performed in RPG. Overflow is the sensing of channel 12 in the printer carriage control tape.

Problem

This example shows the additional specifications necessary to print eight column headings at the top of each printed form. A heading line is a line that contains constants or information from an input record or a constant defined on the Output-Format Specifications form.

Specifications

Figure 15 illustrates the output specifications required for this operation. The numbers in the following list refer to the circled items in Figure 15. The Filename (positions 7-14) of DAILYRPT is the same; adding overflow headings does not change it.

1. On the first line of the form, the H in H/D/T/E (position 15) indicates that the line to be printed is the heading line.

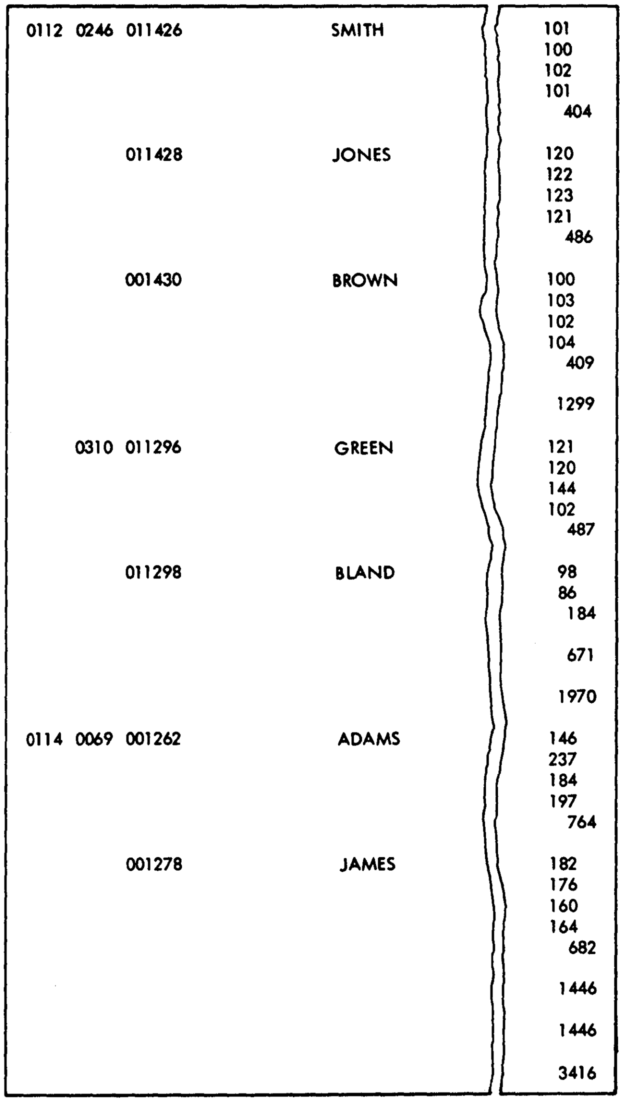
   The 2 in Space After (position 18) provides a blank line after each printed line.

   The 01 in Skip Before (positions 19-20) causes the form to skip to channel 01 in the carriage control tape. This positions the form for printing of the overflow heading line.

   The OF in Output Indicators (positions 24-25) causes the heading line to be printed each time there is a form overflow on the printer.

   On the second line of the form, the letters OR indicate a second condition can cause the heading line to print. The second condition, indicated by 1P (first page) in Output Indicators, causes the heading line to be printed on the first page. This condition is necessary to print the heading on the first page of the report because the overflow condition does not occur until after the first page of the report has been printed.

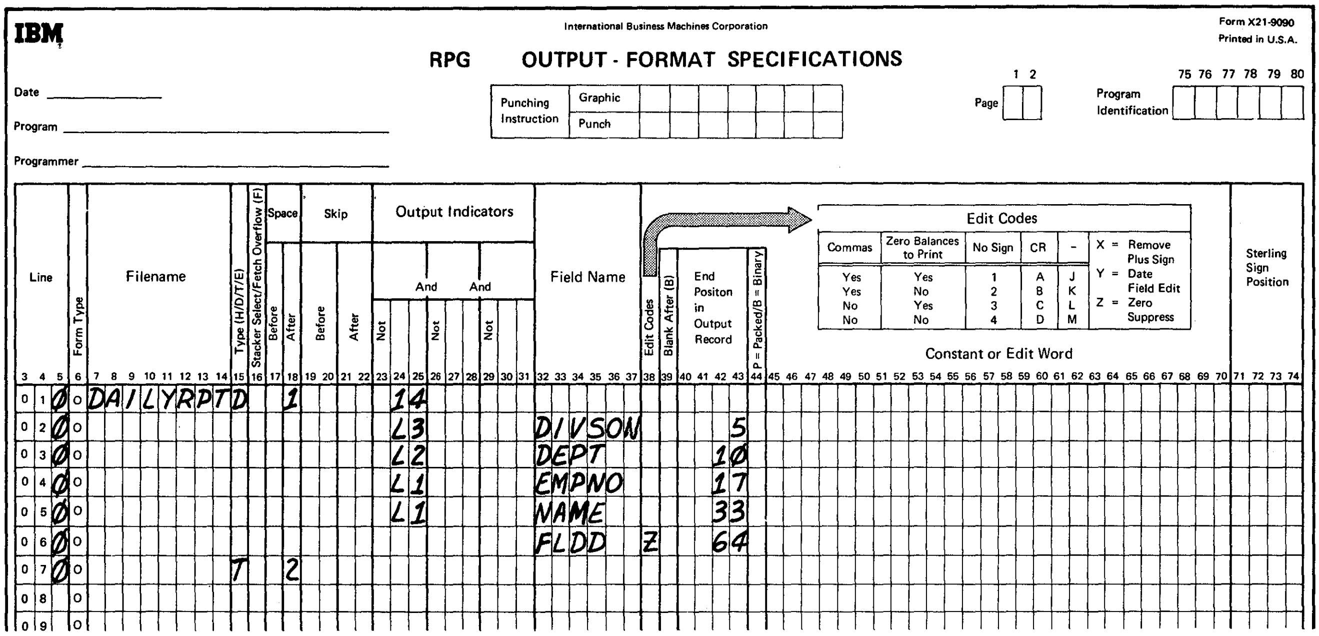
2. The entries on lines 3-10 of Figure 15 specify the actual information or constants to be printed on the report. The actual information is contained within apostrophes.



```
0112 0246 011426          SMITH        101
                                       100
                                       102
                                       101
                                         404

          011428          JONES        120
                                       122
                                       123
                                       121
                                         486

          001430          BROWN        100
                                       103
                                       102
                                       104
                                         409

                                        1299

     0310 011296          GREEN        121
                                       120
                                       144
                                       102
                                         487

          011298          BLAND         98
                                        86
                                         184

                                         671

                                        1970

0114 0069 001262          ADAMS        146
                                       237
                                       184
                                       197
                                         764

          001278          JAMES        182
                                       176
                                       160
                                       164
                                         682

                                        1446

                                        1446

                                        3416
```

53686

Figure 13. Example of a Group-Indicated Report



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 1 2

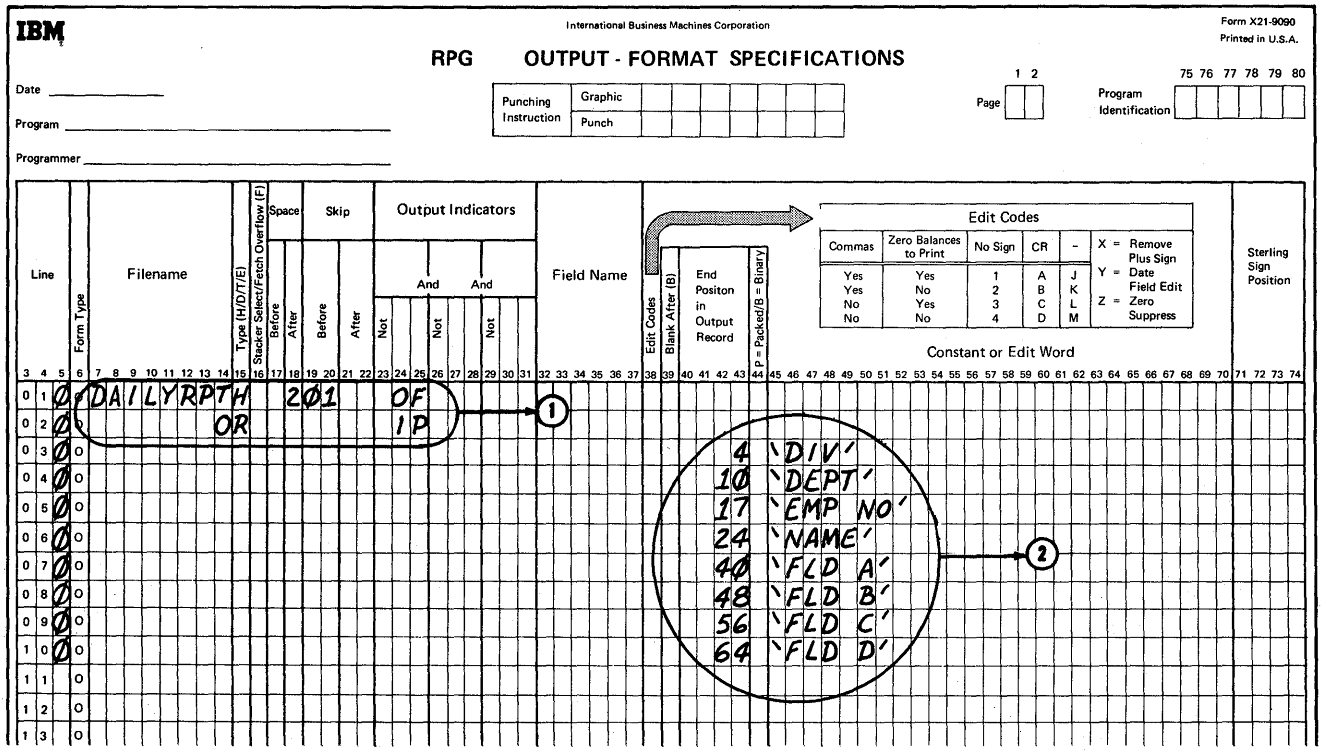
Program Identification 75 76 77 78 79 80

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/Fetch Overflow (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators	Field Name	Edit Codes	Blank After (B)	End Positon in Output Record	P = Packed/B = Binary	Constant or Edit Word	Sterling Sign Position
01Ø	O	DAILYRPT	D			1			14							
02Ø	O								L3	DIVSON			5			
03Ø	O								L2	DEPT			1Ø			
04Ø	O								L1	EMPNO			17			
05Ø	O								L1	NAME			33			
06Ø	O									FLDD	Z		64			
07Ø	O		T			2										
08	O															
09	O															

Figure 14. Specifying a Group-Indicated Report



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 1 2

Program Identification 75 76 77 78 79 80

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/Fetch Overflow (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators	Field Name	Edit Codes	Blank After (B)	End Positon in Output Record	P = Packed/B = Binary	Constant or Edit Word	Sterling Sign Position
01Ø	O	DAILYRPT	H			2	Ø1		OF							
02Ø	O	OR							1P							
03Ø	O												4		'DIV'	
04Ø	O												1Ø		'DEPT'	
05Ø	O												17		'EMP NO'	
06Ø	O												24		'NAME'	
07Ø	O												4Ø		'FLD A'	
08Ø	O												48		'FLD B'	
09Ø	O												56		'FLD C'	
10Ø	O												64		'FLD D'	
11	O															
12	O															
13	O															

Figure 15. Printing Heading Lines

SUMMARY PUNCHING

Problem

Using an IBM 1442, punch a summary total of FLDD for each department together with the appropriate department and division numbers.

Specifications

A blank card must be merged behind each department group before the processing of the object module is executed. Therefore, the input file is a combined file, that is, a file containing cards read into the system and cards used for punching. The blank card is specified on the Input Specifications form as illustrated in Figure 16, line 09.

All other records in the file must contain punches in the card column used in the record identification specification for the blank card.

Figure 17 illustrates the calculation specifications for the summary punching operation.

Line 04 is processed when the blank card is read. It causes accumulation of the level-2 total for the summary card. This specification is necessary when the summary card is merged behind control groups rather than punched from a second file of blank cards. This specification is required because the level-2 control break does not occur until the first card of the next control group is read; this does not occur until after the blank card is read.

Note: The control level L0 has been entered to identify a total calculation.

The summary punching of accumulated value is illustrated on the Output-Format Specification form in Figure 18. The specifications required for this function are circled.

The cards to be punched will become a new output file which is given the name of the combined file DETLABOR. This name is specified in Filename for specification line 12.

The T in H/D/T/E indicates the operation is to be performed at total time.

The 2 in Stacker Select indicates that the summary cards are to be selected into stacker number 2.

Resulting Indicator 16 in Output Indicators indicates that the operation is to occur at the time the blank card is read in.

The first field specified for punching is position 35 which will be punched with an 8 to identify the card as a summary card. The remaining three fields are punched by specifying the name of the field in Field Name and then by specifying the last position to be punched in End Position In Output Record. The format of the summary card is shown in Figure 19.

TESTING FOR ZERO, PLUS, AND MINUS BALANCE

Specifications are executed in the object program in the same sequence in which they are written on the specification form unless a different sequence has been specified.

If specifications had to be followed sequentially in a fixed pattern, a program would follow a single path of operation. The program would not have the ability to choose a predefined alternative to the procedure based upon conditions encountered during processing.

For example, assume that a program has been written containing ten specifications that cause a number of operations to be performed upon a series of quantities. If the first five specifications develop meaningless results when processed with quantities of zero, the processing time of the object program can be reduced if the first five specifications are bypassed whenever the quantity to be processed is zero.

A specification in the RPG program can be used to evaluate a quantity and, depending upon the value of that quantity, direct the program to some other specification.

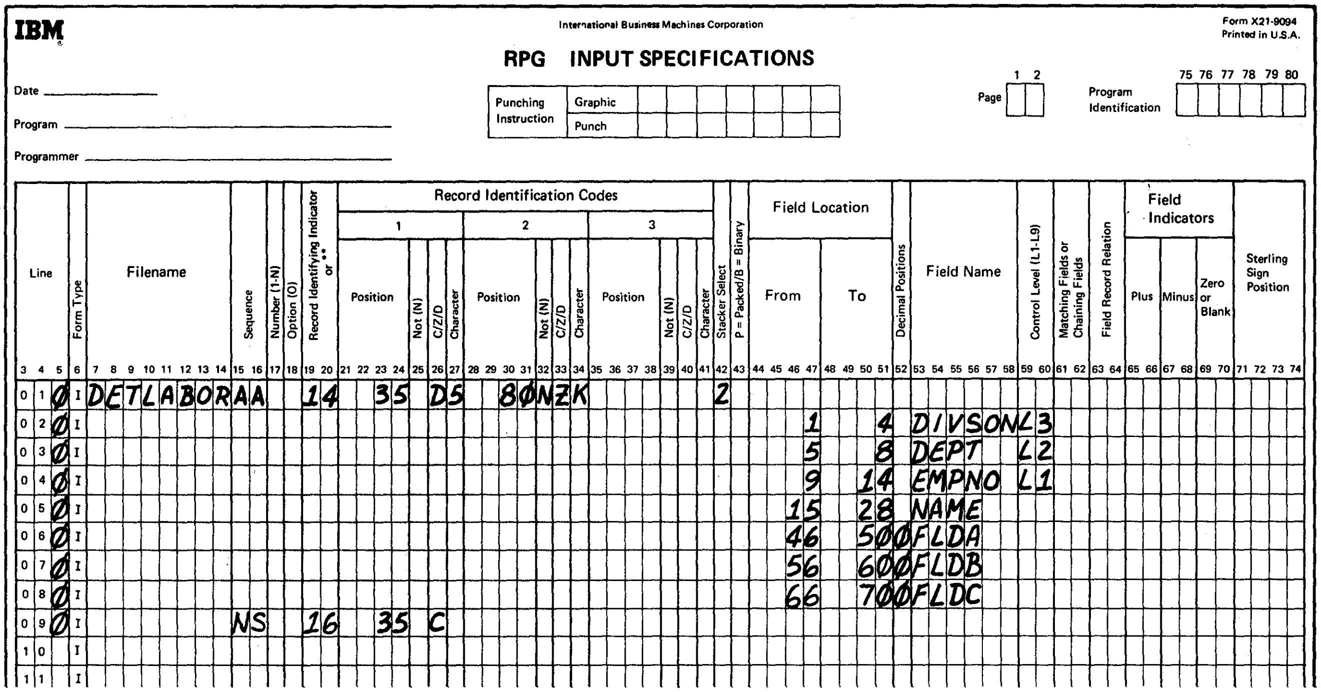
Input Specifications

Three types of tests can be made on the Input Specifications form: tests to determine if the input field contains a plus, minus, zero or blank.

Calculation Specifications

Three types of tests can be made on the Calculation Specifications form: tests to determine whether the result of a calculation is plus, minus, zero or blank.

The program also can compare two fields and can test the result to determine if the contents of one field is greater than, smaller than, or equal to that of the other field.



IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date ________

Program ________

Programmer ________

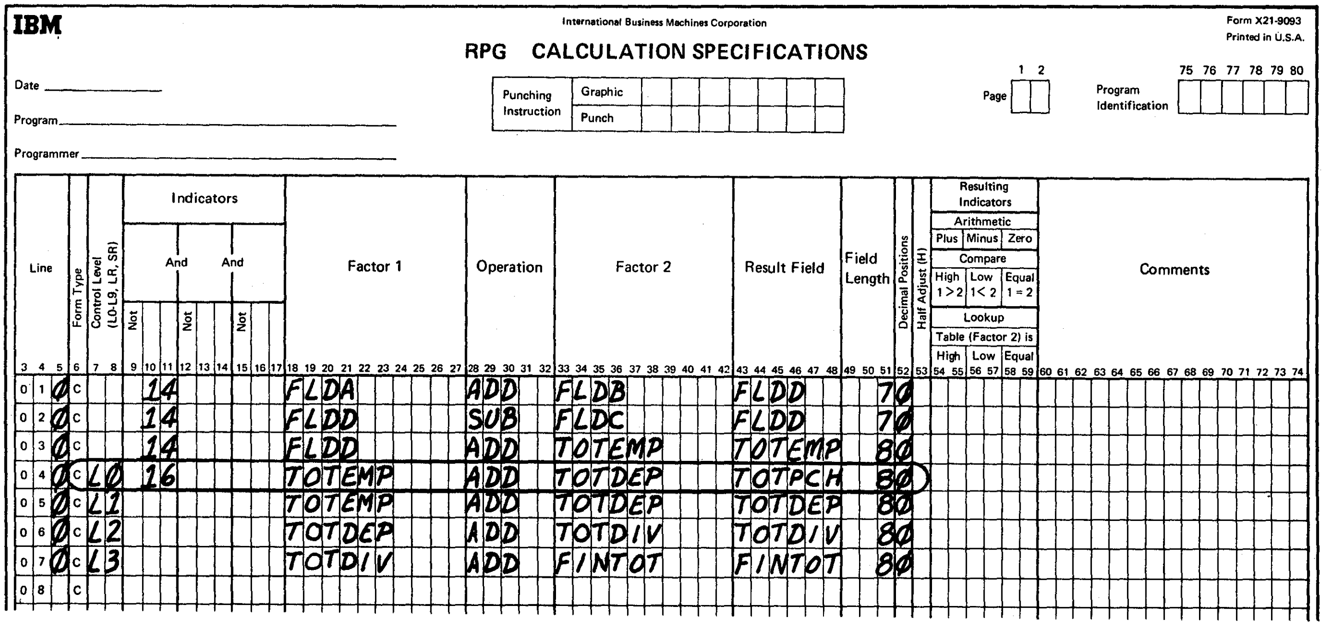
Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1 Position	Not (N)	C/Z/D	Character	Record Identification Codes 2 Position	Not (N)	C/Z/D	Character	Record Identification Codes 3 Position	Not (N)	C/Z/D	Character	Stacker Select	P = Packed/B = Binary	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Minus	Zero or Blank	Sterling Sign Position
01Ø	I	DETLABOR	AA			14	35		D	5	8Ø	N	Z	K					2												
02Ø	I																				1	4		DIVSON	L3						
03Ø	I																				5	8		DEPT	L2						
04Ø	I																				9	14		EMPNO	L1						
05Ø	I																				15	28		NAME							
06Ø	I																				46	5Ø	Ø	FLDA							
07Ø	I																				56	6Ø	Ø	FLDB							
08Ø	I																				66	7Ø	Ø	FLDC							
09Ø	I		NS			16	35		C																						
10	I																														
11	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 ________

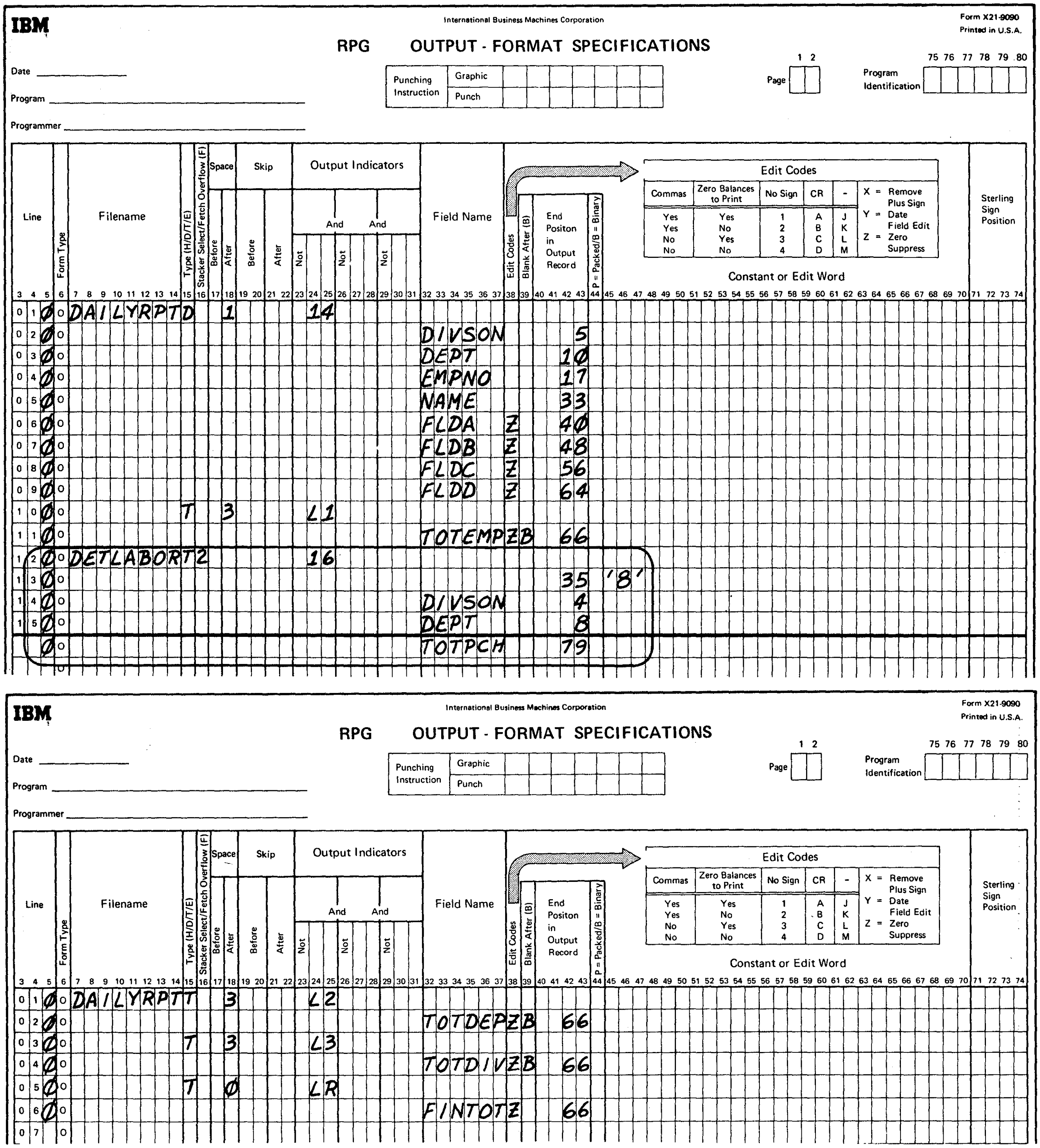
Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators Not	Indicators	And Not	And	And Not	And	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Plus / High 1>2 / High	Minus / Low 1<2 / Low	Zero / Equal 1=2 / Equal	Comments
01Ø	C			14					FLDA	ADD	FLDB	FLDD	7	Ø					
02Ø	C			14					FLDD	SUB	FLDC	FLDD	7	Ø					
03Ø	C			14					FLDD	ADD	TOTEMP	TOTEMP	8	Ø					
04Ø	C	LØ		16					TOTEMP	ADD	TOTDEP	TOTPCH	8	Ø					
05Ø	C	L1							TOTEMP	ADD	TOTDEP	TOTDEP	8	Ø					
06Ø	C	L2							TOTDEP	ADD	TOTDIV	TOTDIV	8	Ø					
07Ø	C	L3							TOTDIV	ADD	FINTOT	FINTOT	8	Ø					
08	C																		

Figure 17. Summary Punching Example, Calculation Specifications



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 (1 2)

Program Identification (75 76 77 78 79 80)

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

Sterling Sign Position

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select/Fetch Overflow (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	Constant or Edit Word
01Ø	O	DAILYRPT	D			1			14					
02Ø	O									DIVSON			5	
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		T			3			L1					
11Ø	O									TOTEMP	Z	B	66	
12Ø	O	DETLABOR	T	2					16					
13Ø	O												35	'8'
14Ø	O									DIVSON			4	
15Ø	O									DEPT			8	
Ø	O									TOTPCH			79	
	O													

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 (1 2)

Program Identification (75 76 77 78 79 80)

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

Sterling Sign Position

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select/Fetch Overflow (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	Constant or Edit Word
01Ø	O	DAILYRPT	T			3			L2					
02Ø	O									TOTDEP	Z	B	66	
03Ø	O		T			3			L3					
04Ø	O									TOTDIV	Z	B	66	
05Ø	O		T			Ø			LR					
06Ø	O									FINTOT	Z		66	
07	O													

Figure 18. Summary Punching Example, Output Specifications

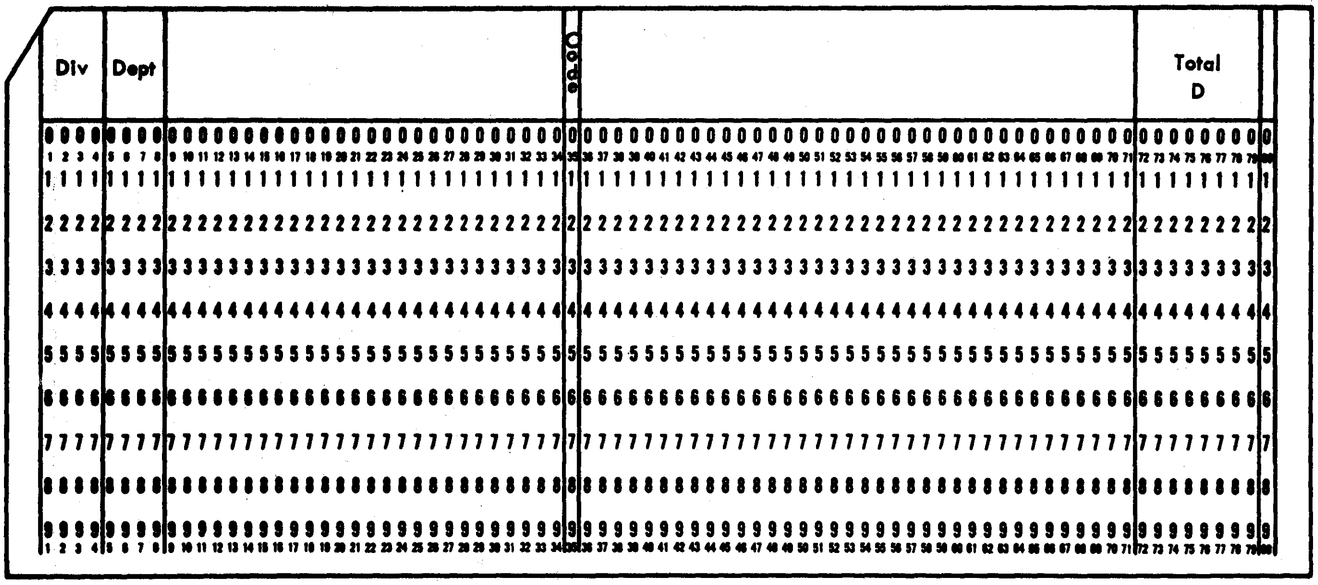


Figure 19. Summary Punching Example, Summary Card Format

53687

Figures 20 and 21 illustrate a test for a zero balance. Figure 22 shows the specifications for a test for a minus balance.

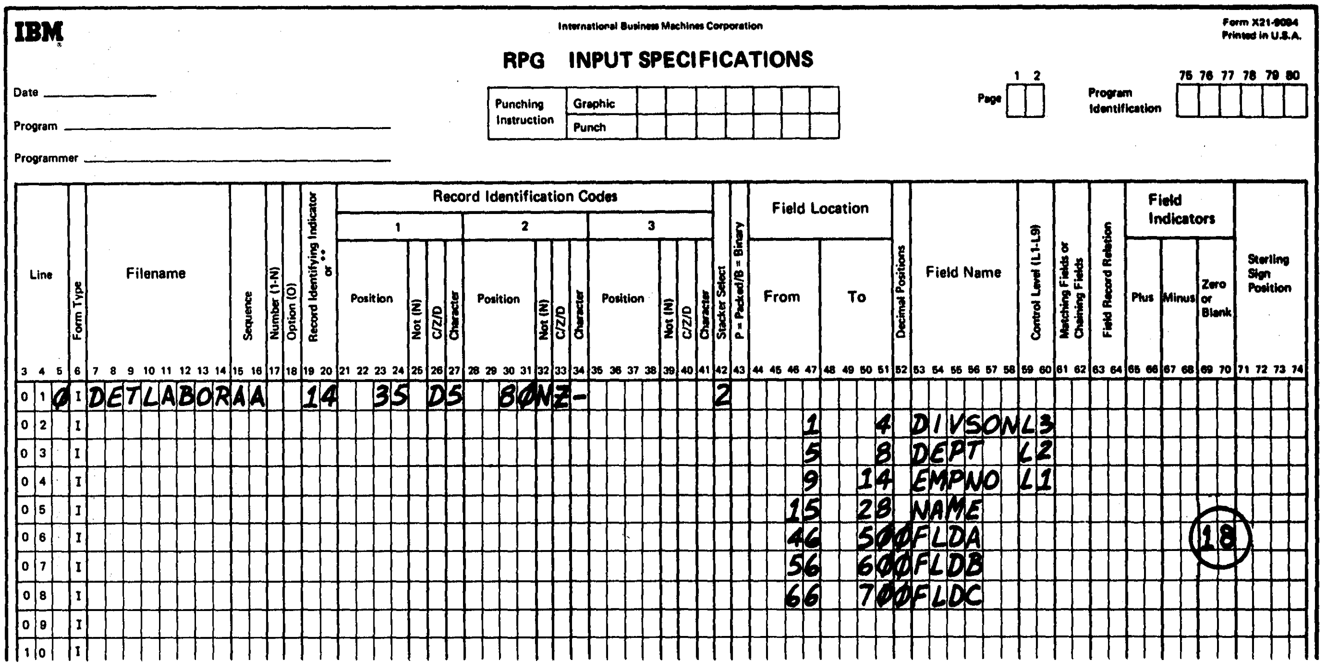
Figure 20 illustrates a typical test of the contents of an input field. If FLDA (line 6) contains zeros, it is unnecessary to perform some of the operations entered on the Calculation Specifications form. To make this test, the programmer places a number from 01 to 99 in Field Indicators: Zero or Blank (positions 69-70). In this example the number is 18. If a card read into the system contains zeros in positions 46-50 (FLDA), indicator 18 is set on.

Setting on an indicator means a special condition has occurred, and the program must consider this condition during the processing of calculation specifications and/or output specifications. This indicator condition is no different from having a resulting indicator set on by a specific record identification code from an input card.

For example, the entries circled in Figure 21 are the additional specifications that bypass the calculation upon detail cards if the value in FLDA is zero or blank.

The specifications 14N18 in Indicators (positions 10-14) mean that the calculation will be performed if indicator 14 is on and indicator 18 is off. (The N in N18 stands for Not.) As written on the Calculation Specifications form in Figure 21, the detail calculations are not performed if the value of FLDA from the input card is zero (Figure 20).

The same indicator specification, N18, could also be used on the Output-Format Specifications form to prevent the printing of detail cards when FLDA is zero (if that was a requirement of the program).



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

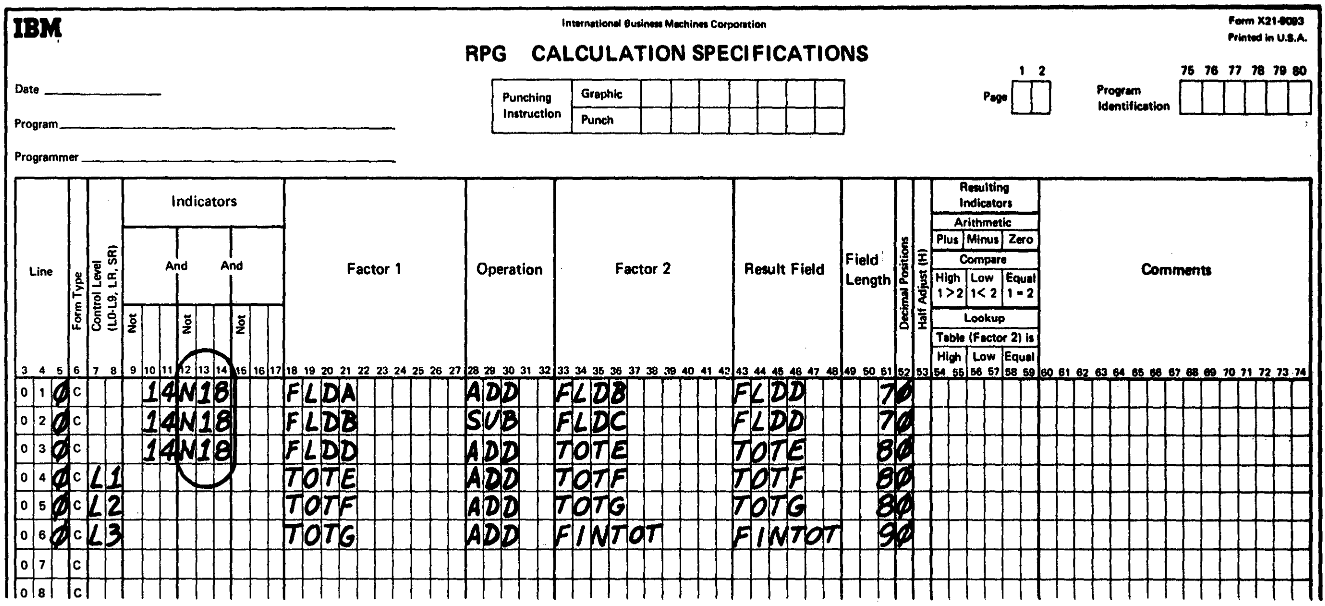
RPG INPUT SPECIFICATIONS

Date ____ Program ____ Programmer ____

Punching Instruction: Graphic / Punch — Page 1 2 — Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Record Identifying Indicator	Position (1)	Not (N)	C/Z/D	Character	Position (2)	Not (N)	C/Z/D	Character	Stacker Select	From	To	Decimal Positions	Field Name	Control Level (L1-L9)	Field Indicators Zero or Blank
0 1 Ø	I	DETLABOR	AA	14	35		D	5	8Ø	N	Z	-	2						
0 2	I													1	4		DIVSON	L3	
0 3	I													5	8		DEPT	L2	
0 4	I													9	14		EMPNO	L1	
0 5	I													15	28		NAME		
0 6	I													46	50	Ø	FLDA		18
0 7	I													56	60	Ø	FLDB		
0 8	I													66	70	Ø	FLDC		
0 9	I																		
1 0	I																		

Figure 20. Specifying a Test for a Zero Balance



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

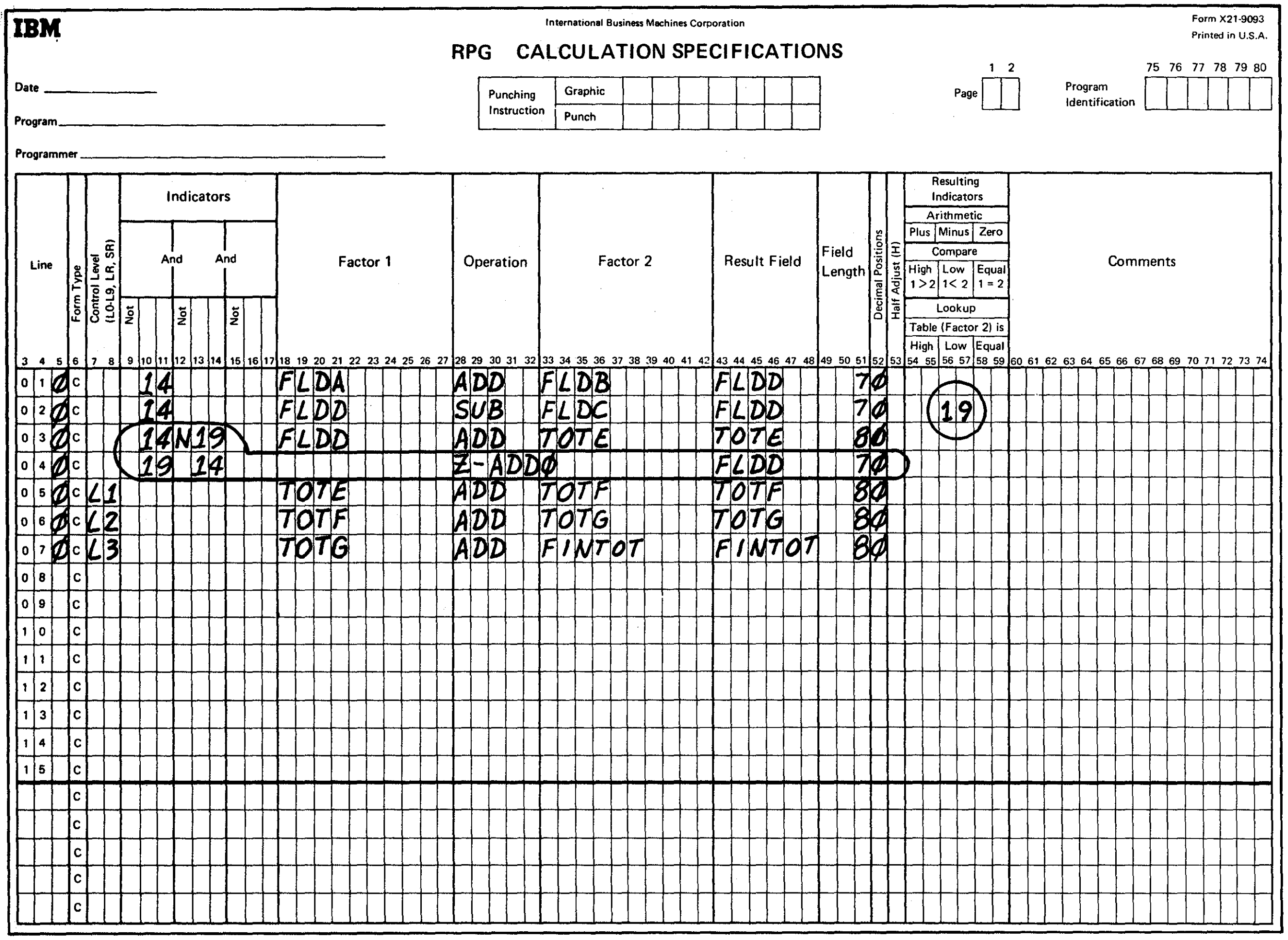
RPG CALCULATION SPECIFICATIONS

Date ____ Program ____ Programmer ____

Punching Instruction: Graphic / Punch — Page 1 2 — Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions
0 1 Ø	C		14 N18	FLDA	ADD	FLDB	FLDD	7	Ø
0 2 Ø	C		14 N18	FLDB	SUB	FLDC	FLDD	7	Ø
0 3 Ø	C		14 N18	FLDD	ADD	TOTE	TOTE	8	Ø
0 4 Ø	C	L1		TOTE	ADD	TOTF	TOTF	8	Ø
0 5 Ø	C	L2		TOTF	ADD	TOTG	TOTG	8	Ø
0 6 Ø	C	L3		TOTG	ADD	FINTOT	FINTOT	9	Ø
0 7	C								
0 8	C								

Figure 21. 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 (1 2) □□

Program Identification (75 76 77 78 79 80) □□□□□□

Line (3-5)	Form Type (6)	Control Level (L0-L9, LR, SR) (7-8)	Indicators: Not (9)	Indicators (10-11)	And: Not (12)	And (13-14)	And: Not (15)	And (16-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field (43-48)	Field Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Arithmetic Plus / Compare High 1>2 / Lookup Table (Factor 2) is High (54-55)	Minus / Low 1<2 / Low (56-57)	Zero / Equal 1=2 / Equal (58-59)	Comments (60-74)
01Ø	C			14					FLDA	ADD	FLDB	FLDD	7	Ø					
02Ø	C			14					FLDD	SUB	FLDC	FLDD	7	Ø			19		
03Ø	C			14	N	19			FLDD	ADD	TOTE	TOTE	8	Ø					
04Ø	C			19		14				Z-ADD	Ø	FLDD	7	Ø					
05Ø	C	L1							TOTE	ADD	TOTF	TOTF	8	Ø					
06Ø	C	L2							TOTF	ADD	TOTG	TOTG	8	Ø					
07Ø	C	L3							TOTG	ADD	FINTOT	FINTOT	8	Ø					
08	C																		
09	C																		
10	C																		
11	C																		
12	C																		
13	C																		
14	C																		
15	C																		
	C																		
	C																		
	C																		
	C																		
	C																		

Figure 22. Testing for a Minus Condition

USING RESULTING INDICATORS

Problem

The calculation specifications in this example illustrate the use of a resulting indicator to test for a minus balance. The result of the test can be used to bypass some specifications and to process other specifications only when the condition tested for is present.

Specifications

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

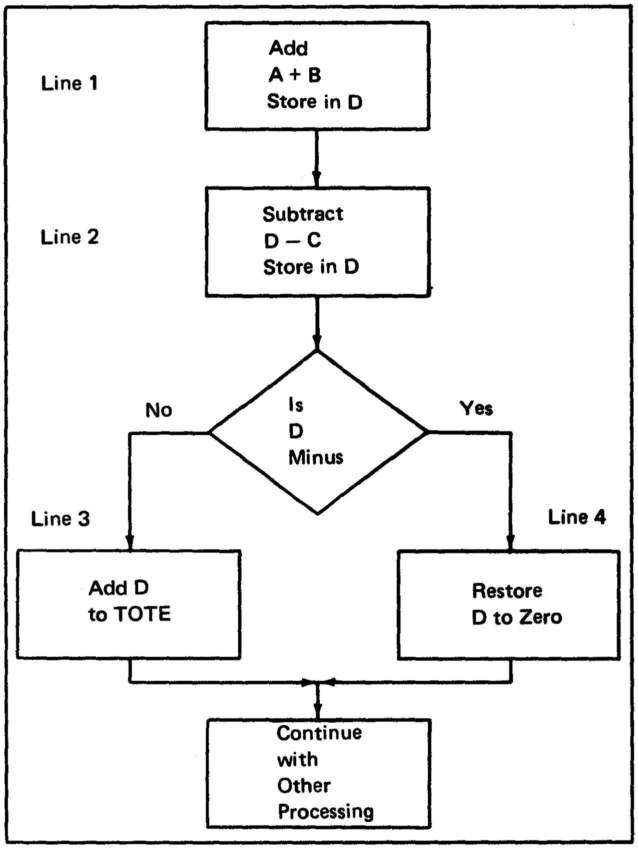


Figure 23. Testing Indicators to Govern Processing

Line 1 specifies that FLDB is to be added to FLDA and that the result is to be placed in FLDD. Line 2 specifies that FLDC is to be subtracted from FLDD, that the result is to be placed in FLDD, and that FLDD is to be tested to determine if the result is minus. In this example it is assumed that if a minus balance occurs, the calculation has no meaning. Therefore, if the result is minus, two things must be done:

1. The result must not be added into field TOTE.

2. The contents of field FLDD must be reset to zeros. (This step might be required if FLDD were used in a subsequent step and the minus balance remaining would give incorrect results.)

This is accomplished on the specification form (Figure 22) by placing an indicator code (19) in Resulting Indicators, Minus on specification line 2. Indicator 19 is set on for a minus condition.

The function of adding TOTE to FLDD is specified on line 3. It is done only if there is no minus condition resulting from the test on line 2. The specifications on line 4 cause FLDD to be reset to zeros only on a minus condition.

The 0 in Factor 2 on line 4 is known as a literal and is used to set FLDD to zeros. (A literal is the actual value to be used in a calculation rather than the name of the location of the data to be used.) The remaining specification on the form in Figure 22 are the same as those from previous examples.

By using Indicator 19, it is possible to suppress detail card printing when the calculation D – C results in a minus balance.

COMPARISON OF TWO FIELDS

The calculation specifications in this example illustrate the ability to compare two fields and govern processing according to the result of the comparison. The comparison may be tested for a high, low, or equal condition. The result of the test can then be used to bypass some specifications or to process other specifications only when a particular condition is present.

Specifications

The calculation specifications for this example are shown in Figure 24. The program logic for the example is shown in Figure 25.

Line 1 specifies that FLDA is compared to FLDB. If the two fields are equal, Indicator 18 is set on, and FLDA is moved to HOLD (line 2). If the two fields are not equal (line 3), FLDB is subtracted from FLDA and the result is placed in SAVE.

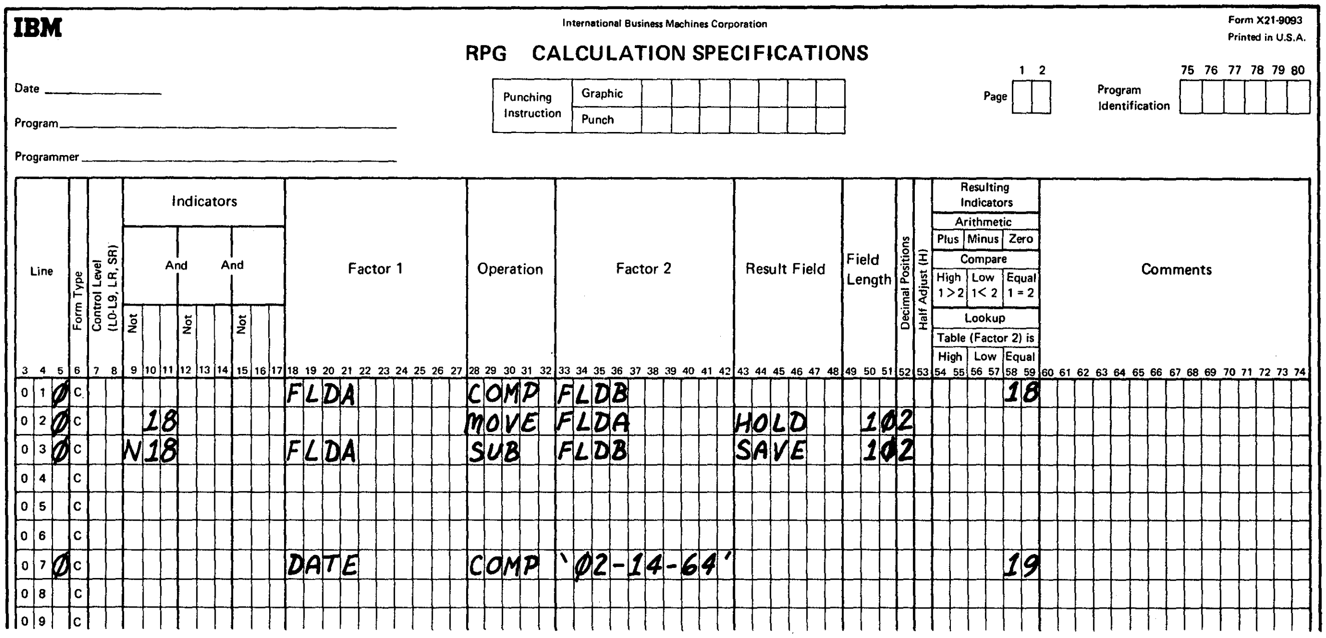
A literal may also be used in a comparison specification. In the lower half of Figure 24, the contents of the input field, DATE, are compared against 02-14-64: indicator 19 is set on if they are equal.

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.

Problem

This example illustrates an inventory card (Figure 26) that must have the quantity multiplied by the price to develop a total cost on a weekly basis. The price of the part is also updated each week so that the price for the following week reflects the average fabrication costs and scrap losses on a year-to-date basis.



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 (3-5)	Form Type (6)	Control Level (LO-L9, LR, SR) (7-8)	Indicators (9-17)	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 High 1>2 / Plus (54-55)	Low 1<2 / Minus (56-57)	Equal 1=2 / Zero (58-59)	Comments (60-74)
01Ø	C			FLDA	COMP	FLDB							18	
02Ø	C		18		MOVE	FLDA	HOLD	1Ø	2					
03Ø	C		N18	FLDA	SUB	FLDB	SAVE	1Ø	2					
04	C													
05	C													
06	C													
07Ø	C			DATE	COMP	'Ø2-14-64'							19	
08	C													
09	C													

Figure 24. Specifying a Comparison

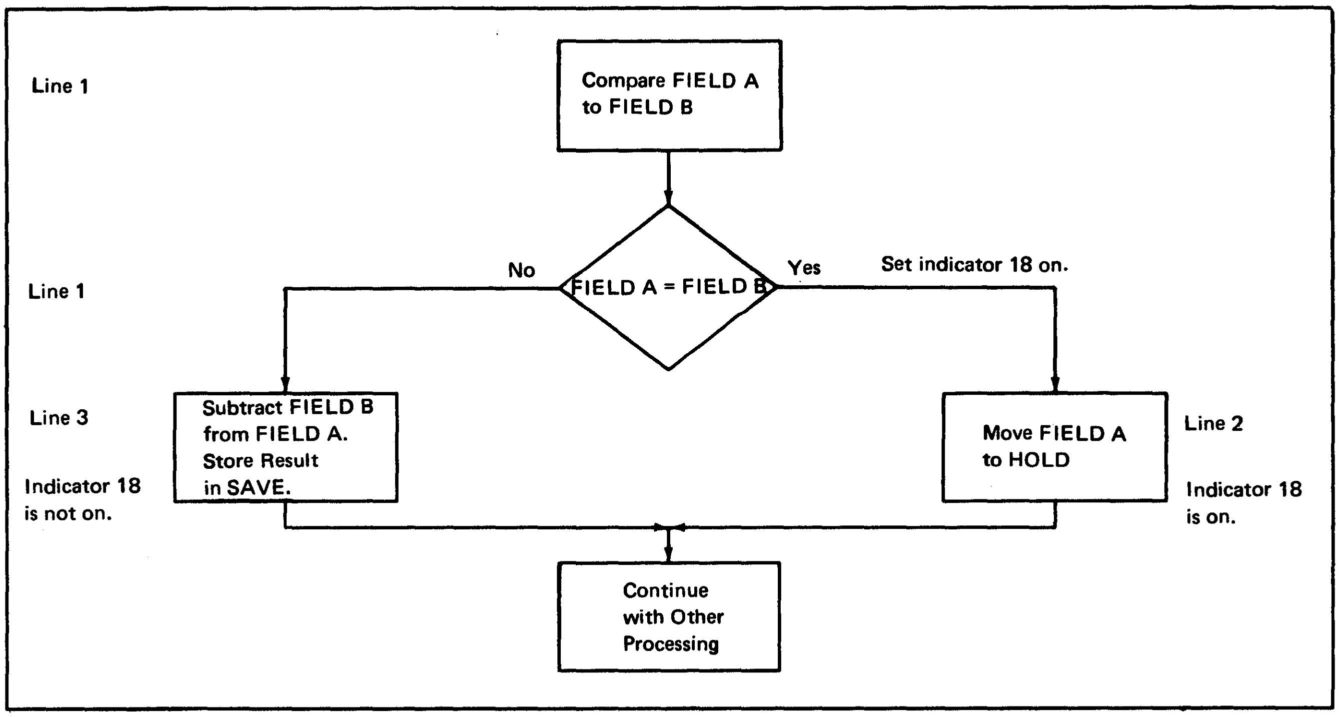


Figure 25. Testing a Comparison

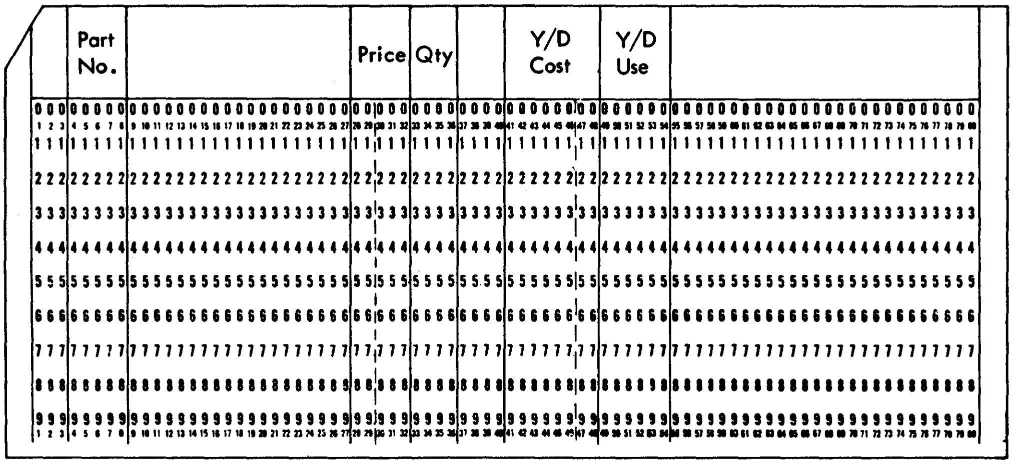


Figure 26. Inventory Card

53690

Specifications

The specifications to accomplish these functions 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 size of each field and the number of decimal places within each field. The fields in this example are:

Part Number	*xxxxx*	*Number of decimal positions (alphameric)*
Price	xx.xxx	3 decimal positions
Quantity	xxxx.	0 decimal positions
Y/D Cost	xxxxxx.xx	2 decimal positions
Y/D Usage	xxxxxx.	0 decimal positions

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

The calculation specifications for this example are shown in Figure 28. Price is multiplied by quantity and the result is placed in a field called COST. The field length of COST is specified as 9. The Result Field is to have two decimal positions and to be half adjusted as specified by the H in Half Adjust (position 53). The following is an example of the arithmetic of this operation, using actual values:

```
Price        1.213
Quantity     x 216
             -----
              7178
             1213
            2426
           -------
           262.008
```

The result field was specified for two decimal positions; therefore, the half adjustment is made to the digit 8 in the units position of the field. Half adjustment is always made to the position to the right of the last position retained as part of the result, as follows:

```
262.008
+     5 Half adjust
-------
262.013
```

The result, 262.01, is stored as the contents of COST. The position that was half adjusted is dropped.

The second line of the Calculation Specifications form provides the specifications for dividing year-to-date costs by year-to-date usage. The result is placed in a field called PRICE. The field length of PRICE is specified as 5, including three decimal positions. The result is to be half adjusted, as specified by the H in Half Adjust. An example of the arithmetic of this operation, using actual values, follows. Year-to-date cost divided by usage equals price.

```
(Y/D                   1.3419        (price)
usage)      1296  | 1739.23xx        (Y/D cost)
```

The result field was specified for three decimal places. Therefore, the half adjustment is made to the digit 9 in the units position. This is the position to the right of the last position retained as part of the result, as follows:

```
1.3419
+    5 Half adjust
------
1.3424
```

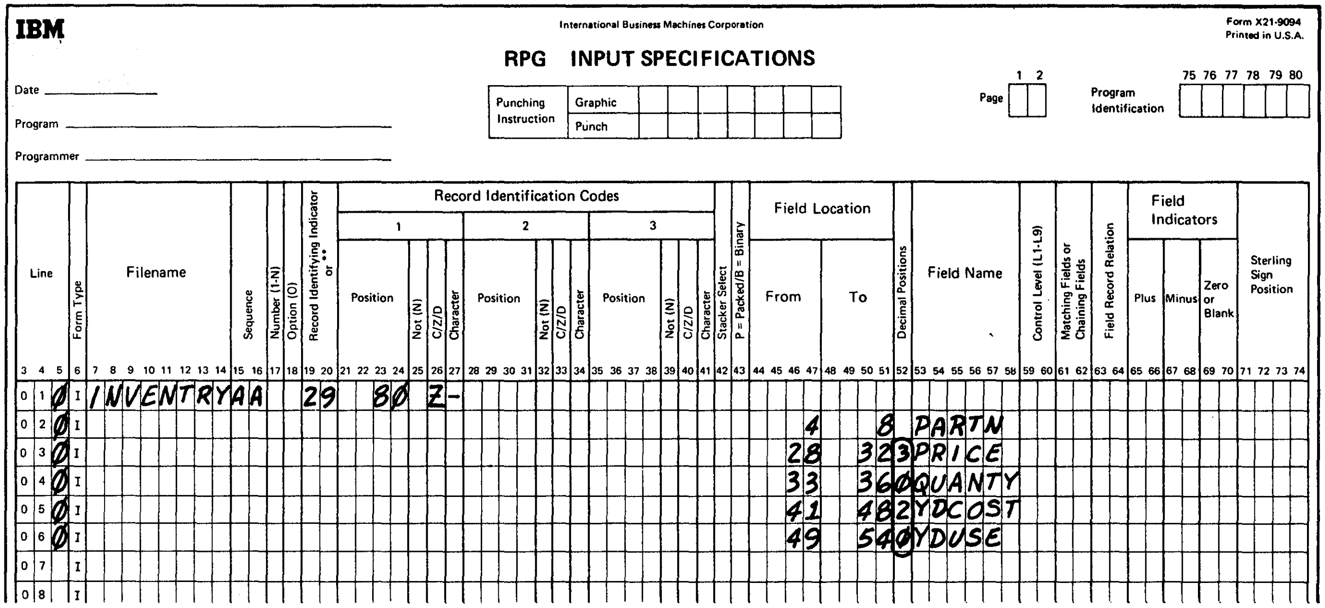
The result 1.342 is stored as the contents of PRICE. The position that was half adjusted is dropped. If half adjust is not used, the rightmost digits are dropped.

SEQUENCE CHECKING

Two types of sequence checking functions can be performed with RPG:

1. Checking the sequence of different record types within a control group.

2. Checking the sequence of control groups. (See the section *Input Specifications Form, Using the Matching Fields Specification for Sequence Checking.*)



IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date

Program

Programmer

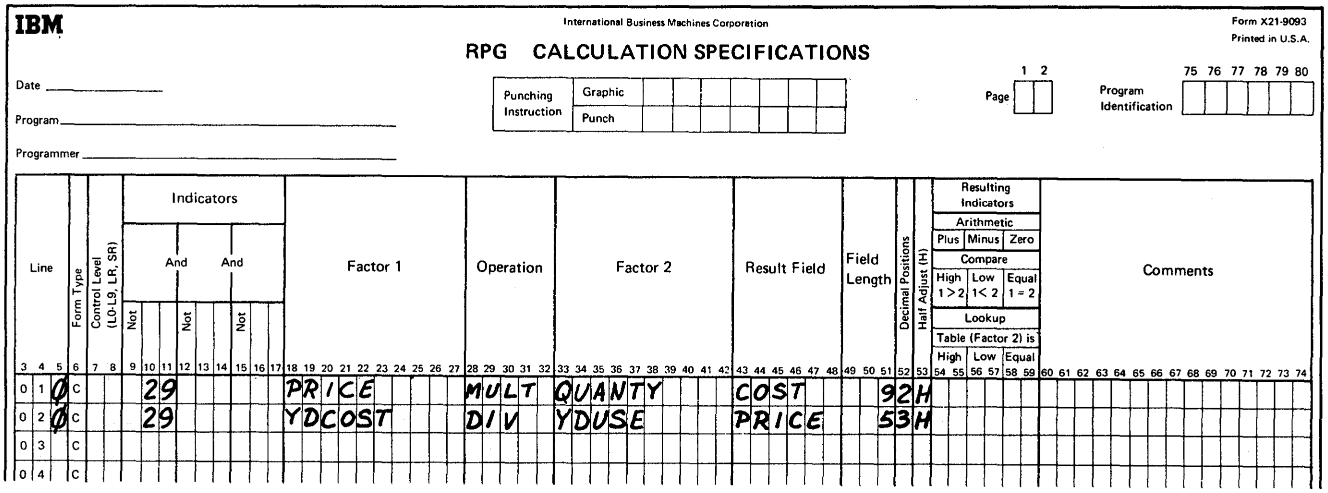
Punching Instruction: Graphic / Punch

Page (1 2)

Program Identification (75 76 77 78 79 80)

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1 Position	Not (N)	C/Z/D	Character	2 Position	Not (N)	C/Z/D	Character	3 Position	Not (N)	C/Z/D	Character	Stacker Select	P = Packed/B = Binary	Field Location From	To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Minus	Zero or Blank	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
01Ø	I	INVENTRY	AA			29	8Ø		Z	-																					
02Ø	I																				4	8		PARTN							
03Ø	I																				28	32	3	PRICE							
04Ø	I																				33	36	Ø	QUANTY							
05Ø	I																				41	48	2	YDCOST							
06Ø	I																				49	54	Ø	YDUSE							
07	I																														
08	I																														

Figure 27. 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 (1 2)

Program Identification (75 76 77 78 79 80)

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 / Compare High 1>2 / Lookup Table (Factor 2) is High	Minus / Low 1<2 / Low	Zero / Equal 1=2 / Equal	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
01Ø	C			29					PRICE	MULT	QUANTY	COST	9	2	H				
02Ø	C			29					YDCOST	DIV	YDUSE	PRICE	5	3	H				
03	C																		
04	C																		

Figure 28. Calculation Specifications

Sequence Checking of Different Record Types Within a Control Group

The application consists of updating an inventory file, which, in this case, contains from one to four of the following types for each inventory part number:

Card	*Code*
Balance forward	5 in column 78
Issue	3 in column 78
Receipt	2 in column 78
Adjustment	8 in column 78

Figure 29 illustrates the specifications required to check the sequence within a group of four card types. In a complete problem the control groups (part numbers) would be designated by use of a control level indicator as explained in a previous example. The numbers on the form refer to the following numbers.

1. The record identification codes for the four cards are specified in the same manner as in previous examples. The C in C/Z/D indicates that the entire character punched in the card is examined to establish the record identification code.

   In the specifications for the last card type (specification line 13 on the form), a D is written in C/Z/D because there is a possibility that some of these card types may have a zone punch. The D specifies that only the digit punches in the card are examined to identify the card type. Thus, zone punches that could result in unequal comparisons are ignored.

2. The sequence established for the file is determined by the sequence in which the specifications for each card type are written on the form and by the numbers placed in Sequence. In this example, the digits 01, 02, 03, and 04 are used. The numbers assigned must begin with 01 in each file and must be consecutive in ascending order.

   Alphabetic characters under Sequence in preceding examples indicate that no sequence checking is to take place. Alphabetic specifications must always be written before numeric specifications.

   These specifications are all that are required to cause the object module to perform a sequence check of the various record types within the part number control groups. If a sequence check error is detected, a special indicator can be tested in the program in order to determine its status.

   Two additional specifications, Number and Option, are used in these types of applications.

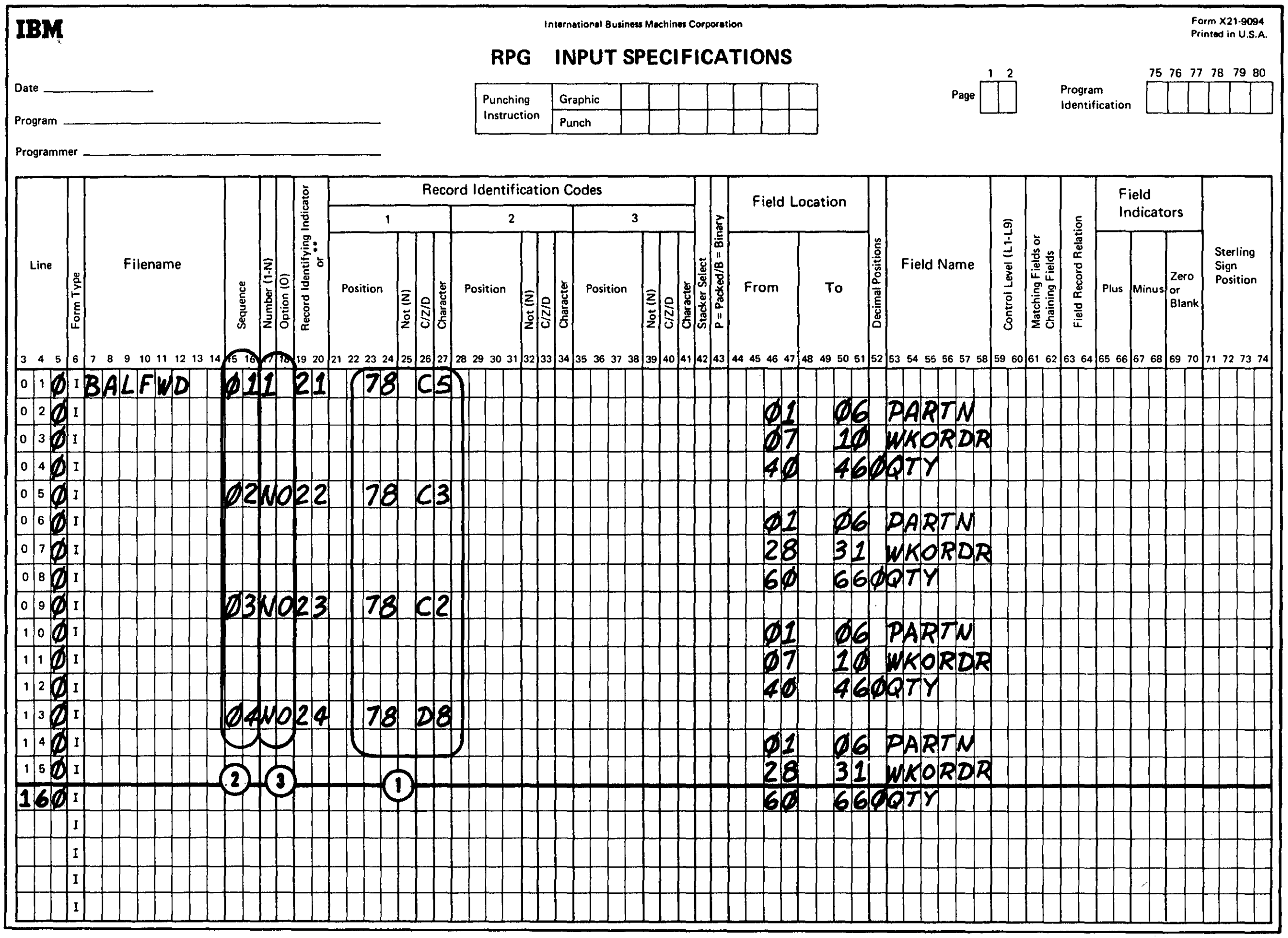
3. If a numeric specification is provided in Sequence, a specification must be provided in Number.

   On the first specification line, the number 1 in Number indicates that one record of that type must be present in each group. In this example only one balance forward card for each inventory part number must be present. If there is no balance forward card, the program recognizes an error in the input file.

   The letter N in Number of the specifications for the other record types means that multiple card types for each part number may be present. In this example, multiple cards for issues, receipts, and adjustments may be present.

   The letter O in Option in the specifications means that the records are optional. That is, a record may or may not be present.

   If the letter O is not specified, it means that the particular record must be present. This requirement applies only if Sequence has been specified as numeric.



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) □□

Program Identification (75 76 77 78 79 80) □□□□□□

Line (3-5)	Form Type (6)	Filename (7-14)	Sequence (15-16)	Number (1-N) (17)	Option (O) (18)	Record Identifying Indicator or ** (19-20)	1 Position (21-24)	1 Not (N) (25)	1 C/Z/D (26)	1 Character (27)	2 Position (28-31)	2 Not (N) (32)	2 C/Z/D (33)	2 Character (34)	3 Position (35-38)	3 Not (N) (39)	3 C/Z/D (40)	3 Character (41)	Stacker Select (42)	P = Packed/B = Binary (43)	Field Location From (44-47)	Field Location To (48-51)	Decimal Positions (52)	Field Name (53-58)	Control Level (L1-L9) (59-60)	Matching Fields or Chaining Fields (61-62)	Field Record Relation (63-64)	Field Indicators Plus (65-66)	Field Indicators Minus (67-68)	Field Indicators Zero or Blank (69-70)	Sterling Sign Position (71-74)
01Ø	I	BALFWD	Ø1	1		21	78		C	5																					
02Ø	I																				Ø1	Ø6		PARTN							
03Ø	I																				Ø7	1Ø		WKORDR							
04Ø	I																				4Ø	46	Ø	QTY							
05Ø	I		Ø2	N	O	22	78		C	3																					
06Ø	I																				Ø1	Ø6		PARTN							
07Ø	I																				28	31		WKORDR							
08Ø	I																				6Ø	66	Ø	QTY							
09Ø	I		Ø3	N	O	23	78		C	2																					
10Ø	I																				Ø1	Ø6		PARTN							
11Ø	I																				Ø7	1Ø		WKORDR							
12Ø	I																				4Ø	46	Ø	QTY							
13Ø	I		Ø4	N	O	24	78		D	8																					
14Ø	I																				Ø1	Ø6		PARTN							
15Ø	I																				28	31		WKORDR							
16Ø	I																				6Ø	66	Ø	QTY							
	I																														
	I																														
	I																														
	I																														

Figure 29. Input Specifications

CORRELATION OF THE RPG SPECIFICATIONS FORMS

Figure 30 shows a file-to-file sample program. (This type of report is often called an 80/80 listing.) It illustrates the relationships of the RPG specification forms.

Assume 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 printed. Three specifications forms are required for this program: File Description, Input, and Output-Format.

File Description Specifications Forms

The two files are described on this form. The card input file is assigned the name INPUT, and the printed output file is named OUTPUT. Page and line sequence numbers are entered in positions 1-5. An F in position 6 indicates that each entry is a file description entry. The file names are entered in positions 7-14 (filename). Position 15 contains an I or an O to indicate whether the file has an input or an output function.

Position 16 of the input file of the File Description Specifications form contains a P because the file described is the primary input file for the job; the E in position 17 indicates that the end-of-file condition for this file occurs when this file is depleted.

Position 19 contains an F and a V to indicate that the file formats are fixed-length and variable-length, respectively.

Block length (positions 20-23) is 80 for the input file because each card is a block of data. Record length (positions 24-27) is also 80 for the input file because each card is an unblocked record. For the output file, the block length and the record length is 132, which is the maximum length of a printer line. The program identification is entered into positions 75-80.

If the input file is read in by an IBM 1442 Card Read-Punch (as it is in this example) the code in Device (positions 40-46) is READ42. The output printer has a Device code of PRINTER.

Input Specifications Form

The Input Specifications form also has the program identification entered in positions 75-80 and the page and line numbers in positions 1-5. An I in position 6 of each card indicates an input specification entry.

The filename is again entered into positions 7-14. Positions 15-16 contain the sequence code AA. Indicator 01, entered into positions 19-20, will be on throughout the job to show that records from its associated file are being processed. The second line describes the field name CARDIN. The field consists of card positions 1-80.

Output-Format Specifications Form

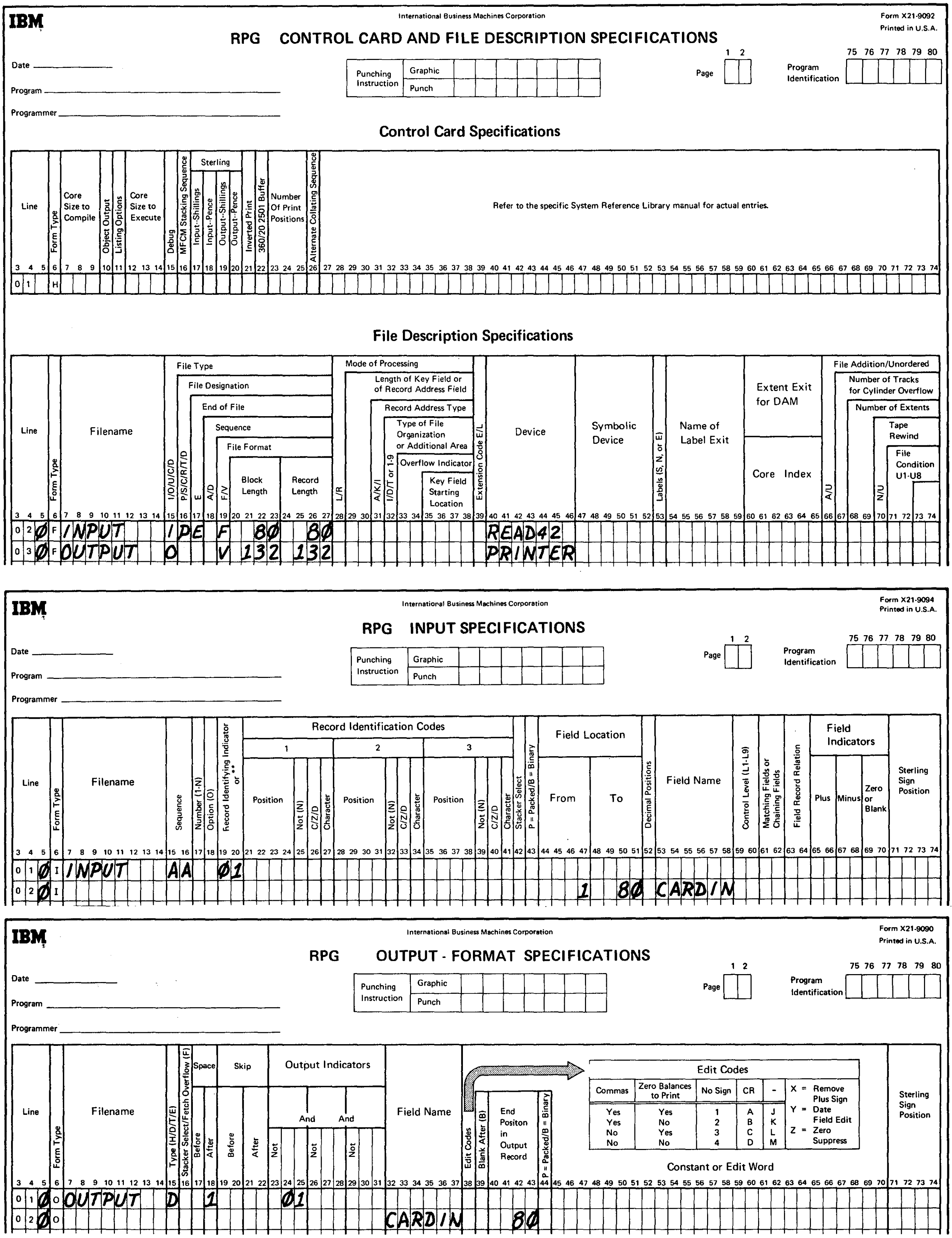
On the Output-Format Specifications form, the filename of the output file is entered in positions 7-14. The D in position 15 indicates that each line printed in this file is a detail line. A single space after printing is specified by the entry in position 18. Indicator 01 from the Input Specifications form is specified in positions 24-25. When indicator 01 is on, a record will be printed.

The second output line has the name of the field to be written in the output record entered in positions 32-37. Data from the field labeled CARDIN is printed (end position of 80) in the output record as specified in positions 40-43.

SUMMARY

This completes the general description of some of the functions that can be performed with RPG. Some of the fields of the specifications forms were not explained and some additional operations that can be performed with RPG remain to be described. At this point, the reader should be able to determine the scope of the RPG program.

See the specific specification form for more information.



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 — Program Identification 75 76 77 78 79 80

Control Card Specifications

Refer to the specific System Reference Library manual for actual entries.

Line (3-5)	Form Type (6)	Core Size to Compile (7-9)	Object Output (10)	Listing Options (11)	Core Size to Execute (12-14)	Debug (15)	MFCM Stacking Sequence (16)	Input-Shillings (17)	Input-Pence (18)	Output-Shillings (19)	Output-Pence (20)	Inverted Print (21)	360/20 2501 Buffer (22)	Number Of Print Positions (23-25)	Alternate Collating Sequence (26)
01	H														

File Description Specifications

Line (3-5)	Form Type (6)	Filename (7-14)	File Type I/O/U/C/D (15)	File Designation P/S/C/R/T/D (16)	End of File E (17)	Sequence A/D (18)	File Format F/V (19)	Block Length (20-23)	Record Length (24-27)	Device (40-46)
02Ø	F	INPUT	I	P	E		F	8Ø	8Ø	READ42
03Ø	F	OUTPUT	O				V	132	132	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 1 2 — Program Identification 75 76 77 78 79 80

Line (3-5)	Form Type (6)	Filename (7-14)	Sequence (15-16)	Record Identifying Indicator (19-20)	Field Location From (44-47)	Field Location To (48-51)	Field Name (53-58)
01Ø	I	INPUT	AA	Ø1			
02Ø	I				1	8Ø	CARDIN

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 1 2 — Program Identification 75 76 77 78 79 80

Line (3-5)	Form Type (6)	Filename (7-14)	Type H/D/T/E (15)	Space After (18)	Output Indicators (24-25)	Field Name (32-37)	End Position in Output Record (40-43)
01Ø	O	OUTPUT	D	1	Ø1		
02Ø	O					CARDIN	8Ø

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 30. File-to-File Program

Each program generated by RPG uses the same general logic, and for each record to be processed the program goes through the same general cycle of operations. Within that cycle, there are two different instances in time when operations specified on the Calculation and Output-Format Specifications forms are performed. These instances are called detail and total time.

For the illustration of this concept, a generalized flowchart of an RPG generated program is shown in Figure 31.

The following numbers correspond to the numbers on Figure 31. A program cycle begins with item 1 and continues through item 11. Steps 6 and 7 are referred to as total time. Steps 1 and 11 are referred to as detail time.

1. Before the first record is read, the program prepares and writes any heading information to be put out on the first page. After the first record has been read, the program prepares and writes heading and detail information which is not conditioned on overflow.

2. The generated program tests for any halt indicators. If any halt indicators are on, the program branches to item 12.

3. The generated program then reads an input record.

4. The generated program tests for the end-of-file conditions. If the end-of-file condition has occurred, the program branches to item 13.

5. All control level indicators and all record identifying indicators (specified) in positions 19-20 of the Input Specifications form are set off. Then, starting with line 1 of the Input Specifications forms and with the record just read, the generated program uses the record identification code to identify the record. When the identification code matches an entry on the Input Specifications form, the program sets on the record identifying indicator that has been specified for the record. When a control field break occurs, appropriate control level indicators are set on.

6. All total calculations are performed. (This step is bypassed for the initial control break which is caused by reading the first input record.)

7. All total output records which are not conditioned on overflow are prepared and put out. (This step is also bypassed for the first control break.)

8. The generated program tests for the last record indicator (LR). If it is on, the program branches to item 14.

9. The object module tests for any overflow condition. If an overflow condition has occurred, the program branches to item 15. Overflow is defined as the condition existing whenever any of the indicators OA-OG, and OV are on. (This step is bypassed for the first control break.)

10. The data fields contained in the input record just read are moved into storage. These fields are specified by field entries on the Input Specifications form.

11. Any detail calculations are performed; processing continues with item 1.

12. Program execution is terminated.

13. The Last Record indicator (LR) is set on and all control level indicators L1-L9 are set on. Then the program branches to item 6.

14. Program execution is terminated.

15. 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.

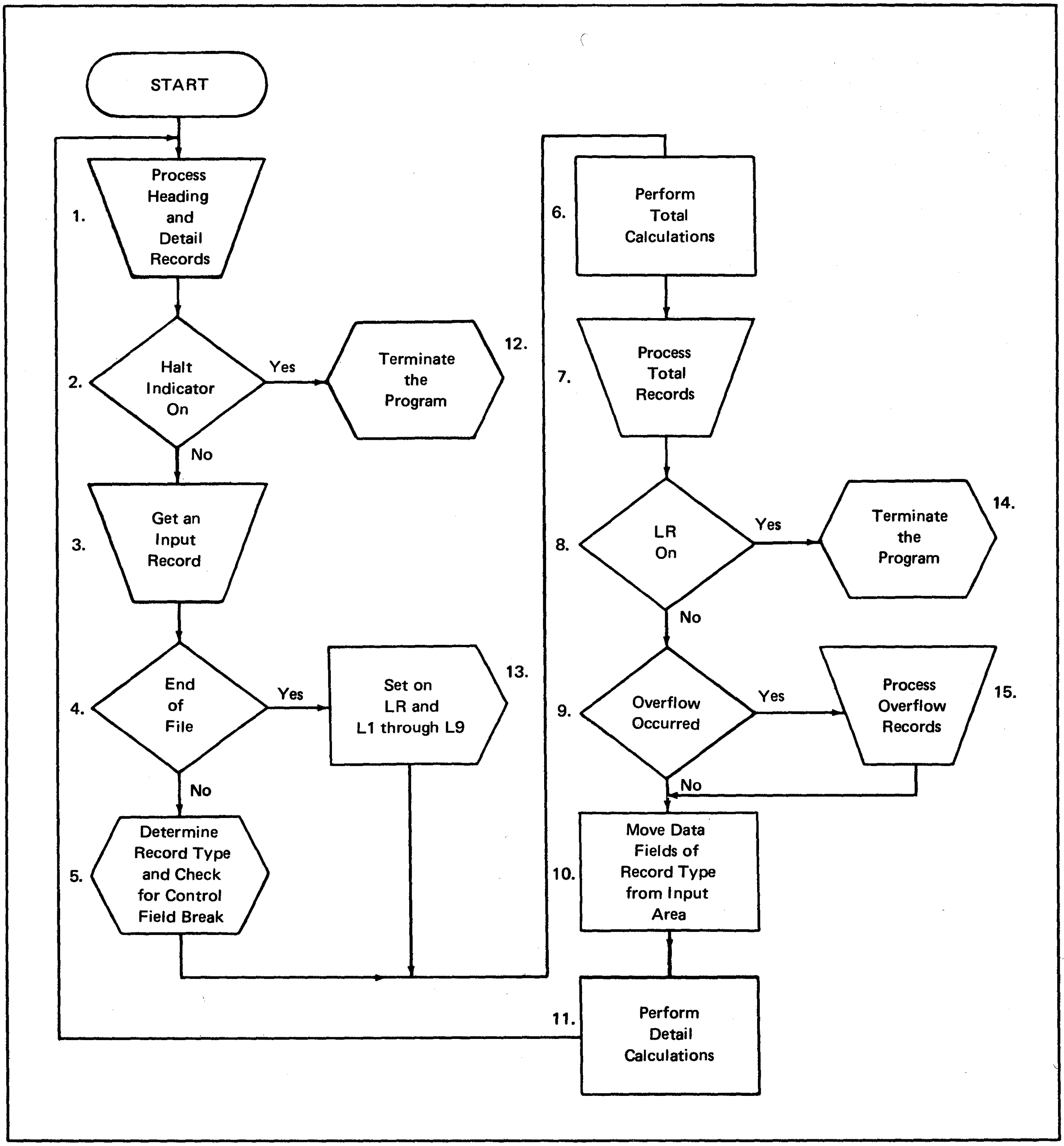


Figure 31. General Logic Flow of a Program Generated by RPG

PROBLEM DEFINITION

The programming examples in the preceding section were intended to introduce the reader to the use of RPG and were therefore kept simple. More complex applications may require a thorough analysis of the existing or proposed system before a program can be written.

This analysis should include a description of source data and its format, and how this data should be processed to develop the report and other necessary output information.

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

1. The available data.

2. The input and output formats to be used.

3. The information required in the input and output formats.

4. The codes to be used to identify the various inputs and outputs and their elements.

5. The handling of the various transactions and exceptions.

After all application data has been gathered, document it for easy reference during the writing of the specification forms. One method of documenting an application is to lay out the complete format of the report on a Printer Spacing Chart. This method also provides a pictorial representation of the final product.

PRINTER SPACING CHART

Before the report specifications are written, the programmer should have a clear picture of what he wants as the final product. If the report is to be printed, he must know the number of fields to be placed on each line of the report, the spacing between lines, and the positioning of the information within each line of the report.

Although no cards for the source deck are punched directly from the entries on this chart, the representation serves as a guide for completing the specification forms. It plays an important role in writing report specifications. If the final product is written on magnetic tape or direct access storage devices or if it is punched in cards, the user must know where the information is to be located. A tape layout chart or a direct access storage device layout chart can be used.

Layout of Lines and Fields

The two most important functions of a printer spacing chart 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. Figure 32 shows an example of the Printer Spacing Chart (Form X24-6436).

The numbers across the top and bottom of the spacing chart represent 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 are spelled out completely in the print positions assigned to them. Variable information is represented by Xs and, where applicable, includes credit symbols, punctuation, etc. The position in an amount field where zero suppression ends is indicated by a zero rather than an X.

GENERAL INFORMATION

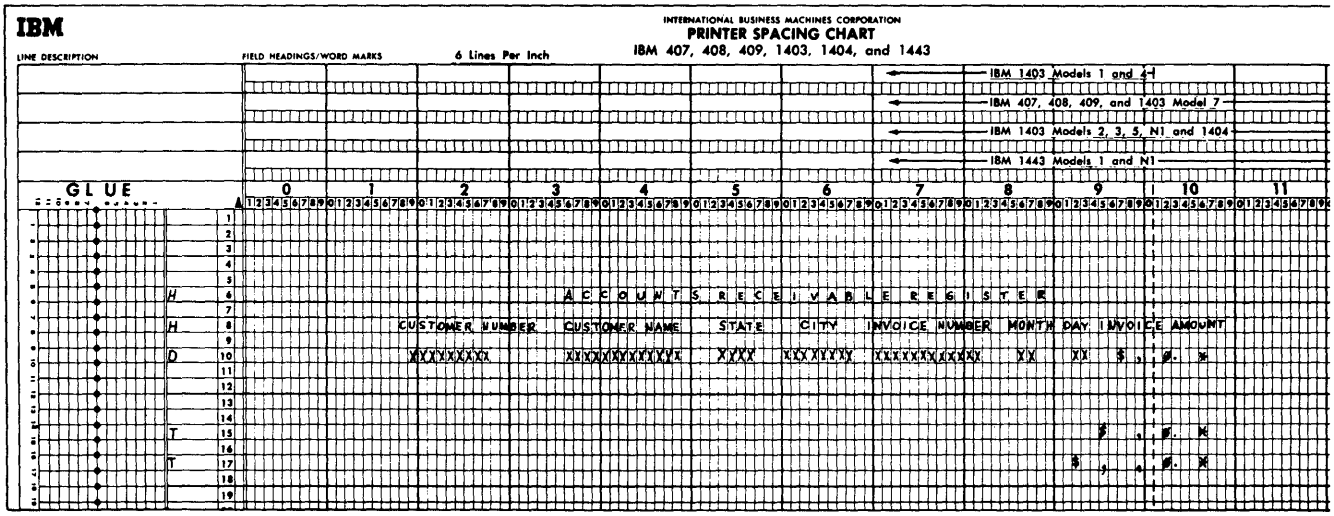
Cross-References

To make this reference manual a more effective learning tool, numerous cross-references have been placed in the manual. They are located wherever it was thought that readers not familiar with disk storage processing and related functions would have difficulty with these unfamiliar subjects.

Disk storage, using ISAM, indexing subroutine, sterling routines, documentation, indicators, debugging, table lookup, matching field, chaining field operations, and related functions are described apart from the detailed descriptions of specifications for them.

These general introductory descriptions, contained at the back of the manual, can be used by the reader as he encounters the related specifications for them throughout the manual.

To facilitate locating them in the manual, all cross-references used are listed in the Index under *Cross-References.*



IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION
PRINTER SPACING CHART
IBM 407, 408, 409, 1403, 1404, and 1443

LINE DESCRIPTION

FIELD HEADINGS/WORD MARKS

6 Lines Per Inch

IBM 1403 Models 1 and 4

IBM 407, 408, 409, and 1403 Model 7

IBM 1403 Models 2, 3, 5, N1 and 1404

IBM 1443 Models 1 and N1

GLUE

Line		Print line
H	6	ACCOUNTS RECEIVABLE REGISTER
H	8	CUSTOMER NUMBER CUSTOMER NAME STATE CITY INVOICE NUMBER MONTH DAY INVOICE AMOUNT
D	10	XXXXXXXXX XXXXXXXXXXXXX XXXX XXXXXXXX XXXXXXXXXXXX XX XX $, 0. *
T	15	$, 0. *
T	17	$, , 0. *

Figure 32. Printer Spacing Chart

53691

The section *Disk Storage Concepts* provides a general introduction to disk file organization and processing including terminology associated with these functions. Readers not familiar with these concepts may wish to review that section before beginning with this section.

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, T for a total line, and E is unused. All lines must be identified as belonging to one of these categories.

Left and Right Justification

When making entries on the RPG forms, it is important that the entry be right justified or left justified as required. Justifying an entry means having it begin in the first position of the specification (left justified), or having it end in the last position of the specification (right justified).

Alphameric entries (composed of both alphabetic and numeric entries) are left justified. Numeric entries are right justified.

Information regarding the correct justification is provided in the description of the entry, in those cases where it may not be clear to the reader as to whether the entry is left or right justified.

Alphabetic Characters

In this manual, the references to alphabetic character designate the letters A through Z, the dollar sign ($), the pound sign (#), and the at sign (@).

Numeric Field Format

The System/360 packed decimal format allows two decimal digits to be represented in one core storage byte. The RPG object program automatically converts all numeric input data from unpacked to packed format. Unless otherwise specified, all numeric data is unpacked before it is put out. In this manual, numeric length refers to the unpacked length although the data is actually stored in the packed format.

Sterling Routines

Sterling routines are included in RPG to provide users with a convenient and time saving means of handling amount fields that are punched in the format of pound sterling monetary units.

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 RPG Control Card form. The other specifications forms are not affected. All calculations are done in the pence unit of measure.

User Date

RPG allows the user to supply a job-related date to his object program at the start of the execution cycle. This data is supplied through the PARM parameter of the EXEC card (refer to *RPG Job Processing, Execution Options*). If the user does not wish to specify a date, he may use the date from the communication region of the system. In either case, this date will be used to initialize the UDATE, UDAY, UMONTH, and UYEAR fields. UDATE is a six-digit numeric field; UDAY, UMONTH, and UYEAR are each two-digit numeric fields. This date must agree with the DATE format specified in the RPG Control Card (see *Control Card Specifications*). The two valid formats for the system date are mmddyy (domestic) where mm=month, dd=day, and yy=year, and ddmmyy (European).

These field names (UDATE, UDAY, UMONTH, and UYEAR) are reserved words and may only be used in the designated positions on the following specifications forms:

- Calculation

1. Positions 18-23 (Factor 1).
2. Positions 33-38 (Factor 2).
3. Positions 43-48 (Result Field) -- RLABL only.

- Output-Format

1. Positions 32-37 (Field Name).

However, UDATE, UMONTH, UDAY, and UYEAR may not be used in Factor 1 with operations TAG and RPGCV, or in Factor 2 with operations EXTCV, GOTO, EXIT, and LOKUP. All other uses will be diagnosed and the specification dropped (including Blank After). These UDATE formats may be edited on output, such as 12/31/68 or 31.12.68, by using the proper edit word.

This section provides detailed explanations of each specification field contained in the six RPG forms.

The forms are listed in the order in which they are used (see Figure 33):

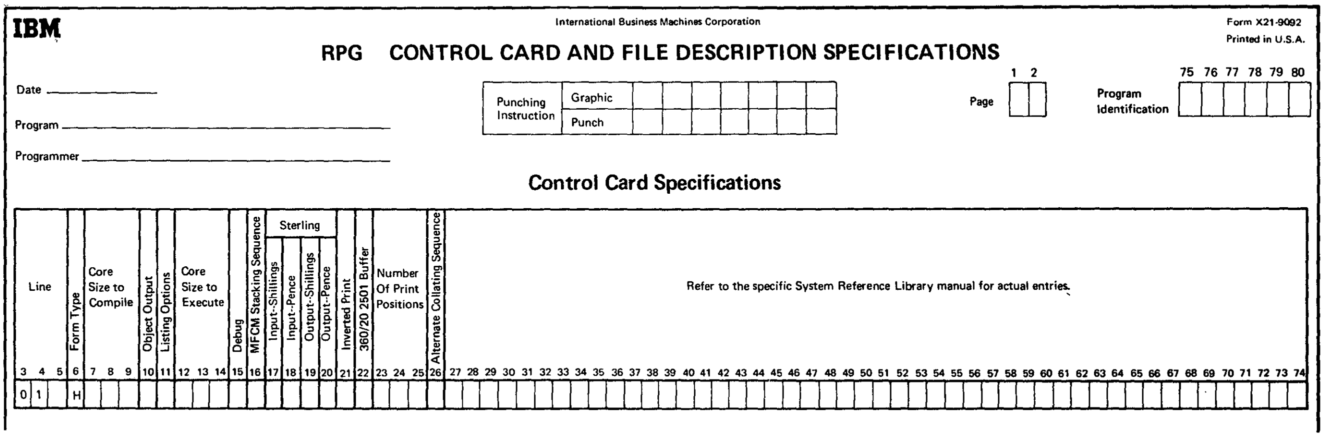
- Control Card and File Description Specifications
- Extension and Line Counter Specifications
- Input Specifications
- Calculation Specifications
- Output-Format Specifications

Control Card Specifications provides information about the job and describes the system to the RPG compiler.

File Description Specifications provides additional information about input and output files that is not included on the input or output forms.

Extension Specifications provides additional information about tables, chaining files, and record address files.

Line Counter Specifications must be used if a report that will ultimately be printed is to be stored on some intermediate device and if the program uses overflow indicators (or automatic skipping).



IBM

International Business Machines Corporation

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

RPG CONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS

Date

Program

Programmer

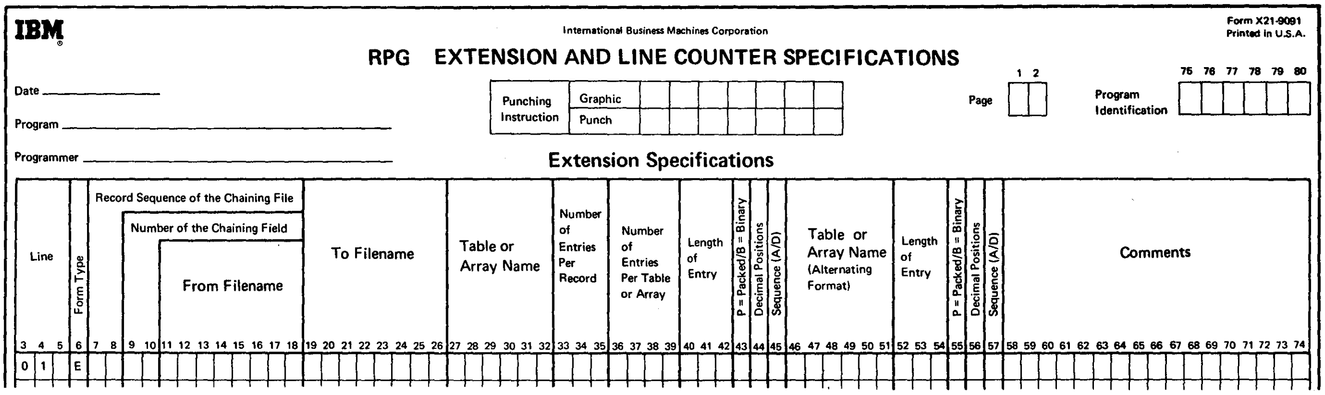
Punching Instruction	Graphic								
	Punch								

Page 1 2

Program Identification 75 76 77 78 79 80

Control Card Specifications

Line	Form Type	Core Size to Compile	Object Output	Listing Options	Core Size to Execute	Debug	MFCM Stacking Sequence	Sterling Input-Shillings	Input-Pence	Output-Shillings	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	H															



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								

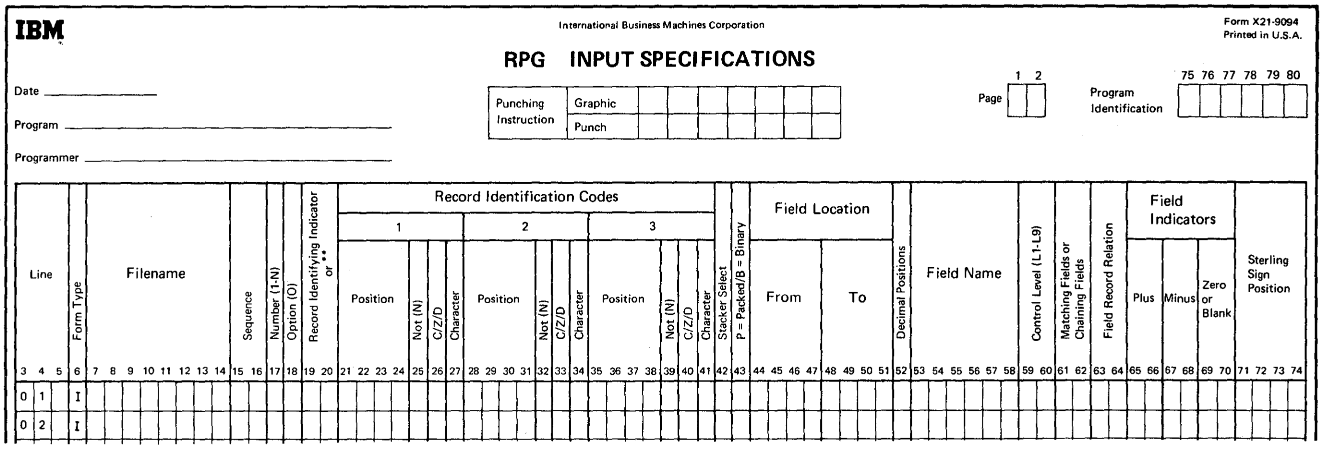
Page 1 2

Program Identification 75 76 77 78 79 80

Extension Specifications

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

Figure 33. Specification Forms (Part 1 of 2)



IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date ____

Program ____

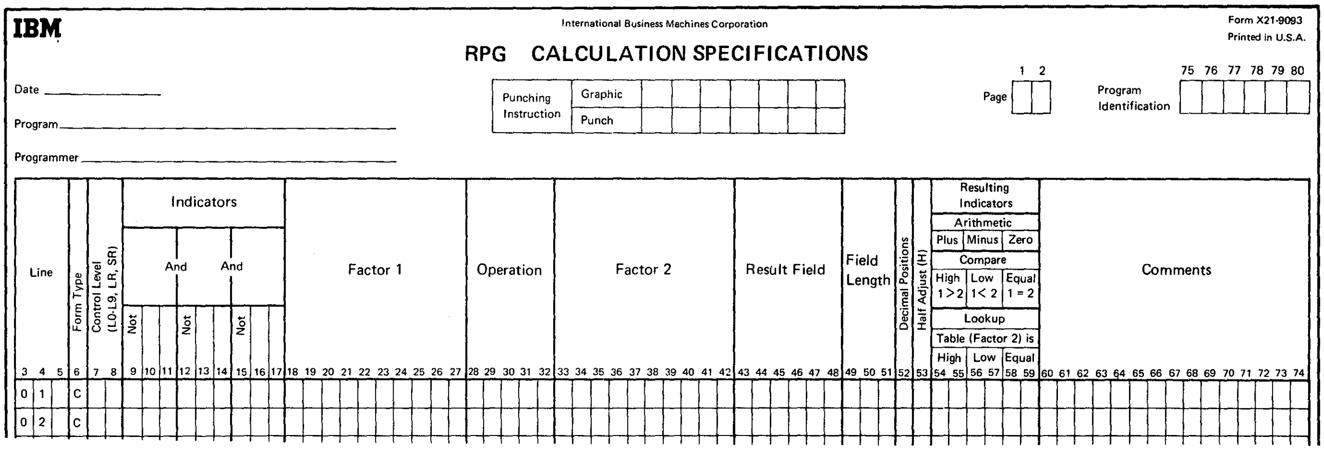
Programmer ____

Punching Instruction	Graphic	
	Punch	

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	Not (N)	C/Z/D	Character	2: Position	Not (N)	C/Z/D	Character	3: Position	Not (N)	C/Z/D	Character	Stacker Select	P = Packed/B = Binary	Field Location: From	To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank	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	I																														
0 2	I																														



IBM

International Business Machines Corporation

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

RPG CALCULATION SPECIFICATIONS

Date ____

Program ____

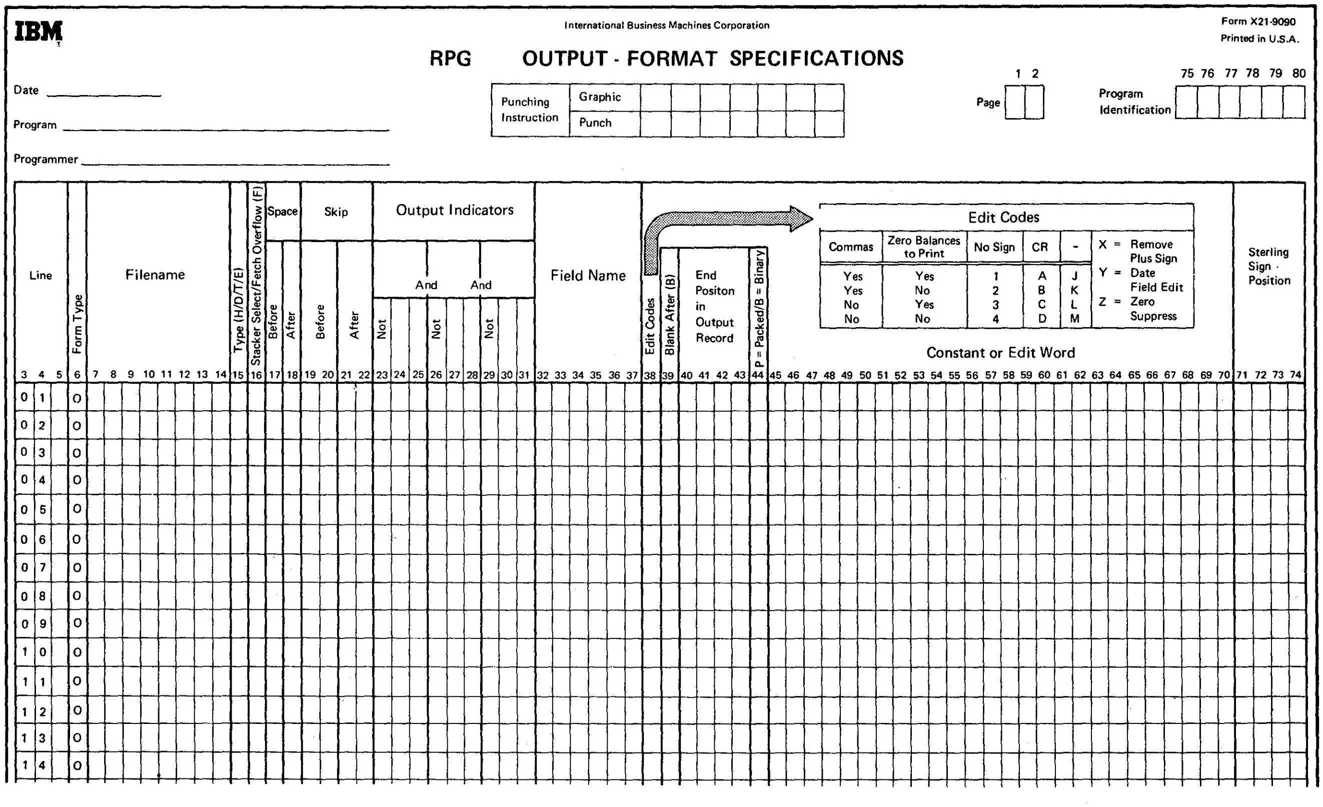
Programmer ____

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: Not		And: Not		And: Not		Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus; Compare High 1>2; Lookup Table (Factor 2) is High	Minus; Low 1< 2; Low	Zero; Equal 1 = 2; Equal	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	C																		
0 2	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 1 2

Program Identification 75 76 77 78 79 80

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/Fetch Overflow (F)	Space: Before	After	Skip: Before	After	Output Indicators: Not		And: Not		And: Not		Field Name	Edit Codes	Blank After (B)	End Positon in Output Record	P = Packed/B = Binary	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	O																				
0 2	O																				
0 3	O																				
0 4	O																				
0 5	O																				
0 6	O																				
0 7	O																				
0 8	O																				
0 9	O																				
1 0	O																				
1 1	O																				
1 2	O																				
1 3	O																				
1 4	O																				

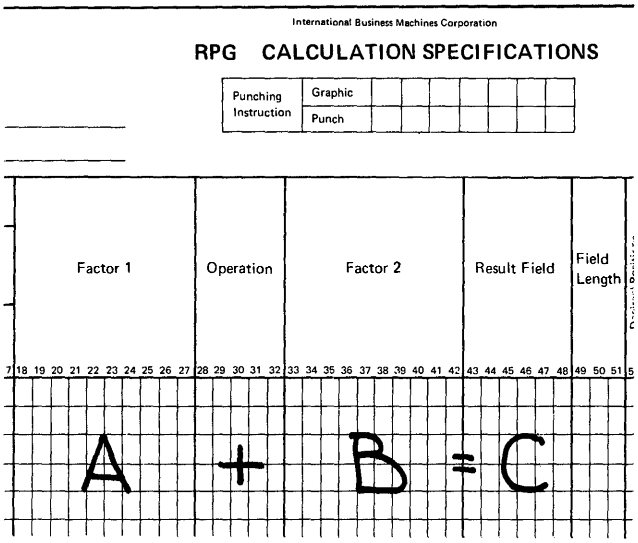
Figure 33. Specification Forms (Part 2 of 2)

Input Specification is used to:

1. Specify the file or files to be read into the system.
2. Identify the different types of records contained in each file.
3. Describe the location of the data fields in each record. See Figure 34.

Calculation Specification specifies the operations to be performed upon the input data and upon data obtained as the result of previous calculations.

Calculation specifications are graphically illustrated as follows:



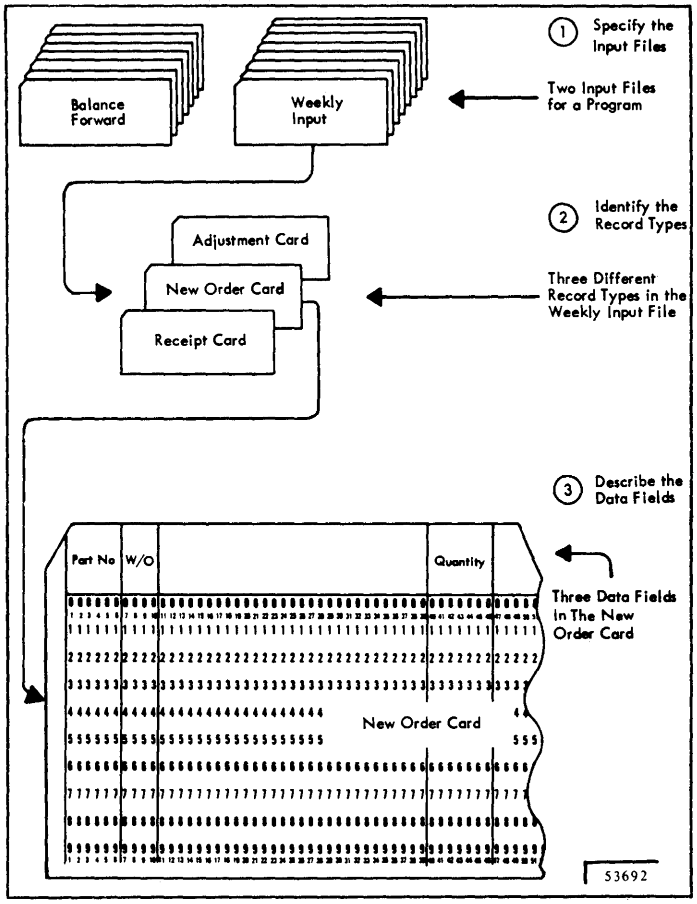
International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

Punching Instruction	Graphic							
	Punch							

Factor 1	Operation	Factor 2	Result Field	Field Length
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
A	+	B	= C	

To perform the operation A+B=C, the A field is specified in Factor 1, the kind of calculation to be performed in Operation, the B field in Factor 2, and the C field in Result Field.



Balance Forward
Weekly Input
1 Specify the Input Files
Two Input Files for a Program
Adjustment Card
New Order Card
Receipt Card
2 Identify the Record Types
Three Different Record Types in the Weekly Input File
3 Describe the Data Fields
Part No
W/O
Quantity
New Order Card
Three Data Fields In The New Order Card
53692

Figure 34. Function of Input Specifications

Output-Format Specification specifies:

1. The kind of output files to be produced (printed reports, summary records, etc.).
2. The location of the data fields in the output reports and records.

See Figure 35.

COMMON FIELDS

There are five entries that have the same function in all forms. These are described first.

Note: The numbers in parentheses indicate position.

Page (1-2)

This specification is located in the upper righthand corner of the forms. Each specification page of the source program may be numbered. The pages are numbered beginning with 01 for the Control Card and File Description Specifications form and continuing in the following sequence:

- Extension and Line Counter Specifications
- Input Specifications
- Calculation Specifications
- Output-Format Specifications

Line (3-5)

Each specification line may be identified by a line number. The first two digits of the line number are preprinted on the form. The third (position 5) is used when it becomes necessary to insert an additional line between two previously written lines. The line to be inserted is written following line 15. It is given an appropriate number (and subnumber in position 5).

The page number and line number have no direct effect on the program and need not be written. These positions are for the convenience of the programmer to indicate the proper order of the RPG source program cards. For example, the specification cards for a program could be placed in numeric sequence (if, for example, they were accidentally dropped or upset) by sorting or arranging them in sequence by page number and line number.

Form Type (6)

Each form has an appropriate type code preprinted in position 6. This code must be punched into all specification cards. The codes are:

- I -- Input specifications
- C -- Calculation specifications
- O -- Output-format specifications
- L -- Line counter specifications
- F -- File description specifications
- E -- Extension specifications
- H -- Control card specifications

Comments (7)

This feature enables the programmer to insert an identifying comment in the specification forms. This facility can be used, for example, to identify the end of one section of a program. These comments are written on the specification line, preceded by an asterisk (*) in position 7. During the generation of the object program, the asterisk in position 7 identifies the comment so that it is not considered a specification.

Program Identification (75-80)

This identification is located in the upper righthand corner of the forms. This entry identifies the specification cards for a particular program or for a specific section of a large program.

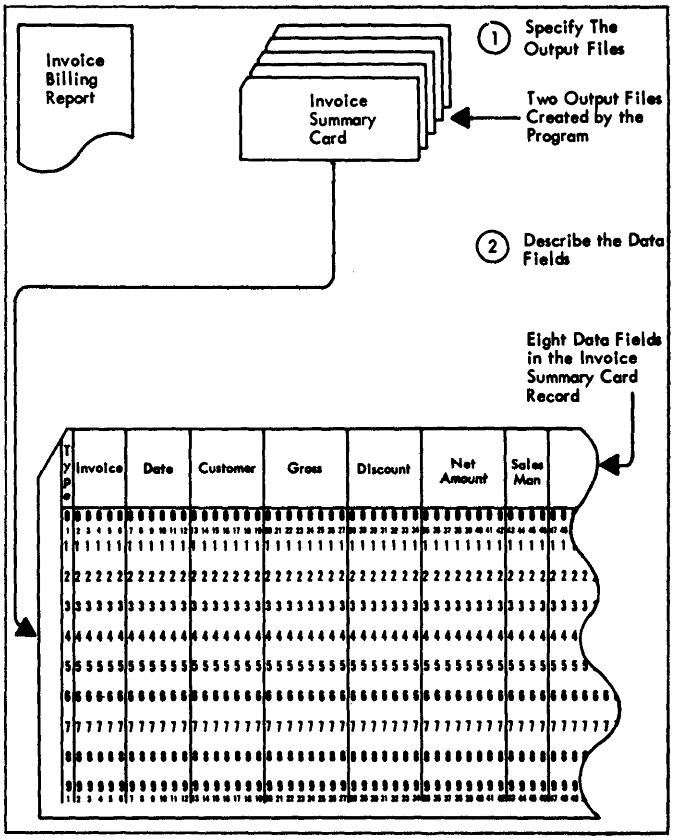

Invoice Billing Report
Invoice Summary Card
① Specify The Output Files
Two Output Files Created by the Program
② Describe the Data Fields
Eight Data Fields in the Invoice Summary Card Record
Type
Invoice
Date
Customer
Gross
Discount
Net Amount
Sales Man

Figure 35. Functions of Output-Format Specifications

53693

The RPG processor requires that the control card (header card) precede the user's source statements. This card must have an H punched in column 6. This card can also be used to provide information about the job and to describe the system to the RPG compiler. One line on the Control Card and File Description Specifications form is provided for coding the control record specifications (Figure 36).

Positions 1-5

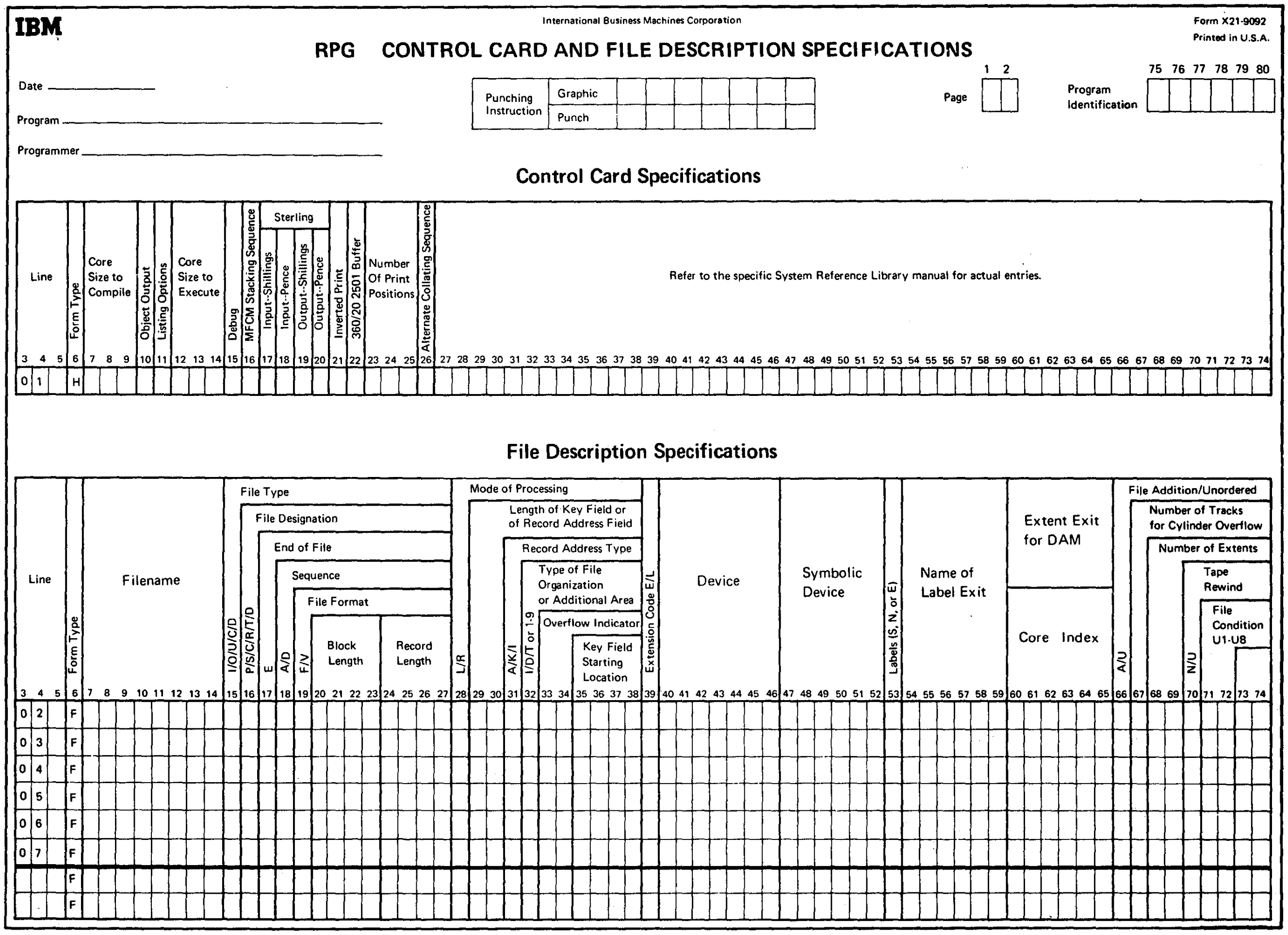
See *RPG Specification Forms, Common Fields.*

Position 6

This position must be punched with an H.

Positions 7-16

Positions 7-16 are not used.



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)

Program Identification (75 76 77 78 79 80)

Control Card Specifications

Line (3-5)	Form Type (6)	Core Size to Compile (7-9)	Object Output (10)	Listing Options (11)	Core Size to Execute (12-14)	Debug (15)	MFCM Stacking Sequence (16)	Sterling: Input-Shillings (17)	Sterling: Input-Pence (18)	Sterling: Output-Shillings (19)	Sterling: Output-Pence (20)	Inverted Print (21)	360/20 2501 Buffer (22)	Number Of Print Positions (23-25)	Alternate Collating Sequence (26)	27-74
0 1	H															Refer to the specific System Reference Library manual for actual entries.

File Description Specifications

Line (3-5)	Form Type (6)	Filename (7-14)	File Type: I/O/U/C/D (15)	File Designation: P/S/C/R/T/D (16)	End of File: E (17)	Sequence: A/D (18)	File Format: F/V (19)	Block Length (20-23)	Record Length (24-27)	Mode of Processing: L/R (28)	Length of Key Field or of Record Address Field (29-30)	Record Address Type: A/K/I (31)	Type of File Organization or Additional Area: I/D/T or 1-9 (32)	Overflow Indicator (33-34)	Key Field Starting Location (35-38)	Extension Code E/L (39)	Device (40-46)	Symbolic Device (47-52)	Labels (S, N, or E) (53)	Name of Label Exit (54-59)	Extent Exit for DAM / Core Index (60-65)	A/U (66)	File Addition/Unordered (67)	Number of Tracks for Cylinder Overflow (68-69)	Number of Extents: N/U (70)	Tape Rewind (71)	File Condition U1-U8 (72-73)	(74)
0 2	F																											
0 3	F																											
0 4	F																											
0 5	F																											
0 6	F																											
0 7	F																											
	F																											
	F																											

Figure 36. Control Card Specifications

Sterling 17-20

Positions 17-20 are used to describe the format of the sterling fields used in sterling currency. Blanks in these positions indicate that sterling currency is not being used. See *Sterling Routines for RPG* for more information.

Input-Shillings (17)

Entry	*Explanation*
0 or blank	Input shilling field is to be printed.
1	Input shilling field is to be in the IBM format.
2	Input shilling field is to be in the BSI format.

Input-Pence (18)

Entry	*Explanation*
1	Input pence field is in the IBM format.
2	Input pence field is in the BSI (British Standards Institution) format.

Output-Shillings (19)

Entry	*Explanation*
0 or blank	Output shilling field is to be printed.
1	Output shilling field is to be in the IBM format.
2	Output shilling field is to be in the BSI format.

The zero is allowed only for purposes of compatibility; it is treated the same as a blank.

Output-Pence (20)

Entry	*Explanation*
0 or blank	Output pence field is to be printed only.
1	Output pence field is to be in the IBM format.
2	Output pence field is to be in the BSI format.

The zero is allowed only for purposes of compatibility; it is treated the same as a blank.

Inverted Printed (21)

The entries in this position specify the domestic or European notation regarding UDATE format and the decimal point/comma use in numeric fields. A blank in position 21 specifies the domestic format of month/day/year and the decimal point notation. The D entry specifies the United Kingdom format of day/month/year and decimal point notation. The I entry specifies the European format of day.month.year and decimal comma notation.

Positions 22-25

Positions 22-25 are not used.

Alternate Collating Sequence (26)

Enter an A in this position if an external subroutine is used to translate the sequence of a matching field to the collating sequence of a System/360. If an external translating subroutine is not used, leave this position blank. The name of the external subroutine is predefined by RPG to be ALTSEQ.

Positions 27-74

Positions 27-74 are not used.

Program Identification (75-80)

If position 75 is blank, the name RPGOBJ is used for program identification.

The first four characters of the identification are punched in positions 73-76 of each card in the object program deck. Positions 77-80 of the object deck cards contain a sequence number.

This form (Figure 45) is used to provide information to RPG about:

1. The input files, defined on the Input Specifications form from which the object module obtains data records.

2. The input files used by the object program, such as record address files, table files, and chaining files.

3. The output files, defined on the Output-Format Specifications form, on which the object module writes data records. Each file used by the object program must be defined.

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 also identifies each file used in the program with the input or output unit with which it is associated. Each line of the form is used to specify one file.

Maximum Number of Files Available

The maximum number of files that can be used in the program is 10. The list that follows defines the maximum number of files for the various types of files that can be used. Any combination of these, up to 10, is permitted.

Type of File	*Maximum Number*
Input	–
Primary	1*
Secondary	8
RA File	1
Chained	9
Table	8
Output	9**
Update	–
Primary	1
Secondary	8
Chained	9
Combined	–
Primary	1
Secondary	1

*Each program must have one (and only one) primary file.
**Up to a maximum of eight printers can be used in a program.

Filename (7-14)

Each file used in the program is identified by writing the name of the file in positions 7-14. The filename must be left justified (that is, it must start in positon 7) and it must begin with an alphabetic character. The remaining characters of the name may be alphameric but must not contain special characters or embedded blanks. (Embedded blanks are blank positions falling between other characters of the name.) The filename may be eight characters or less. The filename entered in these positions must also have been entered on the Input Specifications form (for input data files), on the Output-Format Specifications form (for output data files), or on the Extension form (for Table, RA file, or chaining files). The filename must also be entered on a Line Counter Specification form if this feature is being used. The filename must be the same as the DDNAME on the DD card that describes the file.

If the program operates under the PCP option of the operating system and more than one table file to be loaded from the same card device, all of the table filenames must be the same. This is not true for the MFT or MVT options. The filename must be repeated in the Extension Specifications form for each table name.

File Type (15)

An entry in this position specifies the type of file defined on the line of the form. The following four entries are allowed in this position.

I (Input File)

I identifies the file as an input file (it may be a record address file, table file, or a file containing input data records).

O (Output File)

O identifies the file as an output file (it may be an updated table file to be written out).

U (Update File)

U identifies the file as an update file (direct access storage device only). An update file is both an input and an output file.

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. The over-all length of the record cannot be changed even in a variable length file.

A chained file may be updated at detail time or at total time. All other DASD files can be updated at detail time only.

C (Combined File)

C identifies the file as a combined file (card file only). A combined file is a file containing cards read into the system and cards used for punching. It must reside on an IBM 1442 Card Read-Punch.

Reading and then punching into the same file is accomplished in two different ways:

1. Punching into the same card that is read.

2. Punching into a blank trailer card in the same file.

D (Unused)

Figure 37 illustrates two examples of Filename and File Type specifications.

1. Detail cards are read into the system in one file, summary cards are punched into a second file (which, in this example, might be the punch feed of an IBM 2540 and might contain blank cards only), and the printed report is in a third file.

2. There are two input files in this example. The second file (INPUTSEC) also contains cards that will be used for punching output data; therefore, it is specified with file type C.

File Designation (16)

This specification is used to designate the type of input file defined on this line of the File Description Specification form. The five entries permitted in this position are listed here. Detailed explanations of chained files, table files, record address files, primary files, and secondary files may be found in the sections *Using Tables and Exit Routines in the Object Module* and *Processing Multiple Input Files.*

If the file is an output file, leave this position blank. An entry must be made if the file is an input, update, or combined file. Enter in position 16 the following codes:

Entry *Explanation*

P The file defined is a primary file. Only one primary file may be defined.

S The file defined is a secondary file. Secondary files are processed in the order in which they are specified on the File Description Specification form.

R The file defined is a record address file which relates to a direct access storage file. Only one RA file may be defined. A RA file is specified in the File Description and Extension Specification forms. Entries for RA files are not permitted on the input and output specifications.

C The file defined is a chained file.

T The file defined is a table file.

D Unused.

End of File (17)

Enter an E in position 17 if the input file is a sequential file and is to be checked for an end-of-file condition.

For one input file, the E entry will cause the LR (last record) indicator to be on when the last record of the file has been processed. For multiple input files that use chaining or matching records, the end-of-job condition (LR) will occur when all the input files have reached end of file.

If a matching records job has an E on the primary file and no E on the secondary files, any matched secondaries will be processed before the end of the job occurs.

If this position is left blank for all input files, the end-of-job condition (LR) occurs when all input files have been processed.

Note: An E should not be entered in position 17 (no chaining or matching records specified) if the file is processed randomly or if the file is either an output or table file.

Sequence (18)

This entry is normally made if there is more than one input file and the matching fields specification (positions 61-62 of the Input Specification form) is used. An entry in this position indicates whether the matching fields are in ascending or descending sequence.

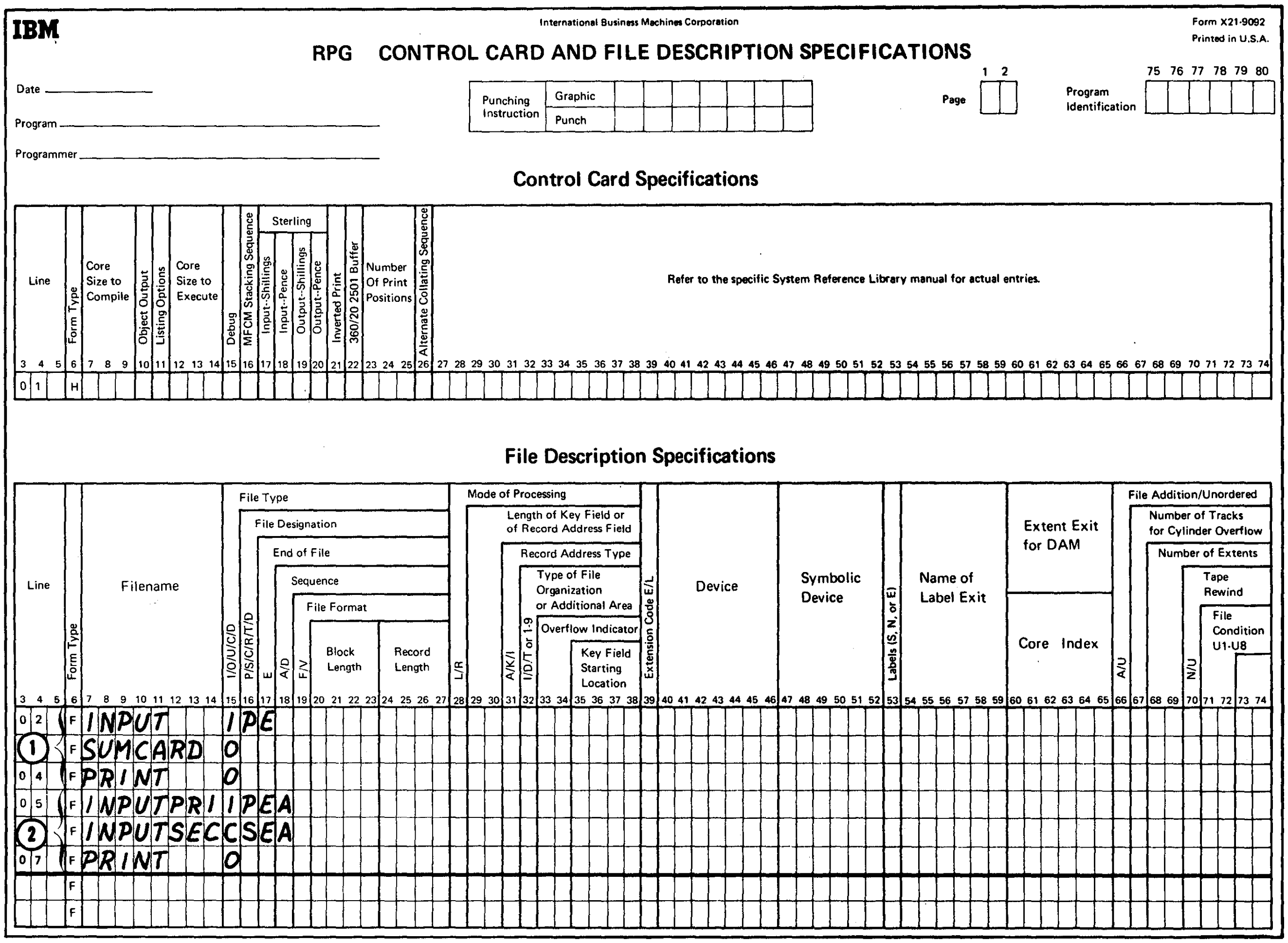
Enter an A in position 18 if the matching fields are in ascending order, or a D if the matching fields are in descending order.

Note: If this position is left blank when matching fields are used, ascending order is assumed.

Sequence Checking of Input Files

If there is only one input file or a chaining file in the program, this specification may be used to sequence check fields of the file to ensure that the file is in sequence. By entering the codes M1, M2, or M3 in positions 61-62 of the Input Specifications form, sequence checking with respect to these fields occurs. In this specification (position 18), either A or D must be entered to specify whether the file is ascending or descending.

The Halt Indicator, H0, is set on when a record of an input file is found to be out of sequence. Unless the H0 indicator is set off by a SETOF operation in the calculation specifications, the program will terminate before the next input record is read.



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

Program Identification

Control Card Specifications

Line	Form Type
01	H

Refer to the specific System Reference Library manual for actual entries.

File Description Specifications

Line	Form Type	Filename	File Type (15)	File Designation (16)	End of File (17)	Sequence (18)
02	F	INPUT	I	P	E	
①	F	SUMCARD	O			
04	F	PRINT	O			
05	F	INPUTPRI	I	P	E	A
②	F	INPUTSEC	C	S	E	A
07	F	PRINT	O			
	F					
	F					

Figure 37. Specifying File Name and File Type

File Format (19)

This position is used to indicate the format of the input or the output records. Enter an F in this position if the records are fixed-length. Enter a V in this position if the records are variable in length.

Enter an S for spanned, unblocked, variable-length records. Enter an M for spanned, blocked, variable-length records. S and M may only be used for sequential input and output files containing records equal to or less than 4000 bytes. If other than sequential files are specified, the entry is diagnosed and variable length records (V) are assumed. Also, a combined or update file (C or U in position 15) is diagnosed and variable length records (V) are assumed.

Block Length (Positions 20-23)

This specification is used to indicate the block length of the input or output records. This is the block length for the problem program. This block length should not be confused with the block length specified for system generation. It is used in conjunction with the next specification, Record Length.

The RPG program provides five techniques for handling records:

1. Fixed-length. All records have the same number of bytes of data.

2. Variable-length. Each record can have a different number of bytes of data.

3. Unblocked. There is only one logical record in each physical record.

4. Blocked. There are two or more logical records in one physical record. Blocked records are not supported by the direct access method.

5. Spanned variable-length (blocked and unblocked). One logical record may be segmented to span more than one block.

Fixed-Length, Unblocked: Each logical record is the same length as the physical record (Figure 38).

Fixed-Length, Blocked: All records and all blocks are of equal length (Figure 39).

Variable-Length, Unblocked: Each logical record corresponds to one physical record (Figure 40). The physical record contains two fields that are not included in the logical record: a block length field (BL) and a record length field (RL). These two fields are used by and created by the RPG program. The block length and record length fields are illustrated here only to show how the program uses and controls the records. These fields are ignored by the user when he establishes his data records and when he specifies record length and block length specifications.

Variable-Length, Blocked: One or more logical records (of variable length) are contained within each physical record (Figure 41).

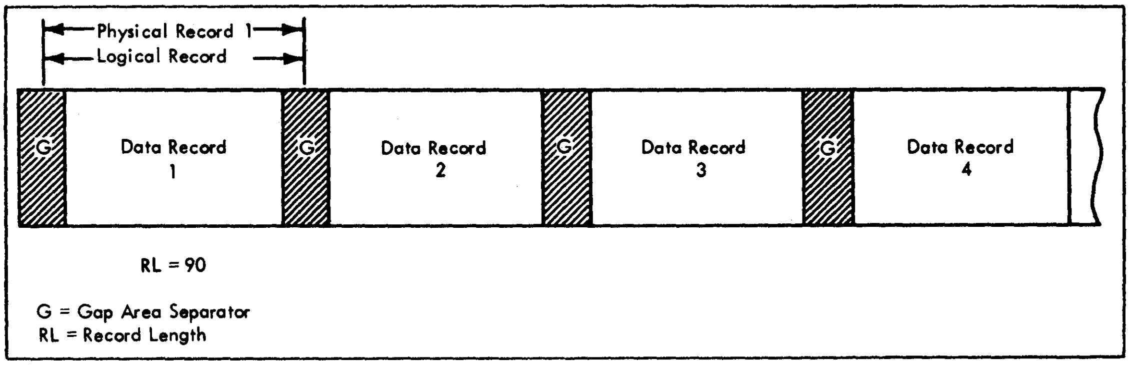


Figure 38. Fixed-Length, Unblocked Record Format

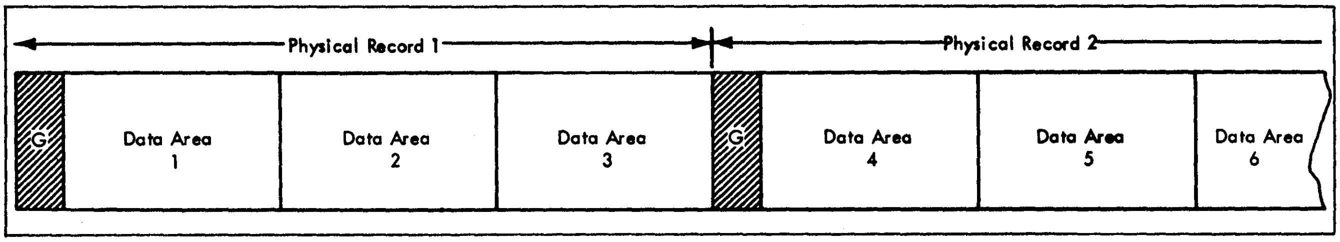


Figure 39. Fixed-Length, Blocked Record Format

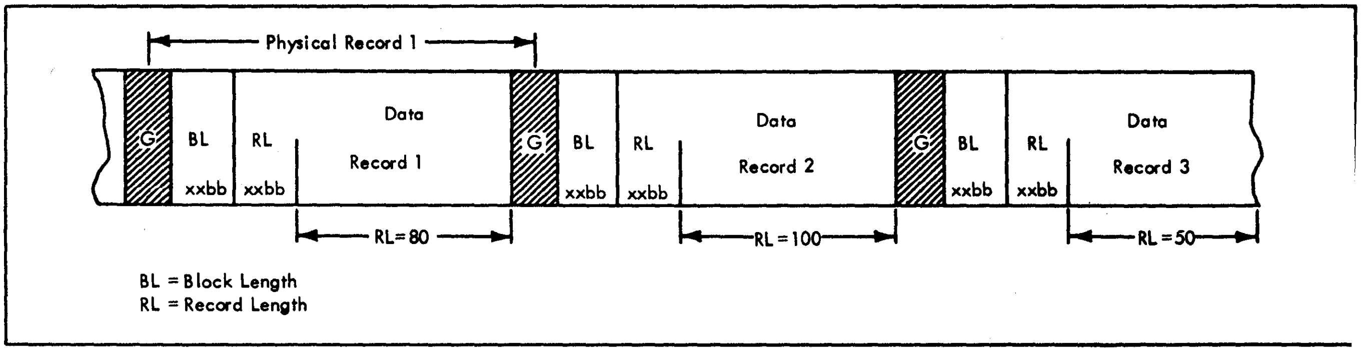


Figure 40. Variable-Length, Unblocked Record Format

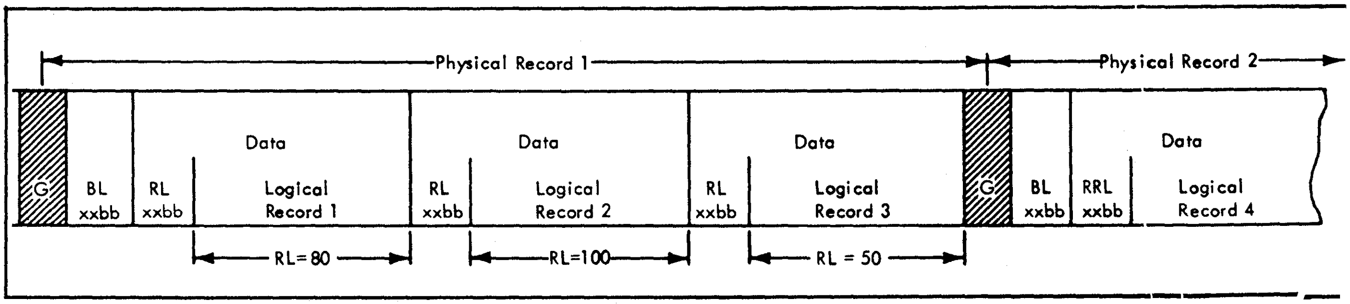


Figure 41. Variable-Length, Blocked Record Format

Spanned Variable-Length, Unblocked and Blocked: The spanning feature of OS used by RPG enables users to create and process variable length records that are larger than one physical block and/or to pack variable length records more densely within blocks (Figures 42 and 43). This is done by splitting the records into segments so that a record can be written into more than one block or a record segment may be used to fill out a block. Thus the block size of a spanned, variable length record may be set to the most suitable size for a particular device type or processing situation.

Current (nonspanned) variable length records are a subset of spanned variable-length records. This means that programs written to process spanned record data sets can process nonspanned record data sets as well.

Record Spanning

The following example shows the basic principles of record spanning. Note the differences between spanned, unblocked and spanned, blocked formats.

Suppose an RPG programmer creates a spanned, blocked, variable-length data set. The record length varies from 50 to 130 bytes, but the block size is specified as 100 bytes. The first five records:

1. RECORD1=70 (bytes)
2. RECORD2=110 (bytes)
3. RECORD3=130 (bytes)
4. RECORD4=50 (bytes)
5. RECORD5=60 (bytes)

are written as shown in Figure 43.

Notice that RECORD2 and RECORD3 are larger than the block size and must be spanned (broken into segments). Notice also that a block may contain any combination of records and/or record segments. Even a record smaller than a block, such as RECORD5, may be segmented to fill a block. However, a single block cannot contain multiple segments of the same record. Though a record is wholly contained on one block, it is treated the same as a record segment.

If the programmer decides to rewrite the data set in spanned unblocked form with the same block size of 100 bytes, RECORD1 through RECORD5 are written as shown in Figure 42. Notice that only records of greater length than the block size are segmented.

Block length and record length for spanned records are specified in the same manner as for variable-length records, except that the record length may be greater than the block length.

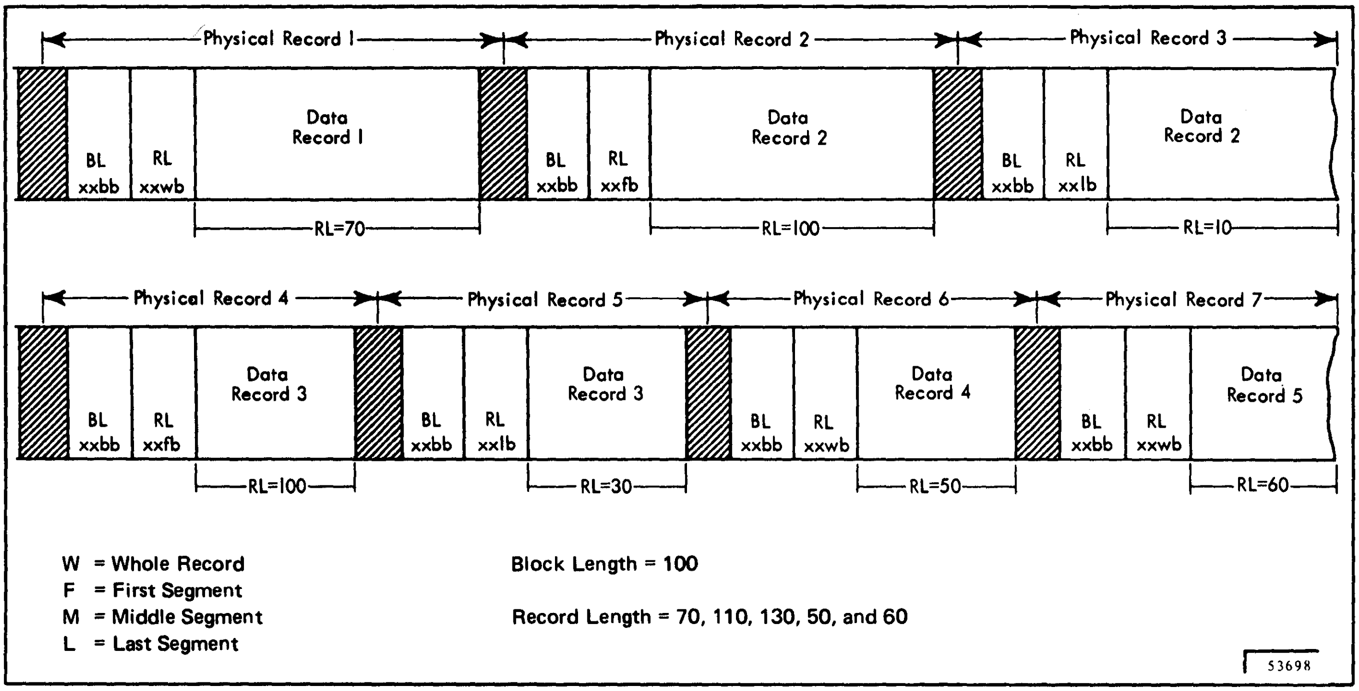


Figure 42. Spanned Variable-Length, Unblocked Record Format

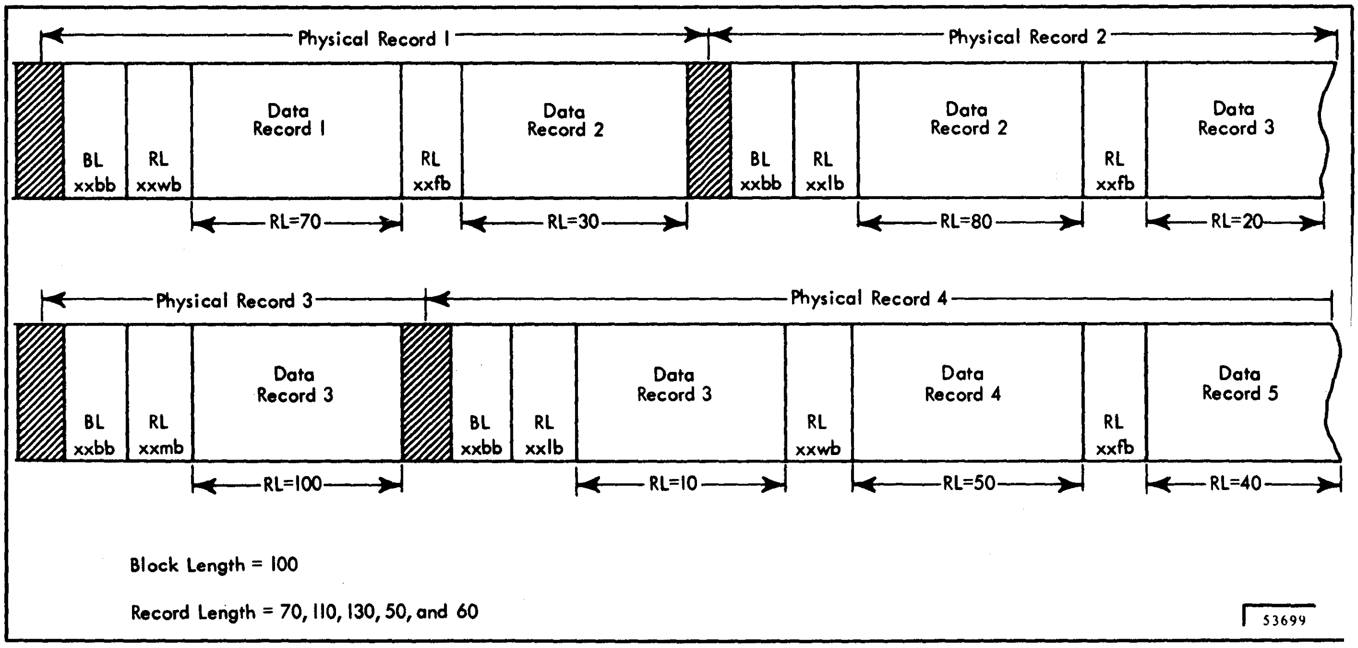


Figure 43. Processing Direct Access Storage Files

Specifications for Block Length

Unblocked

If the records are unblocked, the entry for this specification is the length of a record. If variable-length records are used, it is the length of the longest record. This means that the entry for this specification for unblocked records is always the same as the entry for the Record Length specification, except for spanned, unblocked, variable-length records. Spanned, unblocked, variable-length records allow the Record Length specification to be less than, greater than, or equal to the Block Length specification.

Blocked

If the records are blocked, the entry for the specification is the length of the largest block. For example, if three fixed-length 90-character records are contained in each block, the specification for Block Length is 270.

If there are variable-length records used in the file (Figure 41), the Block Length specification depends upon not only the number of variable-length records in each block, but also upon the maximum size of each variable length record. Spanned, blocked, variable-length records, however, allow the Record Length specification to be less than, greater than, or equal to the Block length specification.

If, in Figure 41, the three variable length records (in physical record 1) never exceed 80, 100, and 50 positions respectively, then the block length is 230.

The entry must be right justified. Leading zeros may be omitted.

For blocked, variable length records, RPG allocates buffer space, depending on the type of I/O device specified, by either of the following formulas:

1. For printer or punch I/O devices,

   $$\text{buffer size}=4+5\left(\frac{\text{block length}}{\text{record length}}+3\right)+\text{block length}$$

2. For all other I/O devices,

   $$\text{buffer size}=4+4\left(\frac{\text{block length}}{\text{record length}}+3\right)+\text{block length}$$

For spanned, unblocked, variable-length records, RPG allocates buffer space, depending on the type of I/O device specified, by either of the following formulas:

1. For printer, punch, or line counter data sets, buffer size = block length + 9

   Note: Records in data sets using control characters (for example, printer, punch, etc.) may not be segmented.

2. For all other data sets, buffer size = block length + 8

For spanned, blocked, variable-length records, RPG allocates buffer space using the same formula as blocked, variable-length records.

For direct access storage devices, it is the user's responsibility to enter a block length that will prevent the buffer size, as assigned by the compiler, from being larger than one track of the desired device.

The maximum block size that may be specified is 9999.

Record Length (24-27)

This specification is used to enter the length of the logical records contained in the file. If the file contains records that are variable in length, enter the length of the largest record. (The entry must be right justified.) The maximum record size that may be specified is 4000.

Note 1: If the block length equals the record length, RPG considers the file to have unblocked records for both fixed length and variable length records.

Note 2: If the block length is greater than the record length, RPG considers the file to have blocked records for both fixed length and variable length records.

Note 3: For spanned records (blocked and unblocked), the block length may be greater than, less than, or equal to the record length.

Mode of Processing (28) Nonsequential DASD Only

This specification is used to indicate the method or mode by which the file is processed. Acceptable entries are listed here.

Entry *Explanation*

L Enter an L in this position if a segment of the file is to be processed. The upper and lower limits of the file are supplied, in this case, by a record address file (RA file). The RA file is supplied by the user. This entry applies to indexed sequential files only.

R Enter an R in this position if the user's records are to be processed randomly. In this case, the records to be processed are obtained by a record address file or a chaining file. This entry applies to files with direct or indexed sequential organization.

Blank If no entry is made in this position for the file, the entire file is processed sequentially. This entry applies to indexed sequential files only.

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

If the file being defined is an indexed sequential file, the length of the key must be specified. The maximum length is 99 positions.

If the file being defined is a record address file, enter the number of positions that each entry in the RA file occupies.

For example, if a six-position part number field is used in the RA file, the entry is 6.

Record Address Type (31) Non-sequential DASD Only

If the records from the file are to be retrieved by using record keys, enter a K in this position. The K indicates that the file defined on this line will be processed using a record key.

If the records are to be retrieved by the record identification, enter an I in this position.

If the records are to be processed sequentially, this entry must be left blank. A is unused.

Type of File Organization (32) Non-sequential DASD Only

Enter an I if the file has indexed sequential organization. Enter a D in this position if the file has direct organization. T and 1-9 are unused.

Rules for DASD Specifications (Positions 28, 31, and 32)

The four rules listed below summarize the entries for the specifications *Mode of Processing, Record Address Type,* and *Type of File Organization.*

1. If a direct access storage device is not used in the system, positions 28, 31, and 32 are left blank.

2. If the type of file organization is sequential, then positions 28, 31, and 32 are again left blank.

3. If the type of file organization is indexed-sequential (I in position 32), then position 31 must contain a K and position 28 must contain either L, R, or blank.

4. If the type of file organization is direct (D in position 32), then position 31 must contain either a K or I, and position 28 must contain an R.

Figure 44 illustrates the code combinations possible for these three specifications.

Overflow Indicator (33-34)

If the file defined on the line is a printer file or an output file with an associated Line Counter Specifications form and overflow indicators are used, enter the overflow indicator associated with the file. A maximum of eight overflow indicators are allowed. The permissible overflow indicators are OA-OG, and OV.

Key Field Starting Location (35-38) DASD Only

This specification indicates the location of the key field within the data record. This specification is provided so that the key field may be located anywhere within the data record.

The entry for this specification is the starting position of the key field. For example, if the key field is in positions 112 through 116, the entry would be 112.

Type of File Organization (column 32)	Record Address Type (column 31)	Mode of Processing (column 28)
Sequential (blank)	Not applicable (blank)	The entire file (between limits) will be processed (blank).
Indexed-sequential (I)	Record key (K)	The entire file will be processed (blank).
		A segment of the file will be processed (L). The limits to be processed are supplied by a record address file (RA file).
		The records will be processed randomly (R). The address are supplied: (a) by an RA file, or (b) by the data contained in the chaining field of an input record.
Direct (D)	Track address with record key (K) or Track address with record identification (I)	The records will be processed randomly (R). The addresses to be converted are supplied by an RA file or by a chaining file. Conversion is required for RA file or chaining file.

Figure 44. Spanned Variable-Length, Blocked Record Format

The entry must be right justified. Leading zeros may be omitted.

This entry is required for indexed sequential files when the key is located in the data; it is blank for directly organized files. If this entry is omitted, fields within the key area for each record may be specified on either the Input or Output-Format Specifications form. Refer to the *Disk Storage Concepts* section for information concerning the relationship of this entry to others necessary for indexed sequential processing applications.

See the section *RPG User's Guide for ISAM Processing* for an explanation of key and record formats.

Note: Because of an operating system restriction, this entry must not be 1 for unblocked indexed sequential files.

Extension Code (39)

This specification is used to indicate to the RPG processor that additional information about the file is coded on the Extension or Line Counter Specifications form.

Enter an E in this position if the file defined on the line is a chaining file, table file, or record address file. These files always have additional specifications on the Extension Specifications form.

Enter an L if the Line Counter Specifications form is used for the output file described on this line. If line counter is specified, TAPE, DISK11, DISK14, CEL01, CEL0111, CEL0114, DADEVT, or PRINTER must be specified in Device (positions 40-46).

Device (40-46)

This specification relates a file to a specific type of input or output unit during program compilation time.

If the output file is a printer, enter PRINTER in positions 40-46. (File Format, position 19, should be V).

If the file is an input or output file and it is associated with a card reader or card punch unit, enter one of the following:

1. READ01 for IBM 2501 Card Reader
2. READ20 for IBM 2520 Card Read-Punch
3. READ40 for IBM 2540 Card Read-Punch
4. READ42 for IBM 1442 Card Read-Punch

If the file is an input or output file and it is associated with a tape unit, enter TAPE in these positions.

If the file is associated with a direct access storage device, enter DADEVT. RPG will also recognize the following as specific devices related to a file:

1. DISK11 for IBM 2311 Disk Storage Drive
2. DISK14 for IBM 2314 Direct Access Storage Facility
3. CEL01 for IBM 2321 Data Cell Drive with master and cylinder indices (MI/CI) on same device as data file
4. CEL0111 for IBM 2321 Data Cell Drive with MI/CI on an IBM 2311 Disk Storage Drive
5. CEL0114 for IBM 2321 Data Cell Drive with MI/CI on an IBM 2314 Direct Access Storage Facility

Symbolic Device (47-52)

Positions 47-52 are not used.

Entries for File Description Form

Figure 45 shows several examples of entries on the File Description Specifications form. The numbers to the right correspond to the numbered explanations that follow.

1. The P in position 16 indicates that input file INPUT is the primary file. The E in position 17 indicates that the end-of-job condition will occur when this file is depleted. The file is ascending (A in position 18). The block length is 80, and each record is 80 characters long. The E in position 39 indicates that the file will be referenced on the Extension form. This file is read in on an IBM 1442 Card Read-Punch. The device code is READ42.

2. The record address file defined on this line has a fixed format (position 19). It has a block length of 80. Each record is 80 characters long and the length of each record address field is 8. The E in position 39 indicates that the Extension form will be used. The record address file is read into the program by an IBM 2501 Card Reader. The device code is READ01.

3. The third file defined on this form is a table file. The T in position 16 indicates that it is a table file. It has a fixed format, a block length of 200, and a record length of 100. Position 39 (E) indicates that it will be further defined on the Extension form. The file is read in on magnetic tape; therefore the device code is TAPE.

4. MASTCUST is an input file that will be processed under the control of another file. It is a chained file (C in position 16). It will be processed randomly (R in position 28). The record addresses that will be referenced by the chaining file are record keys (K in position 31), and the file is organized indexed-sequentially (I in position 32). The key field begins in position 46 of the record and is 10 bytes long (positions 35-38, 29-30). This file is located on a direct access storage device and is given the device code of DADEVT.

5. The update file DISKUPDT (U in position 15) is used for input, and it is updated after processing of each record has been completed. It is an indexed-sequential file and is processed randomly. New records may be added within the existing file or to the beginning or end of the file (A in position 66). Note that Add(s) can be performed only in conjunction with random file processing (R in position 28).

   The output specifications must identify which record is to be added with ADD in positions 16-18. The C in position 16 indicates that the file is a chained file. In this example, the record length is 200 with a block length of 400. The key field begins in position 1 of the record.

   This file is located on a direct access storage device and is given the device code of DADEVT.

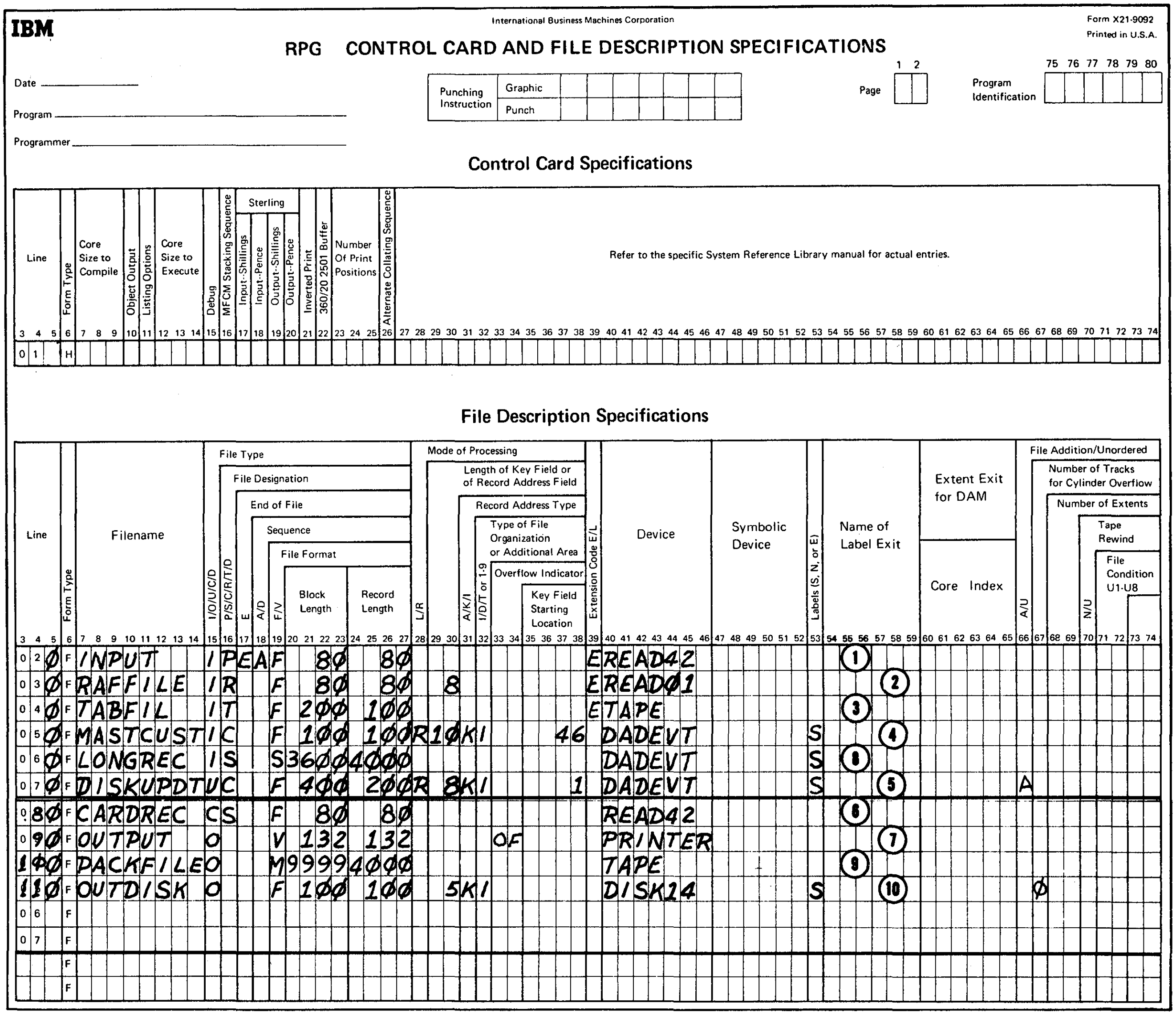
6. The file CARDREC is a combined file (C in position 15). Assume that the file is to be used as input and additional information is to be punched in the input cards during processing. It is a secondary file (S in position 16). This combined file is read in and punched out on an IBM 1442 Card Read-Punch. The device code is specified as READ42.

7. The file OUTPUT is a printed report in this example. It is variable in length; the longest record is 132 characters. The overflow indicator for this file is OF (positions 33-34). This printed report has a device code of PRINTER.

8. The secondary file LONGREC (S in position 16) is an input file (I in position 15). The records are to be processed as spanned records (S in position 19) with a record length of 4000 and a block length of 3600. The block size corresponds to the direct access device type track size. In this case the device code is given as DADEVT.

9. The output file PACKFILE (0 in position 15) is to be put on tape (device code of TAPE). The records are to be processed as spanned blocked with a record length of 4000 and a block length of 9999.

10. The output disk file (OUTDISK) is an indexed-sequential file which consists of unblocked records. No key starting location is specified, which means fields within the key area must be defined on the Output-Format Specifications form. (See the section *RPG User's Guide for ISAM Processing* for an explanation of key and record formats.) The file is located on an IBM 2314 Direct Access Storage Facility. The blank in position 66 indicates that a new file is being created. The zero in position 67 indicates that no tracks are to be used for cylinder overflow.



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) □□ Program Identification (75 76 77 78 79 80) □□□□□□

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)	Debug (15)	MFCM Stacking Sequence (16)	Sterling: Input-Shillings (17)	Input-Pence (18)	Output-Shillings (19)	Output-Pence (20)	Inverted Print (21)	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. (27–74)
0 1	H															

File Description Specifications

Line (3 4 5)	Form Type (6)	Filename (7–14)	File Type I/O/U/C/D (15)	File Designation P/S/C/R/T/D (16)	End of File E (17)	Sequence A/D (18)	File Format F/V (19)	Block Length (20–23)	Record Length (24–27)	Mode of Processing L/R (28)	Length of Key Field or of Record Address Field (29–30)	Record Address Type A/K/I (31)	Type of File Organization or Additional Area I/D/T or 1-9 (32)	Overflow Indicator (33–34)	Key Field Starting Location (35–38)	Extension Code E/L (39)	Device (40–46)	Symbolic Device (47–52)	Labels (S, N, or E) (53)	Name of Label Exit (54–59)	Extent Exit for DAM / Core Index (60–65)	File Addition/Unordered A/U (66)	Number of Tracks for Cylinder Overflow (67–68)	Number of Extents (68–69)	N/U (70)	Tape Rewind (71–72)	File Condition U1-U8 (73–74)
02Ø	F	INPUT	I	P	E	A	F	8Ø	8Ø							E	READ42			1							
03Ø	F	RAFFILE	I	R			F	8Ø	8Ø		8					E	READØ1			2							
04Ø	F	TABFIL	I	T			F	2ØØ	1ØØ							E	TAPE			3							
05Ø	F	MASTCUST	I	C			F	1ØØ	1ØØ	R	1Ø	K	I		46		DADEVT		S	4							
06Ø	F	LONGREC	I	S			S	36ØØ	4ØØØ								DADEVT		S	8							
07Ø	F	DISKUPDT	U	C			F	4ØØ	2ØØ	R	8	K	I		1		DADEVT		S	5		A					
08Ø	F	CARDREC	C	S			F	8Ø	8Ø								READ42			6							
09Ø	F	OUTPUT	O				V	132	132					OF			PRINTER			7							
1ΦØ	F	PACKFILE	O				M	9999	4ØØØ								TAPE			9							
11Ø	F	OUTDISK	O				F	1ØØ	1ØØ		5	K	I				DISK14		S	10			Ø				
06	F																										
07	F																										
	F																										
	F																										

Figure 45. Entries for File Description Form

Labels (53)

When label processing is used for a tape or disk, the program checks the labels on the input file to see if the correct file is on-line. The output files are checked and, if the label has expired, a new label is written.

If nonstandard labels are used for magnetic tape files, nonstandard label processing routines must be provided by the user and incorporated in the operating system at system generation time. (For details refer to *IBM System/360 Operating System, System Programmer's Guide.*) The appropriate nonstandard label processing routine is selected and executed by the operating system operating system.

If label processing is to be bypassed, this must be specified at system generation time in the PCP system. If this option is specified during execution and was not specified at system generation time, the system assumes that there are no labels. In the MVT or MFT systems, bypassing of label processing is specified in the PARM field of the reader cataloged procedure.

Label Options in RPG

The specification label (position 53) provides three options for label processing in the RPG program:

1. S = standard labels. Label processing is provided by the RPG program. No additional programming is required by the programmer.
2. E = standard labels followed by user-standard labels. RPG provides processing for the standard labels and then provides an automatic exit to a user subroutine for the processing of the user-standard labels. (See next specification *Name of Label Exit.*) This option is not available for indexed-sequential files.
3. blank = no labels. An entry of blank indicates no label processing is to be performed by the RPG program. This option is not available for DASD files.
4. N = unused.

Name of Label Exit (54-59)

This specification must contain the name of the routine written by the user to process user standard labels (E in position 53). The name can be either alphabetic or numeric, but the first character must be alphabetic. If the entry is shorter than six characters, it must be left justified.

Refer to *IBM System/360 Operating System Supervisor and Data Management Services,* Form C28-6646 for label exit register conventions.

Extent Exit for DAM--Core Index (60-65)

Positions 60-65 are not used.

File Addition (66)

This specification, A in position 66, is used only when new records are added to an indexed sequential file. This entry may be combined with I, O, or U File Type entries (File Description Specifications, position 15) only when random processing (R) is specified (File Description Specifications position 28). This position must be left blank if the file being defined is not an indexed sequential file or if no add function is desired. See Figure 46 for a summary of functions performed for the different combinations of entries in positions 15 and 66.

Number of Tracks for Cylinder Overflow (67)

For the indexed sequential load function, enter the number of tracks (between 0 and 8 for 2311; 0 and 9 for 2314/2321) per data cylinder that are to be used for overflow records. This information may also be supplied through the DD card DCB parameter at execution of the object program.

Note: The user may also choose whether cylinder overflow, independent overflow, or both are to be used when the DCB parameter is used on the file's DD card and no entry is made in position 67. See the section *RPG User's Guide for ISAM Processing* for a complete discussion of cylinder and independent overflow.

Number of Extents (68-69)

These positions are not used by OS RPG and must be left blank.

Tape Rewind (70)

This position is not used by OS RPG, but tape positioning may be controlled by appropriate parameters in the DD card.

File Condition (71-72)

Positions 71-72 are not used.

Position (73-74)

Positions 73-74 are not used.

File Description Specification Column 15	File Description Specification Column 66	Function
0	blank	Create a new file or extend existing file (LOAD).
0	A	Add new records to existing file (ADD). Valid with RANDOM file processing only.
I	blank	Process file.
I	A	Process file and add new records. Valid with RANDOM file processing only.
U	blank	Process file and update records.
U	A	Process file, update records, and add new records (ADDRTR). Valid with RANDOM file processing only.

Figure 46. Processing Functions for Indexed-Sequential Files

Entries made on the Extension Specifications form provide RPG with information, such as chaining files, tables used in the object module, and record address files. These functions are illustrated in Figure 47.

In the sections *Using Tables and Exit Routines in the Object Module* and *Processing Multiple Input Files,* detailed information and examples show how to use these functions.

The entries allowed on the Extension Specifications form are discussed in the following section.

Record Sequence of the Chaining Field (7-8)

This specification is used only for chaining files. The entry for this specification is the same entry that is made for the chaining file in Sequence (positions 15-16) on the Input Specifications form.

Number of the Chaining Field (9-10)

This specification is used only for chaining files. The entry for this specification is the identifying number of the chaining field (C1 through C9). This number is entered in Chaining Field (positions 61-62) on the Input Specifications form.

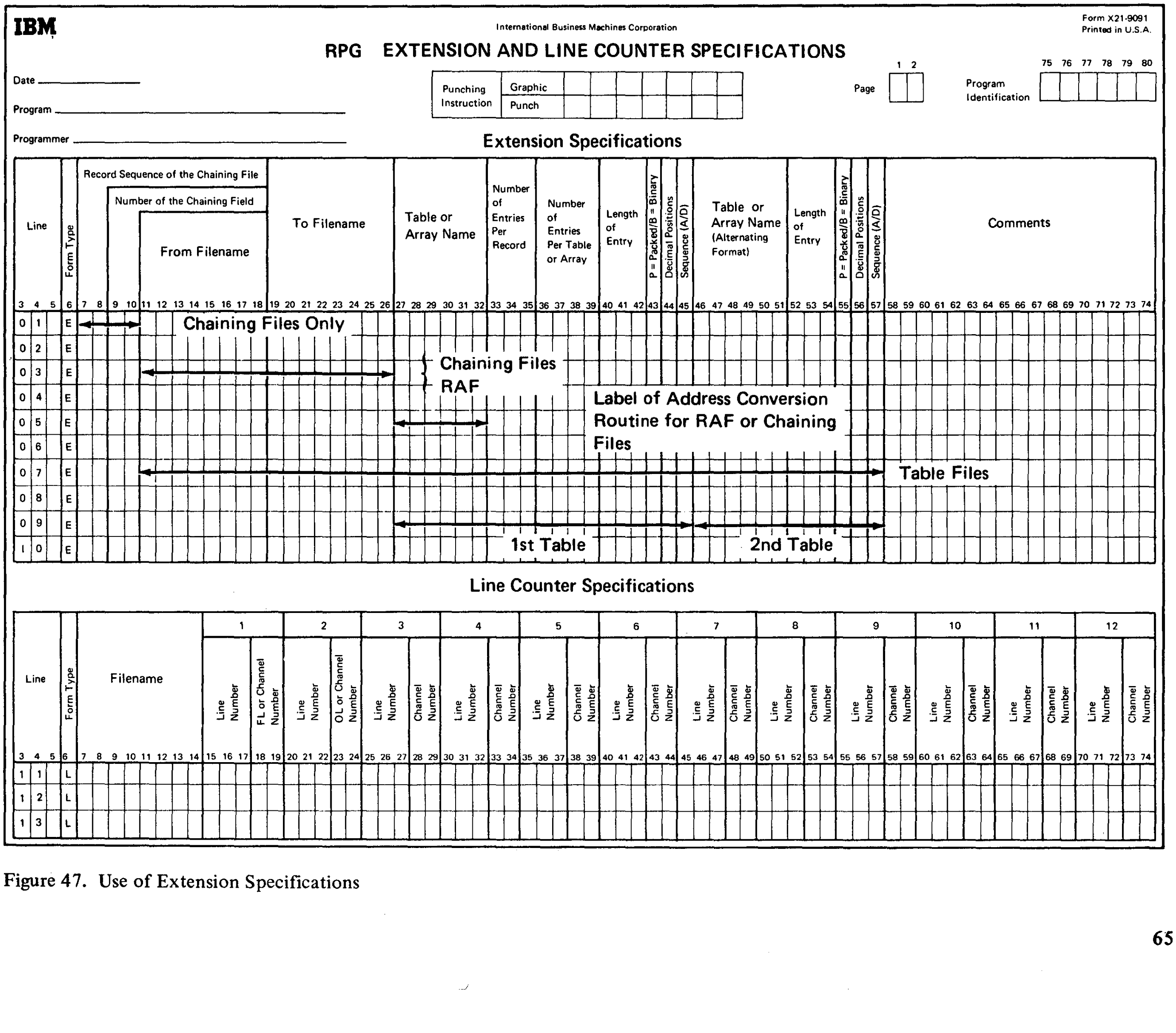


Figure 47. Use of Extension Specifications

From Filename (11-18)

This specification is used in conjunction with the next specification, To Filename (19-26). The purpose of these two specifications is to identify, for the RPG program, the relationship between two files. For example, they provide the name of a chaining file and the name of the file that is chained to it. Both the From Filename and To Filename are taken from Filename (positions 7-14) of the related entry from the File Description Specifications form.

Figure 48 illustrates the entries for those two specifications.

To Filename (19-26)

The entries for this specification are described in *From Filename* and in Figure 48.

Table or Array Name (27-32)

This specification and the remaining specifications on the form (positions 33-57) are used to describe table files. Table files are processed in the same order in which they appear on the Extension Specifications form. This specification is also used to specify the name of the address conversion routine for an RA file or chaining file.

A table file is composed of a table of arguments and a table of functions. If both the arguments and functions are entered on one input unit, the table file is known as an alternating table file. That is, the input record contains an argument field followed by a function field, followed by the next argument field, etc. In this case, the argument table is described in positions 27 through 45, and the function table is described in positions 46 through 57 on the same specification line.

If only an argument table is used, it is still described in positions 27-45; positions 46-57 are left blank.

It is possible to have the arguments contained in one input file and the functions in another input file. In this case, each file is described on a separate specification line in positions 27 through 45; positions 46-57 are left blank.

Note: It is not a requirement of the program that the arguments must be specified first. The function entries may be listed first. However, for the following specification descriptions, the manual assumes the argument entries are specified first.

Specifications for Table or Array Name

Arrays are not used in OS/360 RPG.

This specification contains the name of the argument table. The name must be in the form TABnnn. The entries nnn may be any alphameric characters.

When a file that has direct organization is processed under control of a record address file or a chaining file, the entry for these positions is the label of the user's conversion routine.

Figure 49 illustrates the extension specifications for a record address file and the master customer file it is used with.

The concepts of chaining and record address files have not been discussed at this point in the publication. For a complete discussion of chaining, address conversion, and record address files, see *Processing Multiple Input Files.*

Number of Entries Per Record (33-35)

Enter in positions 33-35 the maximum number of table entries (that is, arguments or functions) that are contained in each input record. The entry must be right justified.

Number of Entries Per Table or Array (36-39)

In these positions enter the exact number of table entries (arguments or functions) contained in the table. The entry must be right justified.

Note: The above two entries refer to tables, not to files. Thus, in alternating table files the entries are the total number of arguments or functions, not the sum of the two.

Length of Entry (40-42)

Enter in positions 40-42 the length of each table entry. The maximum size of a numeric entry is 15 characters, of an alphameric entry is 256 characters. The entry must be right justified.

Packed/Binary (43)

If the data for the table is in the packed decimal format, enter P in this position. Otherwise, leave this position blank. B is unused.

Decimal Positions (44)

If the data contained in the table is numeric, enter the number of decimal positions (1-9). Enter a zero if there are no decimal positions. If the data is alphameric, leave this position blank.

Sequence (45)

If the data contained in the table is in ascending sequence, enter an A in this position. If the data contained in the table is in descending sequence, enter a D in this position. Leave this position blank if the data contained in the table is not in ascending or descending sequence or if this specification is not required.

If a high or low LOKUP operation is performed, an entry must be made in this position.

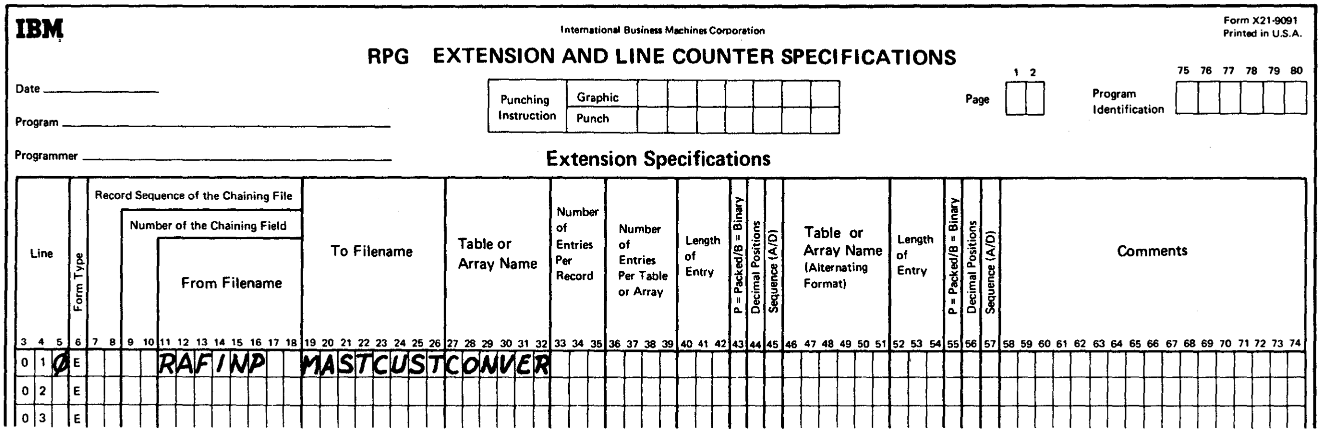
Note: The next four specifications (positions 46-57) are used only if alternating arguments and functions are read in on one input unit. The entries for these specifications are written on the same specification line as the entries in positions 33-45.

Table or Array Name (46-51)

If alternating arguments and function tables are used, enter the second table name in these positions. It must be of the form TABnnn. The entry must be left justified.

Type of File	*From Filename (11-18)	* To Filename (19-26)
Chaining file	Chaining filename. This is the file that has the data record containing the chaining field. The name of the file is taken from positions 7-14 of the file description specifications form.	Chained filename. This is the file from which the data record is obtained. The name of the file is taken from positions 7-14 of the file description specifications form for the chained file.
Record address file	The name of the record address file is entered in this specification.	The name of the file that contains the data record to be processed is entered in this specification.
Table files	If a table file is being defined (positions 27-57), enter the name of the file that contains the table.	If the table file being defined will be put out after it is updated, enter the name that was assigned to it on the output-format specifications form. If it is not being put out, leave blank.

Figure 48. From and to Filenames



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 (1 2) □□

Program Identification (75 76 77 78 79 80) □□□□□□

Extension Specifications

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

Figure 49. Extension Specifications, Conversion Routine Label

Length of Entry (52-54)

Enter in these positions the length of each table entry. The maximum size of a numeric entry is 15 characters; of an alphameric entry 256 characters. The entry must be right justified.

Packed/Binary (55)

Enter a P if the data for the table is in the packed decimal format. Otherwise, leave this position blank. B is unused.

Decimal Positions (56)

If the data contained in the table is numeric, enter the number of decimal positions (1-9). Enter a zero if there are no decimal positions. If the data is alphameric, leave this position blank.

Sequence (57)

If the data contained in the table is in ascending sequence, enter an A in this position. If the data contained in the table is in descending sequence, enter a D in this position. Leave this position blank if the data contained in the table is not in ascending or descending sequence or if this specification is not required.

Comments (58-74)

Leave positions 58-74 blank, unless comments are entered in these positions.

ENTRIES ON THE EXTENSION FORM

Figure 50 shows several examples of entries that can be made on the Extension Specifications form. The numbers to the right of the entries correspond to these explanations:

1. In this example, INPUT is a card file containing the record key that will be used to process records in the DASD file MASTCUST. The file INPUT is the chaining file. That is, it is the file that links or chains to another file (in this case MASTCUST). The field contained in INPUT, which is used to link the two files, is the chaining field.

   The record sequence of the input file is taken from the Input Specifications form. C1 is the number of the chaining field. Thus, INPUT is chained to MASTCUST by using a field defined on the input specifications that contains C1 in positions 61-62. A complete discussion of chaining may be found in *Processing Multiple Input Files, Chaining.*

2. In this example, RAFFILE is a record address file that supplied the addresses of the records to be processed in the file DISKUPDT.

3. TABFIL is the name of a table file that contains both a table of arguments and a table of functions.

   The arguments in the file are identified by the table name of TABARG. The argument table is described in positions 33-45. There are 10 arguments in each record. The number of arguments in the table is 150 and each argument is 10 characters long. The arguments are numeric and there are no decimal positions, thus the entry is 0 (position 44).

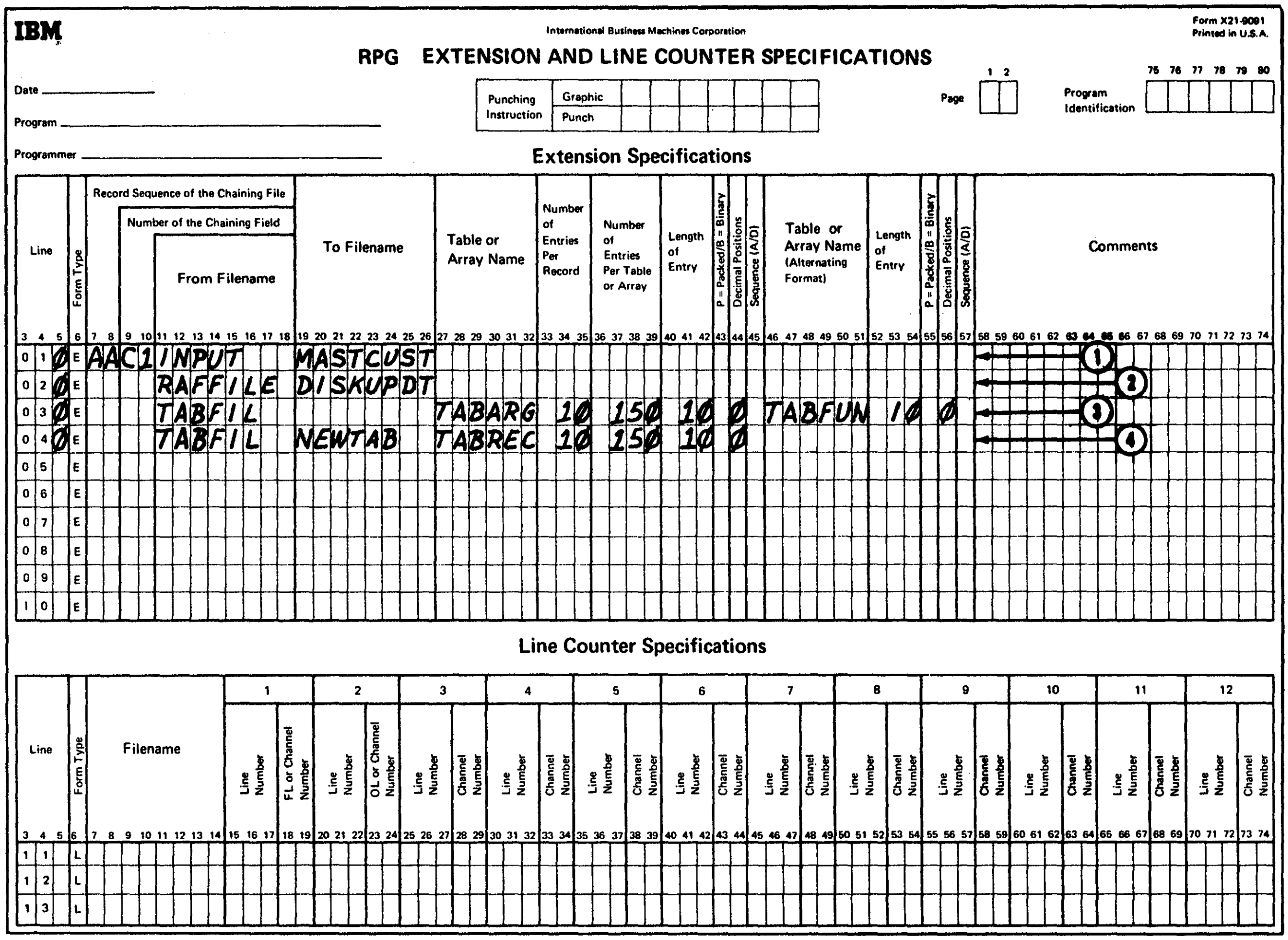
   The functions in the file are identified by the table name TABFUN. The function table is described in positions 52-57. Each function is 10 characters long and each function is organized in the form: argument-function. Therefore, TABARG is specified first.

4. This example shows the specifications for a table file that contains only arguments. After the table of arguments is updated, the table is to be written on an output unit.

   TABFIL is the name of the table file.

   NEWTAB is the name given to the table file when it is being written on the output unit (output operations).

   The arguments in the file are identified by the table name of TABREC. The argument table is described in positions 33-45. Ten table entries are in each record. The number of table entries in the table is 150, and each table entry is 10 characters long. The data is numeric, but there are no decimal positions, thus the entry is 0. Positions 46 through 57 are left blank.



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 (1 2) □□

Program Identification (75 76 77 78 79 80) □□□□□□

Extension Specifications

Line (3-5)	Form Type (6)	Record Sequence of the Chaining File (7-8)	Number of the Chaining Field (9-10)	From Filename (11-18)	To Filename (19-26)	Table or Array Name (27-32)	Number of Entries Per Record (33-35)	Number of Entries Per Table or Array (36-39)	Length of Entry (40-42)	P = Packed/B = Binary (43)	Decimal Positions (44)	Sequence (A/D) (45)	Table or Array Name (Alternating Format) (46-51)	Length of Entry (52-54)	P = Packed/B = Binary (55)	Decimal Positions (56)	Sequence (A/D) (57)	Comments (58-74)
01Ø	E	AA	C1	INPUT	MASTCUST													1
02Ø	E			RAFFILE	DISKUPDT													2
03Ø	E			TABFIL		TABARG	1Ø	15Ø	1Ø		Ø		TABFUN	1Ø		Ø		3
04Ø	E			TABFIL	NEWTAB	TABREC	1Ø	15Ø	1Ø		Ø							4
05	E																	
06	E																	
07	E																	
08	E																	
09	E																	
10	E																	

Line Counter Specifications

Line (3-5)	Form Type (6)	Filename (7-14)	1 Line Number (15-17)	1 FL or Channel Number (18-19)	2 Line Number (20-22)	2 OL or Channel Number (23-24)	3 Line Number (25-27)	3 Channel Number (28-29)	4 Line Number (30-32)	4 Channel Number (33-34)	5 Line Number (35-37)	5 Channel Number (38-39)	6 Line Number (40-42)	6 Channel Number (43-44)	7 Line Number (45-47)	7 Channel Number (48-49)	8 Line Number (50-52)	8 Channel Number (53-54)	9 Line Number (55-57)	9 Channel Number (58-59)	10 Line Number (60-62)	10 Channel Number (63-64)	11 Line Number (65-67)	11 Channel Number (68-69)	12 Line Number (70-72)	12 Channel Number (73-74)
11	L																									
12	L																									
13	L																									

Figure 50. Entries for the Extension Form

The Line Counter Specifications form provides the ability to store formatted reports, which will be printed later, on an intermediate device such as tape, disk, or data cell. This ability is required if the System/360 Operating System you are working with is MFT, MVT, or PCP.

The carriage control tape on the printer must be punched to agree with the spacing on the Line Counter Specifications form. If this is not done, erratic and unpredictable spacing or skipping can occur.

The line number associated with channel 12 on the Line Counter Specifications form causes the setting of the overflow indicator (if you do not skip to channel 12) as specified in positions 33-34 on the File Description Specifications form. When the line counter has reached the specified line number, the overflow indicator is set. Depending upon the absence or presence of overflow indicators on the File Description and Output-Format Specifications forms, one of the following actions will result for line counter files:

File Description (positions 33-34)	*Output-Format (positions 23-31)*	*Action*
No entry	No entry	Automatic skip to next page at overflow time
No entry	Entry	Error
Entry	No entry	Continues printing (user controls overflow)
Entry	Entry	Performs normal overflow

A standard IBM System/360 write/control or independent carriage control character is added as the first byte of each record that is put out on the intermediate device. For example, a 132-character record becomes 133 characters long. All space before and skip before controls are written out as an independent carriage control character in a separate record that does not print. Space after and skip after controls are written out as modifier bits in conjunction with the write command for the line. A line that has space or skip before control produces two records on the intermediate device; a line with skip or space after control produces only one record. An auxiliary program must be used to retrieve these records for printing.

HOW TO USE LINE COUNTER SPECIFICATIONS

Line counter specifications are written on the Extension and Line Counter Specifications form (Figure 51).

Positions 1-6

See *RPG Specification Forms, Common Fields.*

Filename (7-14)

Positions 7-14 indicate the output file to be used. This is the same filename used on the File Description Specifications form and must apply to the device used.

Line Number and FL or Channel Number (15-19)

These positions are used to relate the line on the output form to a channel punch in the carriage tape. The line number is coded in positions 15-17 and the channel number in positions 18-19. Channels 1 and 12 must be specified on the Line Counter Specifications form.

Positions 20-74

Each set of five positions following position 19 is used in the same way as positions 15-19.

Example: The program in Figure 52 shows how line counter specifications are used to relate the line of the printed page of a report to its corresponding channel punch in the carriage control tape. On the Line Counter Specifications form, channel 01 (positions 18-19) corresponds to line 006 (positions 15-17), and channel 12 (overflow) corresponds to line 059.

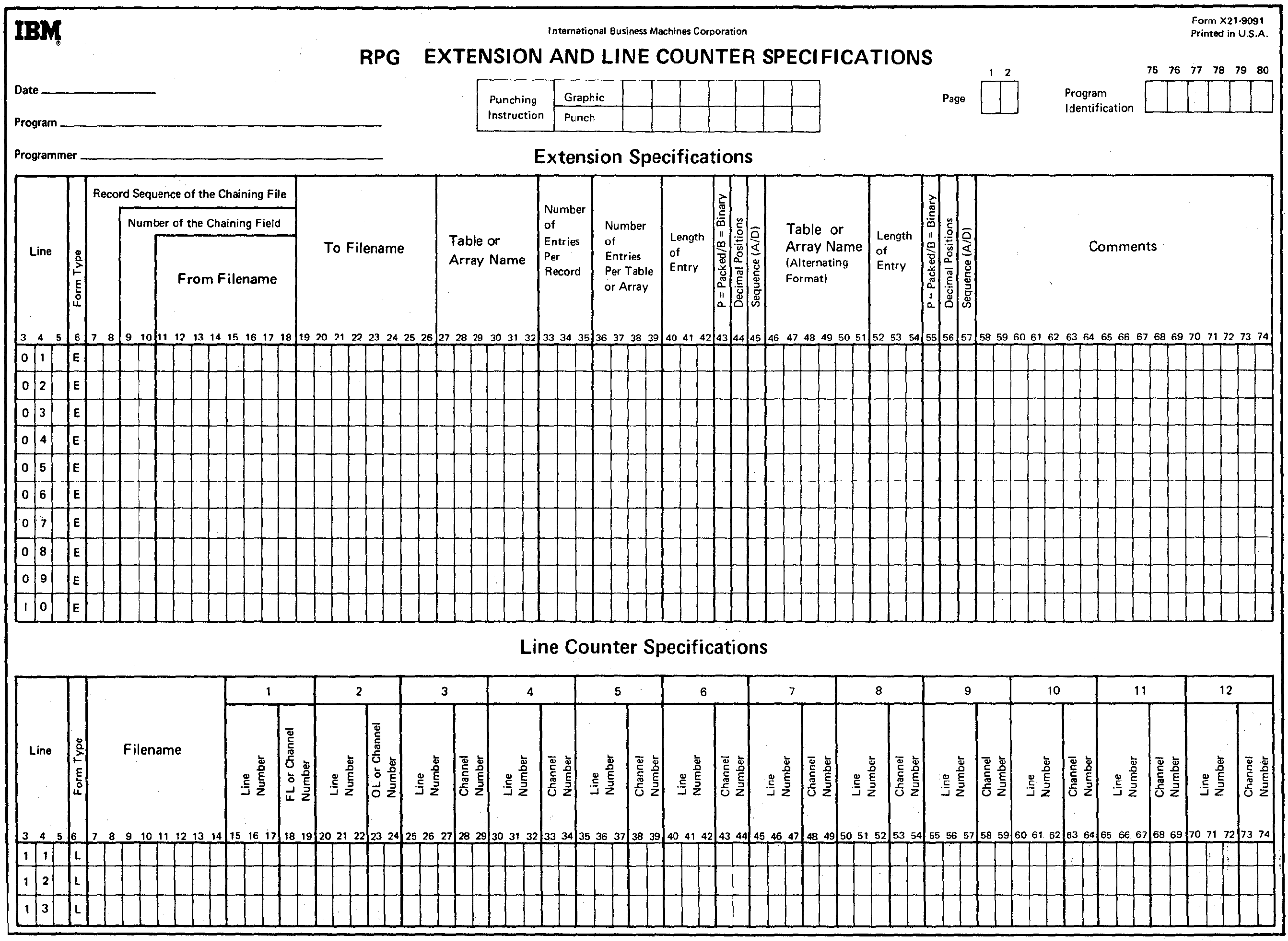
When this program is executed, the internal line count is set at 006. As each record is written on the output device, the line count is increased by one. When the line count equals 059, the overflow indicator (OF) is set on, overflow output occurs, and a record is written on the output device with an associated skip to channel 01. Whenever a skip to channel 01 occurs, the internal line count is reset to 006.

One output record is created for every line printed. This record also includes the entry for any space or skip after requirements. An output record is also created for each specification of space or skip before.

The output filename specified in positions 7-14 of the Line Counter Specifications form is the same file specified on the File Description Specifications form (positions 7-14). This filename must apply to the intermediate device used. The intermediate device used for this program is a disk (positions 40-46 on the File Descriptions Specifications form).

Figure 53 shows part of the output disk records for this file and the printer output lines created from this disk. On the first page of the printed report, the first two members of Department 851 are printed; then overflow occurs. A heading line is printed on the next page, and then the rest of Department 851 is printed.

Figure 54 shows the coding necessary to print the line counter tape prepared from Figure 52 by use of the Disk to Print Utility program. With System/360 Operating System the "RECORD FIELD=" parameter must be 121 or less, including the control character. For further information on utility programs consult *IBM System/360 Operating System Utilities,* Form C28-6586.



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 (1 2)

Program Identification (75 76 77 78 79 80)

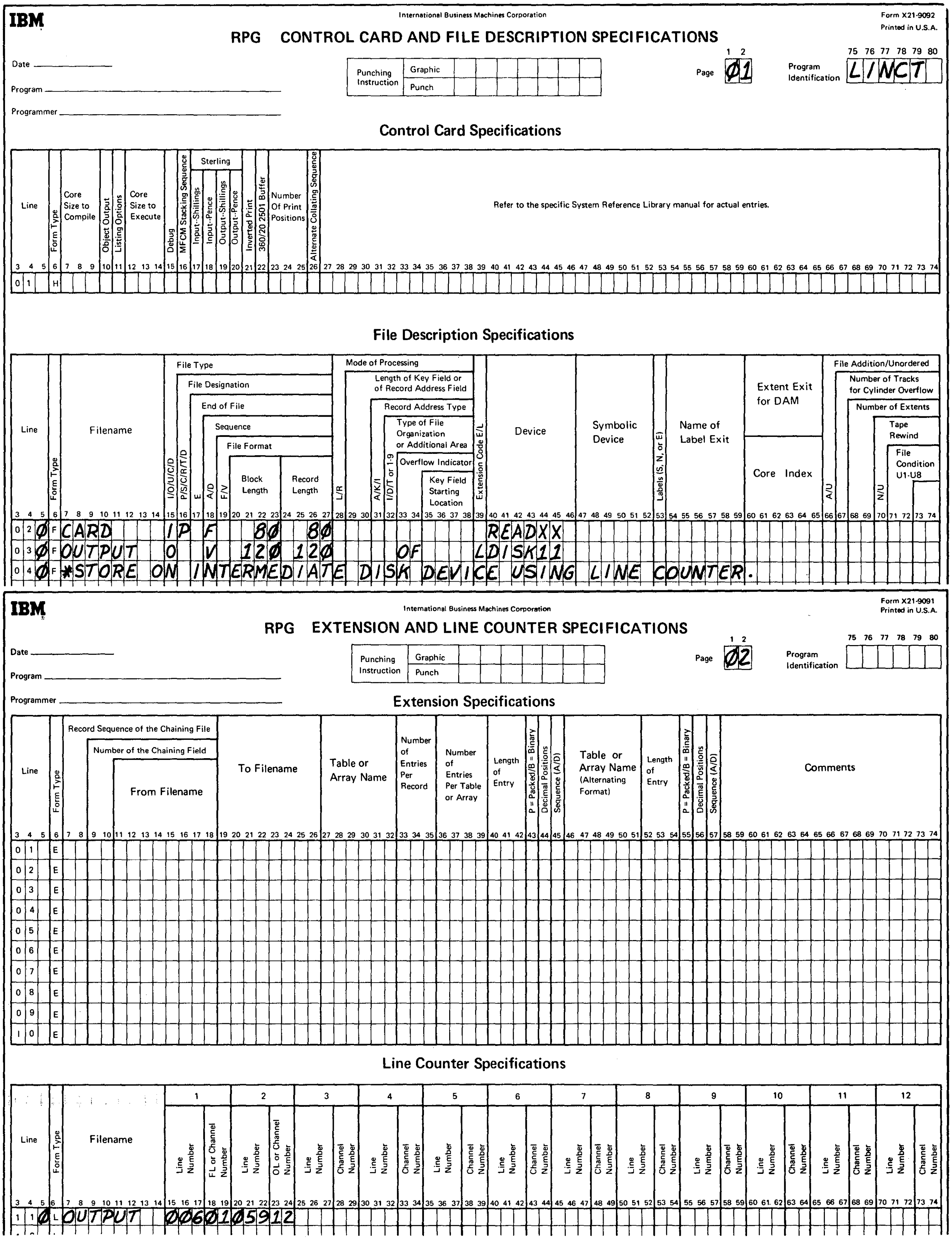
Extension Specifications

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

Line Counter Specifications

			1		2		3		4		5		6		7		8		9		10		11		12	
Line	Form Type	Filename	Line Number	FL or Channel Number	Line Number	OL or Channel Number	Line Number	Channel Number	Line Number	Channel Number	Line Number	Channel Number	Line Number	Channel Number	Line Number	Channel Number	Line Number	Channel Number	Line Number	Channel Number	Line Number	Channel Number	Line Number	Channel Number	Line Number	Channel Number
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
1 1	L																									
1 2	L																									
1 3	L																									

Figure 51. Extension and Line Counter 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 (1 2): Ø1 Program Identification (75–80): LINCT

Control Card Specifications

Line (3–5)	Form Type (6)	Core Size to Compile (7–9)	Object Output (10)	Listing Options (11)	Core Size to Execute (12–14)	Debug (15)	MFCM Stacking Sequence (16)	Sterling (17–20)	Inverted Print (21)	360/20 2501 Buffer (22)	Number Of Print Positions (23–25)	Alternate Collating Sequence (26)	27–74
01	H												Refer to the specific System Reference Library manual for actual entries.

File Description Specifications

Line (3–5)	Form Type (6)	Filename (7–14)	File Type I/O/U/C/D (15)	File Designation P/S/C/R/T/D (16)	End of File E (17)	Sequence A/D (18)	File Format F/V (19)	Block Length (20–23)	Record Length (24–27)	Mode of Processing / Key Field / Record Address Type / Organization (28–32)	Overflow Indicator (33–34)	Key Field Starting Location (35–38)	Extension Code E/L (39)	Device (40–46)	Symbolic Device (47–52)	Labels (53)	Name of Label Exit (54–59)	Extent Exit for DAM / Core Index (60–65)	A/U (66)	File Addition/Unordered (67–74)
02Ø	F	CARD	I	P			F	8Ø	8Ø					READXX						
03Ø	F	OUTPUT	O				V	12Ø	12Ø		OF		L	DISK11						

04Ø F *STORE ON INTERMEDIATE DISK DEVICE USING LINE COUNTER.

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 (1 2): Ø2 Program Identification (75–80): ____

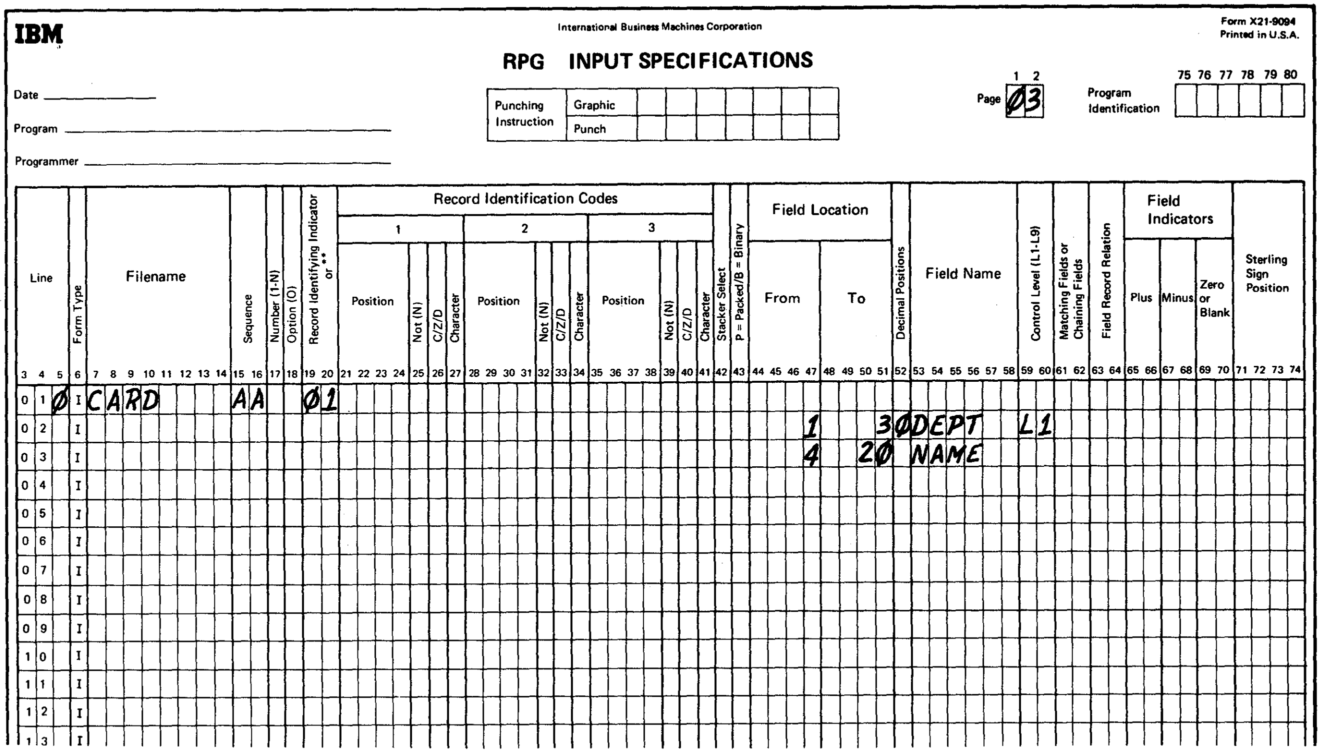
Extension Specifications

Line	Form Type	Record Sequence of the Chaining File / Number of the Chaining Field / From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P = Packed/B = Binary	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P = Packed/B = Binary	Decimal Positions	Sequence (A/D)	Comments
01	E															
02	E															
03	E															
04	E															
05	E															
06	E															
07	E															
08	E															
09	E															
10	E															

Line Counter Specifications

Line (3–5)	Form Type (6)	Filename (7–14)	1 Line Number (15–17)	1 FL or Channel Number (18–19)	2 Line Number (20–22)	2 OL or Channel Number (23–24)	3 Line Number	3 Channel Number	4 Line Number	4 Channel Number	5 Line Number	5 Channel Number	6 Line Number	6 Channel Number	7 Line Number	7 Channel Number	8 Line Number	8 Channel Number	9 Line Number	9 Channel Number	10 Line Number	10 Channel Number	11 Line Number	11 Channel Number	12 Line Number	12 Channel Number
11Ø	L	OUTPUT	ØØ6	Ø1	Ø59	12																				

Figure 52. Using Line Counter Specifications (Part 1 of 2)



IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date

Program

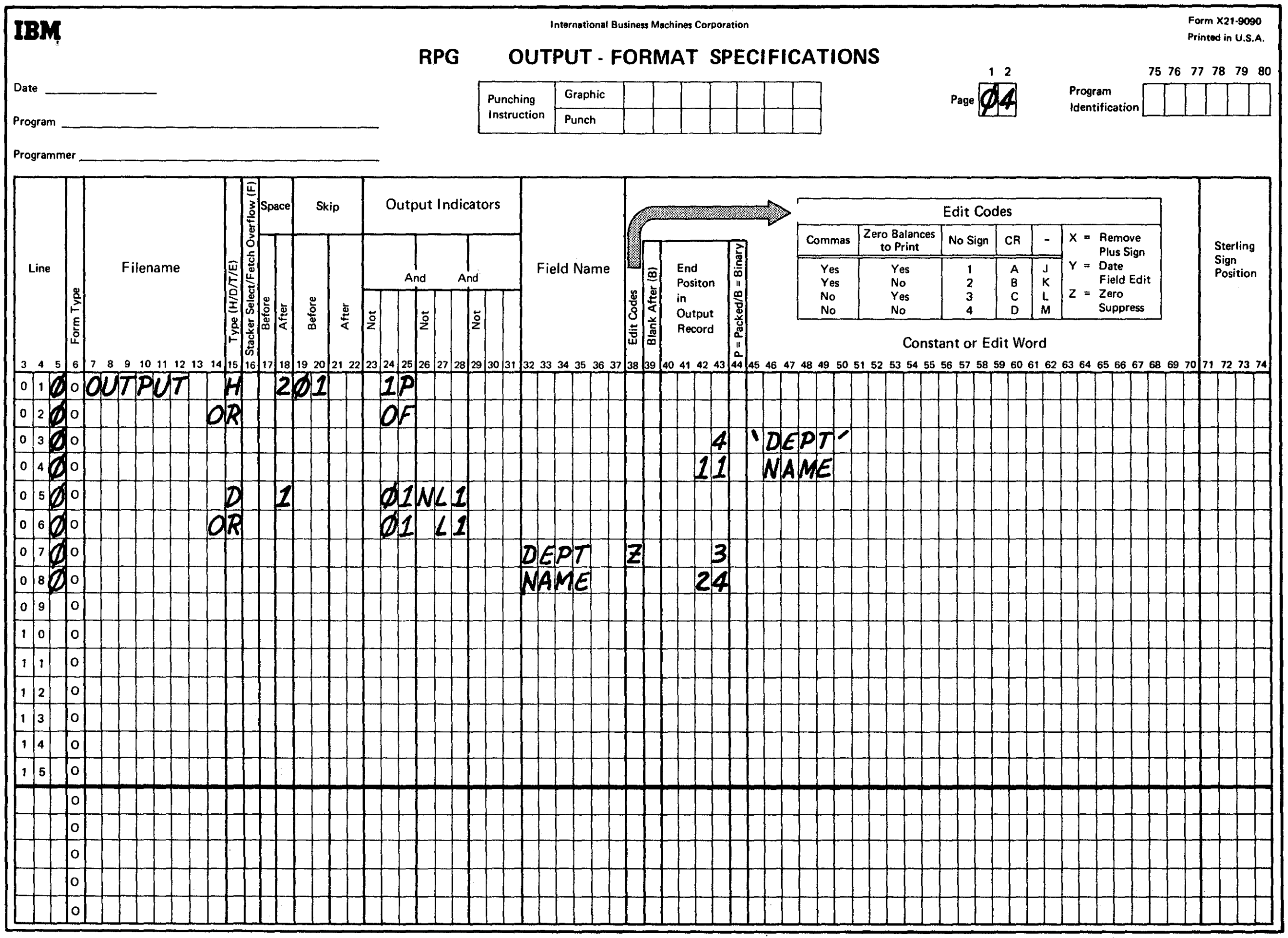
Programmer

Punching Instruction: Graphic / Punch

Page Ø3

Program Identification

Line	Form Type	Filename	Sequence	Record Identifying Indicator	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)
01	Ø I	CARD	AA	Ø1					
02	I				1	3	Ø	DEPT	L1
03	I				4	20	Ø	NAME	
04	I								
05	I								
06	I								
07	I								
08	I								
09	I								
10	I								
11	I								
12	I								
13	I								



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 Ø4

Program Identification

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)	Space After	Skip Before	Output Indicators	And	And	Field Name	Edit Codes	End Position in Output Record	Constant or Edit Word
01	Ø O	OUTPUT	H	2	Ø1	1P						
02	Ø O	OR				OF						
03	Ø O										4	'DEPT'
04	Ø O										11	NAME
05	Ø O		D	1		Ø1	NL1					
06	Ø O	OR				Ø1	L1					
07	Ø O								DEPT	Z	3	
08	Ø O								NAME		24	
09	O											
10	O											
11	O											
12	O											
13	O											
14	O											
15	O											

Figure 52. Using Line Counter Specifications (Part 2 of 2)

```
            (DATA)
bbbbrrrrc851         BAILEY
bbbbrrrrc851         MARTIN
bbbbrrrrc(skip before record)
bbbbrrrrcDEPT        NAME
bbbbrrrrc851         SMITH
bbbbrrrrc851         THOMPSON
```

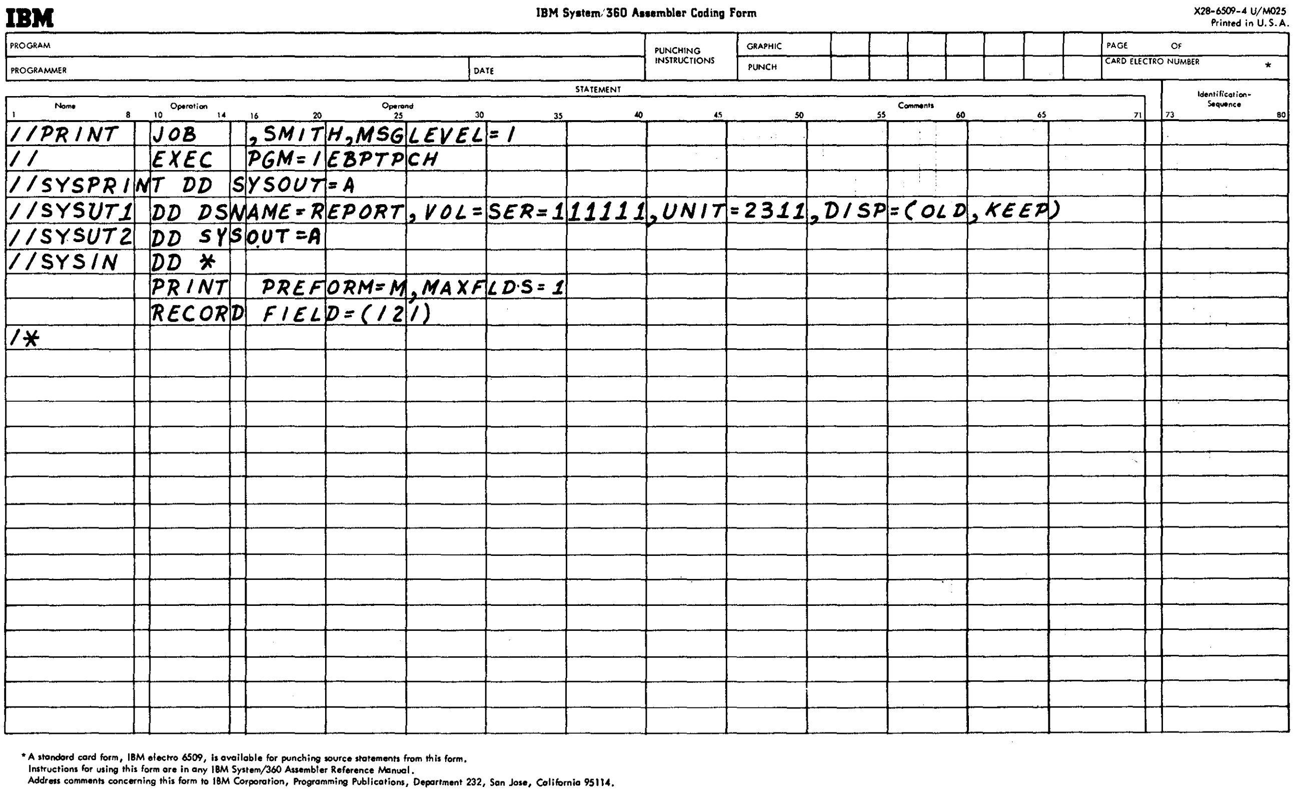
b = Block Length (4 bytes)

r = Variable-Record Length (4 bytes)

c = Printer Control Character (1 byte)

```
                                            Page 1
          DEPT               NAME

          824                BENNET
          824                FOSTER

          843                MARTIN
          843                PETERSON

          847                ADAMS
          847                ALLEN
          847                ANDERSON
          847                BROWN
          847                HILL
          847                HOFFMAN
          847                JONES

          851                BAILEY
          851                MARTIN
                                            Page 2

          851                SMITH
          851                THOMPSON

          857                ANDRIST
          857                CLEMENT
          857                CARROL
          857                GRAY
          857                MOBERG
```

Figure 53. Records on Disk and Output from Line Counter Program



IBM

IBM System/360 Assembler Coding Form

X28-6509-4 U/M025
Printed in U.S.A.

PROGRAM		PUNCHING INSTRUCTIONS	GRAPHIC
PROGRAMMER	DATE		PUNCH

PAGE OF

CARD ELECTRO NUMBER *

STATEMENT

Name 1 8 · Operation 10 14 · Operand 16 20 25 30 35 40 45 50 · Comments 55 60 65 71 · Identification-Sequence 73 80

```
//PRINT  JOB  ,SMITH,MSGLEVEL=1
//       EXEC PGM=IEBPTPCH
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSNAME=REPORT,VOL=SER=111111,UNIT=2311,DISP=(OLD,KEEP)
//SYSUT2 DD SYSOUT=A
//SYSIN  DD *
         PRINT  PREFORM=M,MAXFLDS=1
         RECORD FIELD=(121)
/*
```

*A standard card form, IBM electro 6509, is available for punching source statements from this form.
Instructions for using this form are in any IBM System/360 Assembler Reference Manual.
Address comments concerning this form to IBM Corporation, Programming Publications, Department 232, San Jose, California 95114.

Figure 54. Disk To Print Utility

The specifications for this form are divided into two categories as shown in Figure 55.

1. Record Identification (positions 7-42). These entries identify the input record (by specifying the identifying record codes it contains) and specify the relationship of the record to other records in the file.

   One line of the form is used to describe one record type.

   When the specifications are being written, it is not necessary to indicate the specific input units used in the program. The unit used for each file is specified on the file description form. Each filename, therefore, is related to a specific input unit. By merely writing the filename on the input form, the input unit has, in effect, been specified.

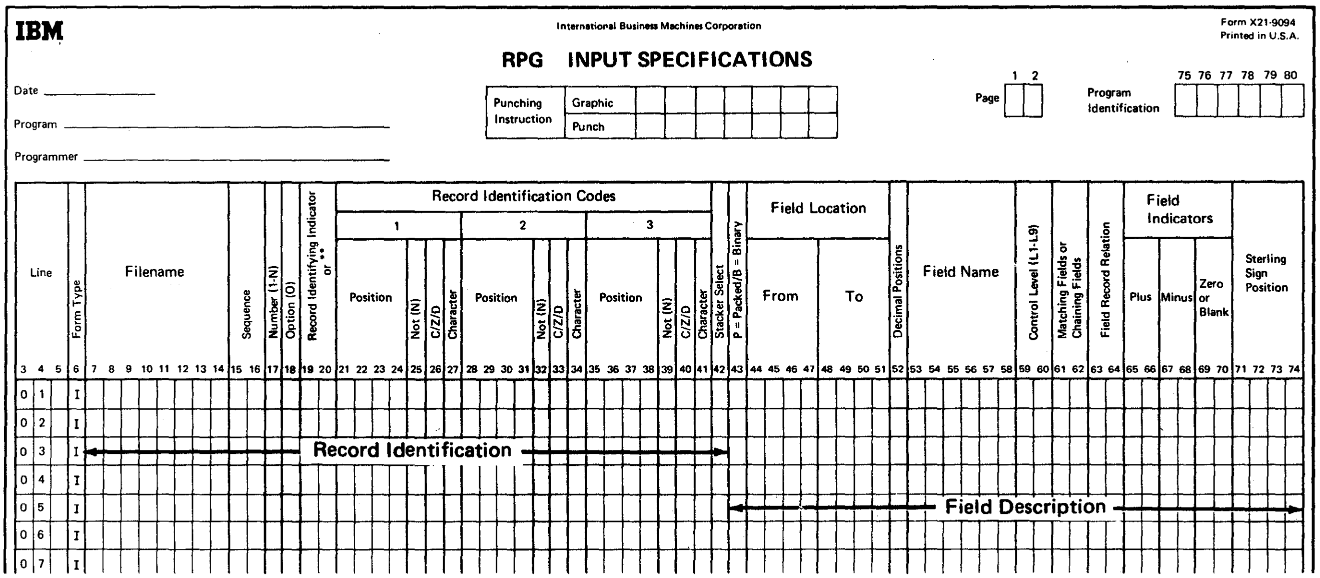
2. Field Description (positions 43-74). These entries describe the fields of the input record used in the report.

Sequence of Input Records

To save processing time, input records that occur only rarely in the program should always be specified at the end, and input records that occur frequently should be specified at the beginning of each specification list. (Specification list is a term used to describe the specifications from one or more of the same type of specification forms.)

On each detail cycle, the specifications are examined in the same sequence in which they are written on the Input Specifications form.

For example, assume a card file containing 3000 cards is to be processed and that there are five different exception procedures that must be followed for some of the cards. If the program is written so that the specifications concerning the exception cards are at the beginning of the input list, then, as each card is read, all specifications for the exceptions must be examined first before the specifications for the normal processing are found. Thus a great amount of processing time would be wasted if the card file contained only two or three exception cards but the exception specificatons had to be examined for all 3000 cards.



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

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	Not (N)	C/Z/D	Character	2: Position	Not (N)	C/Z/D	Character	3: Position	Not (N)	C/Z/D	Character	Stacker Select	P = Packed/B = Binary	Field Location From	To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Minus	Zero or Blank	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	I																														
0 2	I																														
0 3	I																														
0 4	I																														
0 5	I																														
0 6	I																														
0 7	I																														

Figure 55. Input Specifications Form

RECORD IDENTIFICATION ENTRIES

Filename (7-14)

The filename must be given to each input file. The filename must be left justified (that is, it must start in position 7) and it must begin with an alphabetic character. The remaining characters of the name may be alphameric, but must not contain special characters or embedded blanks. The filename may be eight characters or fewer. (Embedded blanks are blank positions falling between other characters of the name.)

Note: In this publication, alphabetic character refers to the latters A through Z, the dollar sign, pound sign, and the at sign ($, #, and @).

The filename must be entered only with the first record identification line of the appropriate file.

Sequence (15-16)

This specification is used to check the sequence of cards within a control group. Figure 56 illustrates a card file containing three types of cards for each part number control group. In this example, to assure correct accumulation of the values, the balance forward card must be the first card in the control group, the receipt card the second, and the new order card the last.

The specifications for checking the sequence of these cards is shown in Figure 57. (Field description specifications are not shown.)

If the file is not in sequence, halt indicator H0 is set on. Unless this H0 indicator is set off by the SETOF operation (see *Calculation Specifications Form, Setting Indicators On or Off)*, the program will terminate before the next input record is read.

The cards are specified on the form in the same sequence in which they are to be read by the object module.

Note: The entries in Sequence must begin with 01 in each file and be consecutive in ascending order.

Alphabetic codes must be placed in Sequence if the input records do not have to be in sequence within a control group or if it is not necessary to stop processing when the records are not in sequence. Any two alphabetic characters can be used.

Within a given file, header cards or other cards that are not in sequence must be specified on the form before specifications that must be sequence checked.

Note: If a numeric specification is given in Sequence, then specifications must also be provided in Number and Option.

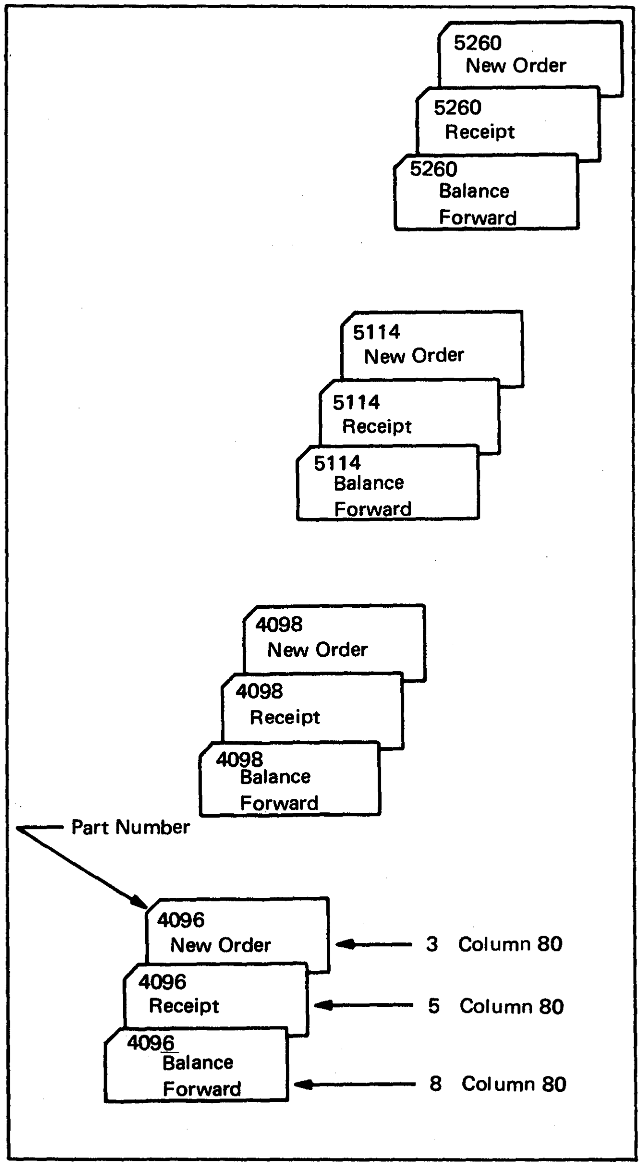


Figure 56. Card Types within Control Groups

Number (17)

If a numeric code is assigned under Sequence, an entry must also be made in Number (position 17). If an alphabetic code has been assigned under Sequence, this position must be blank.

This specification indicates whether only one record of a specific record type should exist in each control group or whether one or more than one record of a specific record type may exist in each control group. For example, Figure 58 illustrates two control groups of cards. In this example, there can only be one balance forward record in each control group, but there may be one or more new orders or receipt records.

The entry for the specification Number is either:

1. 1 if only one record of a type may exist in a control group.
2. N if one or more records of a type may exist in a control group.

The specifications required for the example in Figure 58 are illustrated in Figure 59.

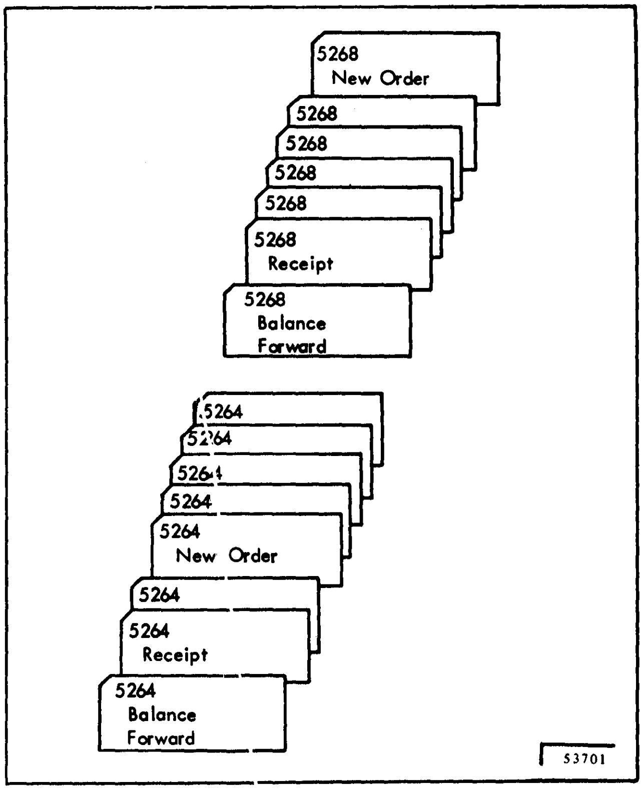
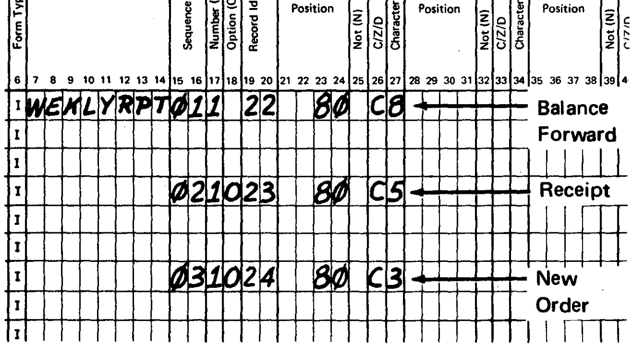


Figure 58. Number of Rec ord Types within a Control Group



International Business Machi

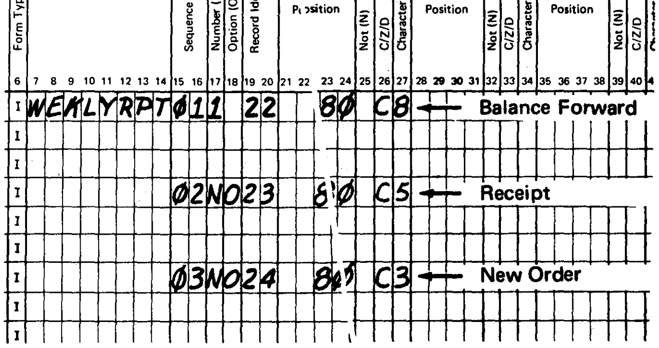
RPG INPUT SPEC

Punching Instruction	Graphic		
	Punch		

ıer

Form Type (6)	Filename (7–14)	Sequence (15–16)	Number (1-N) (17)	Option (O) (18)	Record Identifying Indicator or ** (19–20)	1 Position (21–24)	1 Not (N) (25)	1 C/Z/D (26)	1 Character (27)	2 Position (28–31)	2 Not (N) (32)	2 C/Z/D (33)	2 Character (34)	3 Position (35–38)	3 Not (N) (39)
I	WEKLYRPT	Ø1	1		22	8Ø		C	8	←				Balance Forward	
I															
I															
I		Ø2	1	O	23	8Ø		C	5	←				Receipt	
I															
I															
I		Ø3	1	O	24	8Ø		C	3	←				New Order	
I															
I															

Figure 57: Example of Sequence-Checking within Control Groups



International Business Machines

RPG INPUT SPECI

Punching Instruction	Graphic		
	Punch		

ıer

Form Type (6)	Filename (7–14)	Sequence (15–16)	Number (1-N) (17)	Option (O) (18)	Record Identifying Indicator or ** (19–20)	1 Pı ɔsition (21–24)	1 Not (N) (25)	1 C/Z/D (26)	1 Character (27)	2 Position (28–31)	2 Not (N) (32)	2 C/Z/D (33)	2 Character (34)	3 Position (35–38)	3 Not (N) (39)	3 C/Z/D (40)
I	WEKLYRPT	Ø1	1		22	8Ø		C	8	←				Balance Forward		
I																
I																
I		Ø2	N	O	23	8Ø		C	5	←				Receipt		
I																
I																
I		Ø3	N	O	24	8Ø		C	3	←				New Order		
I																
I																

Figure 59. Example of Using Numbeı· Specification

Option (18)

This specification is used only with numeric sequenced record types.

If the presence of a record is optional, the letter O is entered in this position. If a specific record type must be present in order to perform an operation or if records are nonsequential, this position must be left blank.

In Figure 57 and 59, the specifications under Option indicate that there must be a balanced forward card for each control group, but there may or may not be a new order or receipt card.

Record Identifying Indicator (19-20)

This specification is used in conjunction with the next specification Record Identification Codes (21-41). It has two purposes:

1. To establish a two-digit code for each input record type.

2. To set up a special condition in the object program each time the input record is read into the system. The object program may consider this condition during the processing of the calculation and output specifications.

As an example of the first function, assume that a certain card type is identified by the following codes:

1. Digit 5 in position 40

2. 11-punch in position 79

3. No 12-punch in position 80

By assigning a two-digit record identifying indicator to represent all of these codes, it is much easier to refer to this card type during the writing of the calculation and output specifications.

As an example of the second (and more important) function of this specification, record identifying indicators can be compared to selectors in punched card machines, or to internal or external switches on electronic data processing machines. The use of record identifying indicators (like the use of selectors and switches) is to permit certain operations to occur on specific conditions.

Figure 60 illustrates, by symbols, how record identifying indicators are used in the object program. In this example, a payroll file contains three types of cards.

Card Type	*Record Identifying Indicator*
Current earnings	14
Deduction	15
Adjustment	16

When one of these cards is read into the system during the object run, the appropriate record identifying indicator is set on, and these specifications pertaining to the record are performed. The detail specifications for these record types are indicated on the Calculation and Output-Format Specifications forms and are controlled by one of these three indicators. Specifications associated with other record types are not performed.

The input specifications required to establish the three indicators shown above are illustrated in Figure 61. (Field description specifications are not shown.)

Record identifying indicators from input records are set on and off during the processing of the object program as the various record types are read by the system. However, only one record identifying indicator can normally be on at one time. When a record identifying indicator is set on, all other record identifying indicators are set off. (See *Processing Multiple Input Files, Chaining* for an exception to this rule.)

Other indicator conditions that can be established in the program are Field Indicators (positions 65-70) of the input specifications and Record Identifying Indicators (positions 54-59) of the calculation specifications. These functions are described later, but one aspect of their use is of interest at this time.

All three types of indicators are assigned a two-digit number in the range of 01 through 99. Any of these 99 codes can be assigned to any one of the three types of indicators. Also the indicator codes do not have to be assigned in any sequence. For example, four different card types that are read into the system could be assigned codes 40, 62, 99, 02.

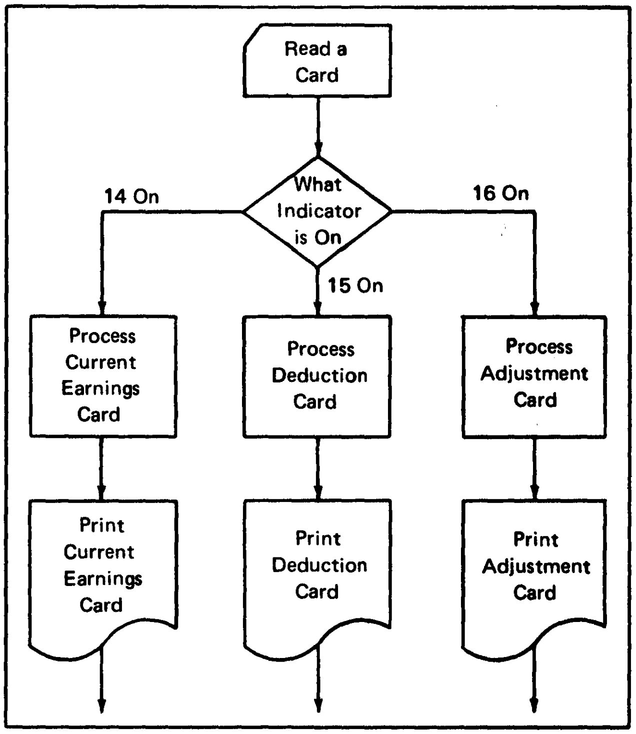
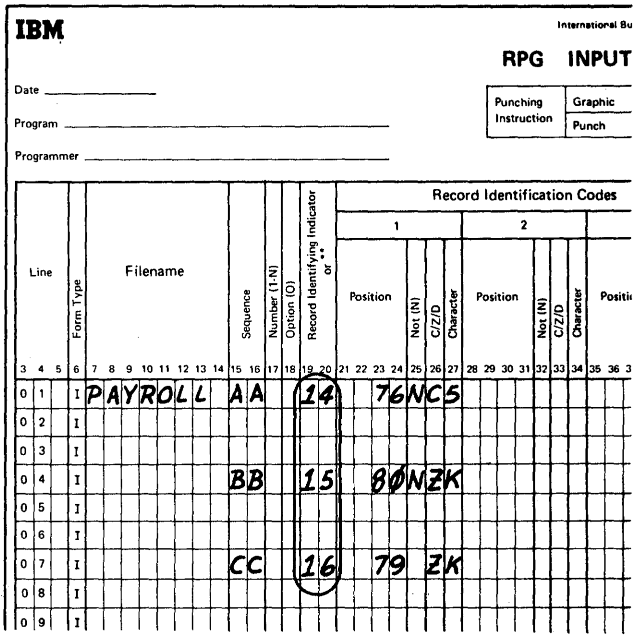


Figure 60. Sample Logic Flow Using Record Identifying Indicators



IBM

International Bu

RPG INPUT

Date ______

Program ______

Programmer ______

Punching Instruction	Graphic
	Punch

							Record Identification Codes								
							1				2				
Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Position	Not (N)	C/Z/D	Character	Position	Not (N)	C/Z/D	Character	Positi
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 3
0 1	I	PAYROLL	AA			14	76	N	C	5					
0 2	I														
0 3	I														
0 4	I		BB			15	8Ø	N	Z	K					
0 5	I														
0 6	I														
0 7	I		CC			16	79		Z	K					
0 8	I														
0 9	I														

Figure 61. Specifying Record Identifying Indicators

Record Identification Codes (21-41)

This specification provides a way of identifying each different record type used in the generated program. As mentioned previously, once the record type has been defined on the Input Specifications form, references to the record are made by its record identifying indicator.

These positions provide for the entry of one of three identifying codes as indicated by the number 1, 2, and 3 on the Input Specifications form. It is possible to specify more than three record identification codes by using more than one line. If only one record type is in the input file, the identification codes can be left blank, but a record identifying indicator and a sequence number must be specified.

Each of the three sets of entries is the same, so only the first set (positions 21-27) is described. Each set is divided into four categories:

1. Position (21-24)

2. Not (25)

3. C/Z/D (26)

4. Character (27)

Position (21-24)

Enter in these positions the placement in each data record of the character that contains the identifying code. The placement must be right justified in positions 21-24; leading zeros may be omitted.

Not (25)

Enter an N in this position if the code described must not be present in the specified record position. Otherwise, leave this position blank.

C/Z/D (26)

The object module identifies the different record types of a program by comparing the character written in Character against the codes contained in the records. The entry in position 26 specifies whether the entire character (C), against only the zone portion (Z), or against only the digit portion (D). Enter a C, Z, or D in position 26.

Zone Record Identification: Testing a zone means that the zone of the character located in the place specified in positions 21-24 of the input form is used to identify the record. (Zone in this case refers to the meaning of this word in punched card data processing systems. It does not refer to bits 0-3 of the System/360 EBCDIC character coding.)

The ampersand, the minus, and the blank are exceptions. An ampersand is identified as a 12-zone, a minus is identified as an 11-zone, and a blank is identified as a no-zone. The four common zones are:

1. 12-zone or plus zone ($\overset{+}{0}$, A-I, and &)

2. 11-zone or minus zone ($\bar{0}$, J-R, and minus)

3. 0-zone (S-Z)

4. No zone (0-9 and blank)

Note: When a blank, ampersand, or minus is used in the character portion of a zone test, other characters which contain the same machine zone (that is, 1, $, or %, respectively) will not satisfy the zone test. Any character outside these four groups compares equal to any other character outside these four groups.

Digit Record Identification: Testing a digit means that the digit portion of the character located in the place specified in positions 21-24 of the Input Specifications form is used to identify the record. (Digit in this case refers to the meaning of this word in punched card data processing systems. It does not refer to bits 4-7 of the System/360 EBCDIC character coding.) A valid digit test is performed only if the zone portion of the character is one of the following:

1. 12-zone or plus zone ($\overset{+}{0}$, A-I, and &)

2. 11-zone or minus zone ($\bar{0}$, J-R, and minus)

3. No-zone (0-9 and blank)

Character (27)

Enter in this position the identifying character that will be compared to the character specified in the input record. The character used in this position may be any letter A through Z, any number 0 through 9, or any special character.

EXAMPLES OF RECORD IDENTIFICATION CODES

Examples of record identification specifications are shown in Figure 62. The explanation of each entry is given here.

1. This entry specifies an 11-zone in record position 48. Any of the letters J-R could be placed in Character because they all contain an 11-zone. A Z must be placed in position 26 so that only the zone portion is checked. If a C were placed in this position, the object program would compare the letter K against the input record code instead of an 11-zone. A minus could be placed in this field to represent an 11-zone, and an ampersand could represent a 12-zone.

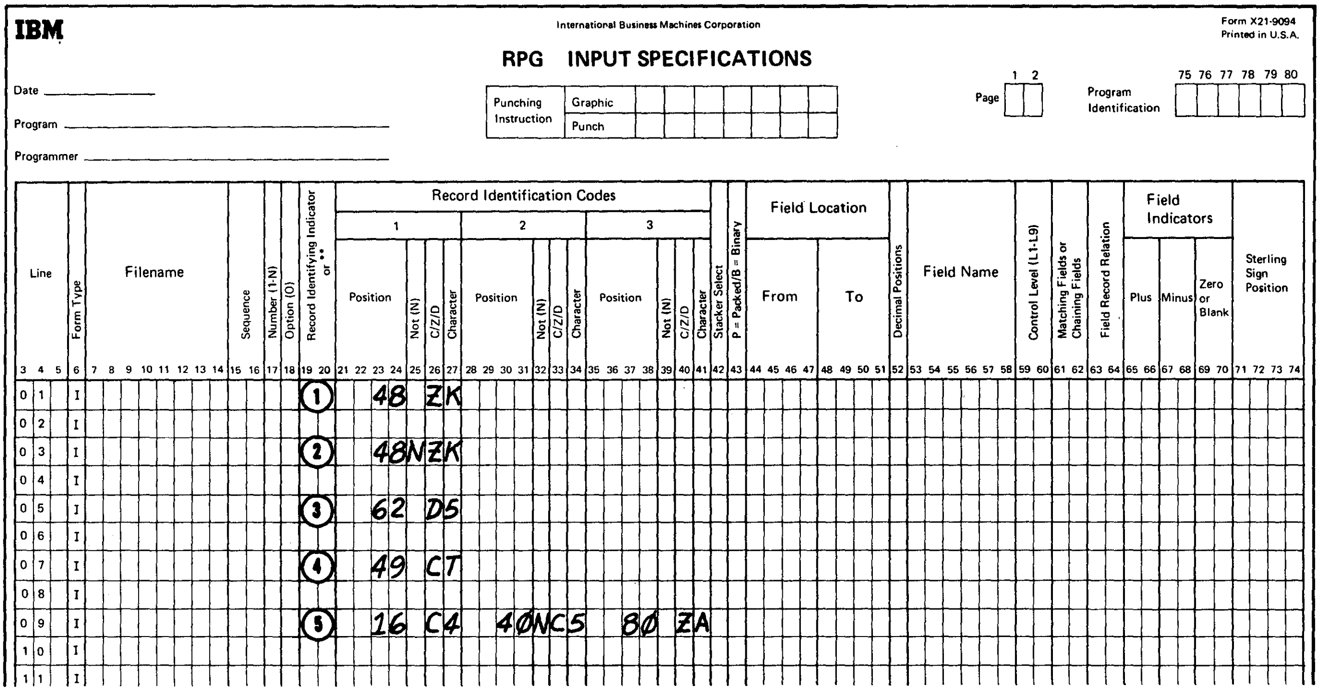
2. This entry specifies that no 11-zone may be present in record position 48.

3. This entry specifies that the digit portion of the code is checked. A character whose digit portion is 5 must be in record position 62. For example, a 5, N, V, or E would fulfill this requirement.

4. This entry specifies that the letter T must be present in record position 49.

5. These entries specify that all three codes must appear in the same record. A 4 must appear in position 16. Position 40 must not contain the number 5, and position 80 must contain a 12-zone.

Note: When more than one record type is specified for a file, the record identification codes of all record types should be mutually exclusive; that is, it should not be possible for an input record to satisfy the identification codes of more than one record type.



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)

Program Identification (75 76 77 78 79 80)

Line (3-5)	Form Type (6)	Filename (7-14)	Sequence (15-16)	Number (1-N) (17)	Option (O) (18)	Record Identifying Indicator or ** (19-20)	Record Identification Codes 1: Position (21-24)	Not (N) (25)	C/Z/D (26)	Character (27)	2: Position (28-31)	Not (N) (32)	C/Z/D (33)	Character (34)	3: Position (35-38)	Not (N) (39)	C/Z/D (40)	Character (41)	Stacker Select (42)	P = Packed/B = Binary (43)	Field Location From (44-47)	To (48-51)	Decimal Positions (52)	Field Name (53-58)	Control Level (L1-L9) (59-60)	Matching Fields or Chaining Fields (61-62)	Field Record Relation (63-64)	Field Indicators Plus (65-66)	Minus (67-68)	Zero or Blank (69-70)	Sterling Sign Position (71-74)
01	I					①	48		Z	K																					
02	I																														
03	I					②	48	N	Z	K																					
04	I																														
05	I					③	62		D	5																					
06	I																														
07	I					④	49		C	T																					
08	I																														
09	I					⑤	16		C	4	4Ø	N	C	5	8Ø		Z	A													
10	I																														
11	I																														

Figure 62. Record Identification Codes

AND Relationship

The last example illustrates the way to specify more than one code on one line. If an input record contained five different codes, they could be specified as shown in Figure 63. This is known as an AND relationship. It implies that the card is identified by:

1. A 4 in column 16.
2. No 5 in column 40.
3. An 11-zone in column 80.
4. A 2 in column 20.
5. A 3 in column 25.

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 positions 14-16 and blanks in positions 17-20 and 42.

OR Relationship

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 positions or in different positions.

A complete description of OR relationships is presented after the description of Field Location (positions 44-51).

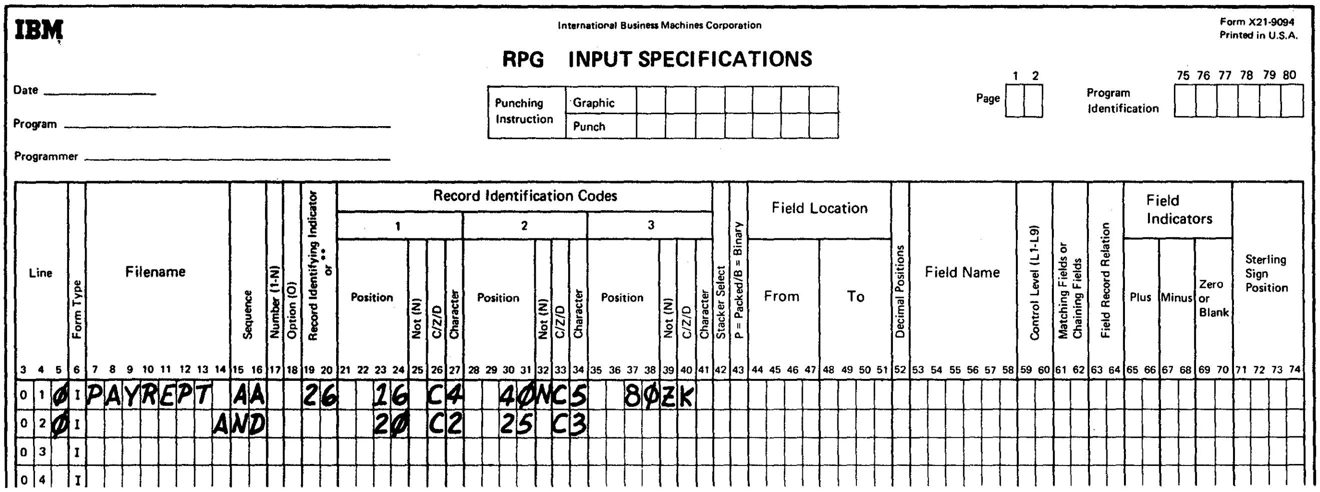
Omitting Record Identification

When input records are to be processed alike without regard for their identifying codes, positions 21-41 may be left blank; however, Sequence (positions 15-16) and Record Identifying Indicator (positions 19-20) must be specified.

When only certain record types within a group are listed first with their identifying record codes, the remaining record types may be bypassed or processed as a group. In either case, they are specified as the last record type, and they must have a record identifying indicator specified for them. Positions 21-41 of the Input Specificaitons form can then be left blank.

If a record type that has not been identified on the Input Specifications form appears in an input file during the processing of the generated program, the special indicator, H0, is set on. Unless the H0 indicator is set off by a SETOF operation (see *Calculation Specifications Form, Setting Indicators On or Off),* the program terminates before the next input record is read.

Note: If the records are to be bypassed, they should not be referred to on the Calculation or Output–Format Specifications forms.



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 □□

Program Identification 75 76 77 78 79 80 □□□□□□

							Record Identification Codes														Field Location							Field Indicators			
							1				2				3																
Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Position	Not (N)	C/Z/D	Character	Position	Not (N)	C/Z/D	Character	Position	Not (N)	C/Z/D	Character	Stacker Select	P = Packed/B = Binary	From	To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	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	PAYREPT	AA			26	16		C	4	4Ø	N	C	5	8Ø		Z	K													
0 2 Ø	I	A	ND				2Ø		C	2	25		C	3																	
0 3	I																														
0 4	I																														

Figure 63. Specifying More Than Three Record Identification Codes

Variable Length Input Records

If the record length of the shortest variable length record is less than the highest position tested in Record Idnetification, a blank cannot be used as a means of identifying a record in those positions that exceed the length of the minimum record. This is illustrated in the following example:

Record Identification Codes	*Record Length*
X-14	30
X-30, X-41 NX-60 NX-49	60
X-21, NX-79	80
X-49, NX-34	50

The minimum variable-length record is 30 positions; therefore, a blank cannot be used as a record identification code in any position higher than 30.

Stacker Select (42)

This specification causes cards to be selected into stackers of the input/output units. It is used when an input/output unit with more than one stacker is attached to the system.

If no entry is made in Stacker Select, the cards from the input file are selected to the first stacker pocket depending on the input/output unit attached to the system.

For input cards that are to be punched by the program (that is, combined file), stacker selection must be specified on the Output-Format Specifications form.

The stacker pockets and their acceptable codes are shown in Figure 64.

The stacker select entry is made on the same line with the record identification. If different records in an OR relationship are to be stacker selected, the appropriate stacker select entry must be written on each OR record identification line.

FIELD DESCRIPTION ENTRIES

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

On the record identification portion of the Input Specifications form, one line represents the specifications for one record type. On the field description portion of the form, one line represents the specifications for one field of a record.

The following information concerns the individual field descriptions of one record type. Field descriptions are always written on the specification line immediately below the specification line that identifies the record type.

Field description entries describe the fields of the input record to be used in the report. Each field of the record requires one line on the Input Specifications form. Positions 7-42 of the line must be blank.

Note: Unused record fields should not be described since this could waste core storage and processing time.

Input	Stacker Number	Stacker Select Code
IBM 1442 Card Read-Punch	1 2	1 or Blank 2
IBM 2501 Card Reader	–	Blank
IBM 2520 Card Read-Punch	1 2	1 or Blank 2
IBM 2540 Card Read-Punch	R1 R2	1 2

Figure 64. Summary of Stacker Select Specifications (Input)

Packed (43)

Packed format in the System/360 means that two decimal digits can be represented in one core storage byte. This is the data format used for numeric fields in RPG. Because input data is usually represented in the unpacked format (one digit in one core storage byte), the RPG program automatically converts numeric input data from the unpacked format to the packed decimal format. Because the packed decimal format permits greater use of storage capacities (card-tape-disk), the RPG program permits numeric data to be put out in the packed decimal format. (See *Output-Format Specifications Forms*).

In order to use this numeric output data in subsequent processing runs, RPG permits data in packed decimal format to be read into the RPG program.

Enter a P in this position if the numeric input field is in the packed decimal format. Otherwise, leave this position blank. (The letter P causes the RPG program to bypass the normal conversion of unpacked format to packed decimal format.)

The implied field length for determining the length of fields in calculation specifications for input in packed decimal format is:

> 2n – 1, where n = number of input record positions used.

Note: When an unpacked numeric field is packed internally by RPG, a 'C' sign bit configuration is developed for nonnegative fields. Therefore, when creating a file, care should be taken that any packed fields which will be used in a binary compare or chaining operation be created and maintained with a 'C' rather than 'F' sign bit configuration. If the 'F' is used, unequal compares and no record found conditions will result.

Field Location (44-51)

Positions 44-51 of the Input Specifications form are used to describe the location of each field in the record. The maximum field length for a numeric field is 15 digits. The maximum field length for an alphameric field is 256 characters.

From (44-47)

This specification contains the location of the first position (leftmost position) of the field. This entry must be right justified. Leading zeros may be omitted.

To (48-51)

This specification contains the location of the last position (rightmost position) of the field. This entry must be right justified. Leading zeros may be omitted.

Note: For indexed-sequential files; fields within the key area may be specified if the data records are unblocked and no Key Field Starting Location (positions 35-38, File Description Specifications form) has been specified. Fields within the key area may be indicated by entering a K to the immediate left of the high-order digit. If a K is specified in the From entry, then it must also be specified in the To entry, and vice versa. At least one nonzero digit must be to the right of any K entry. Figure 67 exemplifies a key field specification. The section *User's Guide for ISAM Processing* contains an explanation of key and record formats.

Figure 65 illustrates a card input record. The field location specifications necessary to read this card into the system are shown in Figure 66.

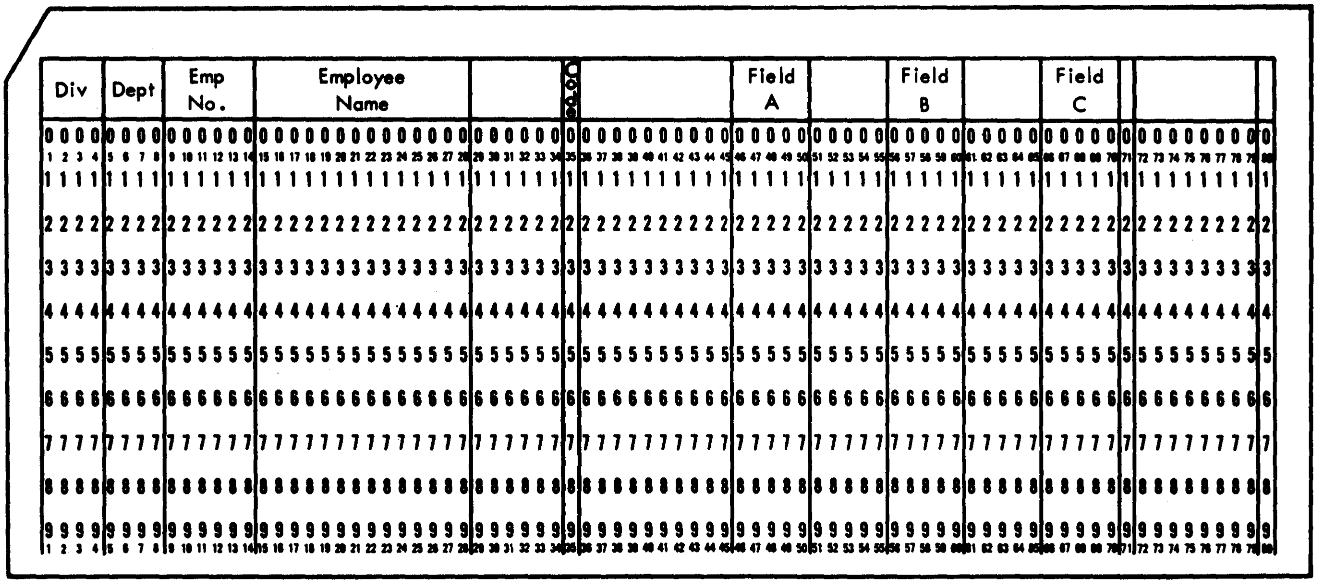
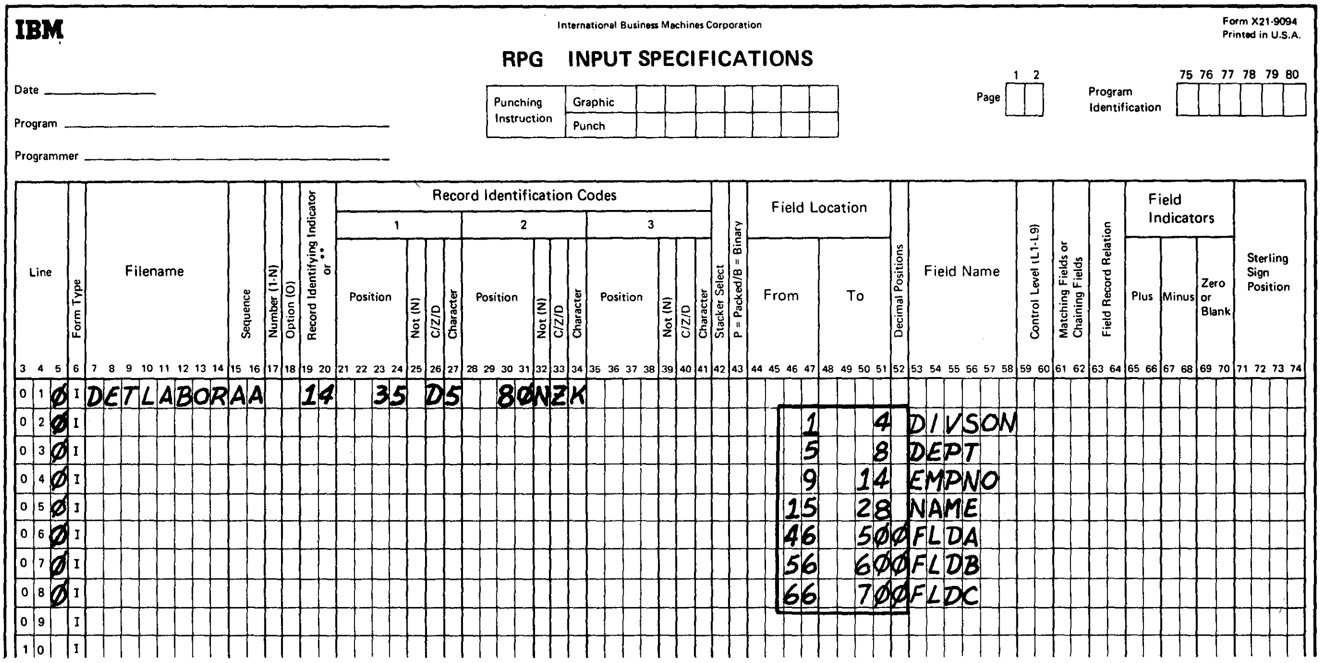


Figure 65. Sample Input Card



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

RPG INPUT SPECIFICATIONS

Date ____ Program ____ Programmer ____

Punching Instruction: Graphic / Punch   Page 1 2   Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	Not (N)	C/Z/D	Character	2: Position	Not (N)	C/Z/D	Character	3: Position	Not (N)	C/Z/D	Character	Stacker Select	P = Packed/B = Binary	Field Location: From	To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank	Sterling Sign Position
01Ø	I	DETLABOR	AA			14	35		D	5	8Ø	N	Z	K																	
02Ø	I																				1	4		DIVSON							
03Ø	I																				5	8		DEPT							
04Ø	I																				9	14		EMPNO							
05Ø	I																				15	28		NAME							
06Ø	I																				46	5Ø	Ø	FLDA							
07Ø	I																				56	6Ø	Ø	FLDB							
08Ø	I																				66	7Ø	Ø	FLDC							
09	I																														
10	I																														

Figure 66. Example of Field Location Specifications

The fields of the record may be listed in any sequence. (In this example they are shown in the same sequence as they appear on the card to make the example easier to understand.)

The specifications in Decimal Positions and Field Name are also included in Figure 66 because all three fields are closely related. These specifications are explained following the description of Field Location.

Records in an OR Relationship

It is possible to specify two different record types with just one set of specifications. This is known as an OR relationship. There are three types of OR relationships:

1. One or more record types have the same fields in the same positions of the record. (For example, all fields in one record type are in the same relative positions in another record type.)

2. One or more record types have the same fields, but the fields are in different positions of the record. (For example, unit cost is in positions 21-25 in one record type and in positions 31-35 in another record type.)

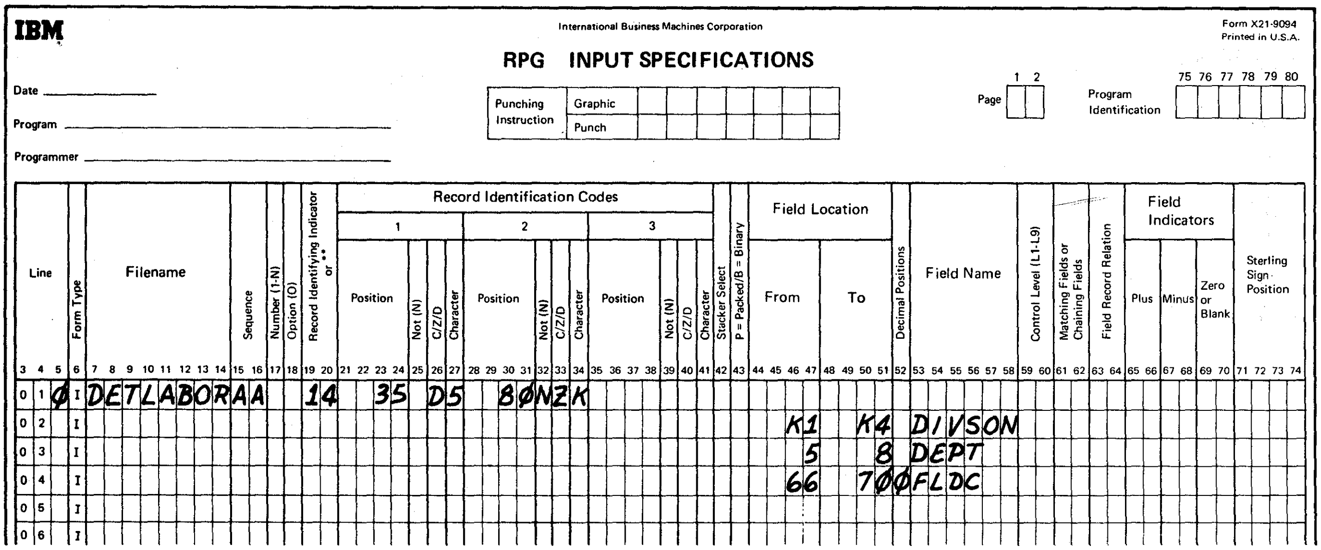
3. One or more record types have different fields in the same positions or in different positions of the record. (For example, two record types with ten fields in the same relative positions, but with three fields in different positions.)

It is of value to specify an OR relationship for the types in items 2 and 3 above only if there are more fields that are alike than fields that are unlike.

As an example of the first type of OR relationship, assume that there were two detail labor cards in the preceding example; perhaps the second was created during the previous week's reporting. If the second card has the same fields as the first but with a different record identification code, it is necessary to repeat all of the field specifications for the second card.

Figure 68 shows the only additional specifications required in order to specify both detail labor cards. The specification OR is entered in positions 14 and 15, and positions 16-18 are left blank. The last OR line (if there is more than one OR relation) is followed by the field description entries.

Figure 69 and 70 illustrate how to specify two records in an OR relationship when the field locations are not the same. The input record in Figure 69 is similar to the record in Figure 65 except that FLDC is located in positions 61-65 instead of positions 66-70. By specifying an OR realtionship, it is possible to specify both record types with one set of specifications.



IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date

Program

Programmer

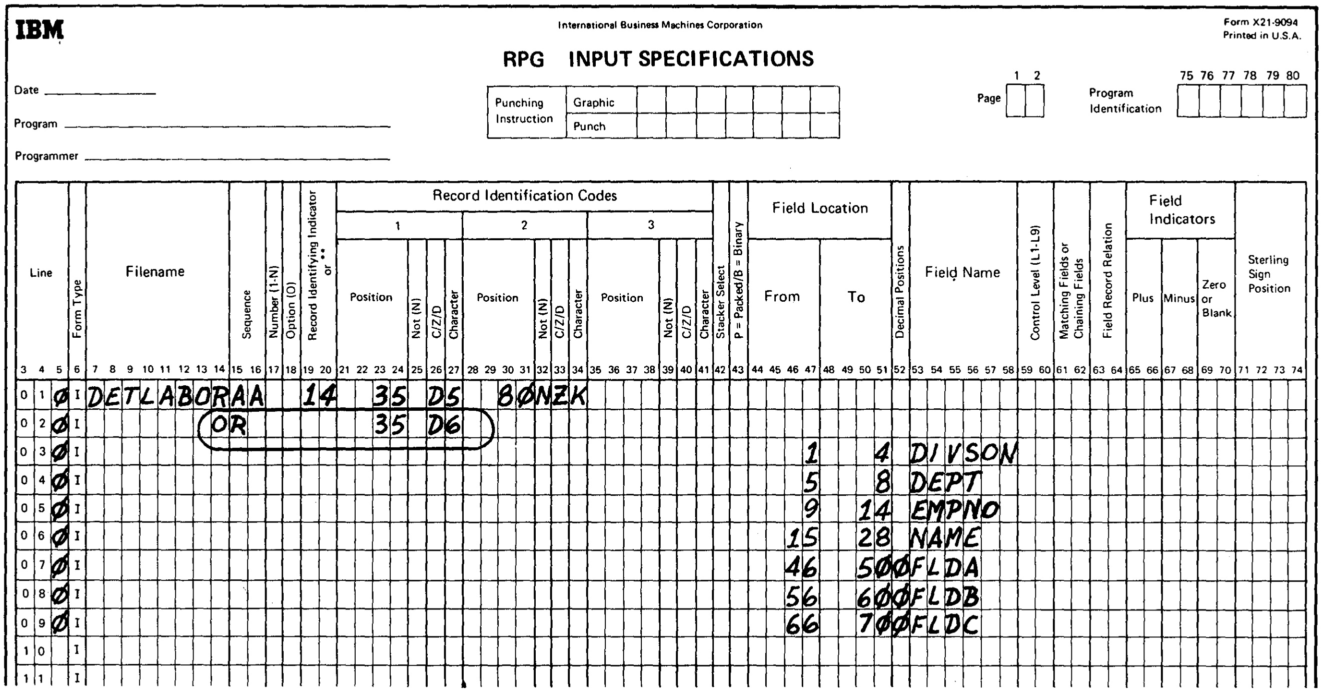
Punching Instruction | Graphic | Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes												Stacker Select	P = Packed/B = Binary	Field Location		Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators			Sterling Sign Position
							1				2				3																
							Position	Not (N)	C/Z/D	Character	Position	Not (N)	C/Z/D	Character	Position	Not (N)	C/Z/D	Character			From	To						Plus	Minus	Zero or Blank	
3 4 5	6	7 8 9 10 11 12 13 14	15 16	17	18	19 20	21 22 23 24	25	26	27	28 29 30 31	32	33	34	35 36 37 38	39	40	41	42	43	44 45 46 47	48 49 50 51	52	53 54 55 56 57 58	59 60	61 62	63 64	65 66	67 68	69 70	71 72 73 74
0 1 Ø	I	DETLABOR	AA			14	35		D	5	8Ø	N	Z	K																	
0 2	I																				K1	K4		DIVSON							
0 3	I																				5	8		DEPT							
0 4	I																				66	7Ø	Ø	FLDC							
0 5	I																														
0 6	I																														

Figure 67. Specifying Key Fields



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 Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	1 Position	1 Not (N)	1 C/Z/D	1 Character	2 Position	2 Not (N)	2 C/Z/D	2 Character	3 Position	3 Not (N)	3 C/Z/D	3 Character	Stacker Select	P = Packed/B = Binary	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Field Indicators Minus	Field Indicators Zero or Blank	Sterling Sign Position
01Ø	I	DETLABOR	AA			14	35		D	5	8Ø	N	Z	K																	
02Ø	I	OR					35		D	6																					
03Ø	I																				1	4		DIVSON							
04Ø	I																				5	8		DEPT							
05Ø	I																				9	14		EMPNO							
06Ø	I																				15	28		NAME							
07Ø	I																				46	5Ø	Ø	FLDA							
08Ø	I																				56	6Ø	Ø	FLDB							
09Ø	I																				66	7Ø	Ø	FLDC							
10	I																														
11	I																														

Figure 68. Records in an OR-Relationship, Identical Field Locations

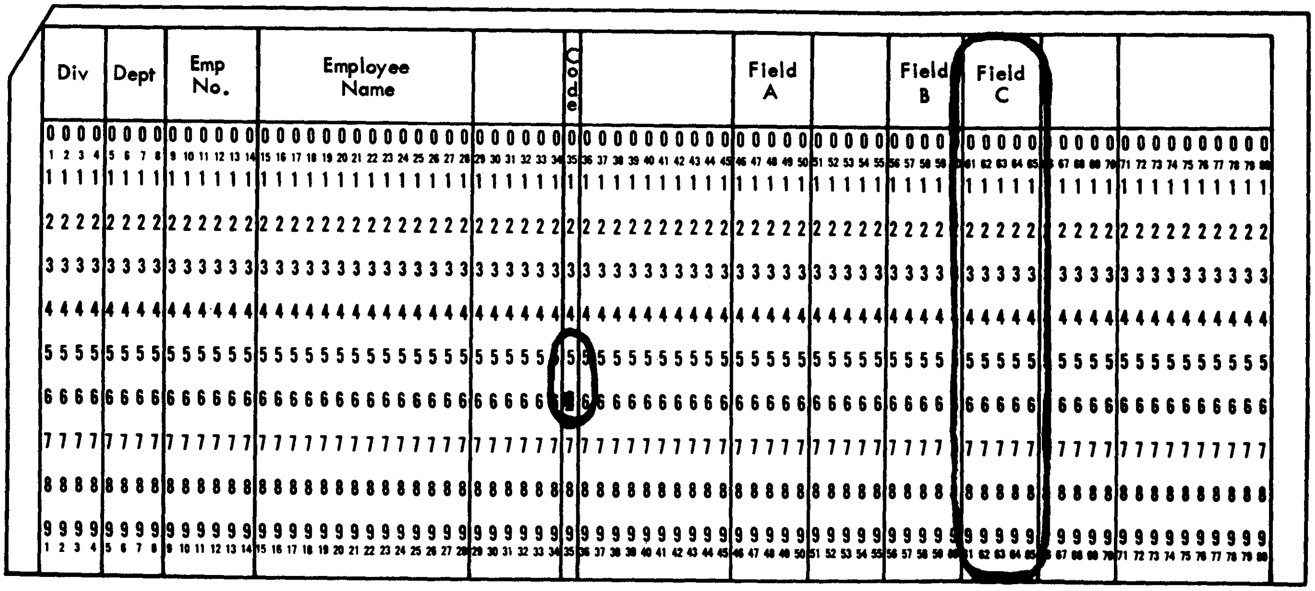


Figure 69. Sample Input Record, Differing Field Locations

53703

Figure 70 illustrates the specificatons for this example. The numbers in the margin of the subsequent text refer to the numbers circled in Figure 70.

1. When field locations are not the same for both types, it is necessary to provide a separate record identifying indicator code for the second record type.

2. Each field that is not located in the same positions in both record types must be specified twice. Each of the two specifications must be related to the appropriate record type. This is accomplished by specifying the appropriate resulting indicator code in Field Record Relation.

For example, FLDC located in positions 66-70 is related to record identifying indicator 14 (the record type identified by a 5 in position 35), and field FLDC in positions 61-65 is related to record indicator 16 (the record type identified by a 6 in position 35). Thus, if record identifying indicator 14 is on, FLDC will be taken from positions 66-70; if record identifying indicator 16 is on, FLDC will be taken from positions 61-65.

It is also possible to take advantage of an OR relationship to specify control fields that are not in the same positions in two different record types. In addition, specifying record types in an OR relationship is not limited to just two records. As many records as required can be specified as having an OR relationship.

Decimal Positions (52)

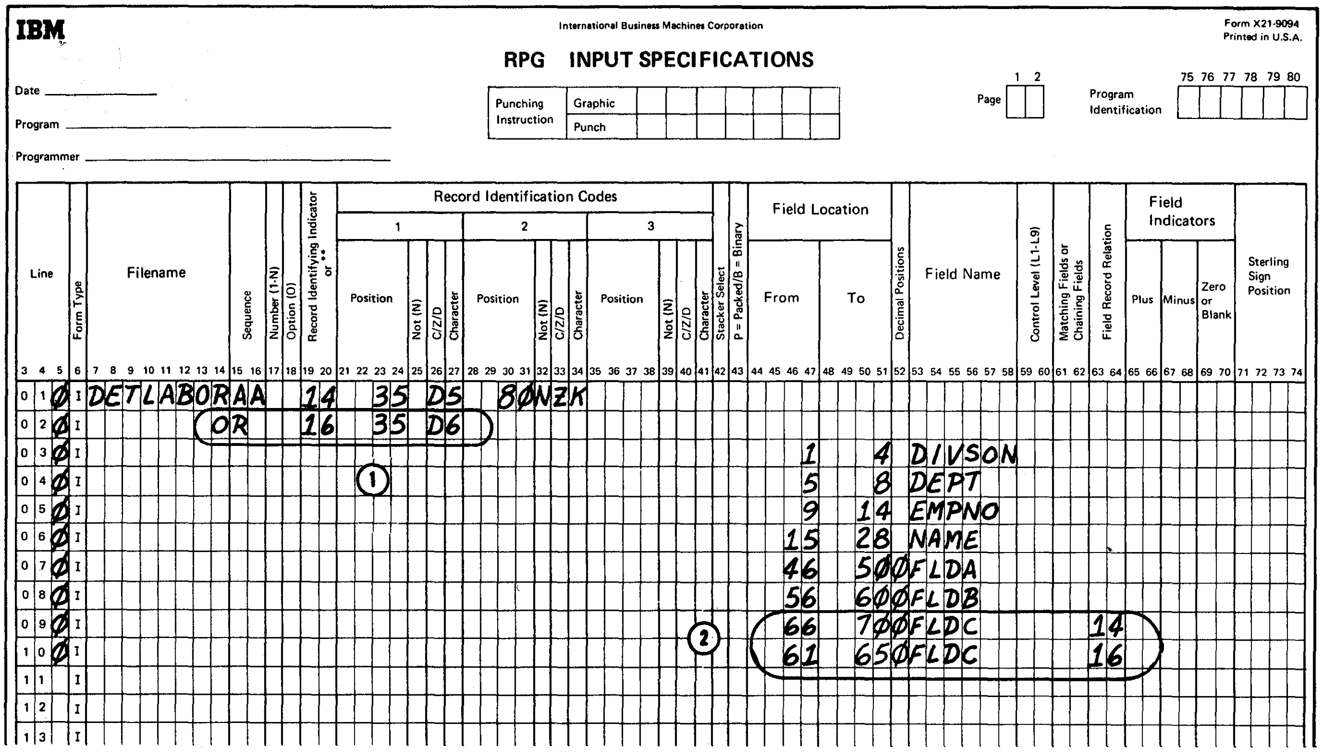
This specification performs two functions:

1. It is used by the object program to determine the number of decimal positions contained in the field specified in Field Location.

2. It causes the field specified in Field Location to have 0-(zero), 11-, or 12-zone bits removed from all positions except the rightmost position.

If the field specified in Field Location is to have calculations or edit functions performed upon it or if it will be zero suppressed in the output, there must be a specification in Decimal Positions. If there are no decimal positions in an arithmetic field, a zero must be specified. This specification must be left blank if the field is alphameric.

In Figure 71, the fields DIVSON, DEPT, and EMPNO contain numeric information but, because they will not have arithmetic operations performed upon them, they are specified as alphabetic fields by leaving Decimal Positions blank. The NAME field is alphabetic and therefore is also blank in Decimal Positions. The fields FLDA, FLDB, and FLDC are numeric arithmetic fields with decimal specifications of zero.



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

RPG INPUT SPECIFICATIONS

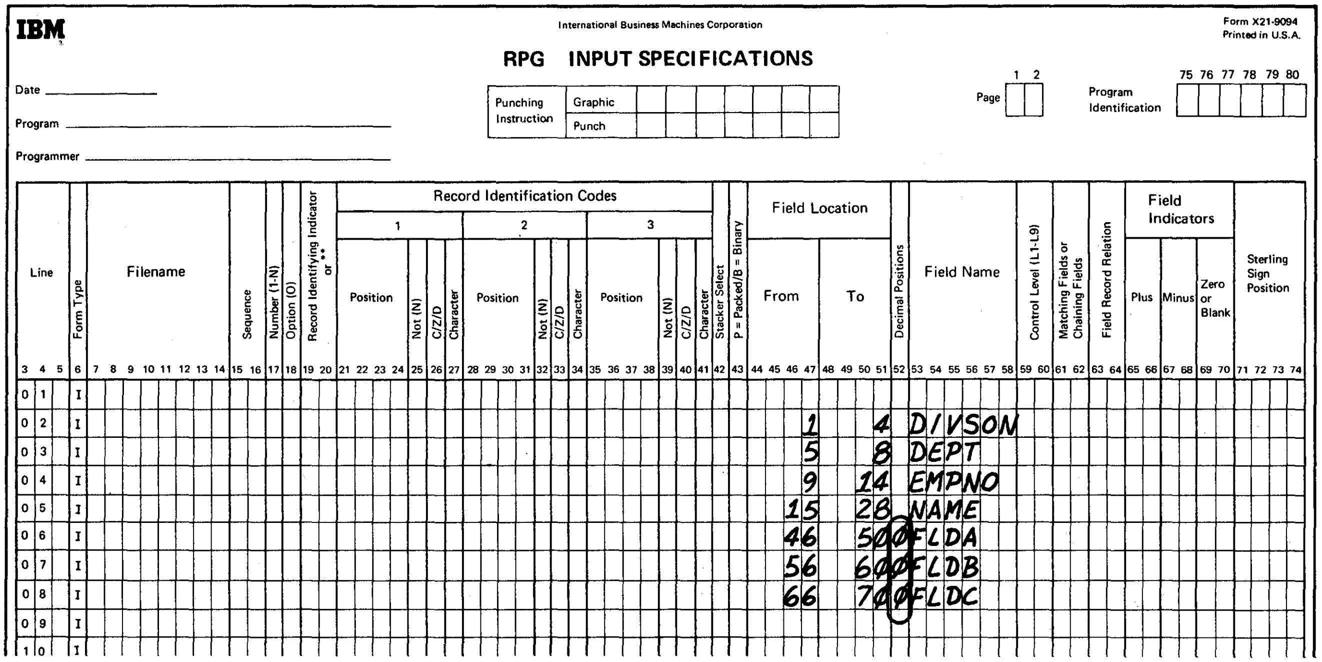
Date ____ Program ____ Programmer ____

Punching Instruction: Graphic | Punch   Page 1 2   Program Identification 75 76 77 78 79 80

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

```
01ØIDETLABORAA  14  35 D5  80NZK
02ØI       OR   16  35 D6
03ØI                                        1   4 DIVSON
04ØI                                        5   8 DEPT
05ØI                                        9  14 EMPNO
06ØI                                       15  28 NAME
07ØI                                       46  50ØFLDA
08ØI                                       56  60ØFLDB
09ØI                                       66  70ØFLDC      14
10ØI                                       61  65ØFLDC      16
11 I
12 I
13 I
```

Figure 70. Records in an OR Relationship, Differing Field Locations



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   Program Identification 75 76 77 78 79 80

```
01 I
02 I                                        1   4 DIVSON
03 I                                        5   8 DEPT
04 I                                        9  14 EMPNO
05 I                                       15  28 NAME
06 I                                       46  50ØFLDA
07 I                                       56  60ØFLDB
08 I                                       66  70ØFLDC
09 I
10 I
```

Figure 71. Specifying Decimal Positions on Input Form

Field Name (53-58)

Each field defined must be given a field name. Once a name has been assigned to a field, other references to it are made by using the field name, rather than by using the specific record position each time. Thus, the record positions of the input fields are not needed when writing the calculation and output specifications.

The field name must begin with an alphabetic character, and it must start in position 53. The field name may be alphameric, but it may not contain special characters or embedded blanks.

Figure 72 illustrates field names that are easy to read and suggest the function of the fields they represent.

Two input files may have fields with the same field names. A field name is assigned only once by RPG. The two input files use the same storage location for fields with identical names. This procedure is permissible when using RPG, but the programmer should be aware that possible errors in calculations or output may occur when two input files have the same field names. This is especially important when chaining fields are specified on the Input Specifications form.

Specifying the Same Data Field as Alphameric or Numeric

If the same field from an input record is to be used as both an alphameric and a numeric field, the field must be specified twice by assigning two different field names to the same location in the record. (If no decimal positions are specified for the field, it is considered to be an alphameric field.)

Using Input Data Fields as Constants

The term constant is frequently used to describe 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.

The technique for getting this information to the program is to define a special record type and assign a field name that is not otherwise used. This technique will also permit entering of constants that are too large to be specified as literals in the source program.

When only a single input file is processed, the date or constant card(s) may be placed at the front of the file preceding the first data record.

If an additional file is required for entering the constant information, it may be designated as a primary file and the data file as a secondary file. Using this technique, the primary file will be processed first and the constant information will be entered prior to the data records.

If multiple input files are processed, the constant card(s) may appear as the first records on any file. The constant records should be defined in the input specifications for the associated file and should not have matching fields (see *Matching Fields* in this section). In a job with multiple files, the constant file could be designated as secondary and since its associated input specifications have no matching fields, its records will be processed first.

Control Level (59-60)

As the object program is processed, a change in a control field causes all processing indicated by this change to be initiated. Positions 59-60 are used to provide a convenient and simple method for specifying all control functions.

Up to nine control levels can be used by RPG. These levels are designated from low to high as L1, L2, L3, ... L9. An indicator similar to a record identifying indicator is associated with each control level designation. They are used to control functions specified on the Calculation and Output-Format Specifications forms. When the field specified in Field Location is a control field, its appropriate control level must be specified in positions 59-60. The first three lines in Figure 72 show the entries for the three control fields.

The field DIVNO (first entry in the example) is the highest level of control.

Indicator L3 could be used to specify when the calculation specifications for the control field DIVNO are to take place or when the totals for the field DIVNO are to be printed or summarized. More information regarding the use of the control level indicators is presented in the descriptions of the Calculation and Output-Format Specifications forms.

Whenever a record is read that contains a control field specified with a level indicator in positions 59-60, the data in the field location defined as the control field is compared with the data from the same previous control field. If the data differs, the specified level indicator and all defined lower-level indicators are set on.

The level indicators are set on just before total time processing for the record that caused the control break. They are set off after detail time processing of the new record. Therefore, total time calculations and output from the last control group, and detail time calculations and output for the first record of the new control group, can be conditioned with the level indicators.

Note: Whenever control levels are used, a control break will occur on the first record of a record type which has control levels. Total calculations, total lines, and overflow lines are bypassed until after the control break occurs.

Additional Functions of Control Level Specifications

If a field specified in Field Location also has an entry in Control Level, the object program places the field into two storage areas. One area, known as a control field holding area, is used for the controlling functions of the field, and the other is used for any other uses of the field (such as for printing or arithmetic calculations).

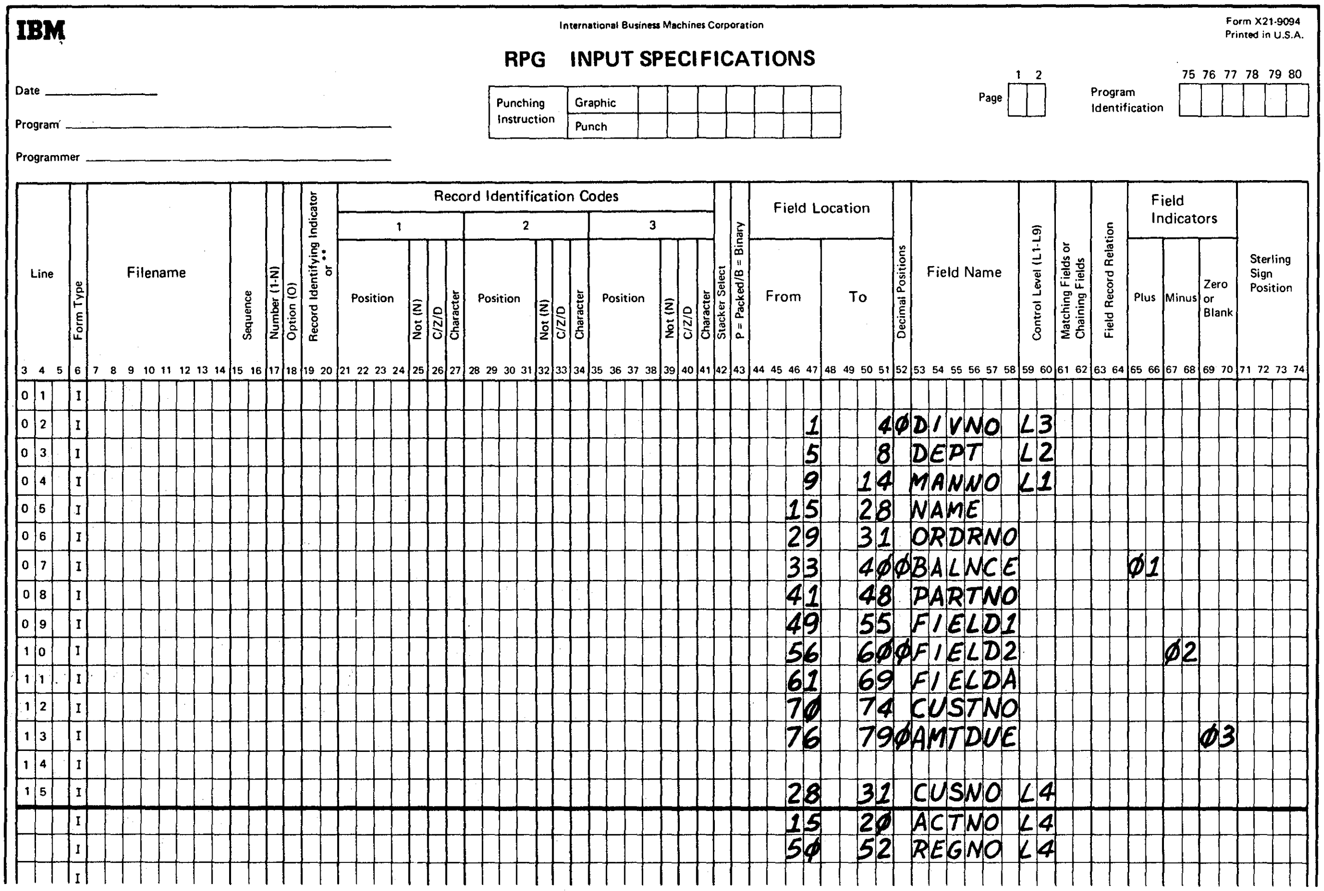
When an alphameric field is used as a control field, the zone bits in the field are used in the comparison of one record to the next. To use an alphameric field for controlling functions without considering the zone bits, specify the field as numeric by making an entry in Decimal Positions. In Figure 72 the field DIVNO is used for controlling functions. The entry L3 causes the value of the field DIVNO to be stored in the control field holding area in packed format with a plus sign of C. The sign of the value is C regardless of the sign of the input field.

Split Control Fields

Several fields in an input record can be specified as one control field. In the lower half of Figure 72, three fields, which are not in adjacent record positons, are specified with the same L4 control level. The three fields are treated as one control field:

CUSNO *ACTNO* *REGNO*

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. All fields are placed according to the sequence in which they are defined on the Input Specifications form.



IBM International Business Machines Corporation

RPG INPUT SPECIFICATIONS

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

Date ____
Program ____
Programmer ____

Punching Instruction: Graphic / Punch

Page 1 2 | Program Identification 75 76 77 78 79 80

Line (3-5)	Form Type (6)	Filename (7-14)	Field Location From (44-47)	Field Location To (48-51)	Decimal Positions (52)	Field Name (53-58)	Control Level (L1-L9) (59-60)	Matching Fields or Chaining Fields (61-62)	Field Record Relation (63-64)	Field Indicators Plus (65-66)	Field Indicators Minus (67-68)	Field Indicators Zero or Blank (69-70)	Sterling Sign Position (71-74)
01	I												
02	I		1	4	ø	DIVNO	L3						
03	I		5	8		DEPT	L2						
04	I		9	14		MANNO	L1						
05	I		15	28		NAME							
06	I		29	31		ORDRNO							
07	I		33	4ø	ø	BALNCE				ø1			
08	I		41	48		PARTNO							
09	I		49	55		FIELD1							
10	I		56	6ø	ø	FIELD2					ø2		
11	I		61	69		FIELDA							
12	I		7ø	74		CUSTNO							
13	I		76	79	ø	AMTDUE						ø3	
14	I												
15	I		28	31		CUSNO	L4						
	I		15	2ø		ACTNO	L4						
	I		5ø	52		REGNO	L4						
	I												

Figure 72. Field Names

Using Split Control Fields with Field Record Relation

The use of the Field Record Relation specification in conjunction with control level indicators permits the programmer to define the same control level for split or nonsplit control fields in various record types. This is illustrated in Figure 73 and Figure 74.

The following points must be considered when using field record relation indicators with split control field specifications.

1. The overall field length for each control level must be the same in each group.

2. The sum of the split control fields cannot be greater than 256 bytes.

3. Split control fields of any one level are arranged in storage in the same sequence as they appear in the input specifications.

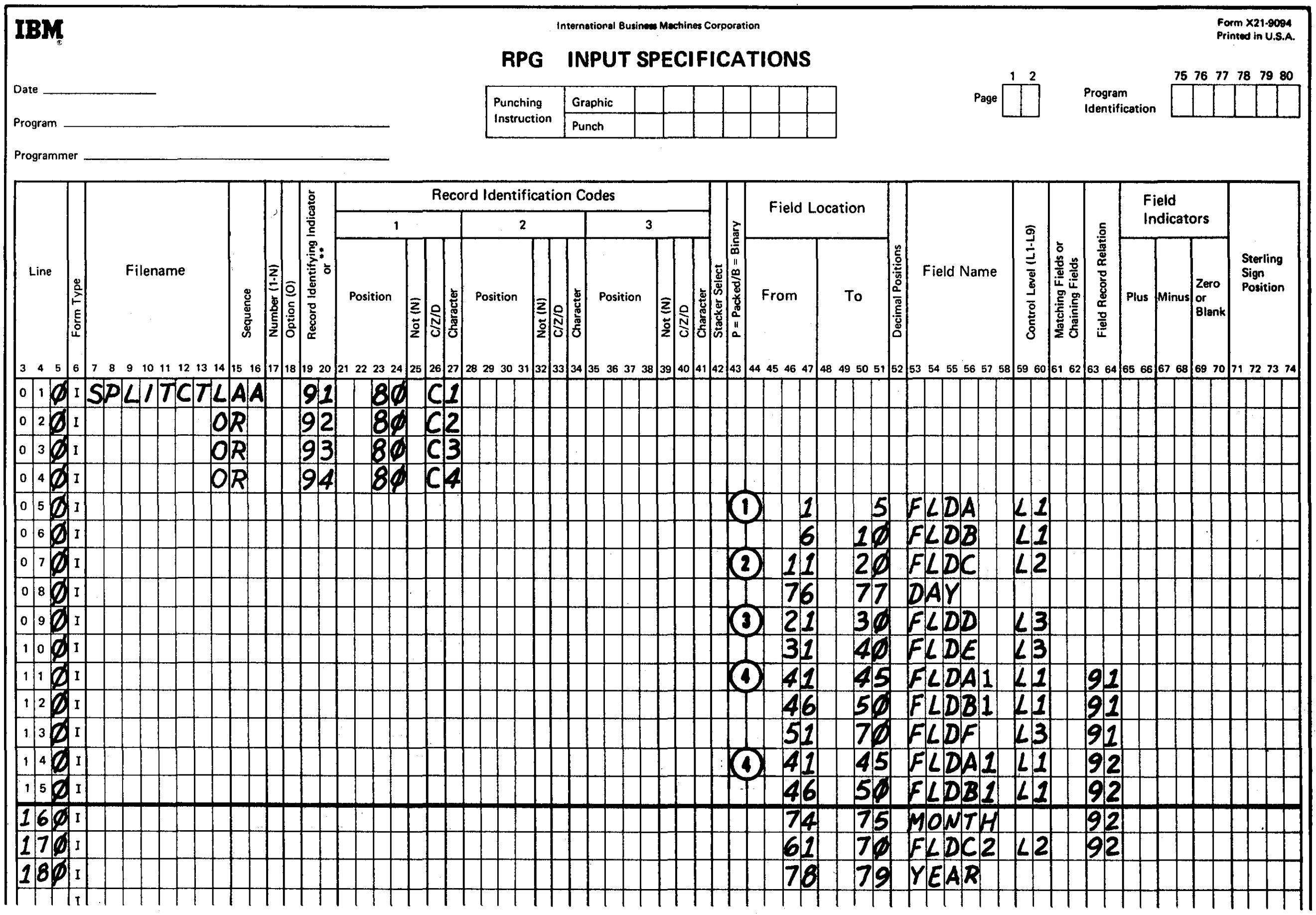
4. If a field record relation indicator is specified for a control level, the related field location is used for control purposes whenever that indicator is on. The examples that are identified by the circled numbers in Figure 73 illustrate this point.

    1. When either indicator 93 or 94 is on, L1 is composed of positions 1-5 and 6-10.

    2. When indicator 91, 93, or 94 is on, L2 is composed of positions 11-20.

    3. When indicator 92, 93, or 94 is on, L3 is composed of positions 21-40. A control level with a blank field record relation indicator is used for control purposes when all indicators that condition any field with the same control level are off.

    4. To specify a portion of a split control field as being common to several record types, repeat that portion of the field definition with each record type indicator.



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

Program Identification 75 76 77 78 79 80

Line (3-5)	Form Type (6)	Filename (7-14)	Sequence (15-16)	Record Identifying Indicator (19-20)	Position (21-24)	C/Z/D (26)	Character (27)		From (44-47)	To (48-51)	Field Name (53-58)	Control Level (59-60)	Field Record Relation (63-64)
01Ø	I	SPLITCTL	AA	91	8Ø	C	1						
02Ø	I	OR		92	8Ø	C	2						
03Ø	I	OR		93	8Ø	C	3						
04Ø	I	OR		94	8Ø	C	4						
05Ø	I							①	1	5	FLDA	L1	
06Ø	I								6	1Ø	FLDB	L1	
07Ø	I							②	11	2Ø	FLDC	L2	
08Ø	I								76	77	DAY		
09Ø	I							③	21	3Ø	FLDD	L3	
10Ø	I								31	4Ø	FLDE	L3	
11Ø	I							④	41	45	FLDA1	L1	91
12Ø	I								46	5Ø	FLDB1	L1	91
13Ø	I								51	7Ø	FLDF	L3	91
14Ø	I							④	41	45	FLDA1	L1	92
15Ø	I								46	5Ø	FLDB1	L1	92
16Ø	I								74	75	MONTH		92
17Ø	I								61	7Ø	FLDC2	L2	92
18Ø	I								78	79	YEAR		

Figure 73. Example of Multiple Split-Control-Field Specifications

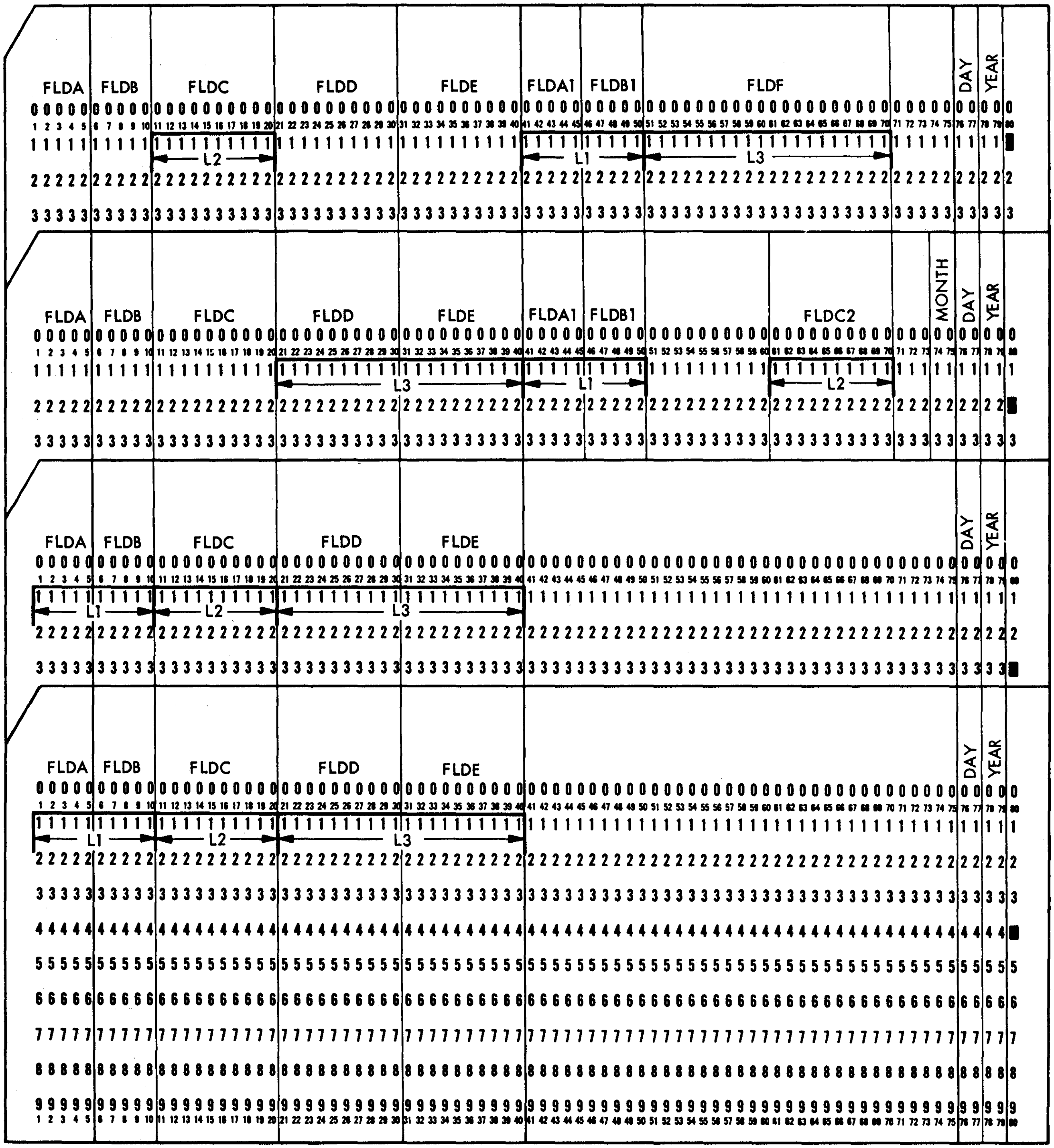


Figure 74. Input Cards with Multiple Split Control Fields

53689

Matching Fields or Chaining Fields (61-62)

This specification is used if the input consists of more than one file. It provides the program with the ability to match or to chain the records of one file with those of another file.

Matching Fields

Up to three matching fields (designated by M1, M2, and M3) are allowed. This entry may be used to match records in different sequential files. The second sample program in this publication uses matching fields in a card input file and a tape input file to govern processing of records. A discussion of this use of matching fields is contained in *Processing Multiple Input Files*.

If a field specified in Field Location also has an entry in Matching Fields, the field is placed into another storage area known as a matching field hold area. Comparisons of matching fields are performed in these hold areas.

Using the Matching Fields Specification for Sequence Checking: If only one input file is specified, fields within the file may be sequence checked by using the Matching Fields specification. Up to three fields within the file may be checked. In Figure 75 three fields within the input file are to be sequence checked. The data from the three fields will be moved by the RPG program to the matching field hold area as shown in Figure 76. When the second record has been read, it will be moved as shown. The compare operation is made on all three fields at the same time. M3 is placed in the highest-order position. M1 is placed in the low-order position.

In position 18 of the File Description Specifications form, the programmer must specify if the file is in ascending or descending sequence. In Figure 76 assume that the file has been specified in ascending sequence. The number 003051008 is lower than 005025003. Thus, the file is in ascending sequence.

Note: A chaining file may also be sequence checked. However, if it is the chaining field that is to be sequence checked, it must be defined twice using two different field names. The chaining indicator is entered in one field definition and the matching fields indicator in the other.

If the file is not is sequence Halt indicator H0 is set on. Unless this H0 indicator is set off by a SETOF operation (see *Setting Indicators On or Off*, the program terminates before the next input record is read.

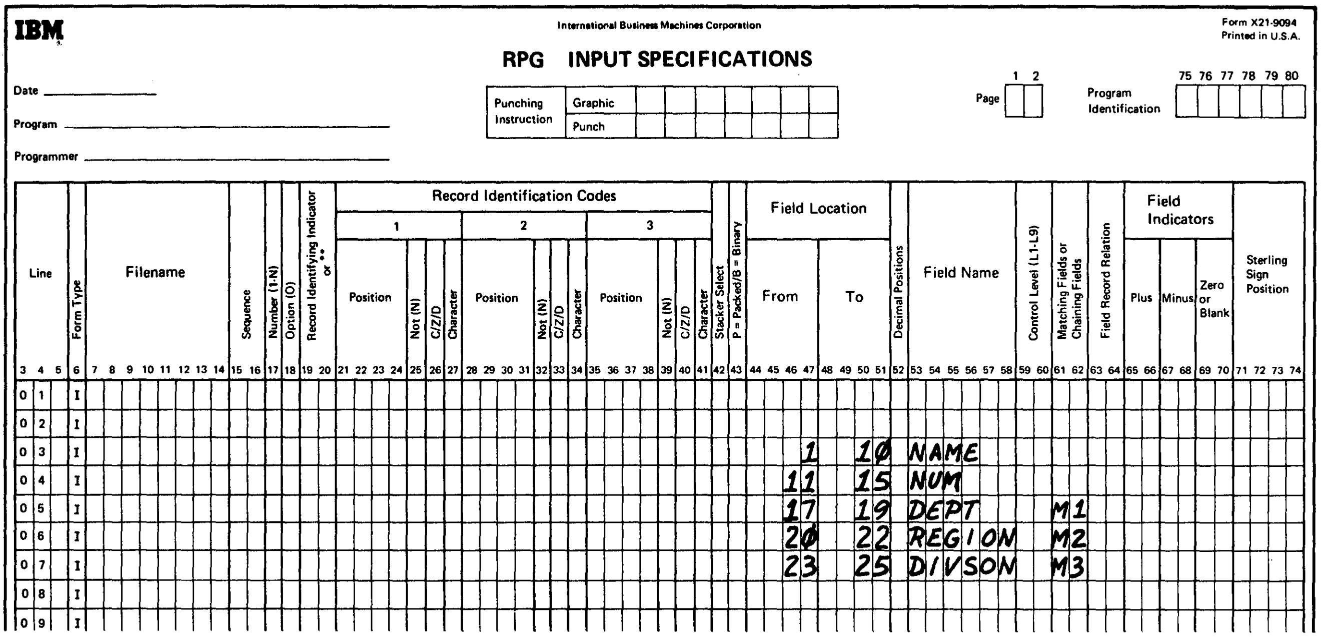
Exit to External Translate Subroutine: If the sequence of the matching fields is not the same as the collating sequence of System/360, the RPG program can provide an automatic exit to an external user subroutine that translates the sequence of the matching fields to the collating sequence of the system.

An entry in the RPG processor control card is all that is required to cause the RPG program to branch to the subroutine. The automatic branch occurs after the input card is read in and before the RPG program checks the sequence of the matching field.

The subroutine to translate the matching fields must use the predefined label ALTSEQ. The register conventions for this subroutine are the same as those for the EXIT operation. (See *Using Tables and Exit Routines in the Object Module, Exit to a User's Subroutine.*) The address of the matching field hold area will be contained in register 1. The subroutine must place the translated fields back into the matching fields hold area before it returns control to RPG.

Chaining Fields

The use of chaining files is explained in the section *Processing Multiple Input Files, Chaining*. Up to nine chaining fields are permitted in a record. In these positions enter the code that identifies the chaining field (C1 through C9).



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

Program Identification 75 76 77 78 79 80

Line	Form Type	Field Location From	Field Location To	Field Name	Matching Fields or Chaining Fields
01	I				
02	I				
03	I	1	1Ø	NAME	
04	I	11	15	NUM	
05	I	17	19	DEPT	M1
06	I	2Ø	22	REGION	M2
07	I	23	25	D/VSON	M3
08	I				
09	I				

Figure 75. Using Matching Fields to Sequence Check in a Single Input File

Data Contained in First Record

DEPT	008
REGION	051
DIVSON	003

Data Contained in Second Record

DEPT	003
REGION	025
DIVSON	005

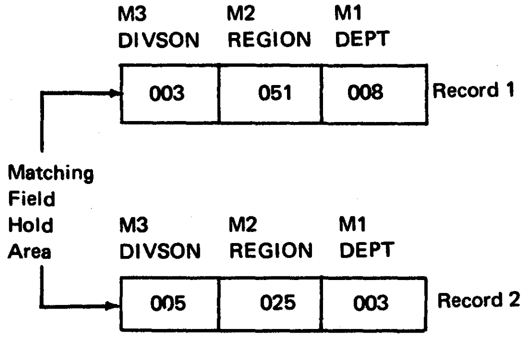


Figure 76. Comparing Matching Fields

Field Record Relation (63-64)

This specification is used when there are records in an OR relationship and the fields of the records are not in the same location. Enter in positions 63-64 the appropriate record identifying indicator which will be on when the field is used. An explanation of the use of this specification was contained in *Field Location, Records in an OR Relationship* in this section, and an example of this specification is provided in Figure 70.

Using Indicators U1-U8 as Field Record Relation Indicators

Indicators U1-U8 may be used for a special purpose in the Field Record Relation positions. If one of these indicators is used, the field is not processed when the indicator is off. If the indicator is on, the field is processed like a normal field. These indicators will be set at object time before any records are processed. See the section *Specifying the Kind of Calculation* for a further explanation of indicators U1-U8.

Note: U1-U8 may not be used as field record relation indicators when a field is a matching field or a control level field.

Using Field Record Relation with Chaining Files

An additional function of this specification is to selectively control chaining operations. (See *Processing Multiple Input Files, Chaining* for a general explanation of chaining. In order to understand this function, readers should be familiar with chaining operations and with the use of the Calculation Specifications form.)

If a chaining field is specified on a field description line and Field Record Relation is blank, the chained record will be obtained whenever the record type (for the chaining field) is present. However, if a record identifying indicator is placed in Field Record Relation, then the chained record will be obtained only if the record type is present and the record identifying indicator (in Field Record Relation) is on.

This feature provides the programmer with the ability to use chaining files on a selective basis. The function of this feature of Field Record Relation is similar to the function of controlling calculations by the status of record identifying indicators.

A variation of the use of this function of Field Record Relation is to control two chained files with one chaining field. For example, the chaining field would be specified twice on the Input Specifications form. Each specification line would be conditioned by a separate record identifying indicator. The particular chained record obtained would depend upon the status (on or off) of the appropriate record identifying indicator.

Field Indicators (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 sets on an indicator that can be used to control calculation and output specifications, or even to stop the processing of the object program.

The entry for the specification is an indicator which will be set on when the field specified on the line is plus, minus, or zero or blank. For more information, see *Indicators.*

Field Status Conditions

Plus: A plus condition occurs when the value of a numeric field is greater than 0.

Minus: A minus condition occurs when the value of the numeric field is less than zero.

Zero or Blank: A zero or blank condition occurs if a numeric field contains all zeros or blanks, or if an alphameric field contains all blanks. It is set on if a numeric field is either +0 or –0.

Note: For alphameric fields, positions 65 through 68 must be blank.

Types of Indicator Codes Used

A two-digit field indicator code is used for this specification. These codes, ranging from 01 to 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 on the Input Specifications form in Record Identifying Indicators or in Field Indicators. This should not be confused with using these indicators. Using these indicators means specifying them in Indicators on the Calculation Specifications form or in Output Indicators on the output form as many times as required. In the latter case, they are merely tested to determine their status and not reset by the test.

Halt Indicators

There are ten additional indicator codes, known as halt indicators that can be used in the RPG program. These indicators, designated as H0 through H9, halt the processing of the object program when error conditions (as determined by the programmer) have been detected. These indicators can also be used in Calculation and Output-Format Specifications forms to control specifications and stop processing. For example, the status of a field can be tested and, depending upon the results of the tests, a halt indicator may be set on which terminates the object program.

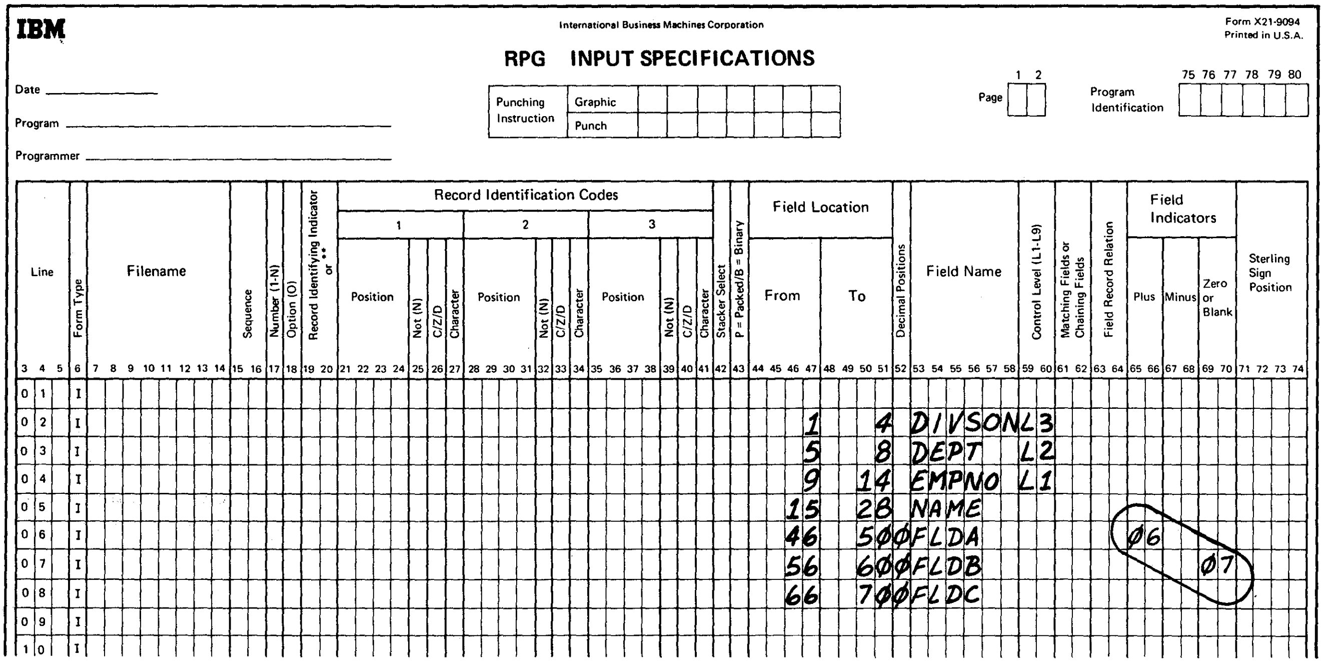
If one of these indicators has been set on during the processing of a record, the object program terminates at the completion of the processing of that record. However, processing will not be interrupted if a halt indicator that has been set on is set off (in the program) before the program attempts to read the next input record.

The H0 indicator can also be set on automatically by the RPG program. The conditons that cause H0 to be set on automatically are listed in Appendix D. Unless H0 is set off by a SETOF operation (see *Setting Indicators On or Off, Calculation Specifications Form*), the program terminates before the next input record is read.

Use of Field Indicators

Field Indicator 07 in Figure 77 is used to determine if FLDB contains zeros or blanks; indicator 06 is used to determine if the value contained in FLDA is positive. Indicator 07 and 06 would both be used to control functions in the calculation and output specifications that must be altered or modified depending on the status of FLDA and FLDB.

Note: Use of indicators H0-H9 as field indicators testing zero or blank (69-70) causes the program to terminate as the first record is read because any indicators used in (69-70) are initialized on.



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

Program Identification 75 76 77 78 79 80

Line (3-5)	Form Type (6)	Filename (7-14)	Sequence (15-16)	Number (1-N) (17)	Option (O) (18)	Record Identifying Indicator or ** (19-20)	Record Identification Codes 1 Position (21-24)	Not (N) (25)	C/Z/D (26)	Character (27)	Record Identification Codes 2 Position (28-31)	Not (N) (32)	C/Z/D (33)	Character (34)	Record Identification Codes 3 Position (35-38)	Not (N) (39)	C/Z/D (40)	Character (41)	Stacker Select (42)	P = Packed/B = Binary (43)	Field Location From (44-47)	Field Location To (48-51)	Decimal Positions (52)	Field Name (53-58)	Control Level (L1-L9) (59-60)	Matching Fields or Chaining Fields (61-62)	Field Record Relation (63-64)	Field Indicators Plus (65-66)	Field Indicators Minus (67-68)	Field Indicators Zero or Blank (69-70)	Sterling Sign Position (71-74)
0 1	I																														
0 2	I																				1	4		DIVSON	L3						
0 3	I																				5	8		DEPT	L2						
0 4	I																				9	14		EMPNO	L1						
0 5	I																				15	28		NAME							
0 6	I																				46	5Ø	Ø	FLDA				Ø6			
0 7	I																				56	6Ø	Ø	FLDB						Ø7	
0 8	I																				66	7Ø	Ø	FLDC							
0 9	I																														
1 0	I																														

Figure 77. Example of Field Indicators Specifications

How Field Indicators are Set Off and On

Indicators used for testing for plus and minus conditions, are set (turned on or off) if their respective conditions occur when a record is read into the system. Each field indicator is related to only one record type. Therefore, the indicators are not reset (turned on or off) until the related record type is read again or until the indicator number is defined in some other specification. One or more field indicators can be set on at one time.

Indicators used for testing for zero or blank conditons are set in the same manner as those used for testing for plus and minus conditions. They can, however, be reset by one other condition. The output specification Blank After causes a field to be set to blanks or zeros (depending upon whether the field is alphabetic or numeric) after execution of the output operation specified. If the field being reset to blanks or zeros is an input field that is being tested for zero or blank, the field indicator specified for it is set on when the field is set to blanks or zeros by the Blank After specification.

Any plus or minus indicators associated with the field are not set off by virtue of the Blank After specification. All zero or blank indicators are initialized on at the beginning of the program to reflect the initial status of the associated fields. (All fields defined in an RPG program are initialized to zero if numeric or blank if alphameric.) The indicators remain on until the status of the associated fields change as a result of an input record being read or a calculation being performed.

Sterling (71-74)

Enter in these positions the place in the record that contains the sign of the sterling field. If the sign is in the normal place, enter an S in position 74. Leading zeros may be omitted. Leave these positions blank if the sterling specification is not used. Additional information on sterling is in *Sterling Routines for RPG.*

SUMMARY OF INPUT SPECIFICATIONS

This concludes the description of the input specifications. Additional information on matching or chaining fields may be found in *Processing Multiple Input Files.* The input specifications listed below are used with the Calculation and Output-Format Specifications forms.

1. Record Identifying Indicator
2. Field Name
3. Control Level
4. Matching Fields
5. Field Indicators

The use and function of these specifications may become more apparent to the reader after the descriptions of the calculation and output specifications have been read.

This form is used to specify the operations to be performed by the generated program upon the input data and upon data obtained as a result of other calculations. Two general rules govern the writing of calculation specifications:

1. Each operation is specified on one line of the form. Operations must be listed in the order in which they are to be performed in the program.

2. Detail calculations must precede all total calculations. Calculations within these two groups must be specified in the order in which they are to be performed on the data.

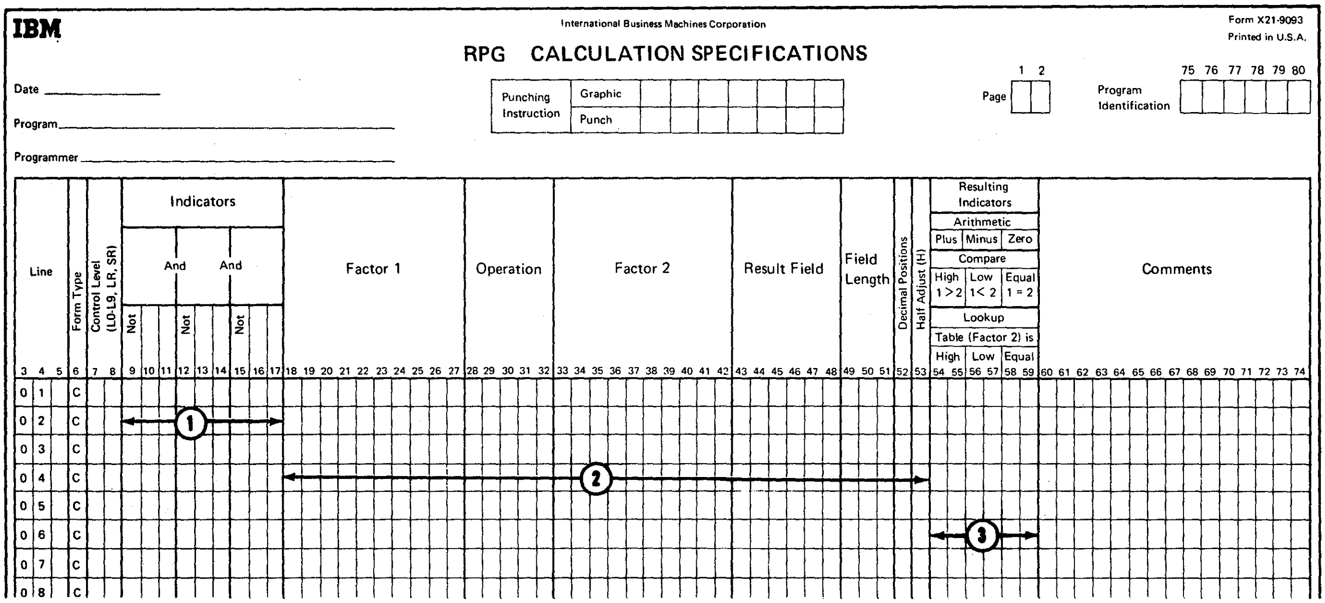
The calculation form is divided into three categories as shown here:

The three categories contain the following kinds of information:

1. When calculations are to be performed. These entries (positions 7-17) determine when calculations are to be performed and upon what conditions they are to be performed.

2. What type of calculations are to be performed. These entries (positions 18-53) determine the type of calculations to be performed, such as add, subtract, multiply, etc. These entries also supply information about the result of the calculations.

3. What tests are to be made on the results of the calculations. These entries (positions 54-59) test the results of the calculations to modify subsequent calculations or output 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 (1 2)

Program Identification (75 76 77 78 79 80)

Line (3 4 5)	Form Type (6)	Control Level (L0-L9, LR, SR) (7 8)	Indicators: Not, And Not, And Not (9–17)	Factor 1 (18–27)	Operation (28–32)	Factor 2 (33–42)	Result Field (43–48)	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 (60–74)
0 1	C											
0 2	C		← 1 →									
0 3	C											
0 4	C			←		2				→		
0 5	C											
0 6	C										← 3 →	
0 7	C											
0 8	C											

SPECIFYING WHEN CALCULATIONS ARE TO BE PERFORMED

The two specifications in this category are Control Level and Indicators.

Control Level (7-8)

Calculations are performed at either detail time or total time. Detail calculations are performed on every record read. Total calculations are performed only when a control level change occurs.

A test for a control change is made after each record is read into the system. Total calculations are performed after this test is made and before the record that caused the control change is processed.

An entry in positions 7-8 indicates the calculation is to be performed at total time. (A blank entry in these positions indicates the calculation is to be performed at detail time.)

The entries for this specification are control level indicators L1 through L9, and indicators L0 and LR.

Control level indicators are set on by control breaks. Whenever a control break occurs, the indicator for the new control level and all indicators for the lower-order control levels are set on at the same time. A change to control level 3, for instance, causes indicators L3, L2, and L1 to be set on.

A control level indicator remains on during total time and during the subsequent detail time, which includes both the calculating and the printing of the detail record.

If an object module has only detail calculations, this specification is left blank. The indicators that control detail calculations are specified in Indicators (positions 9-17). However, positions 9-17 may also be used to control total calculations.

The next two paragraphs describe indicators LR (last record) and L0 (level zero).

LR (Last Record) Indicator

This indicator is set on after the last input record has been read and calculated and after the appropriate detail records are put out. At this time control level indicators L1 through L9 are also set on.

If there is more than one input file, the programmer determines which files are to be checked for the last record. This is accomplished in the File Description Specifications form.

LR is set on when all files with an E in End of File position (17) on the file description specifications have been completely read.

L0 (Level Zero) Indicator

The L0 indicator is on throughout execution of the object module. It specifies total calculations to be performed at a time when no control break has occurred.

The L0 indicator can be used, for instance, to accumulate a total for each page in a report even though there is no control break at the end of a page.

Figure 78 illustrates six control level specifications.

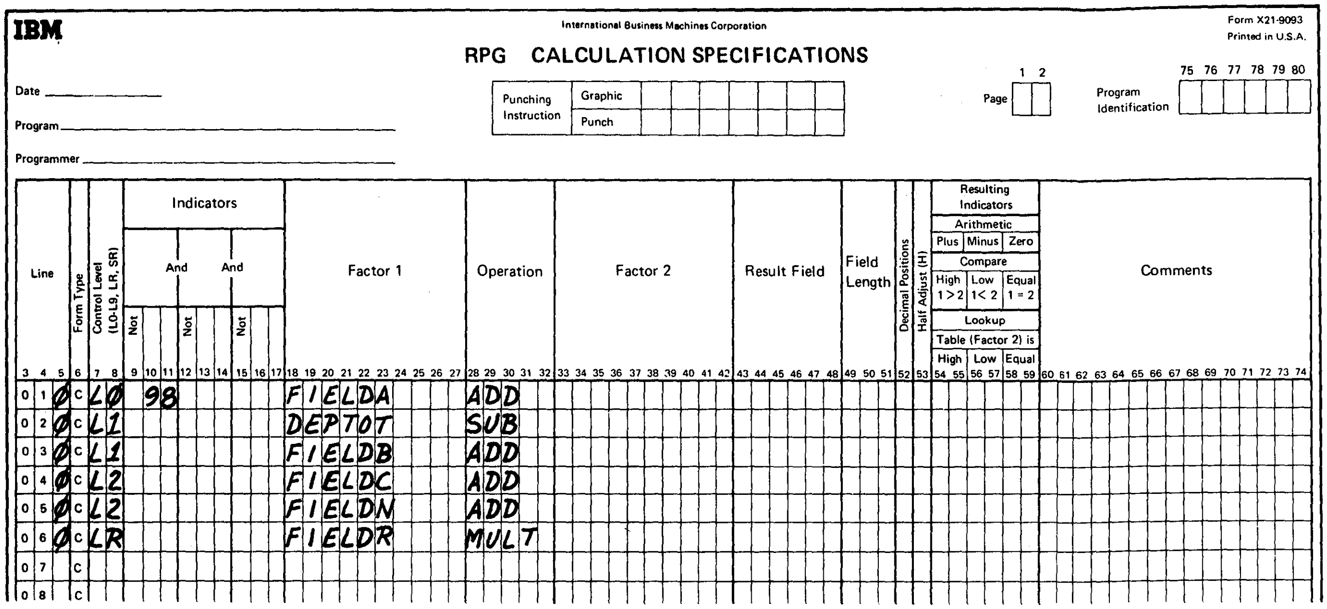
Total calculations (and total output) are processed at the beginning of the execution of an object program as follows:

1. If no control fields are specified on the input specifications for any records, total calculations and total output lines are bypassed for the first record read, because no data is on the first record.

2. If control fields are specified on the input specifications for at least one record type, processing of total calculations and total output is bypassed until the record following the first record with specified control fields is read.

   For example, if the first record read is for control purposes (for example, a date card) and the second record has specified control fields, no total calculations or total output are performed for the second record. The third record causes total calculations and total output to be tested. (This function prevents total calculations and total output for the first record with control levels). Total calculations or total output conditioned by L0 (or total output lines without L1-L9 specified) are performed whether or not a record with control levels has been processed.

3. If the first record processed has a control field of all blanks (or all zeros if a numeric field), a control break will occur when the card is processed.



IBM

International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

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

Date

Program

Programmer

Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line (3-5)	Form Type (6)	Control Level (L0-L9, LR, SR) (7-8)	Indicators: Not (9)	(10-11)	And: Not (12)	(13-14)	And: Not (15)	(16-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field (43-48)	Field Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Arithmetic Plus / Compare High 1>2 / Lookup Table (Factor 2) is High (54-55)	Minus / Low 1<2 / Low (56-57)	Zero / Equal 1=2 / Equal (58-59)	Comments (60-74)
01∅	C	L∅		98					FIELDA	ADD									
02∅	C	L1							DEPTOT	SUB									
03∅	C	L1							FIELDB	ADD									
04∅	C	L2							FIELDC	ADD									
05∅	C	L2							FIELDN	ADD									
06∅	C	LR							FIELDR	MULT									
07	C																		
08	C																		

Figure 78. Using Control Level Specifications

Indicators (9-17)

Entries in positions 9-17 indicate conditions that control the calculations specified in positions 18-59. Indicators can be specified in these positions for both total and detail calculations. From one to three indicator codes can be specified to be in an AND relationship. In an AND relationship, if three indicator codes are specified, all three indicator conditions must be satisfied before the calculation can take place. It is impossible to have an OR condition with this specification.

Enter in positions 10-11, 13-14, and 16-17 the indicators that determine when the calculation is to be performed. If an indicator must be off, enter an N in either position 9, 12, or 15 (whichever is appropriate).

The specifications used in these positions can be arranged in the following categories:

1. If positions 9-17 and positions 7-8 are blank, the calculation specified on the line is performed each time a detail record is read.

2. A record identifying indicator code determines the particular record type on which the calculation is to be performed. This calculation will not be performed on any other record type.

3. A field indicator code controls the calculation according to the status of an input field.

4. A resulting indicator code controls the calculation by conditions that occurred on previous calculations. (This feature is shown on the Calculation Specifications form, positions 54-59 and discussed under *Indicators*.)

5. Control level indicator L1-L9, used with a particular record identifying indicator, permits the calculation to be performed at detail time, but it is performed only on the first record of the control level specified.

6. The MR, matching record, indicator code means that the calculation is performed only if there is a matching record in a second input file.

7. Halt indicators H0 through H9 are normally used to terminate the program or to suppress a calculation when an error has been detected on a previous calculation.

8. The overflow indicators permit the calculation to tak place only if a page overflow has occurred.

9. All total calculations will be bypassed until the first control break has occurred.

10. The use of indicators U1-U8 permits calculations to be controlled by indicators that are set externally. See *Specifying the Kind of Calculation* in this sectio to learn how these indicators are set on and off.

In addition, this specification entry (positions 9-17) may contain a combination of some of the preceding categories. Also, a calculation may be conditioned by the fact that an indicator must not be on.

Figure 79 shows entries that may be made in positions 9-17. The numbers to the right of the entries refer to the following list:

1. The first example is a blank entry. This means that the calculation is performed on every detail record.

2. In this example, indicator 16 could be a record identifying indicator for a specific record type.

3. In this example, indicator 16 could be a record identifying indicator and indicator 18 could be a field indicator, which is specified on the Input Specifications form. If indicator 18 is used to test an input field for blanks, this entry means that the calculation would be performed if record type 16 is present and the contents of the field represented by indicator 18 is not blank.

4. This example is similar to the previous example. However, indicator 19 could be a resulting indicator set on by the previous calculation. The calculation specified on this line in positions 18-59 would not be performed unless indicator 19 was also on.

5. This entry means that the calculation is performed at detail time on control level 1, and only if indicator 24 is on.

6. This entry means that the calculation is performed only if indicator 16 is on and there is a matching record conditon. For example, when fields from detail records are multiplied by a factor contained in a master record, the program must have a way of ensuring that the detail record has been matched with the appropriate master record.

7. The entry NH1 prevents the object program from performing the calculation if an error conditon has occurred. Note that when an error occurs, the job is not terminated until after all processing for the record has been completed. This facility is provided so that the programmer can prevent the calculation of erroneous data.

8. This entry means that the calculation is performed on the detail cycle following an overflow condition.

9. The entry NU1 prevents the object program from performing the calculation if indicator U1 has been externally set on.

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 indicator. Figure 80 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 set off each time or the addition will not be conditioned correctly.

SPECIFYING THE KIND OF CALCULATION

This section describes the kind of calculations to be performed. These specification answer the following four questions:

1. What fields are to be used (Factor 1, Factor 2, Operation, and Result)?

2. What fields are to be acted upon (Factor 1, Factor2, and Result)?

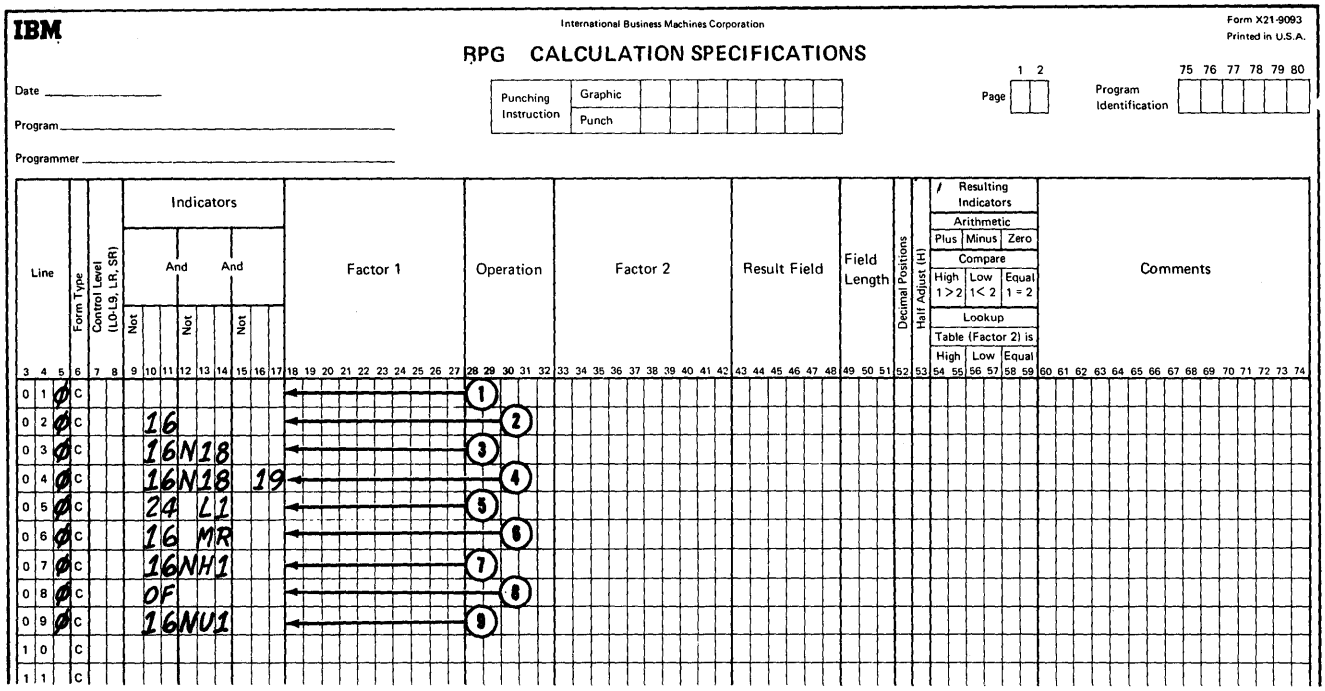
3. What is the operation (add, subtract, multiply, etc.)?

4. What is done with the result?

Factor 1 (18-27)

This specification can be a field name, a literal, or a label and is the same as Factor 2 (positions 28-32). If it is a field name, it must have been defined on the Input Specifications form, or it must be defined in the Result Field of a calculation. The field name must be left justified, and the first character must be alphabetic. The operation code used (positions 28-32) determines which factors are to be used and the type of entries allowed.

Note: The field names UDATE, UMONTH, UDAY, and UYEAR may be used in Factor 1 with the exception of operations TAG, EXTCV, and RPGCV.



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

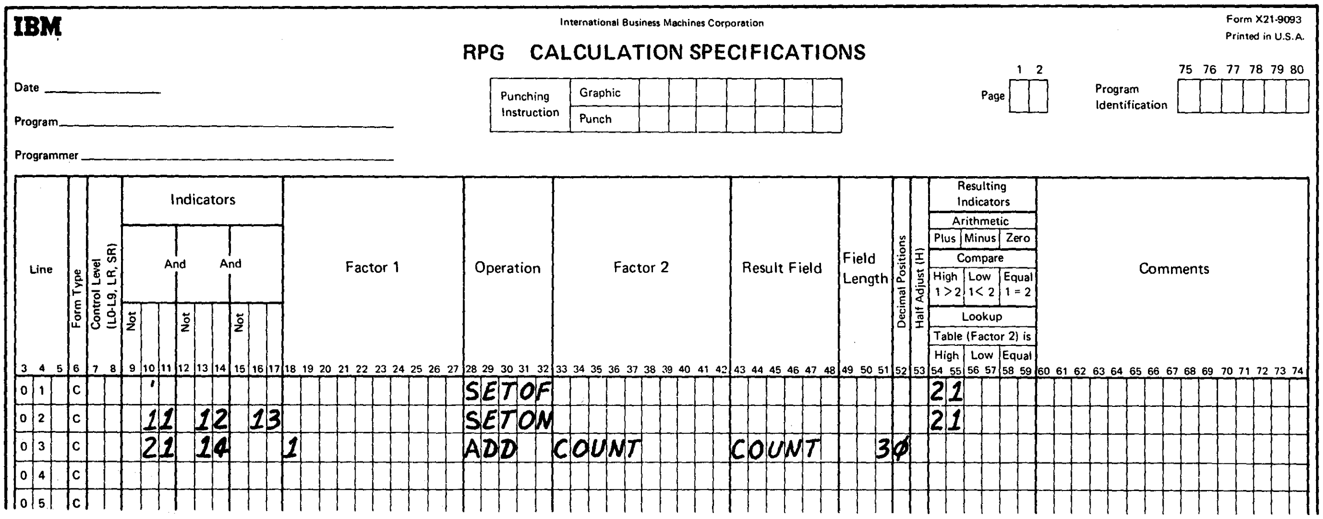
RPG CALCULATION SPECIFICATIONS

Date ______ Program ______ Programmer ______

Punching Instruction: Graphic / Punch — Page 1 2 — Program Identification 75 76 77 78 79 80

Line (3-5)	Form Type (6)	Indicators: Not (9-11)	And: Not (12-14)	And: Not (15-17)	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 Plus/High 1>2 (54-55)	Minus/Low 1<2 (56-57)	Zero/Equal 1=2 (58-59)	Comments (60-74)
01∅	C														
02∅	C	16													
03∅	C	16	N18												
04∅	C	16	N18	19											
05∅	C	24	L1												
06∅	C	16	MR												
07∅	C	16	NH1												
08∅	C	OF													
09∅	C	16	NU1												
10	C														
11	C														

Figure 79. Using Indicators



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 (3-5)	Form Type (6)	Indicators: Not (9-11)	And: Not (12-14)	And: Not (15-17)	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 Plus/High 1>2 (54-55)	Minus/Low 1<2 (56-57)	Zero/Equal 1=2 (58-59)	Comments (60-74)
01	C					SETOF						21			
02	C	11	12	13		SETON						21			
03	C	21	14		1	ADD	COUNT	COUNT	3	∅					
04	C														
05	C														

Figure 80. Indicators in an AND Relationship

General Description of Data Fields

1. A field name to be used in calculations must be defined in either positions 53-58 (Field Name) of the Input Specifications form, or positions 43-48 (Result Field) of the Calculations Speoifications form. The first character of a field name must be alphabetic, and the name must not exceed six characters. Special characters and embedded blanks must not be used.

2. Literals are the actual data to be used, 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, and the label 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 digits 0 through 9. One decimal symbol and/or one plus or minus sign may be used. Other separators (such as a comma between the thousands and hundreds positions) may not be used. The European format of commas to denote decimal points 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 leftmost character. An unsigned literal is considered positive.

4. The decimal symbol may appear anywhere in the literal.

. Numeric Literals must not be enclosed in apostrophes.

'phameric Literals: An alphameric literal consists of ıaracters 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 is coded as 'o''clock'.

Figure 81 illustrates entries for Factor 1. The numbers in circles refer to the items listed below.

1. GROSS could be a field name specified on the input form.

2. NETAMT could have been specified as the result field of the previous calculation.

3. This numeric literal could be used in the program to determine if specific fields in the input records were higher or lower than this number. The position of the decimal symbol must be indicated if the number is not a whole number.

4. Alphameric literals, like the one in this example, can be used to compare against a data field in the input records to perform certain types of calculations upon records representing the month of January.

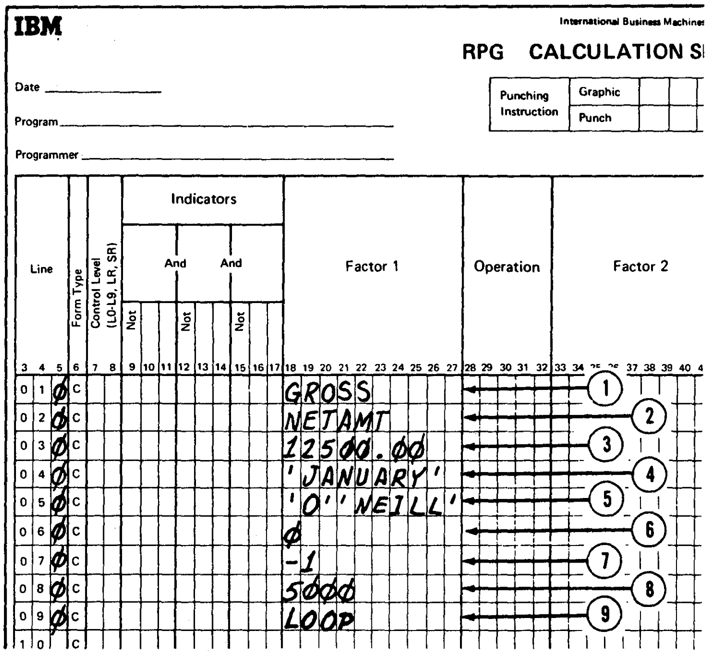
5. This shows an apostrophe within an alphameric literal. The literal is printed as O'NEILL.

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 is used since no decimal point is included.

9. A label name of LOOP is specified. This is used in a TAG statement.



IBM

International Business Machines

RPG CALCULATION S

Date

Program

Programmer

Punching Instruction	Graphic
	Punch

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators	Factor 1	Operation	Factor 2
01Ø	C			GROSS		
02Ø	C			NETAMT		
03Ø	C			125ØØ.ØØ		
04Ø	C			'JANUARY'		
05Ø	C			'O''NEILL'		
06Ø	C			Ø		
07Ø	C			-1		
08Ø	C			5ØØØ		
09Ø	C			LOOP		
10	C					

And
And
Not
Not
Not
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
1
2
3
4
5
6
7
8
9

Figure 81. Examples of Factor 1 Entries

Factor 2 (33-42)

The description of Factor 1 also applies to Factor 2. The field names UDATE, UMONTH, UDAY, and UYEAR may be used in Factor 2 with the exception of operations EXTCV, GOTO, EXIT, and LOKUP.

Summary of Factor 1 and Factor 2

1. Enter either the name or the literal that is Factor 1 or Factor 2 in positions 18-27 or 33-42.
2. If Factor 1 or Factor 2 contains the name of a field, the field must be defined in either:
   a. Positions 53-58 (Field Name) of the Input Specifications form.
   b. Positions 43-48 (Result Field) of the Calculation Specifications form.
3. A name cannot exceed six characters. Special Characters and blanks must not be used.
4. A numeric literal cannot exceed ten characters; an alphameric literal cannot exceed eight characters.
5. Entries in Factor 1 or Factor 2 must be left justified.

Operation (Positions 28-32)

Entries in these positions specify the operations to be performed using Factor 1, Factor 2, and Result Field. Each operation is specified by placing the operation code in Operation (positions 28-32).

Arithmetic Operations	*Code*
Add	ADD
Zero and Add	Z-ADD
Subtract	SUB
Zero and Subtract	Z-SUB
Multiply	MULT
Divide	DIV
Move Remainder	MVR

Move Operations	*Code*
Move	MOVE
Move Left	MOVEL
Move High-to-Low Zone	MHLZO
Move Low-To-High Zone	MLHZO
Move High-to-High Zone	MHHZO
Move Low-to-Low Zone	MLLZO

Testing or Compare Operations	*Code*
Compare	COMP
Test Zone	TESTZ

Branching and Exit Operations	*Code*
Branching (or GOTO)	GOTO
Providing a Label for GOTO	TAG
Exit to a Subroutine	EXIT
RPG Label	RLABL
User Label	ULABL

Setting Indicators On or Off	*Code*
Set Indicators On	SETON
Set Indicators Off	SETOF

Table Operations	*Code*
Table Lookup	LOKUP

Conversion Routine Operations	*Code*
RPG Conversion Routine	RPGCV
End of RPG Conversion	ERPGC
External Conversion Routine	EXTCV
Record Key	KEYCV

Arithmetic Operations

The fields or literals involved in these operations may contain numeric characters only. All arithmetic operations are performed with automatic decimal alignment. Decimal alignment causes numeric fields to be aligned to the decimal point before performing the specified operation as shown in the following examples:

Example A	*Example B*
100.1	15.
+ 1.234	− 1.7
101.334	13.3

The specified result field length and decimal position determine the length and decimal position of the result field value as shown in the following:

Number of positions:	6	5	3
Number of decimals:	3	1	0
Result of Example A:	101.334	0101.3	101.
Result of Example B:	013.300	0013.3	013.

Resulting indicators can be used with all arithmetic operations.

No arithmetic overflow will be sensed by RPG. The length of a field involved in arithmetic operations can be up to 15 digits (this includes decimal alignment when necessary). The resulting field length after decimal alignment must not be greater than 15 digits. 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 Field.

Add (ADD): This operation causes the contents of the field or the literal in Factor 2 to be added to the contents of the field or literal in Factor 1. The result of the operation is placed in the result field specified in Result Field (positions 43-48).

Zero and Add (Z-ADD): This operation causes the field specified in Result Field to be set to zeros. It then causes the data contained in the numeric literal, or the field in Factor 2, to be placed in the Result Field. Factor 1 is not used in this operation. A typical method of setting the Result Field to zero is to use a Z-ADD with Factor 2 having a numeric literal of 0.

Subtract (SUB): This operation causes the contents of the field or literal in Factor 2 to be subtracted from the contents of the field or literal in Factor 1. The result of this operation is placed in the Result Field specified.

Zero and Subtract (Z-SUB): This operation causes the negative of the number contained in the literal or the field in Factor 2 to be placed in the Result Field specified. This operation is performed after the Result Field has been set to zeros. Factor 1 is not used in this operation. This operation is commonly used to change the sign of a field.

Multiply (MULT): This operation causes the contents of the field or literal in Factor 1 to be multiplied by the contents of the field or the literal in Factor 2. The result of this operation is placed in the Result Field specified.

Divide (DIV): This operation causes the contents of the field or literal in Factor 1 to be divided by the contents of the field or literal in Factor 2. The result of this operation (quotient) is placed in the specified Result Field. This may also be thought of as A ÷ B=C. The contents of the field or the literal in Factor 2 cannot be zero.

If Factor 2 is zero, a program check and an abnormal end of job results.

The following field length restrictions apply to this operation:

$$L_1 + (D_2 - D_1 + D_r) \leq 15$$

$$L_2 - (D_2 - D_1 + D_r) \leq 15$$

and if half adjusting is specified:

$$L_1 + (D_2 - D_1 + D_r) \leq 14$$

where

L_1 = length of Factor 1 (dividend)

L_2 = length of Factor 2 (divisor)

D_1 = decimal positions of Factor 1

D_2 = decimal positions of Factor 2

D_r = decimal positions of Result Field.

Note: Invalid results are obtained if the formula is violated.

Any remainder that results from this operation is lost unless the move remainder operation is specified as the next operation in the program.

Note: If a move remainder operation follows a divide operation, the result in the divide operation cannot be half adjusted.

Move Remainder (MVR): This operation moves the remainder from a divide operation to a separate field. If MVR is used, it must immediately follow the divide operation. The divide may not be half adjusted. Figure 82 shows an example of the MVR operation. The remainder is placed in a field named STORE.

The field that is to contain the remainder must be specified in Result Field. Resulting indicators may not be specified on the MVR operation.

The value of the remainder can be determined by the following formula:

R = Dividend – (Divisor x Quotient)

For the above equation to be valid in the divide operation involving factors containing decimal positions, the Result Field that is to contain the remainder must provide for the decimal positions in the remainder based on the sum of $(D_2 + D_r)$ or D_1, whichever is greater.

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 (positions 7-8 blank), the MVR operation must also be performed at detail time.

Example of Divide and Move Remainder: Consider dividing 11 by .76 and requesting an answer in two decimals:

```
         14.47
  76. ┌1100.00
       76
       340
       304
        360
        304
         560
         532
          28 = Remainder
```

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

```
   14.47
   x0.76
    8682
  10129
 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 four decimal places.

The number of decimals placed 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.

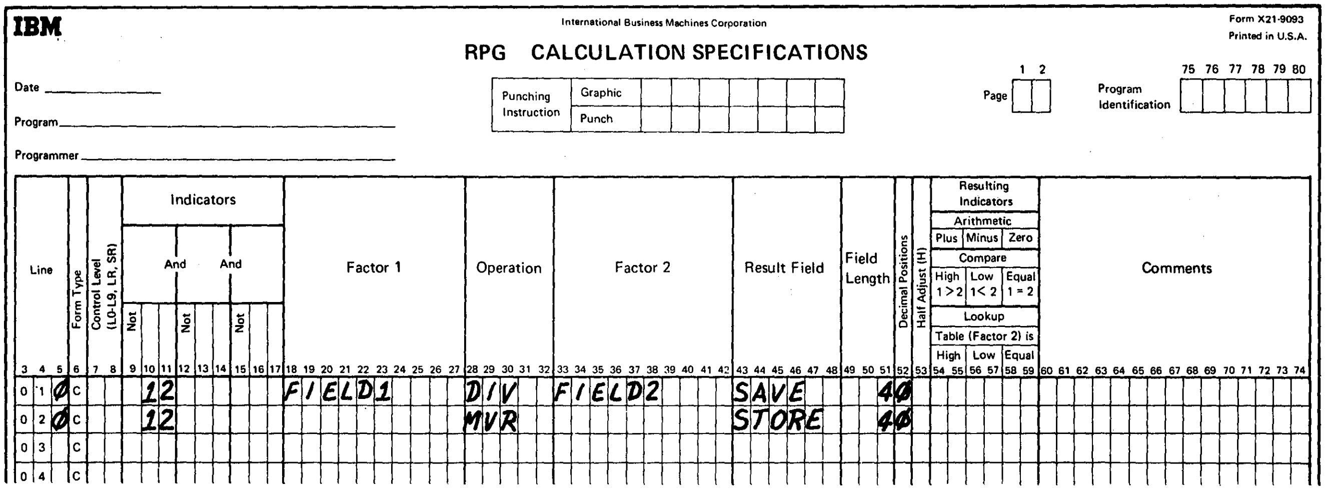
The MVR operation performs decimal alignment. Consequently, the result field could contain the following depending on the field length and decimal positions described:

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 83 illustrates some sample arithmetic operations. The actual result of each operation is shown in the Comments position (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.



IBM

International Business Machines Corporation

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

RPG CALCULATION SPECIFICATIONS

Date ______

Program ______

Programmer ______

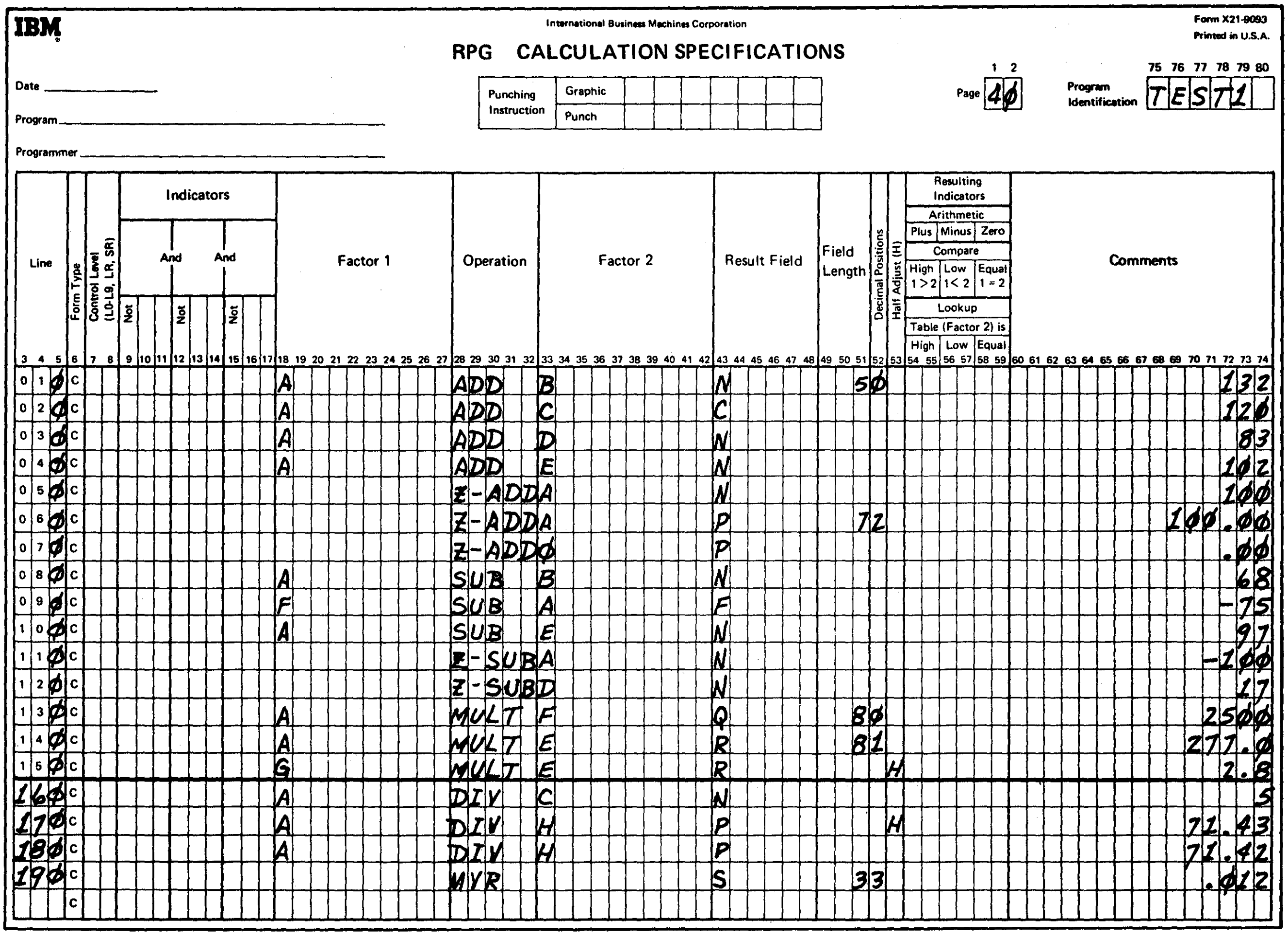
Punching Instruction: Graphic / Punch

Page (1 2)

Program Identification (75 76 77 78 79 80)

Line (3-5)	Form Type (6)	Control Level (L0-L9, LR, SR) (7-8)	Indicators (9-17)	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: Plus / High 1>2 (54-55)	Minus / Low 1<2 (56-57)	Zero / Equal 1=2 (58-59)	Comments (60-74)
01Ø	C		12	FIELD1	DIV	FIELD2	SAVE	4	Ø					
02Ø	C		12		MVR		STORE	4	Ø					
03	C													
04	C													

Figure 82. Using MVR Operation Code



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 TEST1

Line (3-5)	Form Type (6)	Control Level (L0-L9, LR, SR) (7-8)	Indicators (9-17)	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: Plus / High 1>2 (54-55)	Minus / Low 1<2 (56-57)	Zero / Equal 1=2 (58-59)	Comments (60-74)
01Ø	C			A	ADD	B	N	5	Ø					132
02Ø	C			A	ADD	C	C							12Ø
03Ø	C			A	ADD	D	N							83
04Ø	C			A	ADD	E	N							1Ø2
05Ø	C				Z-ADD	A	N							1ØØ
06Ø	C				Z-ADD	A	P	7	2					1ØØ.ØØ
07Ø	C				Z-ADD	Ø	P							.ØØ
08Ø	C			A	SUB	B	N							68
09Ø	C			F	SUB	A	F							-75
10Ø	C			A	SUB	E	N							97
11Ø	C				Z-SUB	A	N							-1ØØ
12Ø	C				Z-SUB	D	N							17
13Ø	C			A	MULT	F	Q	8	Ø					25ØØ
14Ø	C			A	MULT	E	R	8	1					277.Ø
15Ø	C			G	MULT	E	R			H				2.8
16Ø	C			A	DIV	C	N							5
17Ø	C			A	DIV	H	P			H				71.43
18Ø	C			A	DIV	H	P							71.42
19Ø	C				MVR		S	3	3					.Ø12
	C													

Figure 83. Summary of Arithmetic Operation Codes

Move Operations

For the MOVE and MOVEL operations, numeric fields may be changed to alphameric fields, or alphameric fields may be changed to numeric fields. To change a numeric field to an alphameric field, Factor 2 must be numeric, and the Result Field must be specified as alphameric. No decimal alignment is performed when a move operation is used. When a field is changed from numeric to alphameric in a move operation, the sign of the field is included in the Result Field.

When an alphameric field is moved to a numeric field, the sign of the alphameric field is retained. If a particular sign is desired in the numeric field, it must be moved to the sign position from a previously defined numeric field by a move zone operation (MLLZO).

Move (MOVE): This operation code causes data characters (starting at the rightmost position) to be moved from the field or literal contained in Factor 2 to the rightmost positions of the Result Field. Factor 1, half adjust, and resulting indicators are not used in this operation.

If Factor 2 is longer than the Result Field, the excess leftmost positions of Factor 2 are not moved (figure 84).

If the Result Field is longer than the field specified by Factor 2, the positions to the left of the data that is moved remain undisturbed (Figure 85).

Move Left (MOVEL): This operation code causes data characters (starting at the leftmost position) to be moved from the field or literal contained in Factor 2 to the leftmost positions of the Result Field. Factor 1, half adjust, and resulting indicators are not used in this operation.

If Factor 2 is longer than the Result Field, the excess rightmost positions of Factor 2 are not moved (Figure 86).

If the Result Field is longer than the field specified by Factor 2, the positions to the right of the data that is moved remain undisturbed.

When moving data to a numeric field, the determination of the sign is an important consideration. In Figure 87, inserts B and C, the underlined digitis in Factor 2 indicate the position of the sign. If the Result Field is longer than Factor 2, as shown in Figure 87, insert B, the original sign of the Result Field is retained. However, if Factor 2 is as long or longer than the Result Field as shown in Figure 87, insert C, the sign of Factor 2 is assumed by the Result Field and is indicated by the underlined digit in the Result Field.

Factor 1 is not referenced by this operation.

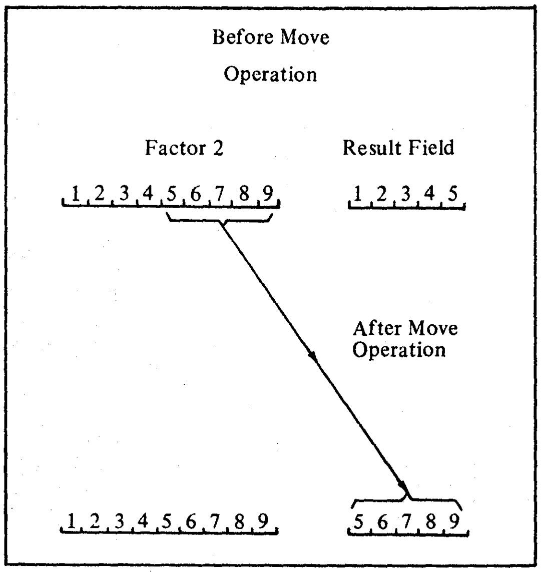


Figure 84. Move Operation-Factor 2 Longer than Result Field

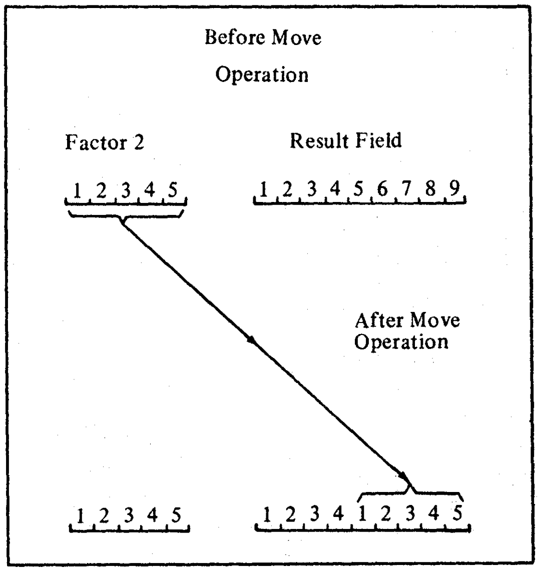
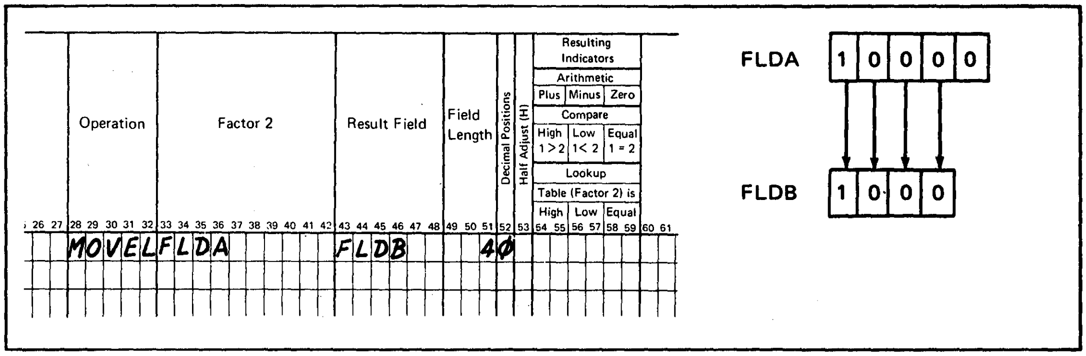
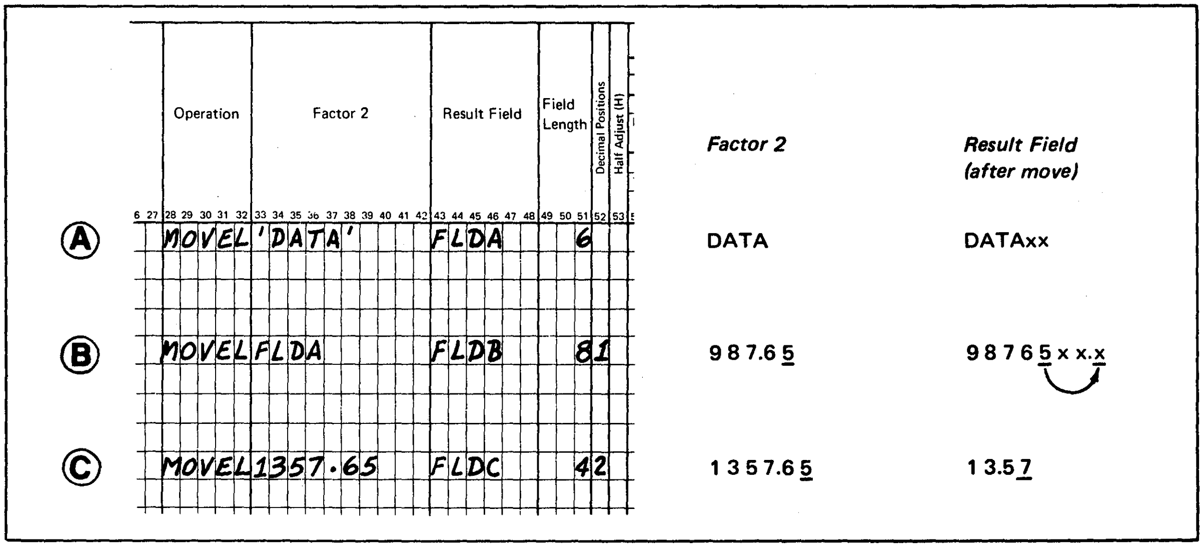


Figure 85. Move Operation-Factor 2 Shorter than Result Field



Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)
MOVEL	FLDA	FLDB	4	Ø	

Figure 86. Using the MOVEL Operation Code



	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	*Factor 2*	*Result Field (after move)*
Ⓐ	MOVEL	'DATA'	FLDA	6			DATA	DATAxx
Ⓑ	MOVEL	FLDA	FLDB	8	1		987.65	98765xx.x
Ⓒ	MOVEL	1357.65	FLDC	4	2		1357.65	13.57

Figure 87. Additional Functions of the 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 specified to use the high zone (H) can only be an alphameric field.

Move High-to-Low Zone (MHLZO): This operation moves the zone at the leftmost position of Factor 2 to the rightmost position of the Result Field.

Figure 88 illustrates the movement of zones for all four move zone operations.

Factor 2 can only be alphameric. If the zone to be moved is located over numeric data, this operation can still be performed; however, the numeric field must have been specified as an alphameric field in the Input Specifications form.

MLLZO Factor 2 Result Field	Alphameric Alphameric	Bits 0-3 of rightmost byte of Factor 2 are moved to bits 0-3 of rightmost byte of Result Field.
Factor 2 Result Field	Alphameric Numeric	Bits 0-3 of rightmost byte of Factor 2 are moved to bits 4-7 of the rightmost byte of the Result Field.
Factor 2 Result Field	Numeric Alphameric	Bits 4-7 of rightmost byte of Factor 2 are moved to bits 0-3 of rightmost byte of the Result Field.
Factor 2 Result Field	Numeric Numeric	Bits 4-7 of rightmost byte of Factor 2 are moved to bits 4-7 of rightmost byte of the Result Field.
MHLZO Factor 2 Result Field	Alphameric Numeric	Bits 0-3 of leftmost byte of Factor 2 are moved to bits 4-7 of rightmost byte of Result Field.
Factor 2 Result Field	Alphameric Alphameric	Bits 0-3 of leftmost byte of Factor 2 are moved to bits 0-3 of rightmost byte of the Result Field.
MLHZO Factor 2 Result Field	Alphameric Alphameric	Bits 0-3 of rightmost byte of Factor 2 are moved to bits 0-3 of leftmost byte of Result Field.
Factor 2 Result Field	Numeric Alphameric	Bits 4-7 of rightmost byte of Factor 2 are moved to bits 0-3 of leftmost byte of the Result Field.
MHHZO Factor 2 Result Field	Alphameric Alphameric	Bits 0-3 of leftmost byte of Factor 2 are moved to bits 0-3 of leftmost byte of Result Field.

Figure 88. Move Zone Operations

53704

The Result Field can be numeric or alphameric. A Result Field specified as numeric contains an F-zone for a plus sign or a D-zone for a minus sign after this operation, regardless of the zone moved.

Figure 89 illustrates a move high-to-low zone operation (alphameric to alphameric).

Move Low-to-High Zone (MLHZO): This operation moves the zone at the rightmost position of Factor 2 to the leftmost position of the Result Field. Factor 2 can be numeric or alphameric, but the Result Field must be alphameric. Any zone may be moved.

Move High-to-High Zone (MHHZO): This operation moves the zone at the leftmost position of Factor 2 to the leftmost position of the Result Field. Factor 2 and the Result Field must be alphameric. Any zone may be moved.

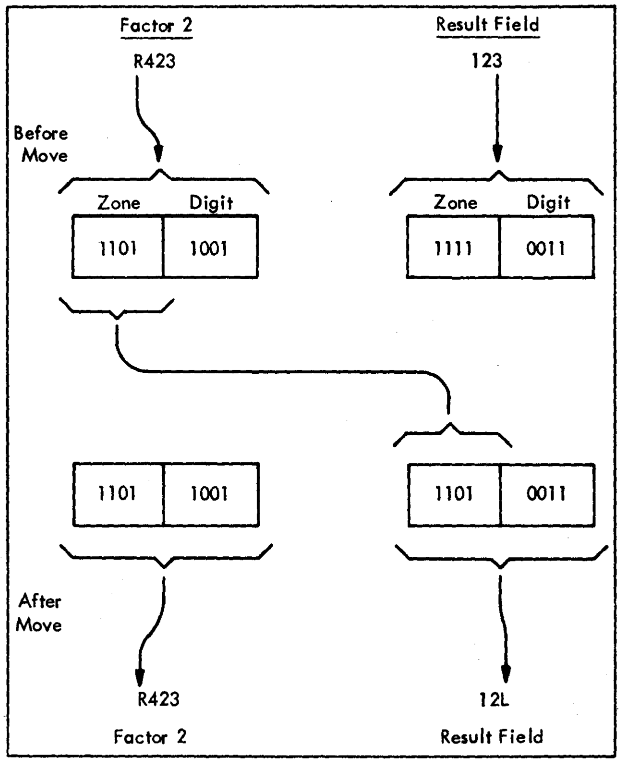


Figure 89. Move High-to-Low Zone Operation

Move Low-to-Low Zone (MLLZO): This operation moves the zone at the rightmost position of Factor 2 to the rightmost position of the Result Field. Factor 2 and the Result Field are alphameric or numeric. A Result Field specified as numeric contains an F-zone for a plus sign or a D-zone for a minus sign after this operation regardless of the zone moved.

Testing or Compare Operations

Compare (COMP): This operation causes the contents of the field or the literal in Factor 1 to be compared against the contents of the field or literal in Factor 2. The outcome of this operation can be used to set on an indicator that has been specified in positions 54-59 (Resulting Indicators High, Low, or Equal). These indicators are set on as follows:

1. High -- Factor 1 is greater than Factor 2.

2. Low -- Factor 1 is less than Factor 2.

3. Equal -- Factor 1 is equal to Factor 2.

This operation is used to make comparisons to alter or modify subsequent calculations. No result field is specified. The following should be considered when using compare operations.

- The Factor 1 and Factor 2 fields are aligned according to whether they are numeric or alphameric. If numeric fields are compared, fields of unequal length are aligned to the implied decimal point.

- Missing digits in numeric fields are assumed to be zeros.

- If alphameric fields are compared, fields of unequal length are aligned to their leftmost characters, and the unused positions are filled with blanks.

- The alphameric compare operation is based upon the internal collating sequence of the system.

- For equal length alphameric fields, the maximum field length is 256 characters.

- For unequal length alphameric fields, the maximum field length is 200 characters.

- An alphameric field and a numeric field should not be compared because the results of such a comparison are unpredictable.

All numeric comparisons are algebraic; the sign is considered in the comparison. (A +5 will compare as greater than −5.) A comparison in which the sign is not considered (an absolute comparison), can be performed by means of a short routine programmed to meet the user's requirements. Figure 90 shows an example of comparing the absolute value of a sum to a literal.

Test Zone (TESTZ): This operation is used to test the zone of the leftmost position of the alphameric field entered in the Result Field. The format of a test zone operation is shown in Figure 91.

If the result of the test is a 12-zone (&, A through I, $\overset{+}{0}$), the indicator specified in positions 54-55 will be set on. If the result of the test is an 11-zone (−, J through R, $\overline{0}$), the indicator specified in positions 56-57 is set on. Any other zone sets on the indicator specified in positions 58-59.

Figure 92 shows an example of this operation. When indicator 25 is set on, the field DATA is tested. If the leftmost position has a 12-zone, indicator 01 is set on. If the position has an 11-zone, indicator 02 is set on.

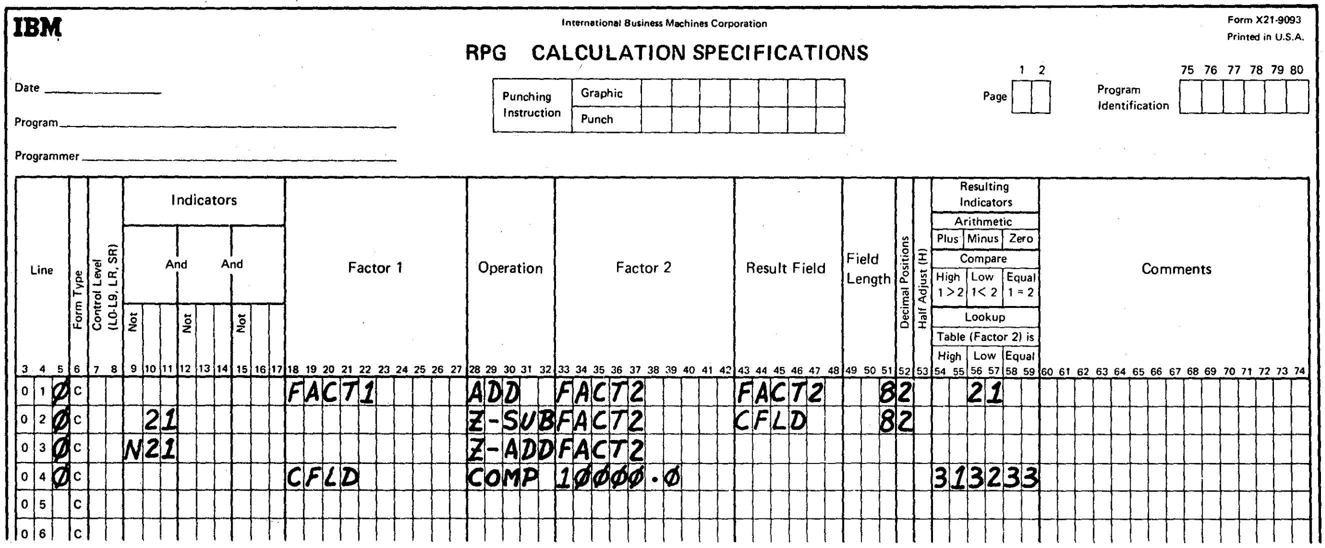
Branching and Exit Operations

Exit to a Subroutine (EXIT): This operation code enables the programmer to transfer control from the RPG program to a user subroutine. Factor 2 contains the name of the subroutine. The name of the subroutine cannot be greater than six alphameric characters; the first character must be alphabetic. Factor 1 and the Result Field are not used. See *Using Tables and Exit Routines in the Object Module, Exit to a User's Subroutine* for a complete discussion of this operation.

RPG Label (RLABL): This operation provides the facility for a subroutine, external to the RPG program, to reference a field in the RPG program. The name of the field to be referenced is entered in Result Field.

The field must be a valid numeric or alphameric field, an indicator, or a table. The use of an indicator or table as an RLABL is explained in the section *Using Tables and Exit Routines in the Object Module.*

Field length and decimal position, or field length only, must be defined for the Result Field of an RLABL entry unless defined by a preceding entry in either the input or calculation specifications. The field name may be from one to six alphameric characters. The first character must be alphabetic. Indicators, Factor 1, and Factor 2 are not used. The fields UDATE, UDAY, UMONTH, and UYEAR may be used in the Result Field of an RLABL operation. It is not necessary for the user to specify the field length or number of decimal positions for these entries. If he does specify them, UDATE must have a field length of six with zero decimal positions, and the others must have a field length of two with zero decimal positions.



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

RPG CALCULATION SPECIFICATIONS

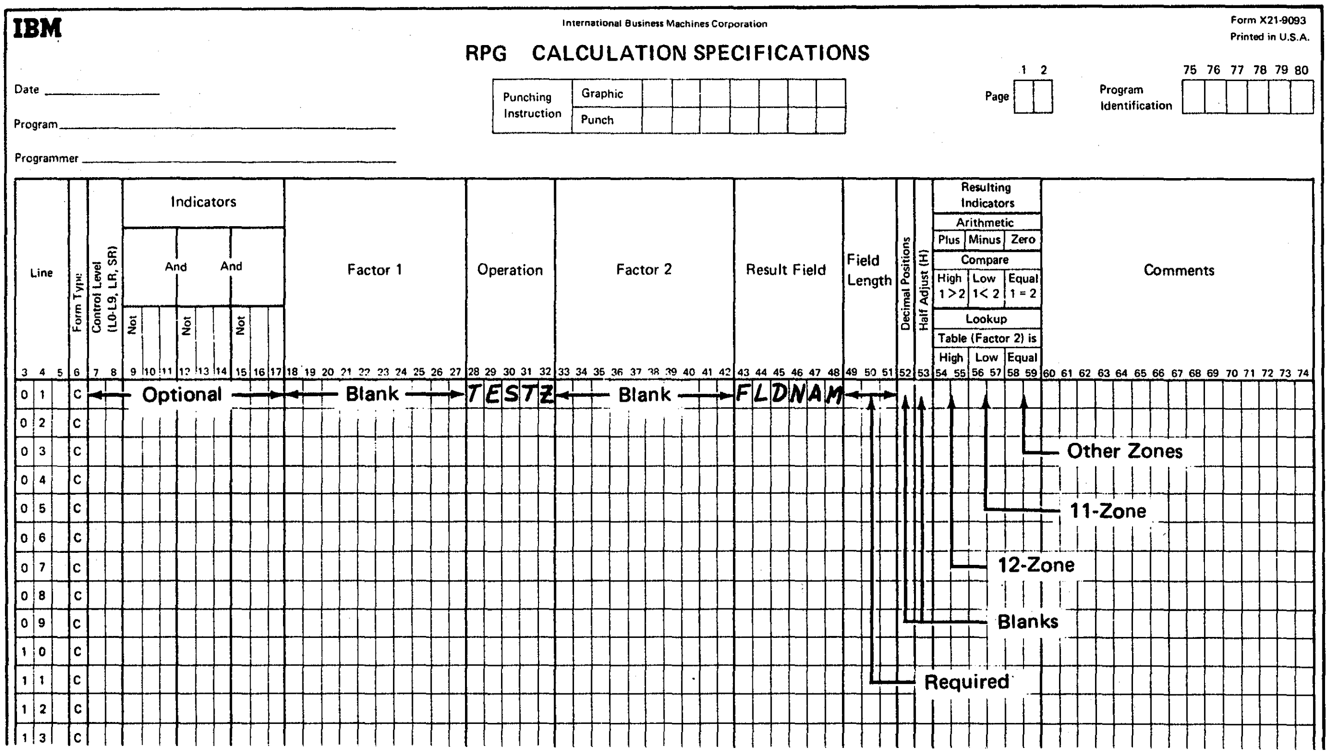
Date
Program
Programmer

Punching Instruction: Graphic / Punch

Page 1 2 Program Identification 75 76 77 78 79 80

Line (3-5)	Form Type (6)	Control Level (L0-L9, LR, SR) (7-8)	Indicators (9-17)	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: Plus / High 1>2 (54-55)	Minus / Low 1<2 (56-57)	Zero / Equal 1=2 (58-59)	Comments (60-74)
01Ø	C			FACT1	ADD	FACT2	FACT2	8	2			21		
02Ø	C		21		Z-SUB	FACT2	CFLD	8	2					
03Ø	C		N21		Z-ADD	FACT2								
04Ø	C			CFLD	COMP	1ØØØØ.Ø					31	32	33	
05	C													
06	C													

Figure 90. Example of an Absolute Compare Routine



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 (3 4 5)	Form Type (6)	Control Level (L0-L9, LR, SR) (7 8)	Indicators (9–17)	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: Plus / High 1>2 / High (54 55)	Minus / Low 1<2 / Low (56 57)	Zero / Equal 1=2 / Equal (58 59)	Comments (60–74)
0 1	C	Optional	Optional	Blank	TESTZ	Blank	FLDNAM							
0 2	C													
0 3	C													
0 4	C													
0 5	C													
0 6	C													
0 7	C													
0 8	C													
0 9	C													
1 0	C													
1 1	C													
1 2	C													
1 3	C													

Other Zones

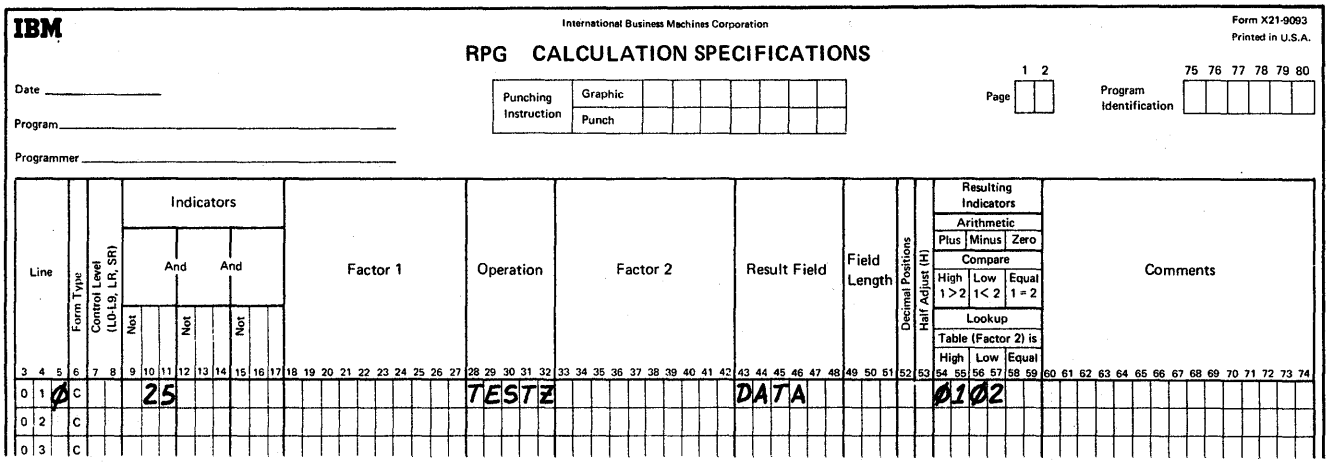
11-Zone

12-Zone

Blanks

Required

Figure 91. Format of the TESTZ 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 (1 2) — Program Identification (75 76 77 78 79 80)

Line (3 4 5)	Form Type (6)	Control Level (L0-L9, LR, SR) (7 8)	Indicators (9–17)	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: Plus / High 1>2 / High (54 55)	Minus / Low 1<2 / Low (56 57)	Zero / Equal 1=2 / Equal (58 59)	Comments (60–74)
0 1 Ø	C		25		TESTZ		DATA				Ø1	Ø2		
0 2	C													
0 3	C													

Figure 92. Using a Test Zone Operation

User's Label (ULABL): This operation enables the RPG program to reference a field contained in a user subroutine. The name of the field to be referenced is entered in Result Field. This name may be from one to six alphameric characters; the first character must be alphabetic. Indicators, Factor 1, and Factor 2 are not used. Field length and decimal positions, or field length only, must be defined. A ULABL field can be specified on the calculation or output specifications but cannot be specified on input specifications. Any identifier declared ULABL in the RPG program must be declared ENTRY in the BAL subroutine which contains the field.

Branching or Go To (GOTO): The operation code GOTO enables branching to occur in the generated program. This means leaving one point in the program to begin operations at some other location in the program. The location of the other routine is identified by a name. This name is entered in Factor 2 and the code GOTO is entered in Operation (positions 28-32). For example, a routine to calculate the employee contribution to the Federal Insurance Contribution Act might be labeled FICA. Branching to this routine would require that Factor 2 of the GOTO operation contain the name FICA, and that the first operation in the routine be a TAG operation (an operation that defines the name of the routine being branched to).

Branching can be performed within detail calculations or total calculations (see *Using the Detail-to-Total Branch*). Branching can be forward or backward, skipping over specifications, or going back over specifications previously skipped or processed.

If GOTO is to occur at total time, Control Level (positions 7 and 8) must be specified (L1 through L9, L0, or LR). If it occurs at detail time, a control specification is not required. Factor 1 and the Result Field are not used in this operation.

Conditioning indicators (positions 9-17) may be used to control when GOTO occurs. The name in Factor 2 cannot be the same as a field name. Figure 93 shows an example of a GOTO and TAG operation. For additional information see *Using the Calculations Specifications Form* in this section.

Providing a Label for GOTO (TAG): The operation TAG provides a name to which the program can branch. Enter this name in Factor 1 and the code TAG in Operation (positions 28-32). The name will be used as Factor 2 of the operation code GOTO.

If the TAG operation occurs at total time, Control Level (positions 7 and 8) of the specification must have a control level specification (L1 through L9, L0, or LR). If it occurs at detail time, a control specification is not required. Factor 2 and the Result Field are not used.

Setting Indicators On or Off

The position headings of Plus, Minus, or Zero and High, Low, or Equal have no meaning during this operation and should be ignored. Positions 54 through 59 are used for this operation code merely to record from one to three indicator codes.

Set Indicators On (SETON): This operation code causes the indicators specified in positions 54-55, 56-57, or 58-59 to be set on.

Specify the first indicator in positions 54-55, the second in positions 56-57, and the third in 58-59. One use of this specification is to set on a halt indicator when input records are out of sequence. Any RPG indicator except L0, 00, and U1-U8 can be set on. Figure 94 shows an example of this facility. Indicator 01 is set on for the first record of a sequence. Indicator L3 is a control level that occurs with the first record of the sequence. If such a situation (L3 and 01) does not occur, halt indicator H1 is set on.

Set Indicators Off (SETOF): This operation code causes the indicators specified in positions 54-55, 56-57, or 58-59 to be set off. Any RPG indicator except U1-U8, L0, MR, 1P, and 00 can be set off.

For example, when an L3 break ocucrs, L1-L3 are set on. If the user does not want L1 on, he can set it off by using the SETOF operation code.

Note: The setting of external indicators U1-U8 is determined by the PARM= entry of the EXEC statement of the job step that will execute the object program. The PARM= entry is coded as follows:

```
PARM= 'U=nnnnnnnn'
```

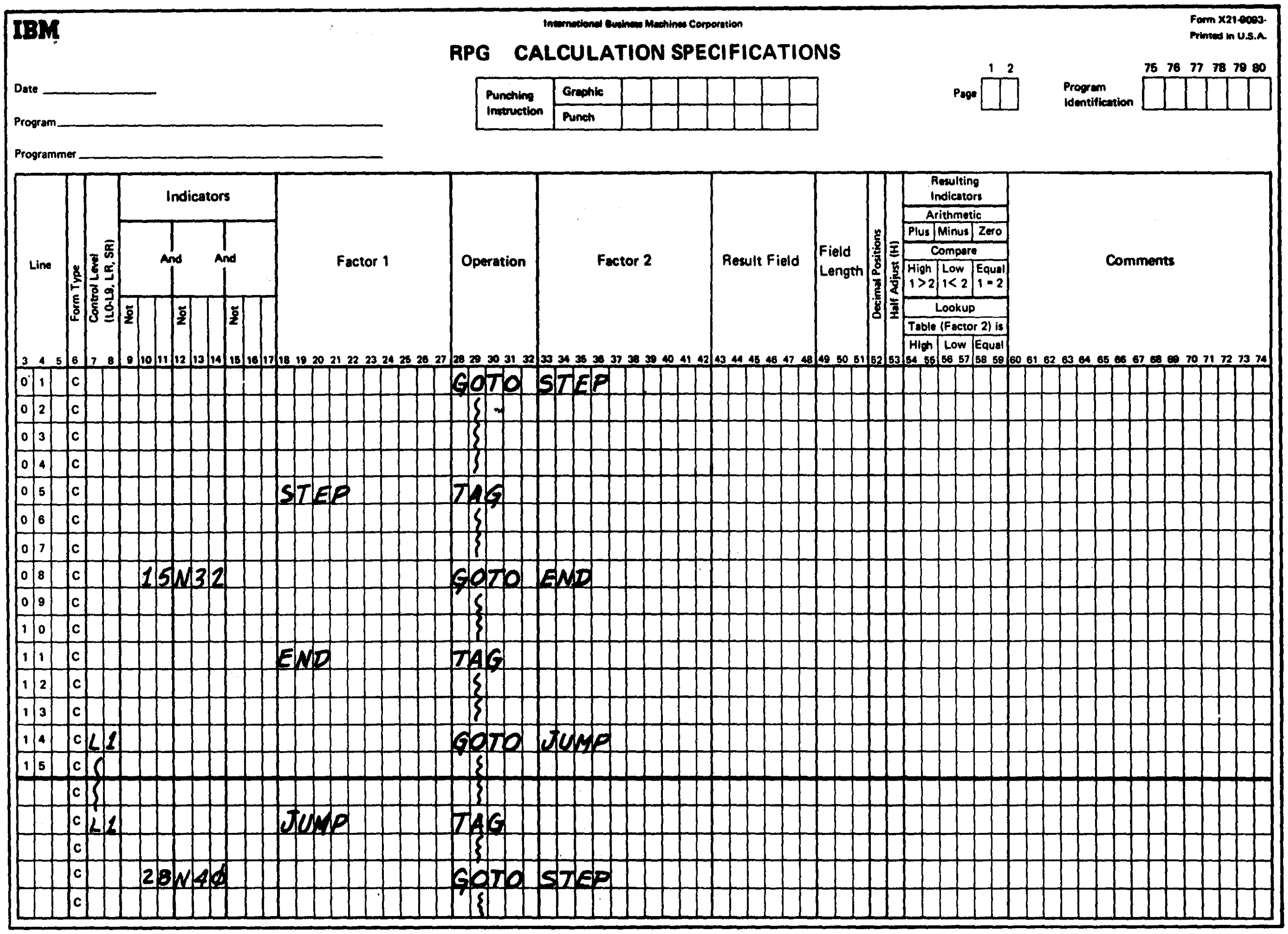
where n is replaced by 0 for each indicator (U1 through U8, left to right) that is to be off, and by 1 for each indicator that is to be on. Therefore, to set U7 and U8 on, the PARM= entry would be:

```
PARM= 'U=00000011'
```

To set U1, U4, U5, and U8 on, the PARM= entry would be:

```
PARM= 'U=10011001'
```

For further information on the EXEC statement, refer to *IBM System/360 Operating System: Job Control Language*, Form C28-6539.



IBM

International Business Machines Corporation

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

RPG CALCULATION SPECIFICATIONS

Date ______

Program ______

Programmer ______

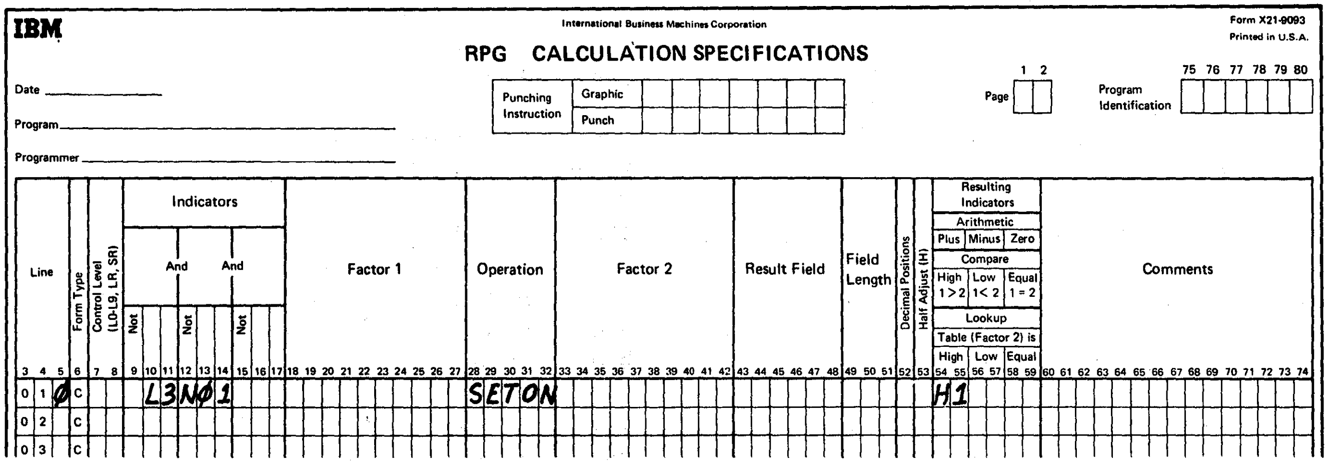
Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators	Comments
01	C				GOTO	STEP						
02	C											
03	C											
04	C											
05	C			STEP	TAG							
06	C											
07	C											
08	C		15N32		GOTO	END						
09	C											
10	C											
11	C			END	TAG							
12	C											
13	C											
14	C	L1			GOTO	JUMP						
15	C											
	C											
	C	L1		JUMP	TAG							
	C											
	C		28N4Ø		GOTO	STEP						
	C											

Figure 93. The GOTO 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 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators	Comments
01Ø	C		L3NØ1		SETON						H1	
02	C											
03	C											

Figure 94. Using SETON for a Record Out of Sequence

Table Operations

Table Lookup (LOKUP): Table lookup allows the RPG program to look up a table contained in core storage and procure from it specific data needed in the calculations. A table may consist of a set of arguments or a set of functions. Tables may be arranged in ascending, descending, or random sequence.

All tables used in an RPG program are loaded into core storage at program object time. They are loaded into the program in the same sequence they appear on the Extension Specifications form.

The LOKUP operation code causes the contents of the field or literal contained in Factor 1 to be used as the search argument. Factor 2 contains the name of the argument table to be searched, and the Result Field contains the table name from which the associated function will be obtained. Decimal alignment is not performed for this operation. (The Result Field may be left blank if the associated function is to be located but not retrieved from the table.)

At least one indicator must be specified in positions 54-59. If an indicator is specified in either the high or low positions, ascending or descending sequence must be specified for the table on the extension specifications.

The use of two resulting indicators causes the RPG program to look up the table entry that is high or equal, or low or equal, in relation to the search argument.

If two indicators are specified, they may not be in both the high and low positions. If different indicators are specified (high and equal or low and equal), the equal indicator takes precedence.

After the lookup operation is completed, the function that is retrieved is placed in a special hold area for the function table. The name of this area is the same as the name of the function table.

To use the function in another operation (for example in an arithemetic operation), the name of the function table is specified in Factor 1 or Factor 2. In this case, the function table name (in Factor 1 and Factor 2) refers to the special hold area in the function table.

To update the function table (for example, a move operation to replace the old function with the new updated function), the name of the function table is specified in the Result Field. The new function is then placed in its proper place in the function table and in the special hold area.

After each table lookup operation, the retrieved function should be used (or moved from the special hold area to another location) before the next table lookup operation is performed. Each subsequent lookup operation overlays the function obtained from the previous lookup operation.

See *Using Tables and Exit Routines in the Object Module* for additional information and examples of table lookup operations.

Conversion Routine Operations

RPG Conversion (RPGCV): This operation code is used to indicate that the track-address conversion routine is coded on the RPG Calculation Specifications form. Factor 1 contains the name (label) of the conversion routine. This name must also be specified on the Extension Specifications form in columns 27-32. The Result Field contains the name of the field that contains the relative track address. This field must be alphameric and must have a length of 3. Indicators and Factor 2 are not used.

End of RPG Conversion (ERPGC): This entry terminates the conversion step entries that have been coded on the Calculation Specifications form. No other entries are necessary. Indicators, Factor 1, Factor 2, and Result Field are not used.

External Conversion Routine (EXTCV): This entry is used to indicate that the track address conversion routine is external to the RPG language.

Factor 1 contains the label specified on the Extension Specifications form in positions 27-32.

The Result Field contains the name of the filed that will contain the track address. This name is defined in the RPG program by the EXTCV operation and must not be defined in the external routine. The field must be alphameric and have a length of 3.

Factor 2 must contain the name of the external conversion subroutine that the RPG program branches to. The Indicators specification is not used.

Record Key (KEYCV): This operation code establishes the name of the field that is to contain the key of the disk record. (It is used only when records are retrieved using record key data.) The code KEYCV is placed in Operation (positions 29-32), and the name of the filed is placed in Result Field (positions 43-48). The field length and decimal positions must be specified if the field has not been defined previously. Indicators, Factor 1, and Factor 2 are not used.

The operation must follow the RPGCV or EXTCV operation. The name of the field that will contain the record key is defined in the RPG program by this operation and must not be defined in the external conversion routine.

Summary of Operation Codes

Figure 95 is a summary of the specification entries required for the operation codes just described.

Result Field (43-48)

This specification sets up a location in storage to contain the result of a calculation. The name of the Result Field must be alphameric and must be left justified. It must not contain blanks, or special characters and the first character must be alphabetic. The sign for numeric fields is always stored in the units position of the Result Field.

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.

The field names UDATE, UMONTH, UDAY, and UYEAR may only be used in the Result Field for an RLABL operation. RPG defines these fields as having zero decimal positions and a field length of six for UDATE, or a field length of two for UMONTH, UDAY, or UYEAR. The user may specify the correct field length and number of decimal positions if he desires.

Figure 96 illustrates Result Field specifications. On the first line, GROSS is multiplied by DRATE and the Result Field is established as DISCNT. This Result Field is used as Factor 2 on the next specification to calculate a new amount. This same Result Field is used as Factor 1 on the next specification line to calculate total discount.

Field Length (49-51)

This entry specifies the length of the result field. The entry must specify the number of positions to be reserved for the Result Field. In Figure 96, DISCNT is eight positions long. The unpacked length must be specified. The maximum numeric field length is 15 digits, and the maximum alphameric field length is 256 characters.

When the same field name is used for more than one calculation, the field length and decimal positions (or field length only), must be the same in all calculations. Field length and decimal positions or field length only, need to be defined only once in the program (either in the input specifications or calculation specifications).

Figure 97 shows the results when a numeric Result Field with different field lengths and decimal positions from Factor 2 is specified. The answer for each example is given in the comments field. Figure 97 shows how the field is truncated to satisfy teh field length and decimal point specifications.

Note: If half adjustment is specified, the field length entry refers to the length of the Result Field after half adjustment.

Decimal Positions (52)

An entry in this position indicates the number of positions to the right of the decimal symbol in the Result Field. An entry must be made in this position for all arithmetic operations if the field has not been defined previously. If the Result Field does not have any decimal positions, the entry must be 0. Up to nine decimal positions can be specified, but, the number of decimal positions cannot exceed the length of the field defined.

If the Result Field is alphameric, this position must be left blank.

This specification is the only entry required to determine the number of decimal places in a calculated result. (The decimal point of input fields is specified on the input specifications). The object program considers the number of decimal positions in both factors of an arithmetic operation and automatically shifts the factors or the results to provide the correct number of decimal positions.

In Figure 96 each Result Field has two decimal positions.

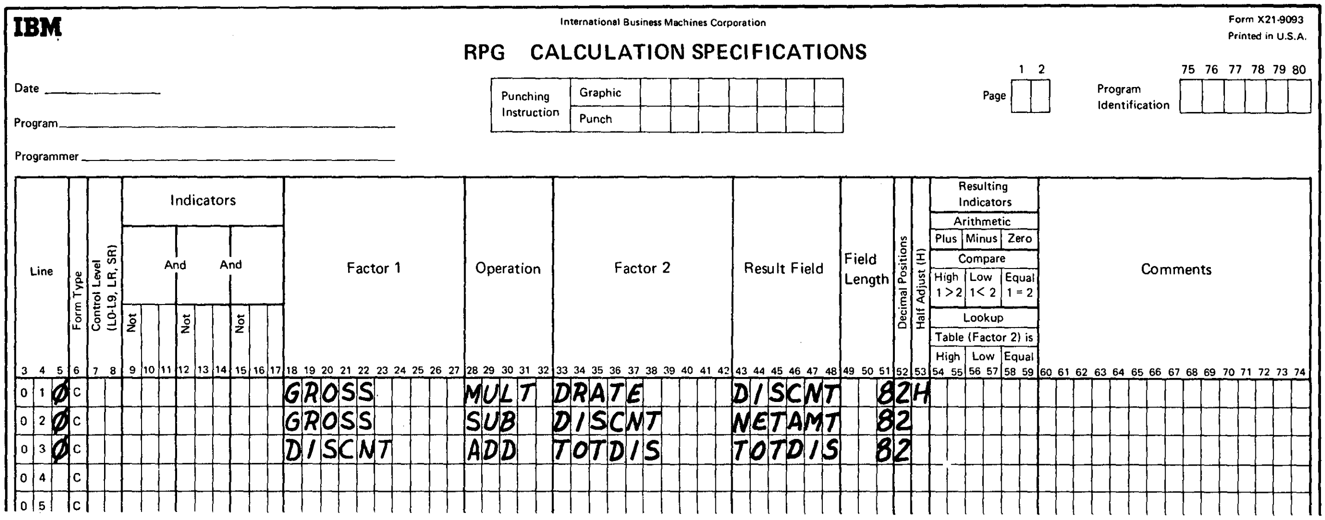
Operation Codes Operation	Control Level	Conditioning Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust	Resulting Indicators
Decimal										
Add	O	O	N	ADD	N	N	O	O	O	O
Zero and add	O	O		Z-ADD	N	N	O	O	O	O
Subtract	O	O	N	SUB	N	N	O	O	O	O
Zero and subtract	O	O		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
Branching										
Branching or GOTO	O	O		GOTO	L					
Providing label for GOTO	O		L	TAG						
Exit and Label										
Exit to a subroutine	O	O		EXIT	L					
RPG label	P			RLABL		E	O	O		
User's label	P			ULABL		E	R	O		
Conversion										
RPG conversion	P		L	RPGCV		A	8			
End of RPG conversion	P			ERPGC						
External conversion routine	P		L	EXTCV	L	A	8			
Record key	P			KEYCV		E	O	O		

Key to Operation Code Chart

If the entry is blank it cannot be filled in.

8	=	Required field length of 8 positions for track address
E	=	Required field must be either alphameric or numeric
A	=	Required field must be alphameric
L	=	Required label name
N	=	Required field must be numeric
O	=	Optional entry
OL	=	Optional label entry
P	=	Optional entry to prevent diagnostic if used at total time
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 95. Summary of Operation Specifications



IBM

International Business Machines Corporation

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

RPG CALCULATION SPECIFICATIONS

Date

Program

Programmer

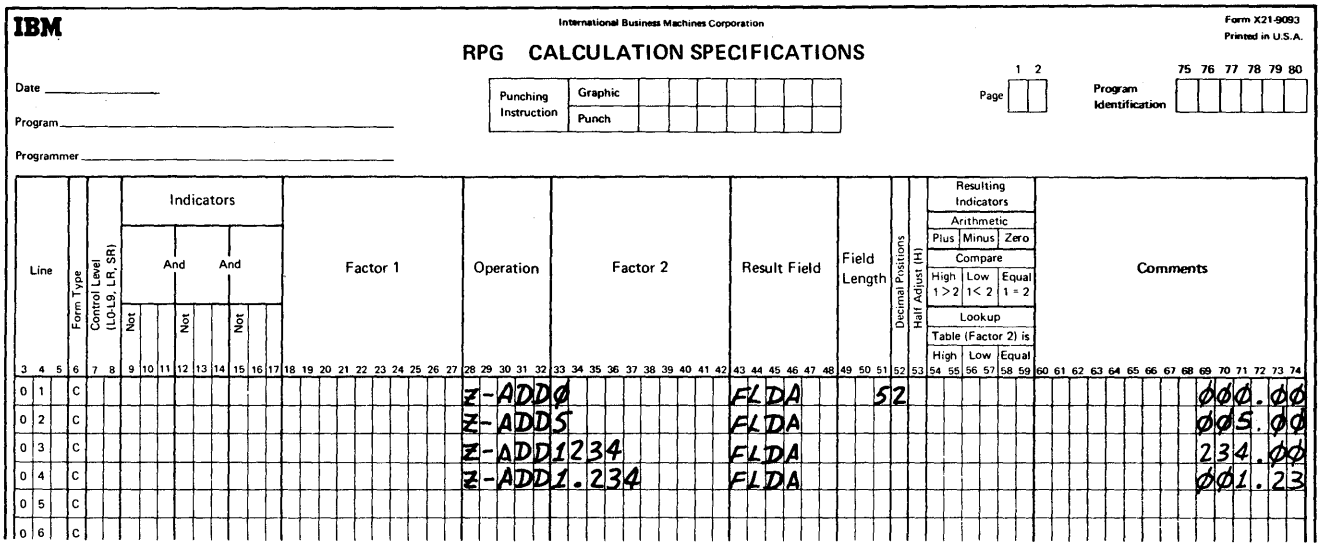
Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

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	Comments
01Ø	C			GROSS	MULT	DRATE	DISCNT	8	2	H		
02Ø	C			GROSS	SUB	DISCNT	NETAMT	8	2			
03Ø	C			DISCNT	ADD	TOTDIS	TOTDIS	8	2			
04	C											
05	C											

Figure 96. Result Field Entry



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 (Not, And, Not, And, Not)	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators	Comments
01	C				Z-ADD	Ø	FLDA	5	2			ØØØ.ØØ
02	C				Z-ADD	5	FLDA					ØØ5.ØØ
03	C				Z-ADD	1234	FLDA					234.ØØ
04	C				Z-ADD	1.234	FLDA					ØØ1.23
05	C											
06	C											

Figure 97. Effects of the Result Field Length

Half Adjust (53)

This specification is used to half adjust, or round, the Result Field. Enter an H in this position if the data in the Result Field is to be half adjusted.

Half adjusting is accomplished in the object program by adding an absolute value of 5 to the right of the last position retained in the Result Field. (This has the effect of rounding away from zero for both positive and negative numbers.)

In Figure 96, DISCNT is half adjusted. If the Result Field is an alphameric field, this specification must be left blank.

If the number of decimal positions in the arithmetic result is less than or equal to the decimal positions specified for the Result Field, the half adjustment specification has no effect.

Using Figure 96, assume the following values in line 010:

GROSS = 2,145.07

DRATE = .09

$$\begin{array}{r} 2{,}145.07 \\ \text{x} \quad .09 \\ \hline 193.0563 \\ + \quad 5 \\ \hline 193.0613 \end{array}$$

DISCNT = 193.06

TESTING THE RESULTS OF CALCULATIONS

The last category of specifications for the Calculation Specifications form is Resulting Indicators.

Resulting Indicators (54-59)

This specification may be used to test the value of a result field after completion of an operation. As the result of this test, an indicator is set on. This indicator can be used to control subsequent calculation operations or to control output operations. The specification is used in five ways:

1. To 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. To test the result of a compare operation to determine if:

   a. Factor 1 > Factor -- High

   b. Factor 1 < Factor 2 -- Low

   c. Factor 1 = Factor 2 -- Equal

3. To define the type of LOKUP operation to determine:

   a. If the argument next higher than the search argument is found.

   b. If the argument next lower than the search argument is found.

   c. If the argument equal to the search argument is found.

Note: An equal search resulting indicator takes precedence over either high or low indicators if an equal table value exists.

4. To define a TESTZ operation and determine what type of zone is to be tested.

5. To define SETOF and SETON operations and determine what indicators are to be set off or on.

   Any one of the following indicators may be used as entries for this specification:

   a. 01-99

   b. H0, H1-H9

   c. L1-L9, LR

   d. OA-OG, OV

When a control level indicator is set on or off, it does not affect the status of the lower control levels (for example, setting on L3 does not cause L1 and L2 to be set on). Control level indicators can be defined one or more times on the form. If these indicators are defined more than once, a subsequent specification of the indicator resets it: that is, if a control level indicator has been defined and is then redefined, it is set on or set off according to the new specification.

Note 1: Defining these indicators means specifying them as record identifying indicators or field indicators in the input specifications or as resulting indicators in the calculation specifications. This should not be confused with using these indicators. Using these indicators means specifying them in Indicators on the Calculation Specifications form or in Output-Indicators on the Output-Format Specifications form as many times as required. In the latter case they are merely tested to determine their status and are not reset by the test.

Note 2: Resulting indicators are reset each time the calculations that use them are performed. More than one resulting indicator can be on at one time.

Note 3: An indicator specified in positions 58-59 (Zero or Blank) for ADD, Z-ADD, SUB, Z-SUB, MULT, and DIV is on when the program begins and remains on until a specification which defines the indicator is processed.

A resulting indicator used to test for a zero balance can be reset by the following condition. The output specification Blank After causes a numeric field to be set to zeros after execution of the output operation specified. If this field is also a Result Field being tested for zero, the resulting field indicator specified is set on when the field is set to zeros by the Blank After specification. If the field is also a Result Field being tested for plus or minus, the resulting field indicators specified are not set off when the field is set to zeros by the Blank After specification.

Figure 98 illustrates the various conditions that cause resulting indicators to be set on.

Condition		Positions 54-55	Positions 56-57	Positions 58-59	Remarks
Arithmetic Operation	If Result Field is	Plus	Minus	Zero	The same indicator may be used in any two positions, 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 positions, but not in all three. If the same indicator is used in any two positions, 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.
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 positions 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 positions 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 and 2, the indicator in positions 58-59 is on.

Figure 98. How Resulting Indicators are SETON

Figure 99 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.

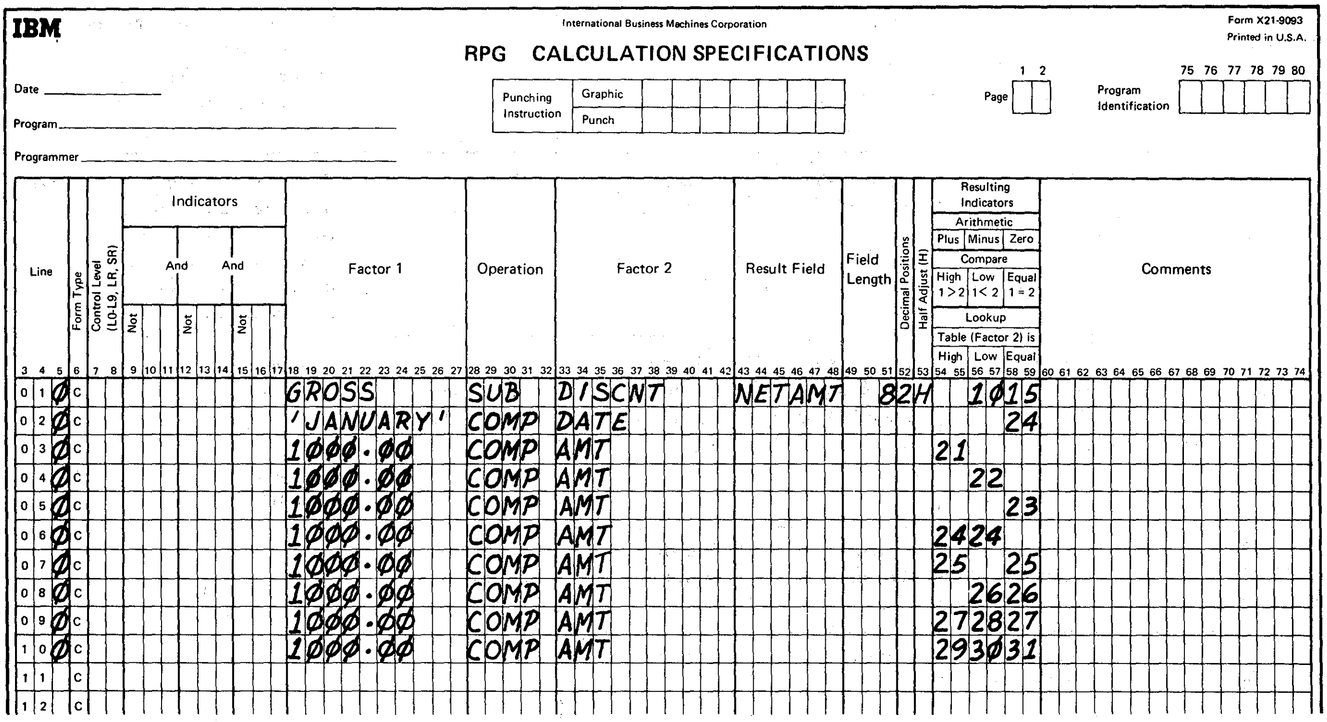
On line 020, the literal JANUARY is compared against the contents of DATE. If the result is equal, indicator 24 is turned on.

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

Lines 030 and 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 is made for either condition. If the first condition is met, the second condition will not set 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 (60-74)

Positions 60-74 of the Calculation Specifications form can be used for comments. Comments are not required by the program; they serve only as a help to you in documenting a program. Any comments you place in these positions are printed out on your source listing. An asterisk in position 7 is not required for this type of comment.



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 (3-5)	Form Type (6)	Control Level (7-8)	Indicators (9-17)	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: Plus / High 1>2 (54-55)	Minus / Low 1<2 (56-57)	Zero / Equal 1=2 (58-59)	Comments (60-74)
01Ø	C			GROSS	SUB	DISCNT	NETAMT	8	2	H		1Ø	15	
02Ø	C			'JANUARY'	COMP	DATE							24	
03Ø	C			1ØØØ.ØØ	COMP	AMT					21			
04Ø	C			1ØØØ.ØØ	COMP	AMT						22		
05Ø	C			1ØØØ.ØØ	COMP	AMT							23	
06Ø	C			1ØØØ.ØØ	COMP	AMT					24	24		
07Ø	C			1ØØØ.ØØ	COMP	AMT					25		25	
08Ø	C			1ØØØ.ØØ	COMP	AMT						26	26	
09Ø	C			1ØØØ.ØØ	COMP	AMT					27	28	27	
10Ø	C			1ØØØ.ØØ	COMP	AMT					29	3Ø	31	
11	C													
12	C													

Figure 99. Using Resulting Indicators

USING THE CALCULATION SPECIFICATIONS FORM

Figure 100 shows entries on the Calculation Specifications form that are used in part of a payroll application. These entries are discussed by line number.

Note: To simplify the illustrations, some lines of calculations have been omitted.

Line Number *Explanation*

01 The program branches from GRSPAY from some other detail calculation (not shown in this example).

02 The number of hours the employee worked is compared with the literal 40. If the employee worked more than 40 hours, indicator 20 is set on. If the employee worked less than 40 hours, Indicator 22 is set on.

03 to 05 If Indicator 20 is on, three calculations are performed. The literal 40 is subtracted from the number of hours worked, and the overtime hours are placed in the field OVERHR, which is a three-position field with one decimal position. RATE is multiplied by OVERHR and the result is placed in the field SAVE, which is a six-position field with two decimal positions. SAVE is multiplied by the literal 15 (which is the overtime premium rate). The result is placed back in SAVE, and it is half adjusted.

06 RATE is multiplied by 40, and the result is stored in GROSS. This operation is performed whether or not the employee worked any overtime. It is not performed-if the employee has worked less than 40 hours.

07 GROSS is added to SAVE if indicator 22 is off.

08 The program branches to the label FICA if indicator 22 is off.

09 Additional calculations.

10 The operation code TAG provides the label FICA. The RPG program branches to this label.

11 GROSS is multiplied by the literal .03 and the result is placed in the field DFICA, which is a six-position field with two decimal positions. The result is half adjusted.

12 DFICA is added to YDFICA and the result is placed in the field HOLD.

13 The contents of HOLD are compared with the literal 144.00, and if HOLD is less than, or equal to, 144.00, indicator 21 is set on.

14 If indicator 21 is on, the program branches to the label ADFICA.

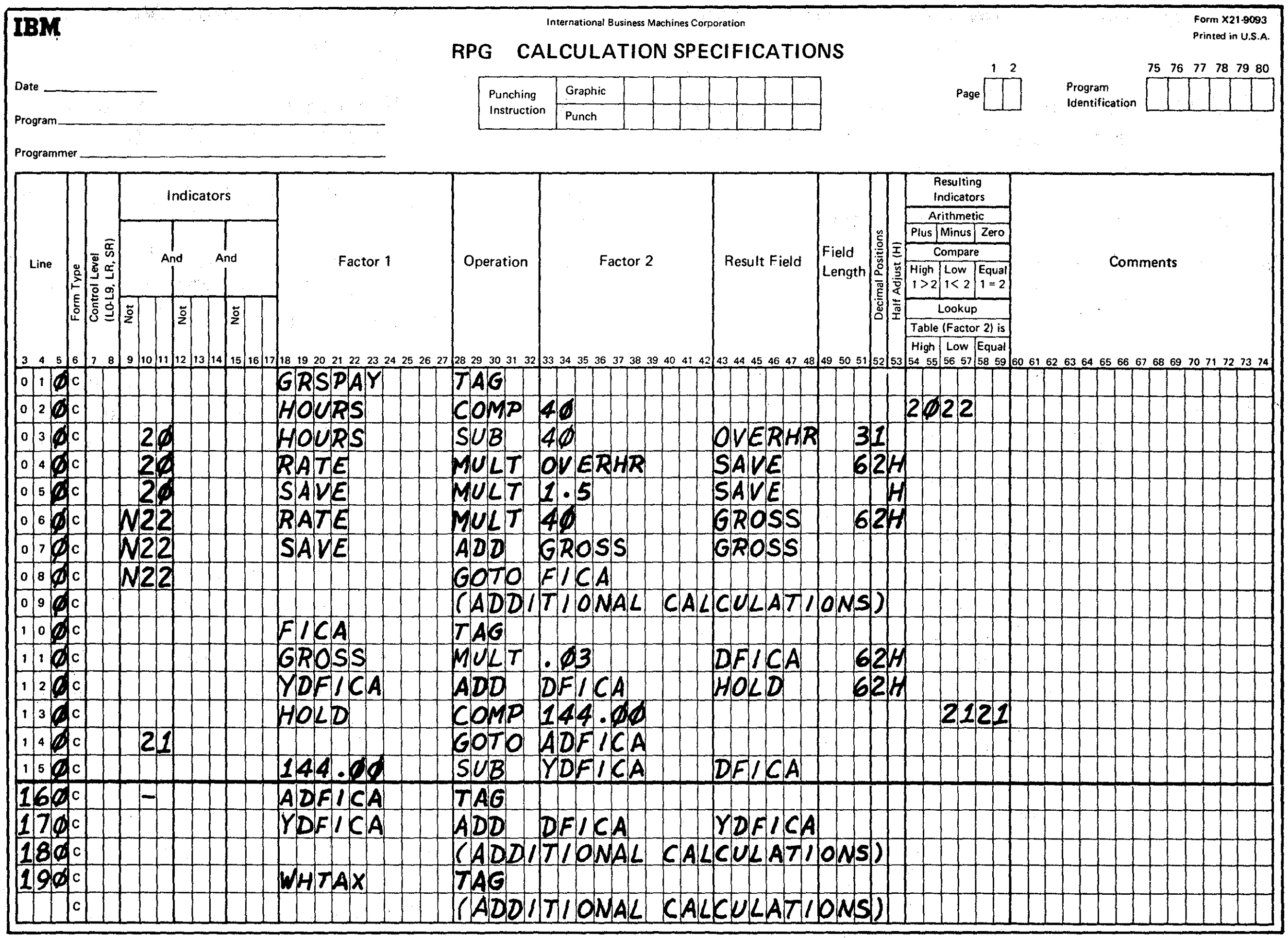
15 YDFICA is subtracted from the literal 144.00 and, the result is placed in DFICA.

16 The operation TAG provides the label ADFICA to which the program can branch (either from the specification on line 14 or sequentially from the specification on line 15).

17 DFICA is added to YDFICA and the result is stored in YDFICA.

18 Additional calculations.

19 The operation code TAG provides the label WHTAX to which the RPG program can branch.



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 (3-5)	Form Type (6)	Control Level (L0-L9, LR, SR) (7-8)	Indicators: Not (9) (10-11)	And: Not (12) (13-14)	And: Not (15) (16-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field (43-48)	Field Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Arithmetic Plus / Compare High 1>2 / Lookup Table (Factor 2) is High (54-55)	Minus / Low 1<2 / Low (56-57)	Zero / Equal 1=2 / Equal (58-59)	Comments (60-74)
01Ø	C					GRSPAY	TAG									
02Ø	C					HOURS	COMP	4Ø					2Ø	22		
03Ø	C		2Ø			HOURS	SUB	4Ø	OVERHR	3	1					
04Ø	C		2Ø			RATE	MULT	OVERHR	SAVE	6	2	H				
05Ø	C		2Ø			SAVE	MULT	1.5	SAVE			H				
06Ø	C		N22			RATE	MULT	4Ø	GROSS	6	2	H				
07Ø	C		N22			SAVE	ADD	GROSS	GROSS							
08Ø	C		N22				GOTO	FICA								
09Ø	C						(ADDITIONAL CALCULATIONS)									
10Ø	C					FICA	TAG									
11Ø	C					GROSS	MULT	.Ø3	DFICA	6	2	H				
12Ø	C					YDFICA	ADD	DFICA	HOLD	6	2	H				
13Ø	C					HOLD	COMP	144.ØØ						21	21	
14Ø	C		21				GOTO	ADFICA								
15Ø	C					144.ØØ	SUB	YDFICA	DFICA							
16Ø	C		–			ADFICA	TAG									
17Ø	C					YDFICA	ADD	DFICA	YDFICA							
18Ø	C						(ADDITIONAL CALCULATIONS)									
19Ø	C					WHTAX	TAG									
	C						(ADDITIONAL CALCULATIONS)									

Figure 100. Using the Calculations Specifications Form

Output-Format Specifications Form

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

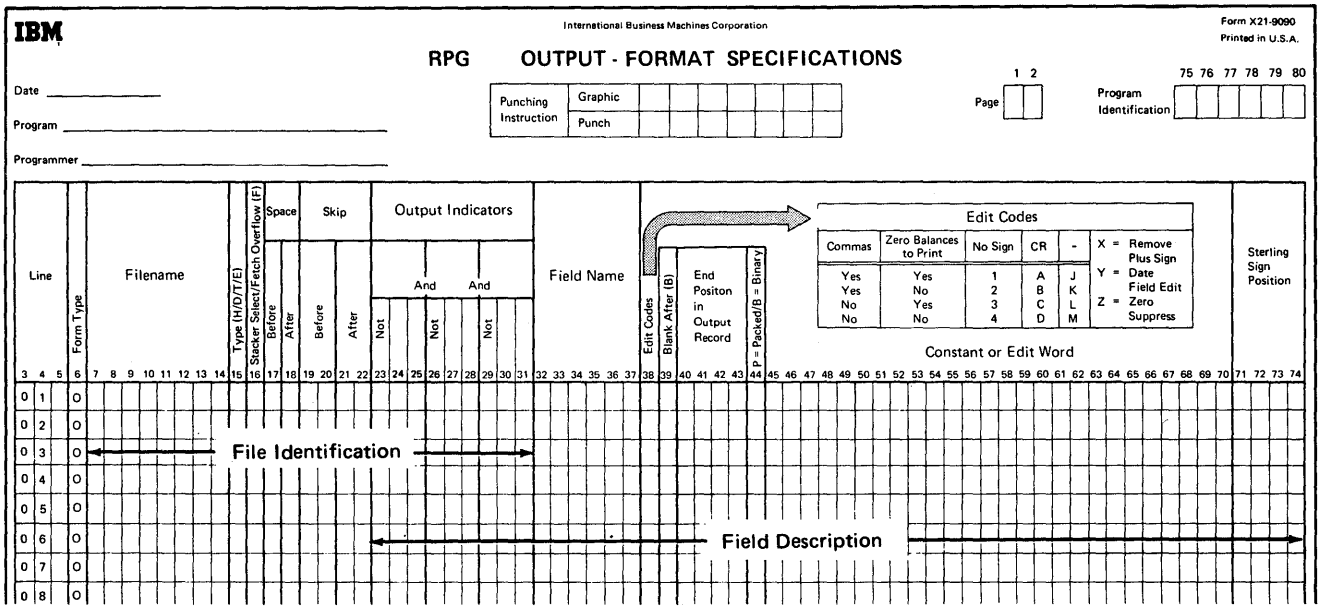
General Information

Output-Format Specifications can be divided into two categories (Figure 101): file identification and control (positions 7-31) and field description (positions 23-74).

File Identification and Control: These specifications identify the output records that make up the files. They also direct cards to the appropriate stackers, provide correct spacing on printed reports, and determine under what conditions records are to be produced.

Field Description: These specifications indicate where and when the individual fields are to be placed in the output record. All field description entries are written on the lines following their corresponding file identification specification. Each field is described on a separate line.

Specifying Output Units: When writing output-format specifications, it is not necessary to indicate specific output units used in the program. The I/O unit for each file is specified on the File Description Specifications form. Each filename is related to a specific output unit. The filename, therefore, specifies the output unit.



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 1 2

Program Identification 75 76 77 78 79 80

Line
Form Type
Filename
Type (H/D/T/E)
Stacker Select/Fetch Overflow (F)
Space
Before
After
Skip
Before
After
Output Indicators
And
And
Not
Not
Not
Field Name
Edit Codes
Blank After (B)
End Positon in Output Record
P = Packed/B = Binary
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 O
0 2 O
0 3 O
0 4 O
0 5 O
0 6 O
0 7 O
0 8 O
File Identification
Field Description

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 101. Output-Format Specifications Form

FILE IDENTIFICATION AND CONTROL

Filename (7-14)

A filename is assigned to each output file. The filename must be left justified and begin with an alphabetic character. The filename may be alphameric, but it must not contain special characters or blanks.

When writing output specifications, the filename need only be entered once. Enter it on the first line to define the file as shown in Figure 102.

Type H/D/T/E (15)

The entry in this position identifies the type of record being specified. The following entries are used for this specification:

1. H -- Heading record

2. D -- Detail record

3. T -- Total record

4. E -- Unused

Note: All heading records for a file must be entered first, followed by all detail records, then by all total records for the same file (Figure 102).

Heading Records

These records usually contain constant information. However, they may also contain information from input records, including the record present at the time the output record is produced.

Detail Records

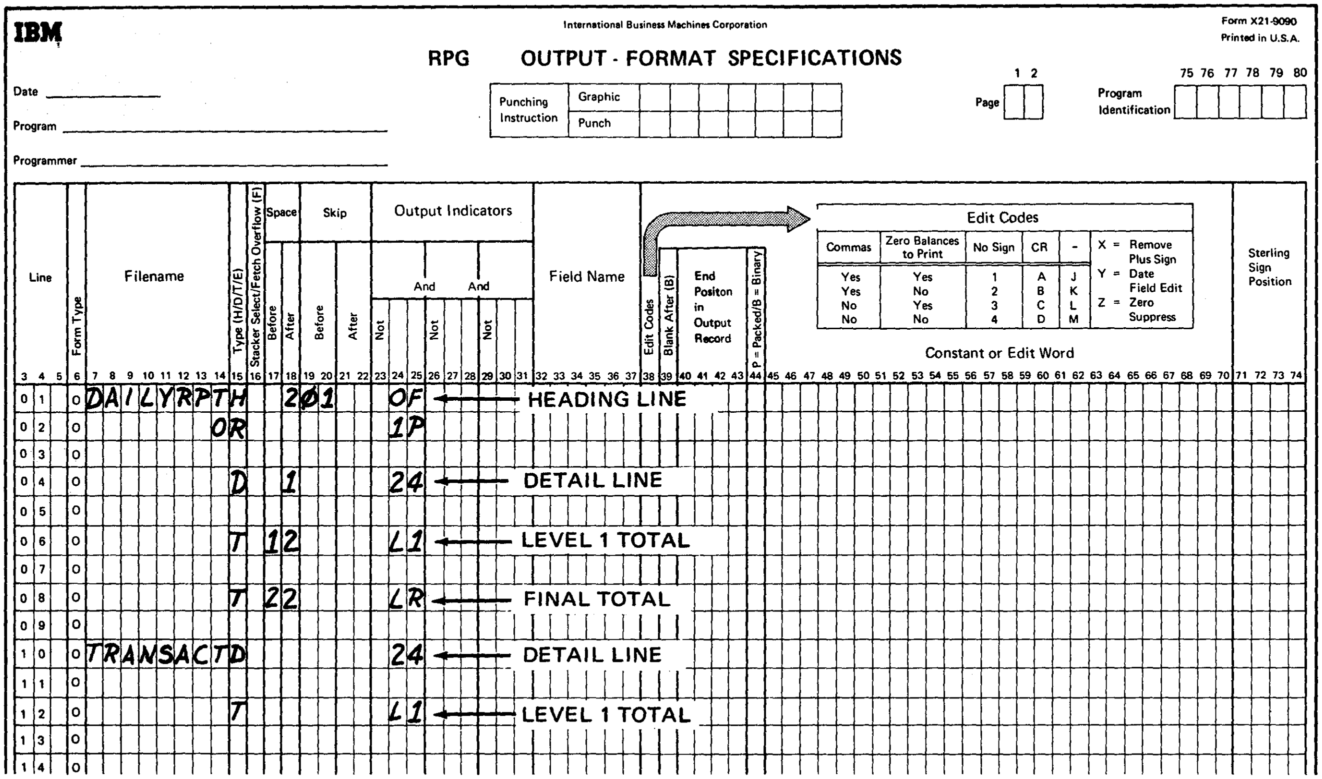
These records have a direct relationship to the input record. Most data in a detail record comes from the input record or from calculations performed at detail time.

Note: The RPG compiler does not differentiate between heading and detail records. They are distinguished only for the convenience of the user.

Total Records

Operations upon fields from the input record are preceded by the test for control field changes, the performance of total time calculations, and the formation of total records. Thus, data from an input record that causes a control field change cannot contribute data to a total records that result from the control change. Heading and detail records, however, can contain data from the input record.

All heading records for a file must be entered first, followed by all detail records, then by all total records for the same file (Figure 102). The output specifications may be written so that all heading records for all files are written first, followed by all detail records, and then all total records (Figure 103).



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 1 2

Program Identification 75 76 77 78 79 80

Edit Codes

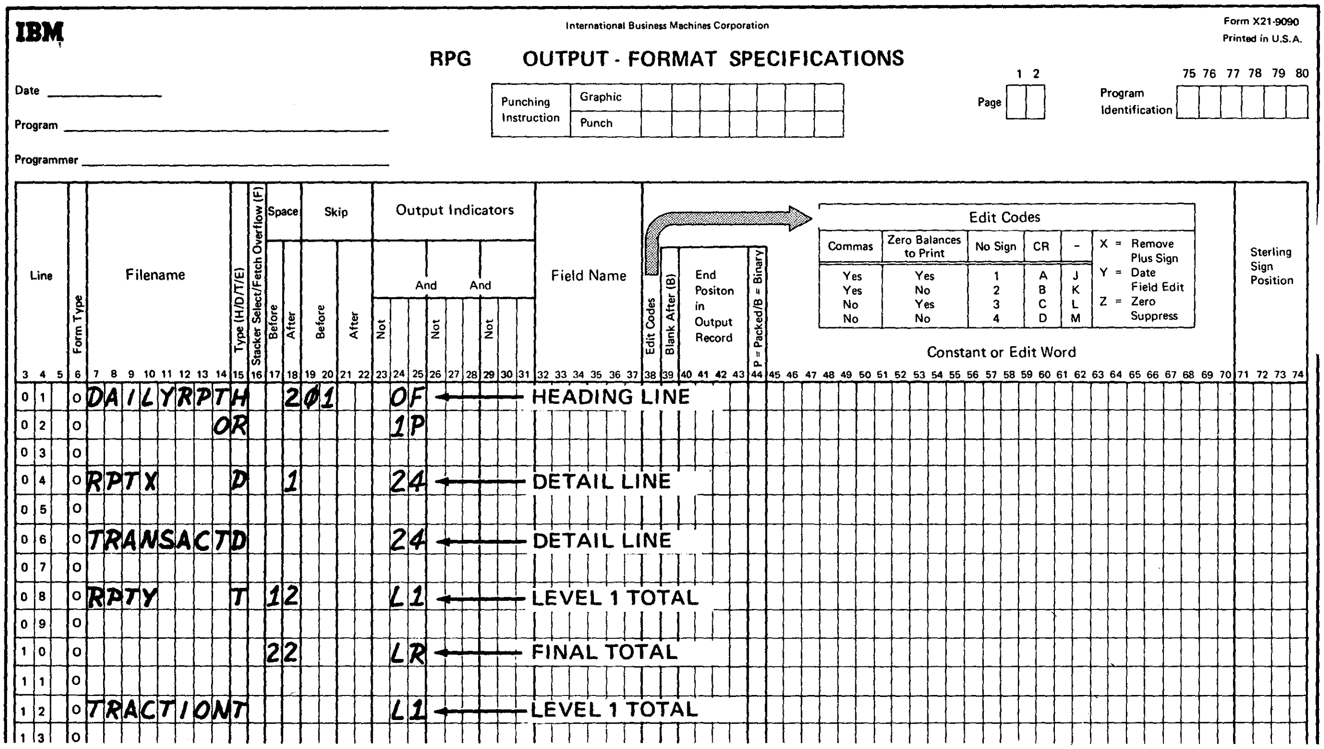
Commas	Zero Balances to Print	No Sign	CR	-	
Yes	Yes	1	A	J	X = Remove Plus Sign
Yes	No	2	B	K	Y = Date Field Edit
No	Yes	3	C	L	Z = Zero Suppress
No	No	4	D	M	

Constant or Edit Word

Sterling Sign Position

Line	Form Type	Filename	Type (H/D/T/E)	Space Before	Space After	Skip Before	Skip After	Output Indicators	
01	O	DAILYRPT	H		2	Ø1		OF	← HEADING LINE
02	O		OR					1P	
03	O								
04	O		D		1			24	← DETAIL LINE
05	O								
06	O		T	1	2			L1	← LEVEL 1 TOTAL
07	O								
08	O		T	2	2			LR	← FINAL TOTAL
09	O								
10	O	TRANSACT	D					24	← DETAIL LINE
11	O								
12	O		T					L1	← LEVEL 1 TOTAL
13	O								
14	O								

Figure 102. Specifying Heading, Detail and Total Lines



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 1 2

Program Identification 75 76 77 78 79 80

Edit Codes

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

Constant or Edit Word

Sterling Sign Position

Line	Form Type	Filename	Type (H/D/T/E)	Space Before	Space After	Skip Before	Skip After	Output Indicators	
01	O	DAILYRPT	H		2	Ø1		OF	← HEADING LINE
02	O		OR					1P	
03	O								
04	O	RPTX	D		1			24	← DETAIL LINE
05	O								
06	O	TRANSACT	D					24	← DETAIL LINE
07	O								
08	O	RPTY	T	1	2			L1	← LEVEL 1 TOTAL
09	O								
10	O			2	2			LR	← FINAL TOTAL
11	O								
12	O	TRACTION	T					L1	← LEVEL 1 TOTAL
13	O								

Figure 103. Specifying Output Lines

Stacker Select/Fetch Overflow (16)

If punched output occurs in the object program, a stacker code is entered in position 16. Cards go into the stacker indicated. Stacker numbers and codes are listed in Figure 104. The stacker select entry must be specified on an OR line if it is to apply to that OR line.

Space (17-18)

This specification and the Skip specifications (positions 19-22) are used to provide proper spacing on printed reports.

Note: If a record is to be printed, at least one entry is required in positions 17-22. If all these positions are blank, one space after is assumed.

Space Before (Position 17)

Enter in this position the number of lines to be spaced before the line is printed. Specify zero, one, two, or three spaces by placing 0, 1, 2, or 3 in position 17. If this position is left blank, no spacing before printing is provided.

Space After (Position 18)

Specify zero, one, two, or three spaces after printing by placing 0, 1, 2, or 3 in position 18. A blank in this position will provide single spacing after printing a line.

Output Unit	Stacker Number	Stacker Select Code
1442 Card Read-Punch	1 2	1 or blank 2
2520 Card Read-Punch	1 2	1 or blank 2
2540 Card Read-Punch	P1 P2 RP3	1 2 3

Figure 104. Summary of Stacker Select Specifications (Output)

Indexed Sequential ADD (16-18)

If records are to be added to an existing indexed sequential file, as specified on the File Description Specifications form, the entry ADD must be placed in these positions. This specification must appear on the same line that contains the TYPE H/D/T/E specificaton. Add may only be specified on H, D, or T lines; it then applies to all OR lines.

Skip (19-22)

This specification provides for the proper spacing of reports. The order in which spacing and skipping are performed is:

1. Skip before
2. Space before
3. Skip after
4. Space after

Skip Before (Positions 19-20)

The entries 01-12 cause the printer carriage to skip to channels 1 through 12 of the carriage tape before the line is printed. In Figure 102, the form skips to channel 1 before the heading line is written.

Skip After (Positions 21-22)

The entries 01-12 cause the printer carriage to skip to channels 1 through 12 of the carriage tape after the line is printed.

Overflow Indicator

Carriage overflow, or the line number associated with channel 12 in the Line Counter Specifications form, sets the indicator specified in the file description specifications (positions 33-34).

Channel 12 is sensed as the corresponding line is printed. The overflow indicator is set on after the next line is printed. Thus, one extra line is always printed after the overflow is detected and before the overflow functions can occur. In planning a printer operation, the carriage tape overflow punch must coincide with the next-to-last line to be printed on the form.

Line counter specifications cause the overflow indicator to be set on when the line counter reaches the specified line number. The overflow indicator remains on for one complete processing cycle. It is set off after the heading and detail lines of the next record are printed. The overflow indicator can be used to control calculation specifications.

A test for the overflow status is made by the object module immediately before each line of the report is printed (but after any Space Before specifications are executed). Two conditions can occur at this time:

1. If the overflow indicator is on before a detail line is printed, the detail line and any other detail lines whose output indicators are on, are printed.

   Any Space After specifications are performed, the next record is read, and a test is made to determine if a control break has occurred. If one has occurred, all total lines (caused by the control break) are printed; then any specified overflow printing is performed.

2. If the overflow indicator is on before a total line is printed, all total lines whose output indicators are on are printed. This is followed by any specified overflow printing.

Overflow Lines

An overflow line is any line conditioned by an overflow indicator (OA-OG, OV). This line is printed during overflow output time in the object program.

A line conditioned negatively with an overflow indicator (for example, NOF or NOV) is not an overflow line. This line is printed at either heading/detail or total time.

Fields Conditioned by Overflow

A field can be conditioned by an overflow indicator. This field is printed as any other field conditioned by an indicator. It is put out only if the record is to put out and the overflow indicator is on.

Automatic Skipping

For any line of a print file, the RPG compiler provides the object module with automatic skipping from channel 12 to channel 01 on that file when:

1. No overflow indicator is specified on the file description specification, and
2. No overflow indicator is specified as one of the first three indicators (positions 23-31 of output specifications).

Printing Lines Conditioned by Overflow

If an output line is conditioned by either an overflow or a control level indicator, the line prints whenever either of these indicators is on. If the overflow indicator and control level indicator occur at the same time during processing, the line prints twice.

This situation can be avoided by using the coding shown in Figure 105. Line 010 of the Calculation Specifications form in Figure 105 specifies that if a level 1 control break does not occur on this cycle but has occurred on the previous cycle, indicator 25 is set off. Line 020 specifies that if a level 1 control break has occurred, indicator 25 is set on. Indicator 25 remains on for an extra total cycle because:

1. If overflow has occurred at detail time, it will be sensed at total time.

2. The overflow routine is executed after the control level 1 indicator (L1) has been set off.

The output line is coded as shown in Figure 105. By making the condition mutually exclusive (that is, both conditions cannot exist at the same time), the line is not printed twice.

No test is made to determine if overflow occurs on a Space After; therefore, the overflow indicator is not set until the next line is printed. If this occurs, the overflow indicator is set on one cycle late, causing the OF or L1 line to print twice. To prevent this from occurring, use the coding shown in Figure 105.

The object module prints lines conditioned by overflow in a certain sequence:

1. When overflow occurs during the printing of a heading or detail line, the object module does not print lines conditioned by overflow. The program does, however, take note that the overflow condition has occurred.

2. Total lines not conditioned by overflow are printed.

3. The object module then tests to see if overflow has occurred.

4. Total lines conditioned by overflow are then printed.

5. Heading and detail lines conditioned by overflow are then printed.

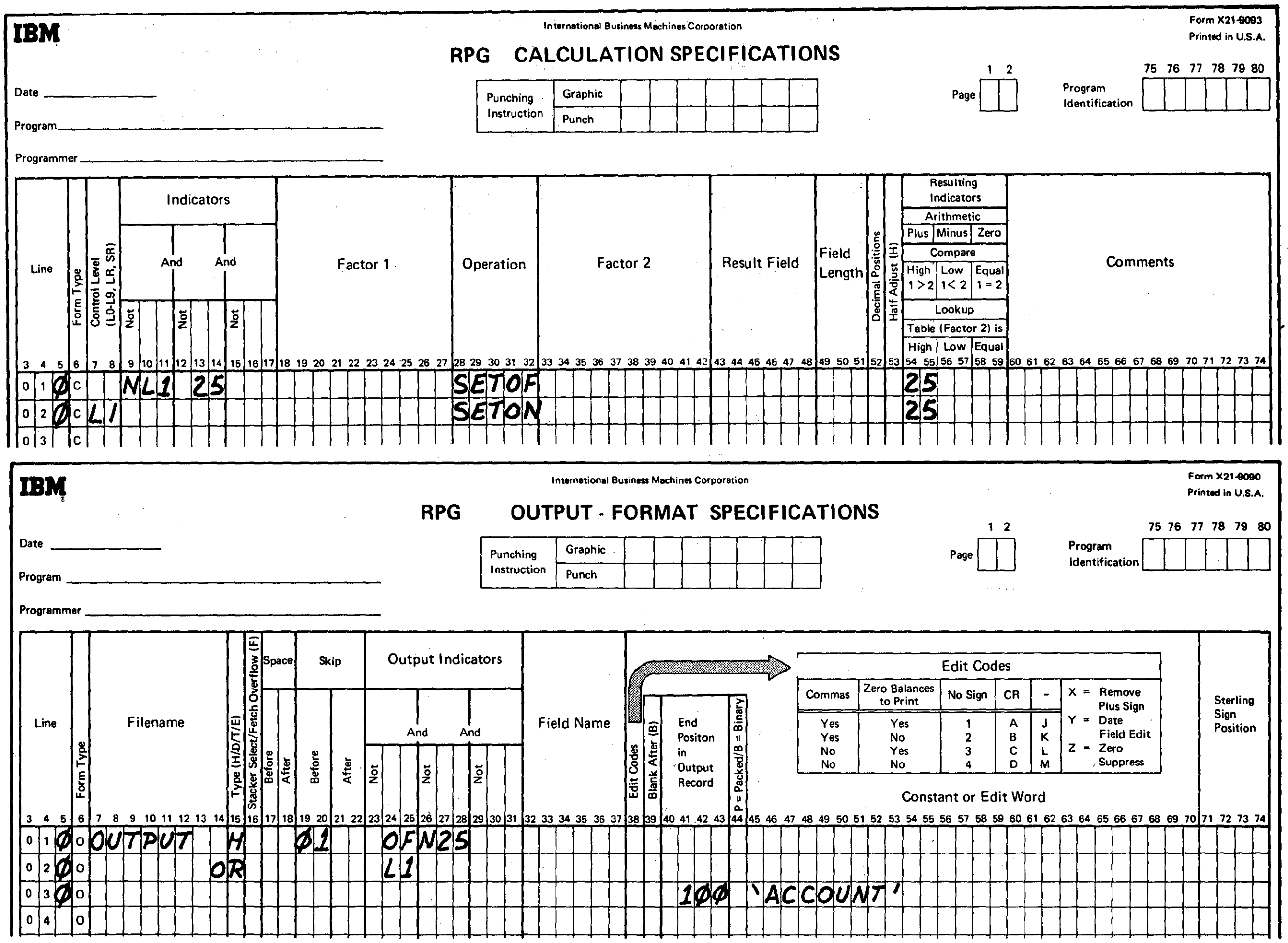
Note: If an overflow condition has been specified, automatic skipping is not performed.

Multiple Printers

It is possible to use a maximum of eight output printer files for each RPG object module. When producing records for more than one printer, the user must specify the printer on which the overflow condition occurs by assigning a unique overflow indicator for each printer.

In positions 33-34 of the File Description Specifications form, one of eight available overflow indicators is entered for each printer file. The following eight indicator codes are used: OA, OB, OC, OD, OE, OF, OG, and OV.

Figure 106 shows the possible conditions under which the overflow indicator is set and overflow lines occur.



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

RPG CALCULATION SPECIFICATIONS

```
01ØC  NL1 25             SETOF                     25
02ØCL1                   SETON                     25
03 C
```

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

RPG OUTPUT - FORMAT SPECIFICATIONS

```
01ØOOUTPUT  H   Ø1   OFN25
02ØO       OR        L1
03ØO                                 1ØØ 'ACCOUNT'
04 O
```

Figure 105. Specifying Output Indicators for Overflow

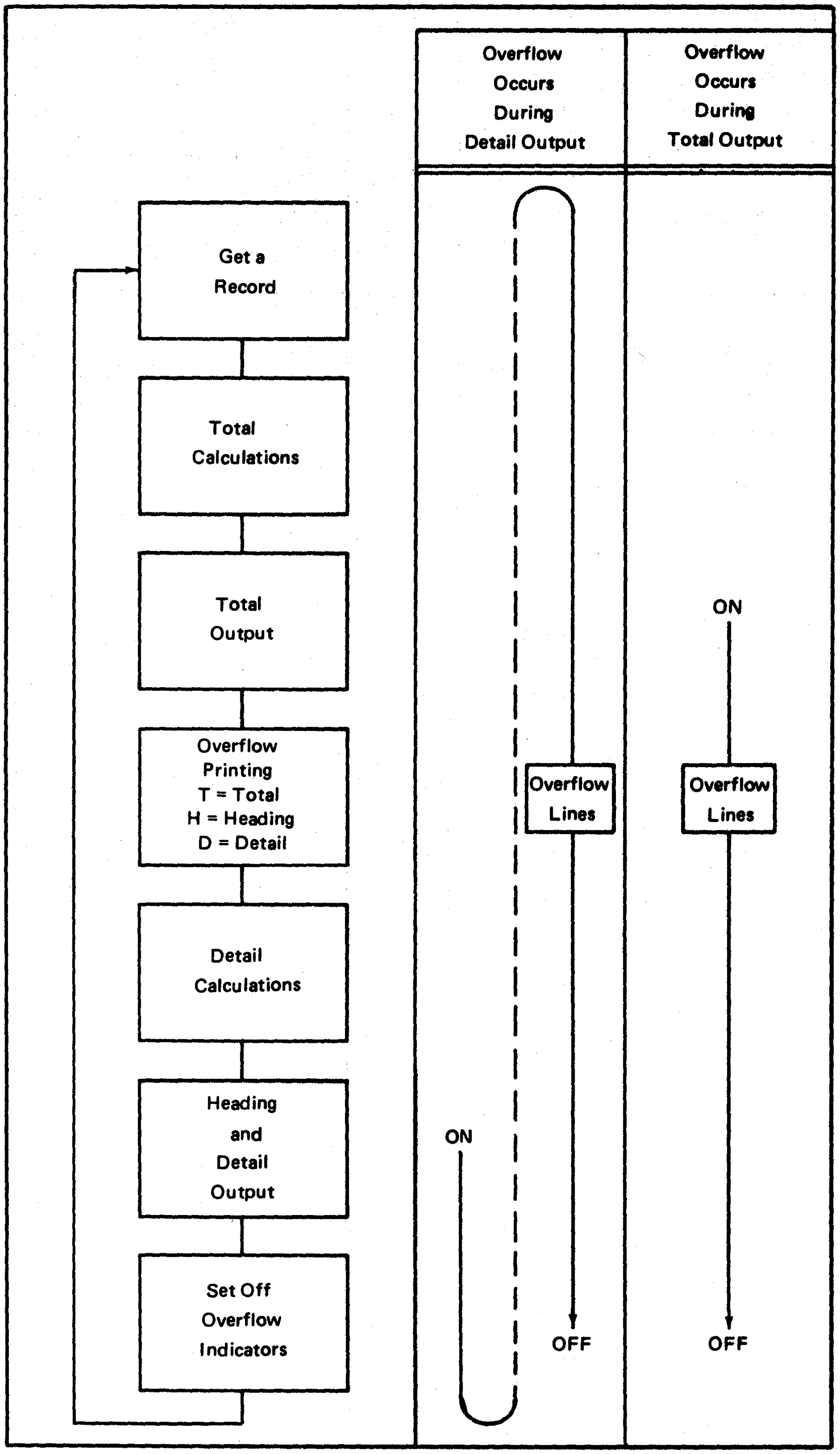

Overflow Occurs During Detail Output
Overflow Occurs During Total Output
Get a Record
Total Calculations
Total Output
Overflow Printing T = Total H = Heading D = Detail
Detail Calculations
Heading and Detail Output
Set Off Overflow Indicators
Overflow Lines
Overflow Lines
ON
ON
OFF
OFF

Figure 106. Setting of the Overflow Indicator and Overflow Printing

Figure 107 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) and the 12-punch is passed by the first line:

1. The overflow indicator is set according to Figure 107.

2. Overflow lines and all other lines for the particular block are printed.

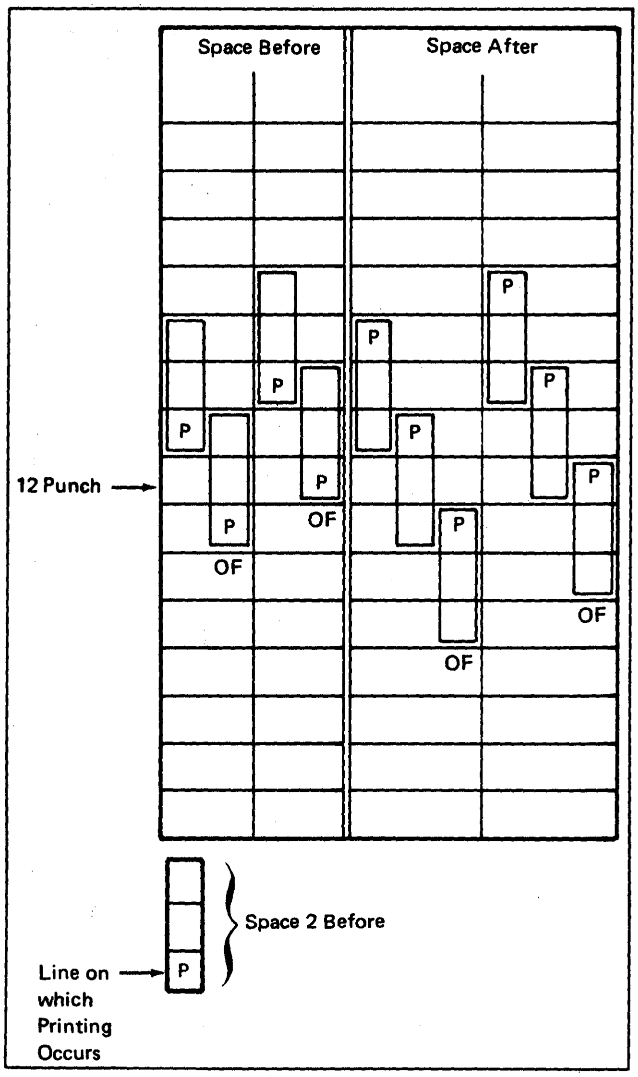


Figure 107. Setting of the Overflow Indicator

Output Indicator (23-31)

This specification can be used to condition either individual fields or an entire file. Entries in this position can specify a maximum of three indicators. These indicators control:

1. When a line is to be put out, or

2. When a particular field is to be written.

The following information applies only to its use in the file identification line.

If more than one indicator is specified on one line, all indicators are considered in an AND relationship. This means that all indicator conditions specified must be satisfied before the output condition can be executed.

If no output indicators are specified for a detail line, the line is put out every time heading and detail lines are processed (see *Program Logic*).

Examples of Output Indicators

The types of indicators that can be entered in positions 23-31 are listed.

1. A record identifying indicator code specifies the particular record type on which the output operation is to be performed. The operation cannot be performed on other record types.

2. A field indicator code controls the output operation by the status of an input field.

3. A resulting indicator code controls the output operation by conditions that have occurred during calculations.

4. A control level indicator (L1 through L9 and LR) causes the output operation to be performed only at the control level specified.

5. The MR indicator code causes the output operation to be performed only if there is a matching record in a second input file.

6. Halt indicators H0 through H9 are normally used to suppress an output operation when an error has been detected in the input data or during a calculation.

7. Overflow indicators cause the output operation to take place only if a page overflow has occurred (overflow line).

8. The 1P (first page) indicator enables overflow headings to be printed on the first page of a report. (This indicator is set on at the beginning of processing before any input records have been read.)

9. An external indicator (U1-U8) controls the output operation by conditions that were set up at object execution time through the PARM field of the EXEC card. See the section on indicators for more information on setting these indicators on and off.

If the object program requires that more than three indicators be in an AND relationship, the letters AND are entered in positions 14-16 of the following line, and the additional indicators are specified on that line. Positions 17-22 must be blank on an AND line. Additional specification lines can specify as many output indicators in an AND relationship as required. Each additional line must begin with AND in positions 14-16. An AND line cannot be used to condition the printing of a field.

If a line is to be conditioned as an overflow line, the overflow indicator must not appear in a specification line having the letters AND in positions 14 through 16.

Output OR Lines

If the output condition is performed in an OR relationship (one or the other of two sets of conditions), the letters OR are entered in positions 14-15 of the following specification line, and the OR indicators are specified on that line.

Additional specification lines can specify as many output indicators in an OR relationship as required. Each additional line must begin with OR in positions 14-15. An OR line cannot be used to condition a field.

When an OR line is specified on a print file, the printer control functions (positions 17-22) can all be left blank. The space and skip entries of the preceding line are then assumed. Different OR lines can have different printer control function specifications, however.

This does not hold true for the stacker select entry (position 16). A blank entry is valid and will select the card to a particular pocket. See *Stacker Select/Fitch Overflow* in this section.

Figure 108 shows seven examples of output indicators when they are used as part of the file identification specifications. The numbers to the right refer to the following discussion:

1. The control carriage skips to channel 01 before printing; the heading line is printed only when an overflow condition occurs or when the 1P (first page) indicator is on. A space 3 occurs after printing.

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 from the input specifications, or they could be resulting indicators from the calculation specifications.

3. The detail line is printed if indicator 40 or 46 is on. Spacing after printing occurs differently depending upon which indicator is on.

4. The detail line is printed on one of two conditions.

   - Indicators 50, 52, and 53 are on and indicator 51 is off.

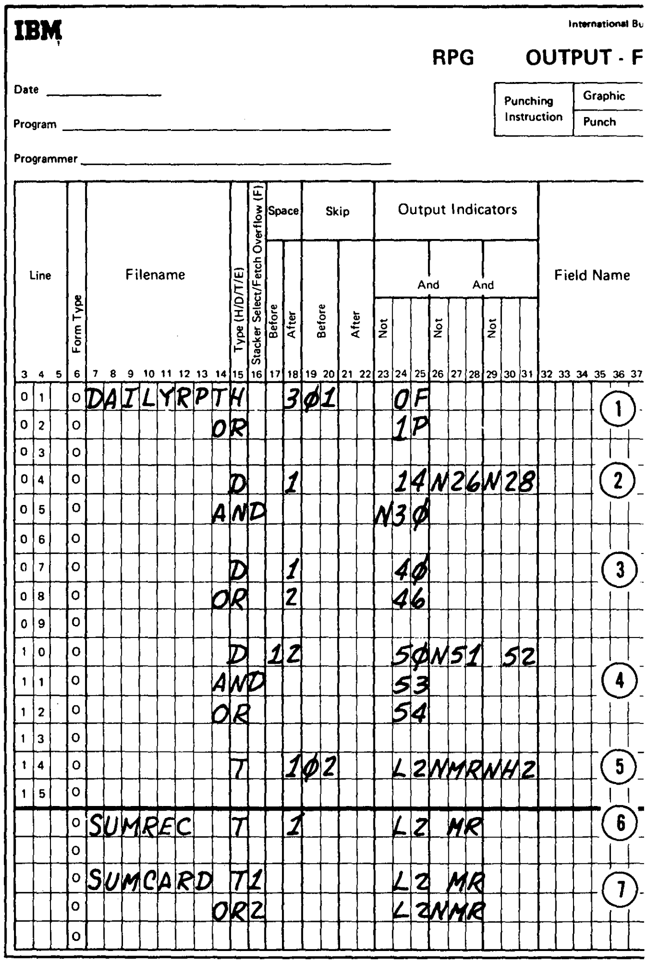
   - Indicator 54 is on.

5. The total line is printed and the control carriage skips to channel 2 before printing only if the level 2 (L2) indicator is on, the MR indicator is off, and H2 is off. The NH2 specification prevents the object program from printing a line if an error condition has occurred.

   The program does not stop until after all processing for the record has been completed. This feature enables the programmer to prevent the printing of erroneous data.

6. The summary record is printed when the L2 and MR indicators are on.

7. A summary card is punched when the L2 indicator is on. Depending upon the status of the MR indicator, the card is directed to a different stacker. This entry assumes a two-stacker punch device.



IBM

International Bu

RPG OUTPUT - F

Date

Program

Programmer

Punching Instruction | Graphic | Punch

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select/Fetch Overflow (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators Not		And Not		And Not		Field Name	
3 4 5	6	7-14	15	16	17	18	19 20	21 22	23	24 25	26	27 28	29	30 31	32-37	
01	O	DAILYRPT	H			3	Ø1			OF						1
02	O	O	R							1P						
03	O															
04	O		D			1				14	N	26	N	28		2
05	O	A	N	D					N	3Ø						
06	O															
07	O		D			1				4Ø						3
08	O	O	R			2				46						
09	O															
10	O		D		1	2				5Ø	N	51		52		
11	O	A	N	D						53						4
12	O	O	R							54						
13	O															
14	O		T			1	Ø2			L2	N	MR	N	H2		5
15	O															
	O	SUMREC	T			1				L2		MR				6
	O															
	O	SUMCARD	T	1						L2		MR				7
	O	O	R	2						L2	N	MR				
	O															

Figure 108. Specifying Output Indicators

FIELD DESCRIPTION

Field Description entries include specifications which:

1. Control the individual fields of a record.

2. Specify the output format of individual fields of a record. The fields of the 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 positions 7-22 of a field description line.

Output Indicators (23-31)

The same types of indicators used for file identification can be used for field description. The maximum number of indicators that can be considered in an AND relationship is three; all must be specified on one field description line. AND/OR entries in positions 14-16 cannot be used with the field description entries.

Figure 109 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 list:

1. Four fields (INVOIC, AMOUNT, CUSTR, and SALESM) are printed from a detail record identified by indicator 44. These fields represent invoice, amount, customer, and salesman information. The entry L1 causes the field, SALESM, to be printed only for the first detail record of each control group. (Remember that a control level indicator remains on during the following detail calculation and print cycle.) The salesman field is group indicated.

2. The second example illustrates how to prevent the printing of just one field of a record. AMOUNT is printed only if indicator 16 is off. This indicator is used to determine if the calculated field, AMOUNT, is zero.

3. This example selectively prints the field headings for an invoice form. This specification prints all the heading information on the first form, but if the information for one customer order continues on two or more forms, only the CUST and INVOIC fields on succeeding forms are printed.

   Printing of the entire line is controlled by indicator 04 or OF. In the field description specifications, the OF indicator is also used to prevent printing of ORDER ORDER, DATE, and SALESM when an overflow condition occurs.

4. The last example illustrates how a field can be controlled for printing by an AND relationship and an OR relationship. DIVSON is controlled by three AND indicators: 16, NH2, and NL3.

   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.

   The letters OR cannot be specified in positions 14-15 of a field description.

Field Name (Positions 32-37)

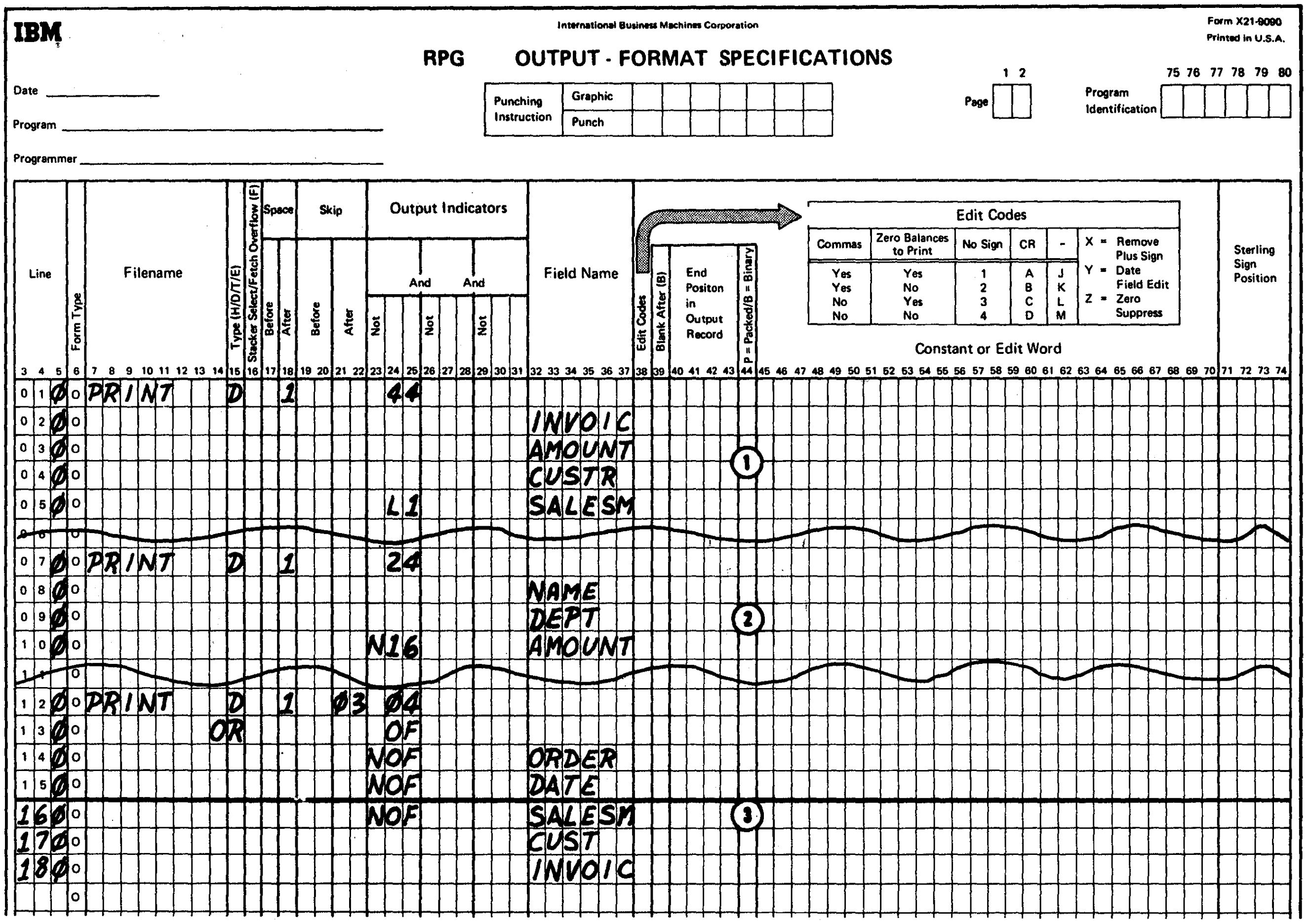
This specification identifies each field of the record to be written. The fields may be listed on the form in any sequence. The location in which each field appears in the output record is determined by the entry in positions 40-43.

The order in which output fields are moved to the output record is determined by the sequence in which they are specified on the Output-Format Specifications form.

Enter in positions 32-37 the name of the field that is defined on the line. The field name must have been previously defined on either the Input or Calculation Specifications form. (For an exception to this, see *Page Numbering* in this section.) If a constant is to be written, it is specified in Constant or Edit Word (positions 45-70), and Field Name is left blank.

The field names UDATE, UDAY, UMONTH, and UYEAR may be used in positions 32-37 if the user date is desired on the output record.

Examples of field names are given in Figure 109.



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 □□ (1 2)

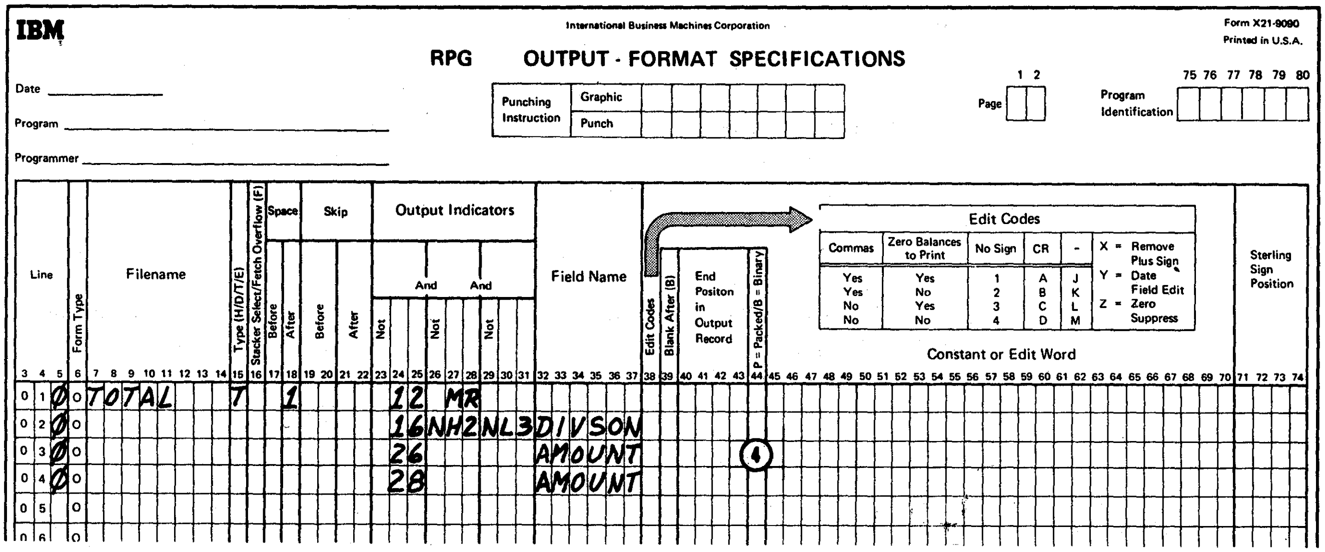
Program Identification □□□□□□ (75 76 77 78 79 80)

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)	Space After	Skip Before	Skip After	Not	Output Indicators	Not	And	Not	And	Field Name	
01Ø	O	PRINT	D	1				44						
02Ø	O												INVOIC	
03Ø	O												AMOUNT	
04Ø	O												CUSTR	1
05Ø	O							L1					SALESM	
07Ø	O	PRINT	D	1				24						
08Ø	O												NAME	
09Ø	O												DEPT	2
10Ø	O						N	16					AMOUNT	
12Ø	O	PRINT	D	1		Ø3		Ø4						
13Ø	O	OR						OF						
14Ø	O						N	OF					ORDER	
15Ø	O						N	OF					DATE	
16Ø	O						N	OF					SALESM	3
17Ø	O												CUST	
18Ø	O												INVOIC	



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 □□ (1 2)

Program Identification □□□□□□ (75 76 77 78 79 80)

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)	Space After	Not	Output Indicators	Not	And	Not	And	Field Name	
01Ø	O	TOTAL	T	1		12		MR				
02Ø	O					16	N	H2	N	L3	DIVSON	
03Ø	O					26					AMOUNT	4
04Ø	O					28					AMOUNT	
05	O											
06	O											

Figure 109. Specifying Field Names

Edit Codes (38)

An entry of Z in this specification causes zero suppression of the field in the output record, which must be a numeric field. Zero suppression means that zeros to the left of the first (high-order) significant digit in the field are replaced by blanks in the output record. Entries other than Z or blank are invalid.

This specification performs an additional function: zone bits in the unit position of a field are removed during zero suppression. This provides a means of zone elimination for amount fields. Normally, this specification is used for printing numeric fields, but not for punching them, because it is usually necessary to punch the sign codes and leading zeros.

If an edit control word is specified for the field, this specification must be blank.

Blank After (39)

An entry of B in this specification causes the output field to be reset to blanks or zeros after the field is placed in the output record. Alphameric fields are set to blanks; numeric fields are set to zeros.

This specification has an additional feature: if the output field being reset by the Blank After entry is also being tested for zero or blank on the input specifications or being tested for zero on the calculation specifications, the corresponding indicator is set on after the output field is reset. However, associated plus or minus indicators are not set off.

If a given field is specified for output at more than one position in the same record, the Blank After entry is made only on the last specification where the field is used.

Note: A Blank After entry also affects any constant in storage. 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 entry is not available for use in subsequent operations.

End Position In Output Record (40-43)

This specification indicates the location of the field in the output record. In positions 40-43, enter only the place in the output record where the rightmost (low-order) character of the field is to be located.

Assume that a 10-character amount field is to be printed in positions 21 through 30 in the output record. The entry in positions 42-43 would be 30. Positions 40-41 are left blank, because the leading zeros may be omitted.

If an amount field is being edited, the program considers the entire edit word as the field to be placed in the output record. For example, assume the letters CR are to be printed at the end of minus amount fields. The R would be printed in position 30, the C in position 29, and the 10-character amount field in positions 19 through 28. Thus the amount field is always printed in positions 19-28, and the credit symbol (if applicable) is printed in positions 29 and 30.

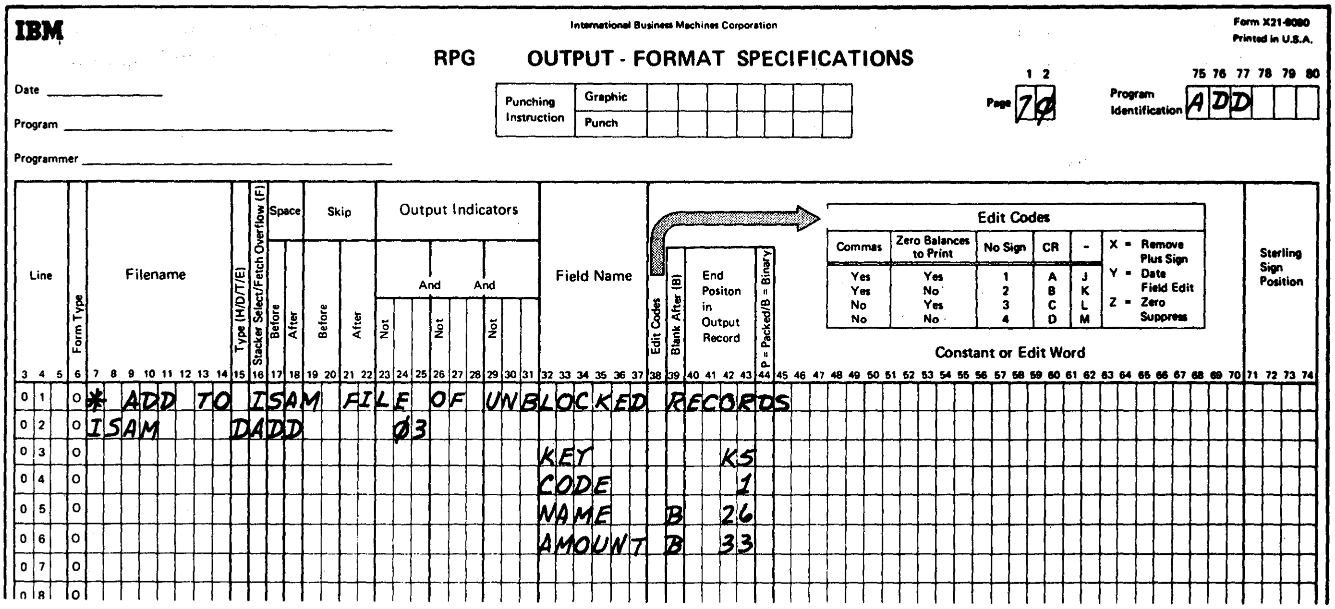
A field within the key area (key not in the data) may be specified, provided the key area is associated with a record in an indexed-sequential load or add file specifying unblocked records with no Key Field Starting Location given (see *File Description Specifications Form*). To specify a key area field, enter a K to the immediate left of the high-order digit in the End Positions in Output Record positions. At least one nonzero digit must appear to the right of any K entry.

Figure 110 shows how records are added to an indexed-sequential file with unblocked record format (ISAM). The K in position 42 of line 030 defines the key field. The key field is put out in the key areas ending in position 5.

Note: Any key area positions left unfilled by field specifications will be filled with blanks. No fields within the key area may be specified for update records.

See the section *RPG User's Guide for ISAM Processing* for an explanation of key and record formats.

Figure 111 illustrates various field description specifications. It shows the specifications for printing a detail line. The three quantity fields ORDQTY, SCRAPQ, and RECPTQ are zero suppressed. SCRAPQ and RECPTQ are reset to zeros by the Blank After specification. The UDATE field is zero suppressed.



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

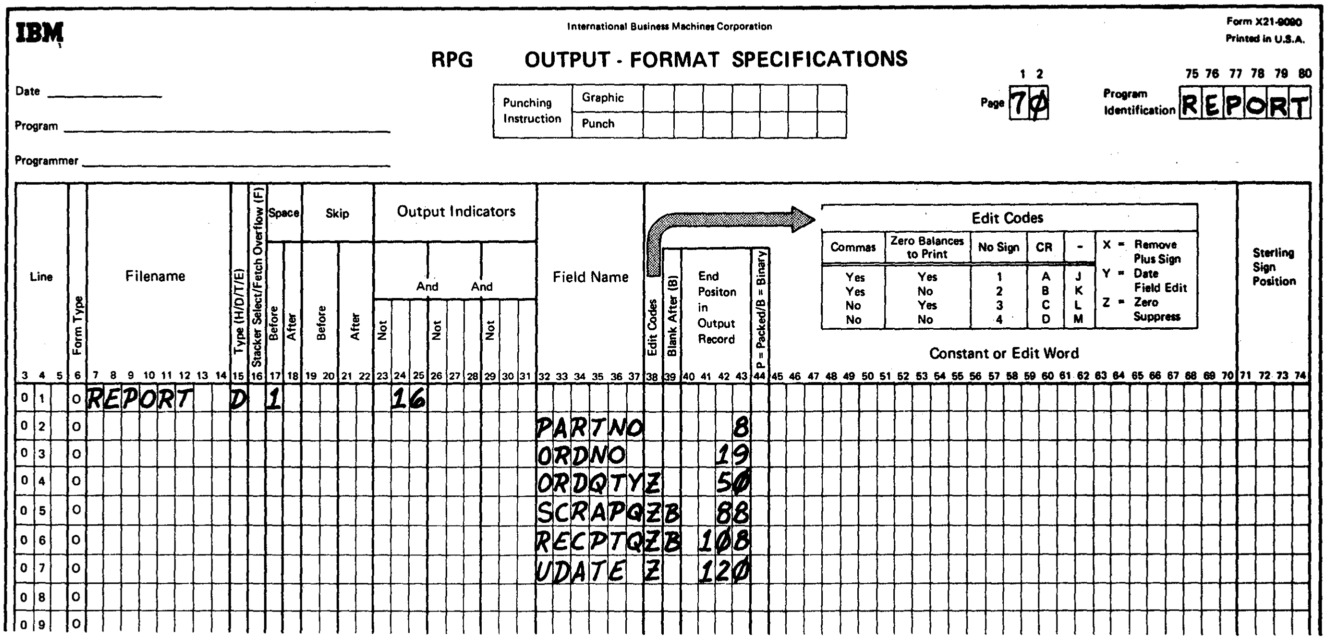
RPG OUTPUT - FORMAT SPECIFICATIONS

Date ____ Program ____ Programmer ____

Page 7Ø Program Identification ADD

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select/Fetch Overflow (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	P = Packed/B = Binary	Constant or Edit Word
01	O	* ADD TO ISAM FILE OF UNBLOCKED RECORDS													
02	O	ISAM	D	ADD					Ø3						
03	O									KEY			K5		
04	O									CODE			1		
05	O									NAME		B	26		
06	O									AMOUNT		B	33		
07	O														
08	O														

Figure 110. Adding Records to an Indexed Sequential File



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

RPG OUTPUT - FORMAT SPECIFICATIONS

Date ____ Program ____ Programmer ____

Page 7Ø Program Identification REPORT

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select/Fetch Overflow (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	P = Packed/B = Binary	Constant or Edit Word
01	O	REPORT	D		1				16						
02	O									PARTNO			8		
03	O									ORDNO			19		
04	O									ORDQTY	Z		5Ø		
05	O									SCRAPQ	Z	B	88		
06	O									RECPTQ	Z	B	1Ø8		
07	O									UDATE	Z		12Ø		
08	O														
09	O														

Figure 111. Field Description Specifications

Figure 112 illustrates an example of field selection specification. These entries punch a summary card at control L1. In this example, a new balance (NEWBAL) and an old balance (OLDBAL) are used in the weekly reports; at the end of the month only the new balance is to be summarized for the following month's report.

Both fields are specified for the same punching positions in the output record. The actual field that is punched, however, is controlled by the setting of indicator 26. The number of fields that can be specified for field selection is not limited.

Packed Field (44)

Packed format in System/360 means that two decimal digits can be represented in one core storage byte. This is the data format used for numeric fields in RPG. Because input data is usually represented in the unpacked format (one digit in one core storage byte) the RPG program automatically converts numeric input data from the unpacked format to the packed decimal format.

Because the packed decimal format permits greater utilization of storage capacities (card-tape-disk), the RPG program permits numeric data to be put out to these devices in the packed decimal format.

Enter a P in position 44 if the numeric output field is to be in the packed decimal format. Otherwise, leave this position blank. The letter P causes the RPG program to bypass the normal conversion of packed decimal format to unpacked format.

The number of bytes occupied in the output record by a numeric field in packed format can be determined in the following manner:

1. Divide the length of the field by 2
2. Eliminate the rightmost decimal fractions
3. Increase the result by one.

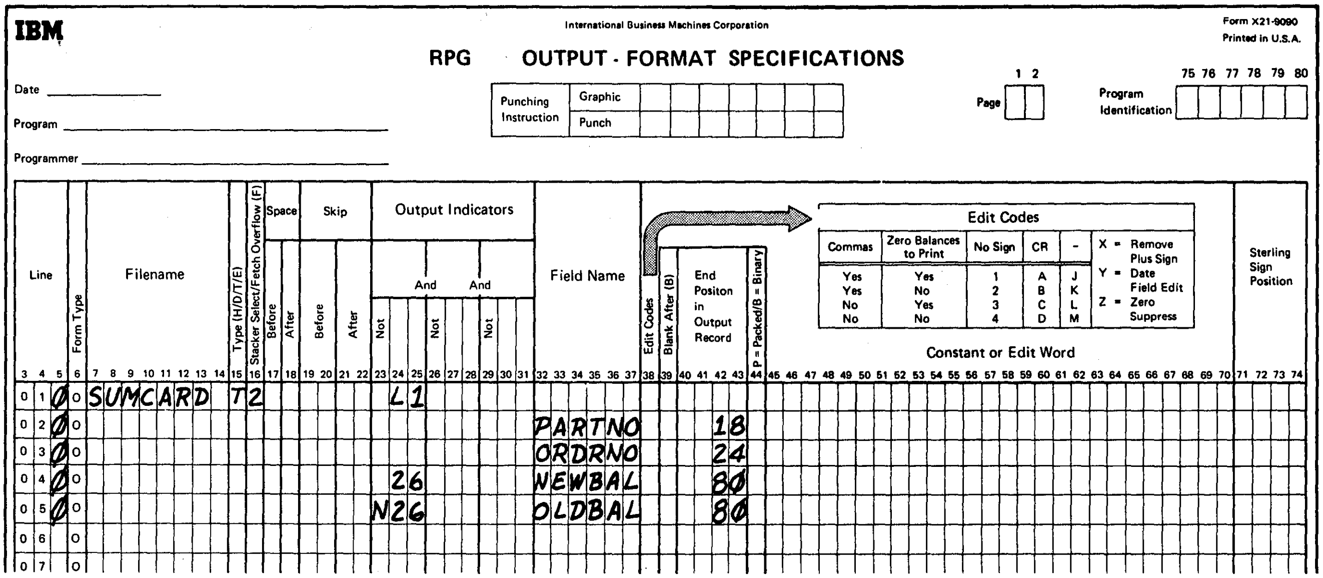
Steps	*6-Position Numeric Field*	*9-Position Numeric Field*
1	6/2=3.0	9/2=4.5
2	3.0=3	4.5=4
3	3+1=4 positions	4+1=5 positions

Position (44) must be left blank for all alphameric fields and for numeric fields used with:

1. Zero suppression
2. Edit control words
3. Sterling

Page Numbering

Page numbering, another automatic feature of RPG, is specified in the field description portion of the output specifications. The PAGE entry (Figure 113, line 080) causes each page of the Output-Format Specifications form to be consecutively numbered in print positions 65 to 68 (the page number is always defined as four positions long unless defined differently on the Input or Calculation Specifications form). The page number is increased by one before it is printed.



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 1 2 □□

Program Identification 75 76 77 78 79 80 □□□□□□

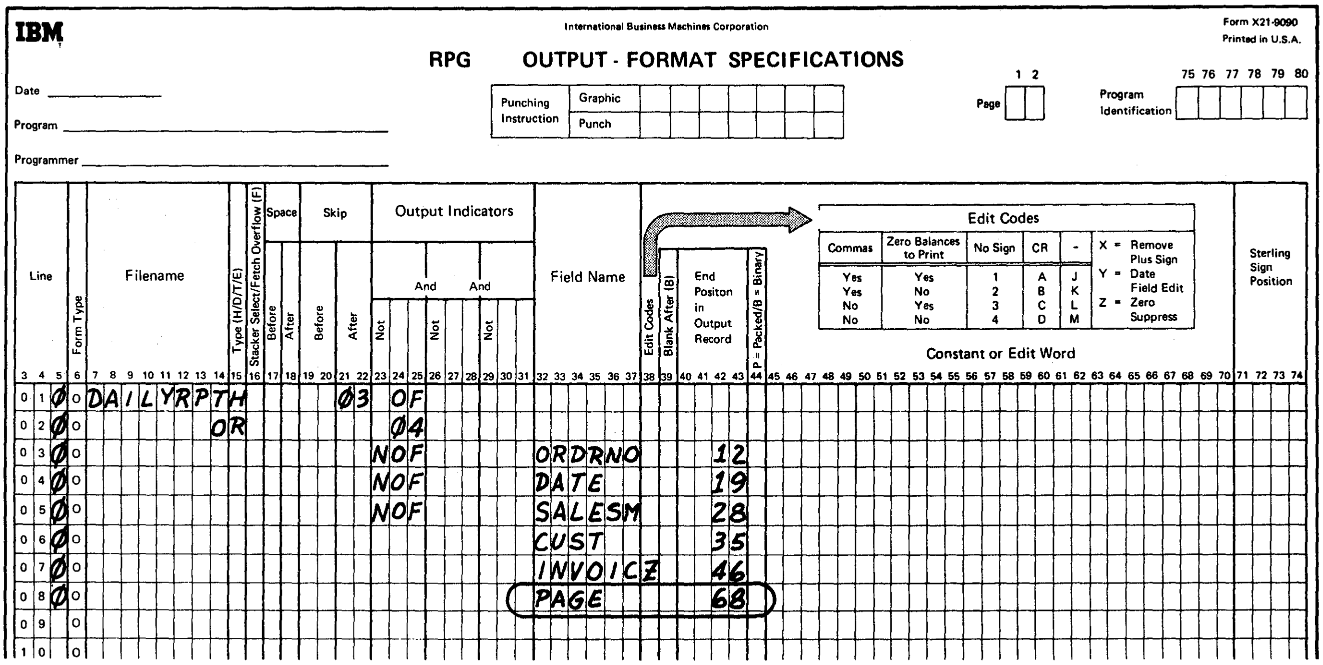
Edit Codes

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 3-5	Form Type 6	Filename 7-14	Type (H/D/T/E) 15	Stacker Select/Fetch Overflow (F) 16	Space Before 17	Space After 18	Skip Before 19-20	Skip After 21-22	Output Indicators: Not 23	24-25	And: Not 26	27-28	And: Not 29	30-31	Field Name 32-37	Edit Codes 38	Blank After (B) 39	End Positon in Output Record 40-43	P = Packed/B = Binary 44	Constant or Edit Word 45-70	Sterling Sign Position 71-74
01ø	O	SUMCARD	T	2						L1											
02ø	O														PARTNO			18			
03ø	O														ORDRNO			24			
04ø	O									26					NEWBAL			8ø			
05ø	O								N	26					OLDBAL			8ø			
06	O																				
07	O																				

Figure 112. Example of Field Selection Specifications



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 1 2 □□

Program Identification 75 76 77 78 79 80 □□□□□□

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 3-5	Form Type 6	Filename 7-14	Type (H/D/T/E) 15	Stacker Select/Fetch Overflow (F) 16	Space Before 17	Space After 18	Skip Before 19-20	Skip After 21-22	Output Indicators: Not 23	24-25	And: Not 26	27-28	And: Not 29	30-31	Field Name 32-37	Edit Codes 38	Blank After (B) 39	End Positon in Output Record 40-43	P = Packed/B = Binary 44	Constant or Edit Word 45-70	Sterling Sign Position 71-74
01ø	O	DAILYRPT	H					ø3		OF											
02ø	O		OR							ø4											
03ø	O								N	OF					ORDRNO			12			
04ø	O								N	OF					DATE			19			
05ø	O								N	OF					SALESM			28			
06ø	O														CUST			35			
07ø	O														INVOIC	Z		46			
08ø	O														PAGE			68			
09	O																				
10	O																				

Figure 113. Example of Page Numbering Specifications

Page numbering normally begins with number 1, but the programmer may start page numbering with any number by preparing a header card containing the starting page number, less 1. The header card may define the PAGE field on the Input Specification form. The PAGE field may also be referenced on the Calculation Specification form.

The field must be defined as numeric (0 in position 52) and can be 1-15 positions in length. In Figure 114 the page number is initialized from sequence AA on the Input Specifications Form (line 010). If page numbering were to begin with the number 500, 499 would be punched in positions 1-4 of the input record that contains the character P in position 80. PAGE is defined on the Output-Format Specifications form as a separate field that prints in position 100. Figure 114 also shows an example of editing the field defined by the reserved word UDATE.

It is possible to reset the page number to zero and start a new series of page numbers during the processing of the program. For example, rather than number each page of a report, it may be required to start page numbering with 01 for each major control break. In this case the major control level indicator must be placed in the Output Indicators on the same line as the Field Name specification PAGE.

Any output indicators specified in a PAGE line are checked before printing. If all conditions are met, the page counter is reset to 0 before being incremented by 1.

Figure 115 shows an example of resetting the PAGE counter field on an L3 control level. A new page is skipped and a line is printed when the L1 indicator is on. The PAGE number field is reset if L3 is on when the line is printed. The PAGE counter field is incremented by one prior to printing.

Page Numbering for Multiple Printers

The individual printer files can initialize page numbering. Eight special PAGE entries (PAGE, PAGE1, PAGE2, PAGE3, PAGE4, PAGE5, PAGE6, and PAGE7) can be initialized to any count in the same manner as described for PAGE.

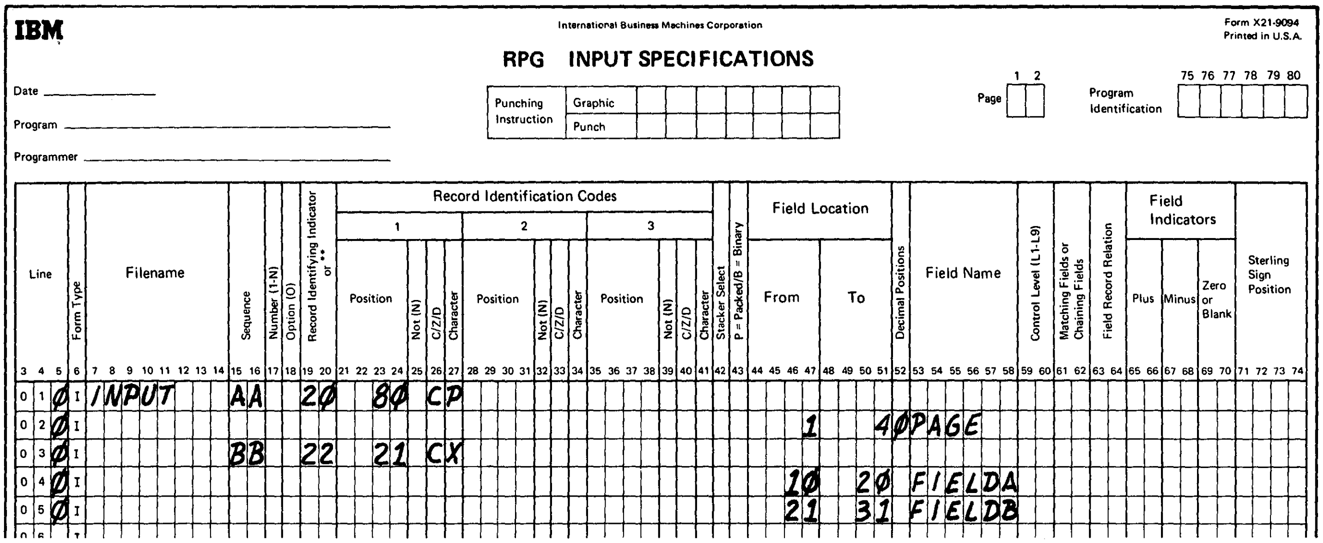
Constant or Edit Word (45-70)

This specification provides constants in the output records or permits editing of numeric fields. Constants and edit words must be left justified.

Constants

This entry provides a convenient method of placing alphameric data into the object program. These alphameric constants or literals do not change from one processing of the program to the next. A literal is the actual data to be used in the output record rather than a name representing the location of data.

A literal of up to 24 alphameric characters may be placed in positions 45-70. The literal must begin with an apostrophe in position 45, and it must be enclosed by a set of apostrophe symbols even if it contains only numeric characters. The literal beginning in position 45 is placed on the output record as defined in End Position in Output Record (positions 40-43).



IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date ____________

Program ____________

Programmer ____________

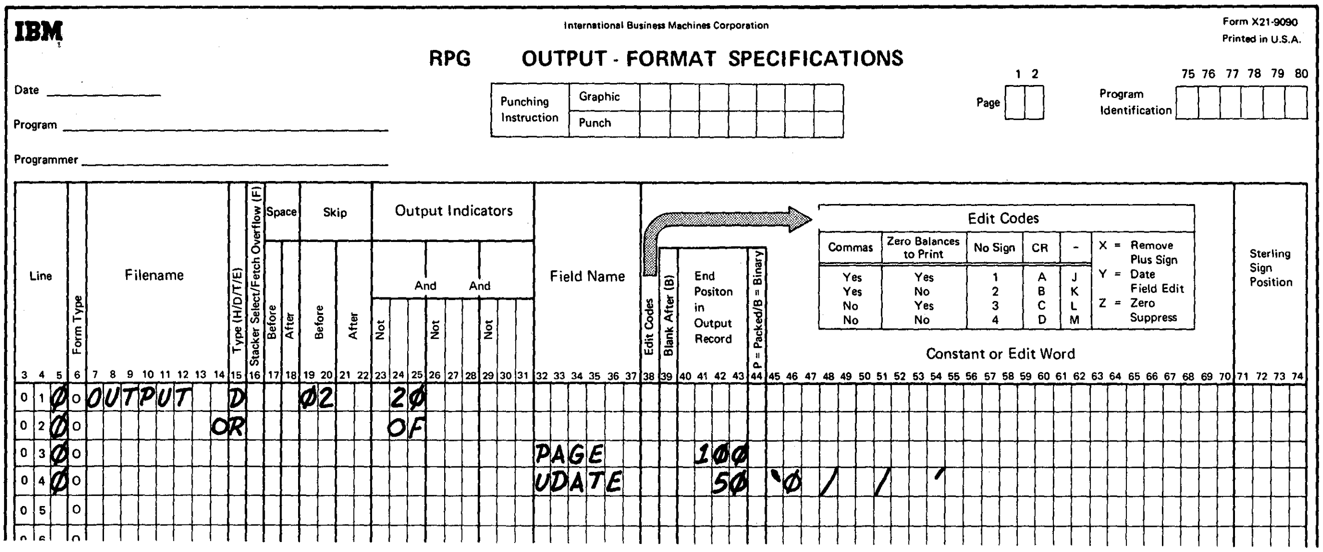
Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line (3-5) | Form Type (6) | Filename (7-14) | Sequence (15-16) | Number (1-N) (17) | Option (O) (18) | Record Identifying Indicator or ** (19-20) | Record Identification Codes 1, 2, 3: Position, Not (N), C/Z/D, Character (21-41) | Stacker Select (42) | P = Packed/B = Binary (43) | Field Location: From (44-47), To (48-51) | Decimal Positions (52) | Field Name (53-58) | Control Level (L1-L9) (59-60) | Matching Fields or Chaining Fields (61-62) | Field Record Relation (63-64) | Field Indicators: Plus (65-66), Minus (67-68), Zero or Blank (69-70) | Sterling Sign Position (71-74)

```
01ØIINPUT   AA  2Ø  8Ø CP
02ØI                                        1   4ØPAGE
03ØI        BB  22  21 CX
04ØI                                       1Ø  2Ø FIELDA
05ØI                                       21  31 FIELDB
06 I
```



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 1 2

Program Identification 75 76 77 78 79 80

Line (3-5) | Form Type (6) | Filename (7-14) | Type (H/D/T/E) (15) | Stacker Select/Fetch Overflow (F) (16) | Space: Before (17), After (18) | Skip: Before (19-20), After (21-22) | Output Indicators: Not, And Not, And Not (23-31) | Field Name (32-37) | Edit Codes (38) | Blank After (B) (39) | End Positon in Output Record (40-43) | P = Packed/B = Binary (44) | Constant or Edit Word (45-70) | Sterling Sign Position (71-74)

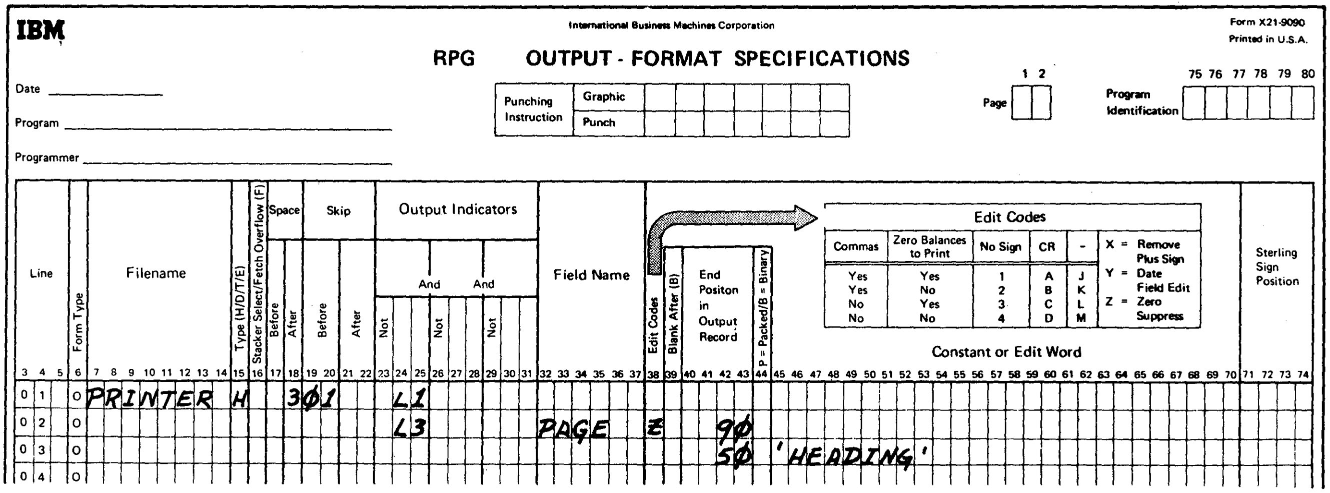
Edit Codes

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

```
01ØOOUTPUT  D   Ø2    2Ø
02ØO       OR        OF
03ØO                         PAGE       1ØØ
04ØO                         UDATE       5Ø 'Ø /  /  '
05 O
06 O
```

Figure 114. Page Numbering and UDATE



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 1 2

Program Identification 75 76 77 78 79 80

Line (3-5) | Form Type (6) | Filename (7-14) | Type (H/D/T/E) (15) | Stacker Select/Fetch Overflow (F) (16) | Space: Before (17), After (18) | Skip: Before (19-20), After (21-22) | Output Indicators: Not, And Not, And Not (23-31) | Field Name (32-37) | Edit Codes (38) | Blank After (B) (39) | End Positon in Output Record (40-43) | P = Packed/B = Binary (44) | Constant or Edit Word (45-70) | Sterling Sign Position (71-74)

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

```
01 OPRINTER H  3Ø1    L1
02 O                 L3      PAGE  Z   9Ø
03 O                                   5Ø 'HEADING'
04 O
```

Figure 115. Resetting the Page Field

Rules for Forming Alphameric Literals in an Output Record

1. Any character in the character set may be used in an alphameric literal.

2. Alphameric literals must be enclosed in a set of apostrophe symbols. An apostrophe symbol may be contained within a literal by entering two consecutive apostrophe symbols within the literal. For example, the literal o'clock would be entered as 'o''clock' in positions 45-54 of the Output-Format Specifications form.

3. Field Name (positions 32-37) must be left blank when an alphameric literal is defined on the line.

Figure 116 shows an example of constants being written. Line 030 shows one constant being written. Lines 060 and 070 show one large constant which requires two specification lines to create. This one line of information is treated as two constants.

Edit Words

An edit word provides for the punctuation of amount fields, including the printing of dollar signs, commas, periods, and sign status. Only numeric unpacked fields may be edited. Position 38 (Edit Codes) and Position 44 (Packed) must be left blank. An edit operation is shown in Figure 117.

When an amount field is to be edited, its edit word is placed in positions 45-70 of the line which specifies the field. An edit word consists of two parts (the body and the status) as shown in Figure 118.

The body of the edit word is the portion beginning with the leftmost character and continuing to the right of the character that governs the transfer of the rightmost positon of the data field.

The status of the edit word is the portion continuing to the right from the body including all characters preceding the closing apostrophe symbol. The CR (credit) or – (minus), if present, may appear anywhere within the status.

A character in the edit word that is replaced by a numeric character from the data field is referred to as a digit position. A blank in the edit word is a digit position as are the characters 0, *, and $ under certain conditions.

Rules for Forming an Edit Word

1. An edit word must be enclosed in a set of apostrophe symbols.

2. A blank in the edit word is replaced with the digit from the corresponding position of the data field specified in Field Name.

3. An ampersand causes a space in the edited field. There is no way to obtain an actual ampersand itself in an edited field.

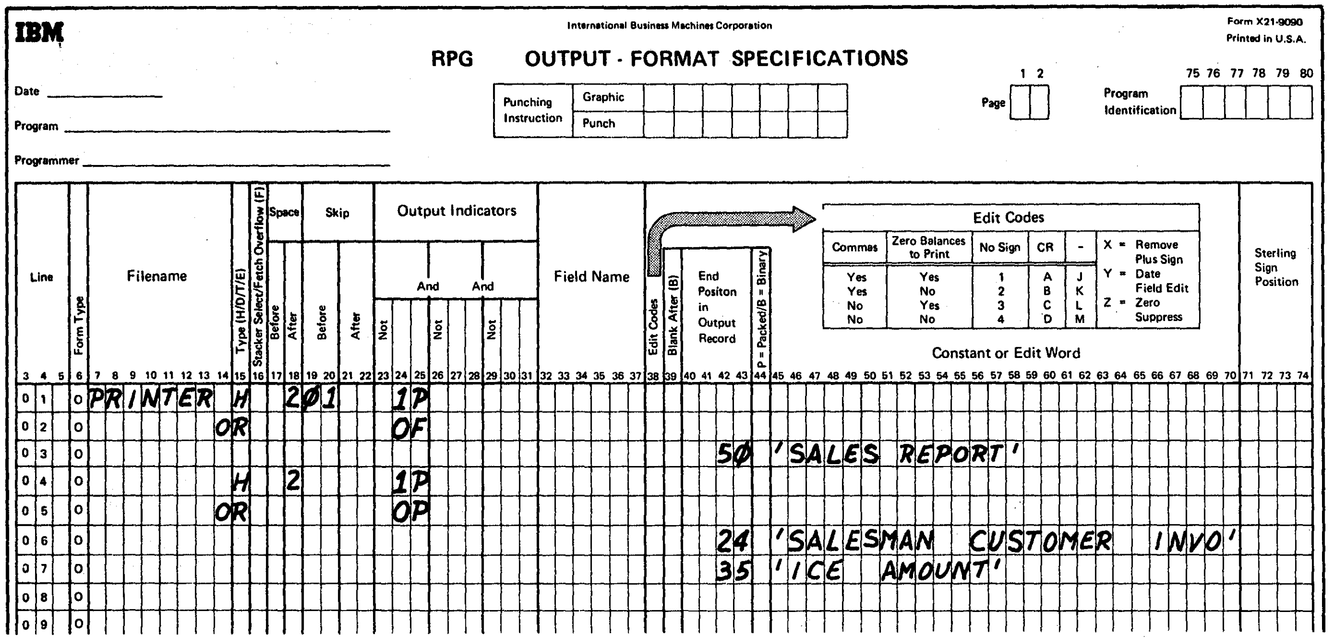
4. A zero is used to stop zero suppression. It is put in the rightmost position where zero suppression is to take place. It is replaced with the character from the corresponding position of the data field unless that character is a zero. At least one leading zero is suppressed.

5. An asterisk in the body of the control word is used for asterisk protection and zero suppression. It is put in the rightmost position where zero suppression is to take place. It is replaced with the character from the corresponding position of the data field unless that character is a zero and there is no significant digit to its left. Each zero that is suppressed is replaced by an asterisk.

6. A dollar sign in the body of the edit word written immediately to the left of the zero suppression code (0) causes the insertion of a dollar sign in the position to the left of the first significant digit. This is called the floating dollar sign. If it is necessary for the dollar sign to appear when all digits in the data are significant, the edit word must start with an ampersand to allow a space in which the dollar sign can print.

   A dollar sign that is entered immediately after the initial apostrophe symbol is fixed. A fixed dollar sign is printed in the same location each time. For example, in the edit word '$0bb' the dollar sign is considered to be fixed and not floating.

7. The decimal point and commas are printed in the edited output field in the same relative positions in which they were written in the edit word unless they are to the left of significant digits. In that case, they are blanked out or replaced by an asterisk. Any character other than the fixed dollar sign which precedes the first digit position is always suppressed.



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 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select/Fetch Overflow (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	P = Packed/B = Binary	Constant or Edit Word	Sterling Sign Position
01	O	PRINTER	H			2	01		1P							
02	O	OR							OF							
03	O												50		'SALES REPORT'	
04	O		H			2			1P							
05	O	OR							OP							
06	O												24		'SALESMAN CUSTOMER INVO'	
07	O												35		'ICE AMOUNT'	
08	O															
09	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 116. Specifying Constants

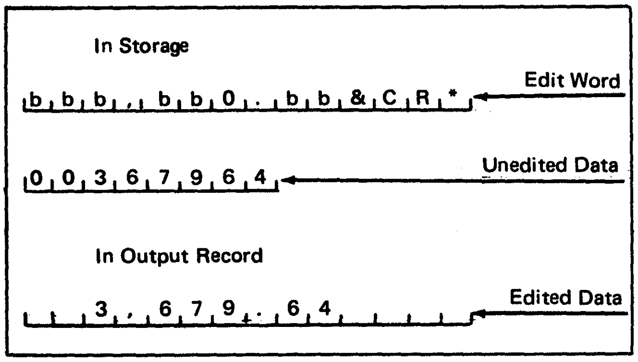


Figure 117. Edit Operation

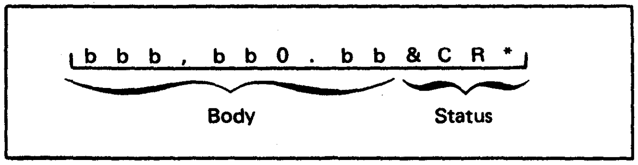


Figure 118. Two Parts of an Edit Word: Body and Status

8. 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 replaced by an asterisk, if asterisk protection is used, or they are blanked out.

9. The entry CR or a minus symbol in the status portion of the edit word is undistrubed if the sign in the data field is minus. If the sign is plus, the CR or minus is blanked out.

10. Asterisks to the right of the CR or minus symbol are undisturbed. They are normally used to indicate a specific class of total. An error results unless a zero (for zero suppression) or an asterisk (for asterisk protection) are present in the body of the edit word.

11. If there are more digit positions in the edit word than there are digits in the field to be edited, leading zeros are added to the field before editing.

12. An edit word can contain a maximum of 15 digit positions. An exception occurs for a sterling field where the maximum is 16 digit positions.

13. If there are multiple minus or CR symbols in an edit word, the first one to the right of the last replaceable character is considered the status. Any other minus or CR symbols are treated as constants.

14. If no zero suppression character is entered, any constants in the body of the edit word are suppressed.

Figure 119 illustrates the use of constants and edit words. The numbers to the left correspond to the following list:

1. The constant 26.75 is in the output record ending in position 96. The Field Name specification must be blank.

2. The constant DEPARTMENT TOTAL is contained in the output record ending in position 96. The Field Name specification must be blank.

3. This example illustrates zero suppression to the left of significant digits. The letters CR are written because the amount field might be minus.

4. In this example, 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 120 shows examples of edit words, the data, and the results after the field is edited. The same end print position is specified for all output fields.

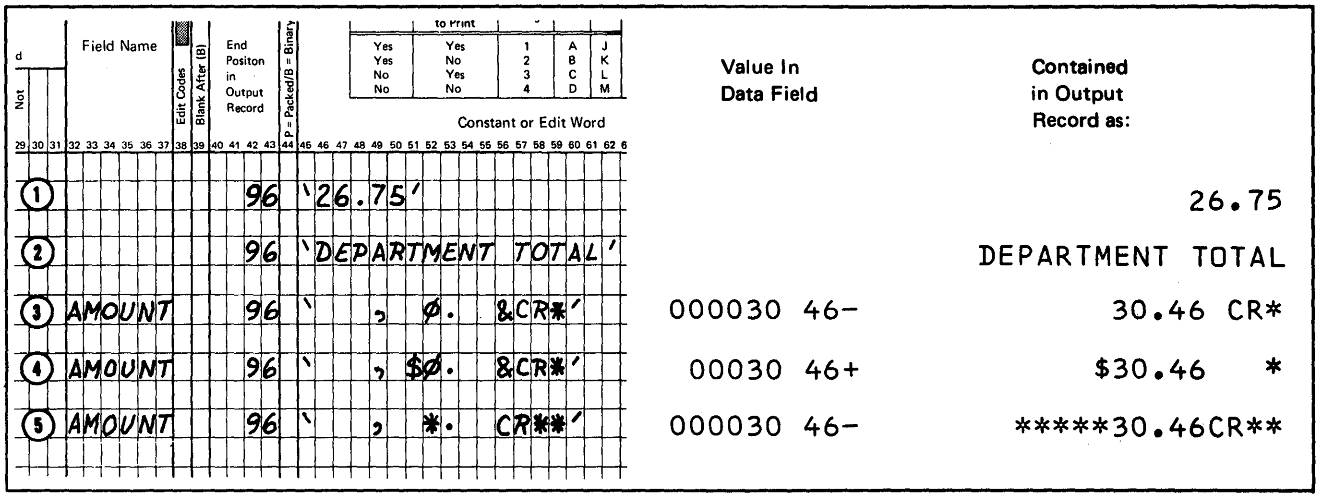
When you use a printer with the UCS feature, data checks can occur during RPG compilation of a program that uses edit words. To suppress these unit checks, one of the following options must be used:

1. Suppress data checks at compile time by omitting the SYSPRINT DD card parameter DCB=(,OPTCD=U). (Refer to *IBM System/360 Operating System, Supervisor and Data Management Macro-Instructions* Form C28-6647.) Cataloged procedures do not include this parameter and can therefore be used to suppress data checks on a printer with UCS.

2. Use the utility program, IEHUCSLD, (*IBM System/360 Operating System Utilities,* Form C28-6586*)* to load X'20' and X'21' into the UCS buffer.

Sterling Sign Position (71-74)

Enter in these positions the location in the record that will contain the sign of the sterling field. Leading zeros may be omitted. Enter an S in position 74 if the sign is in the normal place. For print files the normal position must be used. If the Sterling specification is not required, leave this position blank. Addition information on Sterling is in *Sterling Routines for RPG.*



				to Print			
				Yes	Yes	1	A	J
				Yes	No	2	B	K
				No	Yes	3	C	L
				No	No	4	D	M

Not (29 30 31)	Field Name (32–37)	Edit Codes (38)	Blank After (B) (39)	End Positon in Output Record (40–43)	P = Packed/B = Binary (44)	Constant or Edit Word (45–62)	Value In Data Field	Contained in Output Record as:
①				96		'26.75'		26.75
②				96		'DEPARTMENT TOTAL'		DEPARTMENT TOTAL
③	AMOUNT			96		' , Ø. &CR*'	000030 46-	30.46 CR*
④	AMOUNT			96		' , $Ø. &CR*'	00030 46+	$30.46 *
⑤	AMOUNT			96		' , *. CR**'	000030 46-	*****30.46CR**

Figure 119. Using Constants and Edit Words

Edit Word	Data	Result	
		If Data is Positive	If Data is Negative
' , 0. '	000005	.05	.05
' , 0. '	100000	1,000.00	1,000.00
' AMT0'	005	5	5
' AMT0'	100	10AMT0	10AMT0
' 0AMT '	005	AMT5	AMT5
' 0AMT '	1005	100AMT5	100AMT5
' XX '	005	5	5
' XX '	100	1XX5	1XX5
'. '	05	5	5
' , $0. '	000005	$.05	$.05
' , $0. '	100000	$1,000.00	$1,000.00
' ,$0 . '	000010	$0.10	$0.10
'$0 . '	00100	$ 01.00	$ 01.00
' 0–'	010	10	10–
' 0CR*'	010	10 *	10CR*
'0 – – '	123456789	123–45–6789	123–45–6789
' 0CR--'	05	5 –	5CR–
' 0CREDIT'	05	5 EDIT	5CREDIT
' **'	10	*10*	*10*
' 0**'	10	10**	10**
'$, *. '	000010	$*****.10	$*****.10
'$, *. '	050000	$**500.00	$**500.00
' 0 '	05	5	5
' 0 '	05	05	05
'0 '	05	005	005

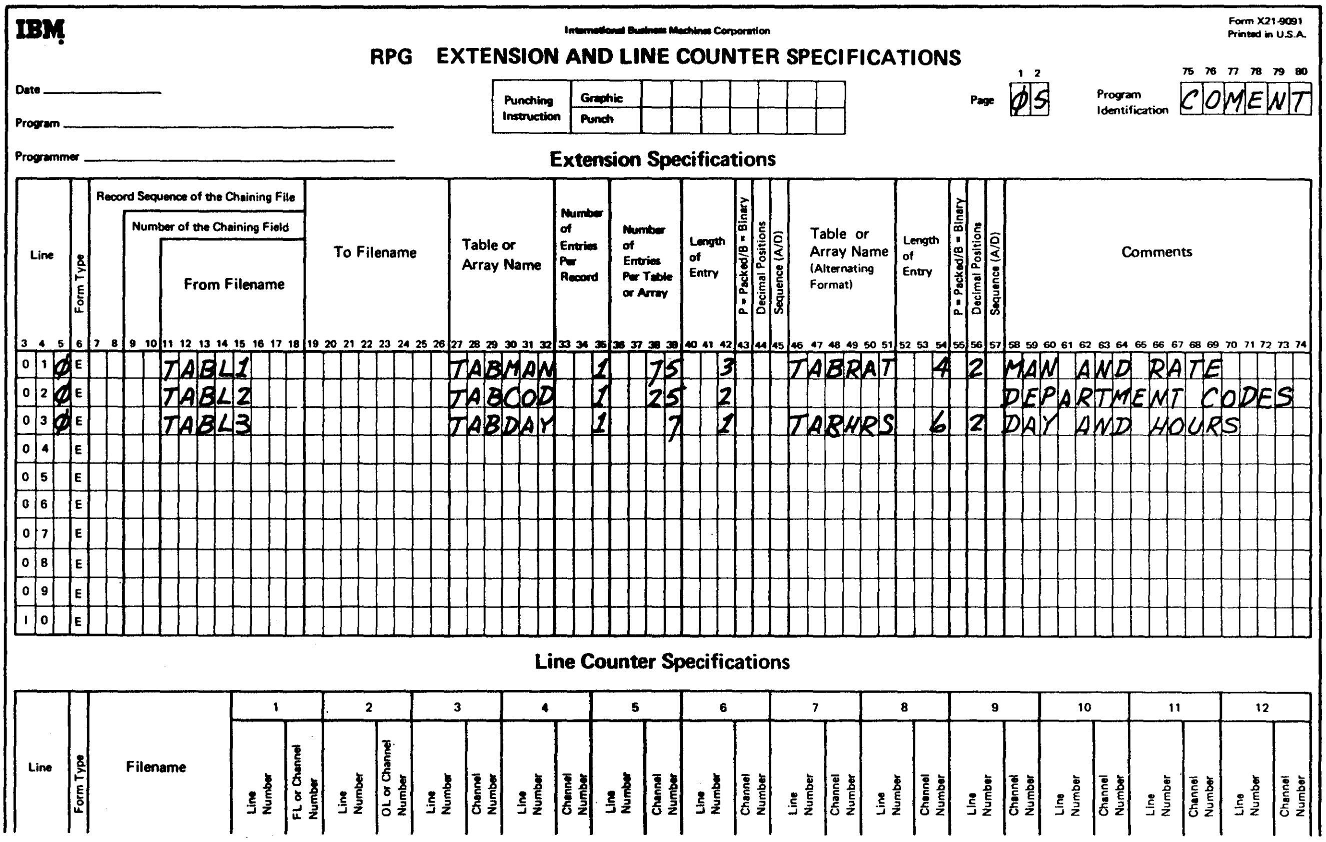
Figure 120. Editing Examples

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 (Figure 121 and 122). 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 position 7. They can be used at the beginning of the program to describe functions of that program (see Figure 123), or they can be used to denote the beginning of major sections of a program (Figure 124).

The indicator Summary Form (Figure 125) can be used to 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 (1 2) ∅5

Program Identification (75–80) COMENT

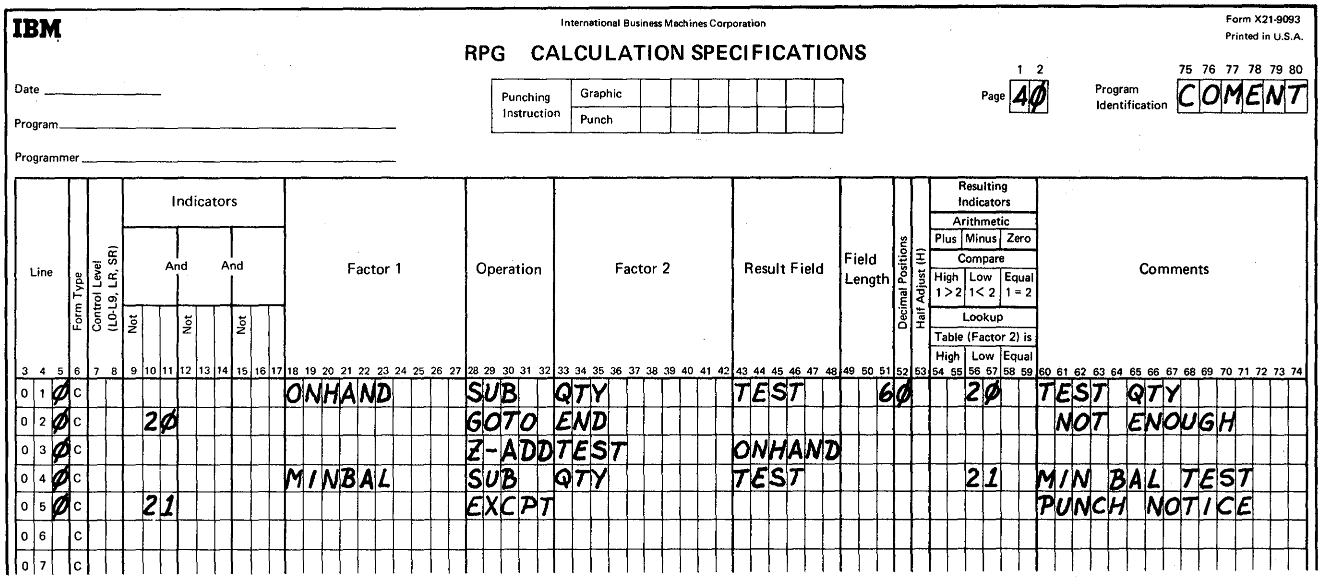
Extension Specifications

Line (3–5)	Form Type (6)	Record Sequence of the Chaining File (7–8)	Number of the Chaining Field (9–10)	From Filename (11–18)	To Filename (19–26)	Table or Array Name (27–32)	Number of Entries Per Record (33–35)	Number of Entries Per Table or Array (36–39)	Length of Entry (40–42)	P = Packed/B = Binary (43)	Decimal Positions (44)	Sequence (A/D) (45)	Table or Array Name (Alternating Format) (46–51)	Length of Entry (52–54)	P = Packed/B = Binary (55)	Decimal Positions (56)	Sequence (A/D) (57)	Comments (58–74)
01∅	E			TABL1		TABMAN	1	75	3				TABRAT	4		2		MAN AND RATE
02∅	E			TABL2		TABCOD	1	25	2									DEPARTMENT CODES
03∅	E			TABL3		TABDAY	1	7	1				TABHRS	6		2		DAY AND HOURS
04	E																	
05	E																	
06	E																	
07	E																	
08	E																	
09	E																	
10	E																	

Line Counter Specifications

Line	Form Type	Filename	1 Line Number	1 FL or Channel Number	2 Line Number	2 OL or Channel Number	3 Line Number	3 Channel Number	4 Line Number	4 Channel Number	5 Line Number	5 Channel Number	6 Line Number	6 Channel Number	7 Line Number	7 Channel Number	8 Line Number	8 Channel Number	9 Line Number	9 Channel Number	10 Line Number	10 Channel Number	11 Line Number	11 Channel Number	12 Line Number	12 Channel Number

Figure 121. Program Documentation



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

RPG CALCULATION SPECIFICATIONS

Date ______________

Program ______________

Programmer ______________

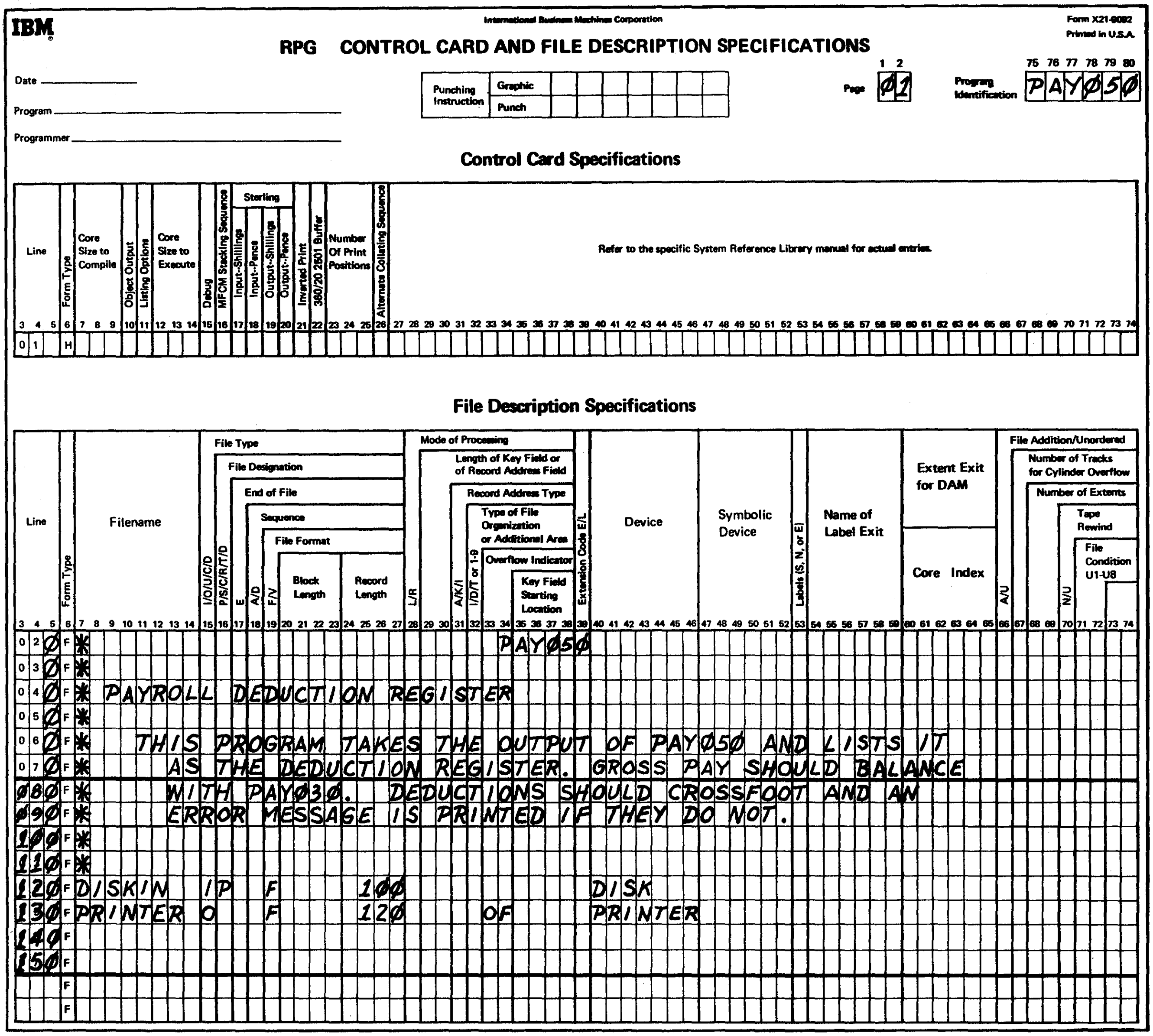
Punching Instruction: Graphic ______ Punch ______

Page (1 2): 4Ø

Program Identification (75 76 77 78 79 80): COMENT

Line (3-5)	Form Type (6)	Control Level (L0-L9, LR, SR) (7-8)	Indicators: Not (9)	Indicators (10-11)	And: Not (12)	And (13-14)	And: Not (15)	And (16-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field (43-48)	Field Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Plus / High 1>2 / High (54-55)	Minus / Low 1<2 / Low (56-57)	Zero / Equal 1=2 / Equal (58-59)	Comments (60-74)
01Ø	C								ONHAND	SUB	QTY	TEST	6	Ø			2Ø		TEST QTY
02Ø	C			2Ø						GOTO	END								NOT ENOUGH
03Ø	C									Z-ADD	TEST	ONHAND							
04Ø	C								MINBAL	SUB	QTY	TEST					21		MIN BAL TEST
05Ø	C			21						EXCPT									PUNCH NOTICE
06	C																		
07	C																		

Figure 122. 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 Ø1 Program Identification PAYØ5Ø

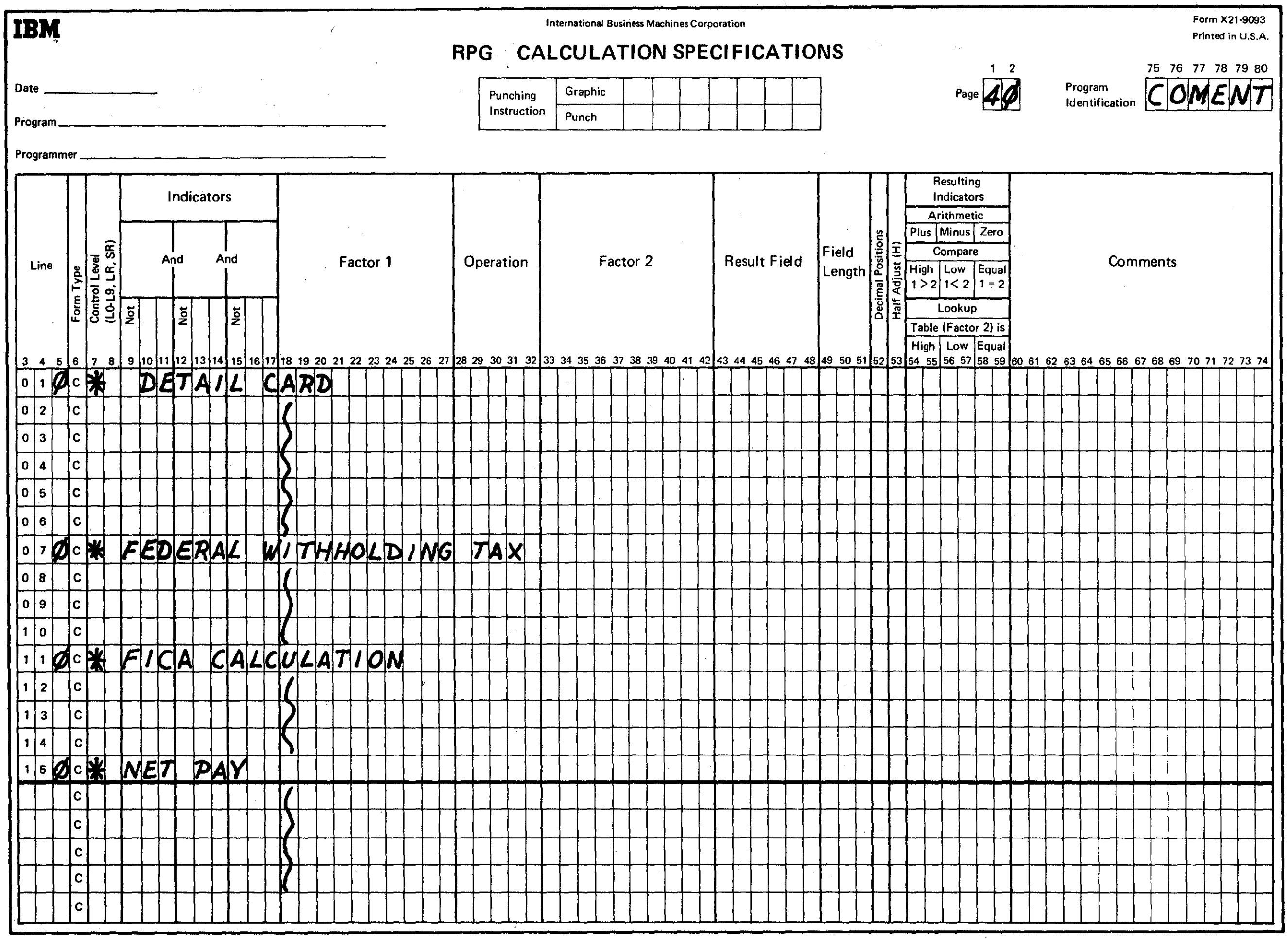
Control Card Specifications

Refer to the specific System Reference Library manual for actual entries.

File Description Specifications

```
  01 H
  02ØF*                          PAYØ5Ø
  03ØF*
  04ØF* PAYROLL DEDUCTION REGISTER
  05ØF*
  06ØF*   THIS PROGRAM TAKES THE OUTPUT OF PAYØ5Ø AND LISTS IT
  07ØF*     AS THE DEDUCTION REGISTER. GROSS PAY SHOULD BALANCE
  Ø8ØF*     WITH PAYØ3Ø.  DEDUCTIONS SHOULD CROSSFOOT AND AN
  Ø9ØF*     ERROR MESSAGE IS PRINTED IF THEY DO NOT.
  1ØØF*
  11ØF*
  12ØFDISKIN  IP  F     1ØØ            DISK
  13ØFPRINTER O   F     12Ø     OF     PRINTER
  14ØF
  15ØF
     F
     F
```

Figure 123. Comment Cards on a Specification Form



IBM

International Business Machines Corporation

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

RPG CALCULATION SPECIFICATIONS

Date

Program

Programmer

Punching Instruction	Graphic									
	Punch									

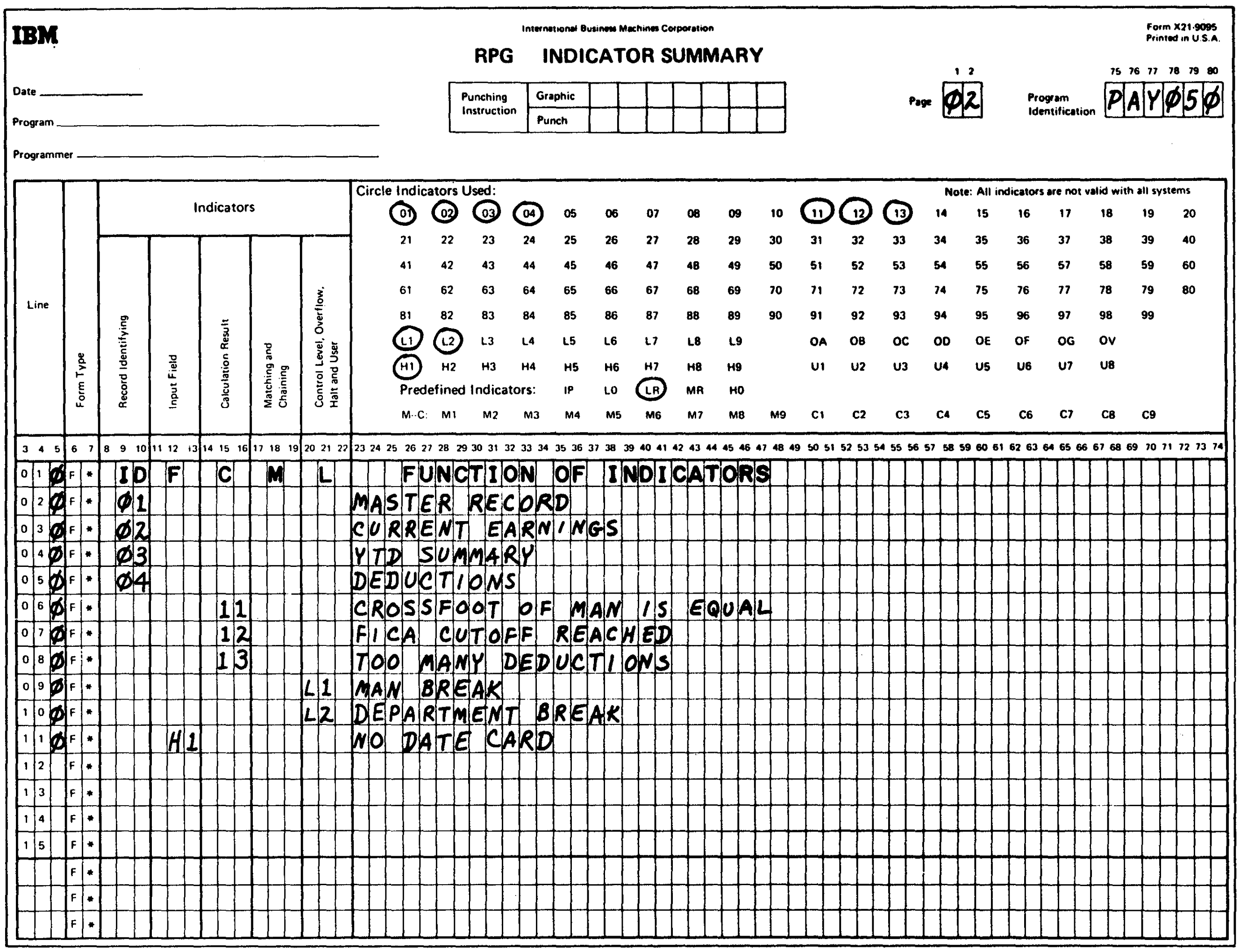
Page (1 2): 4Ø

Program Identification (75 76 77 78 79 80): COMENT

Line (3–5) | Form Type (6) | Control Level (L0-L9, LR, SR) (7–8) | Indicators: Not, And, Not, And, Not (9–17) | 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, Minus, Zero; Compare High 1>2, Low 1<2, Equal 1=2; Lookup Table (Factor 2) is High, Low, Equal (54–59) | Comments (60–74)

```
01ØC*  DETAIL CARD
02 C
03 C
04 C
05 C
06 C
07ØC* FEDERAL WITHHOLDING TAX
08 C
09 C
10 C
11ØC* FICA CALCULATION
12 C
13 C
14 C
15ØC* NET PAY
   C
   C
   C
   C
   C
```

Figure 124. Program Documentation



IBM

International Business Machines Corporation

RPG INDICATOR SUMMARY

Form X21-9095
Printed in U.S.A.

Date

Program

Programmer

Punching Instruction	Graphic								
	Punch								

Page (1 2): Ø2

Program Identification (75 76 77 78 79 80): PAYØ5Ø

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 HO
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	
3 4 5	6 7	8 9 10	11 12 13	14 15 16	17 18 19	20 21 22	23–74
01Ø	F *	ID	F	C	M	L	FUNCTION OF INDICATORS
02Ø	F *	Ø1					MASTER RECORD
03Ø	F *	Ø2					CURRENT EARNINGS
04Ø	F *	Ø3					YTD SUMMARY
05Ø	F *	Ø4					DEDUCTIONS
06Ø	F *			11			CROSSFOOT OF MAN IS EQUAL
07Ø	F *			12			FICA CUTOFF REACHED
08Ø	F *			13			TOO MANY DEDUCTIONS
09Ø	F *					L1	MAN BREAK
10Ø	F *					L2	DEPARTMENT BREAK
11Ø	F *		H1				NO DATE CARD
12	F *						
13	F *						
14	F *						
15	F *						
	F *						
	F *						
	F *						

Figure 125. Using the Indicator Summary Form

Indicators are assigned either by the user or by the RPG program itself. The indicators are represented on the specification form by a two-character code. The point at which the indicator is defined on the specification form determines if the indicator is a record identifying, resulting, or field indicator.

The user can change the status of most indicator by using the operation codes SETON and SETOF. The program sets and resets some types of indicators at certain times during the program cycle.

All indicator types are discussed in this section. The main use of each indicator is given along with the time at which the indicator is set and reset. The set and reset times are related to the general logic cycle shown in Figure 126.

GENERAL INDICATORS (01-99)

General indicators are used to identify record types, to indicate the status of input field, to show the results of calculations and comparisons, to condition calculation and output based on the status of these indicators, and to determine the search in a table lookup operation.

Any of these 99 indicators can be assigned by the user to be associated with the occurrence of a specific condition. When the condition occurs, the indicator is set on. It remains on until a criterion for that indicator (the condition occurring) has been tested again and is not satisfied.

Record Identifying Indicators (01-99, L1-L9, LR, H1-H9)

The identifying characteristics of records are defined in positions 21 to 41 on the Input Specifications form. The record identifying indicator is coded on positions 19-20. The indicator is set on when a record from the file (named in positions 7-14) is used as input and the record identifying conditions are met.

The record identifying indicator can also be used elsewhere in the program. It can be set on or off with the SETON or SETOF operation code. All record identifying indicators are set off by the RPG program before the next record is read (at GET A RECORD in Figure 126). The record identifying indicator is set on immediately after the record is read.

When a chaining file is present in the input, a record is retrieved from the chained file. If a record is found in the chained file the record identifying indicator is set on along with the indicator for the chaining file. When the chained record can not be found, the record identifying indicator for the chained file is not set on.

Input Field Indicators (01-99, H1-H9)

The input field indicators are specified in positions 65-70 of the Input Specifications form. Their purpose is to test the status of a field when it is read into the system. Depending upon the status of the field (plus, minus, zero, or blank), the associated indicator is set on.

The input field indicators are set on if their respective conditions occur when a record is read into the system. Each input field indicator is normally related to only one record type. Therefore, the indicators are not reset until the related record type is read again. One or more field indicators can be on at one time.

The status of indicators used for testing for zero or blank can be changed by specifying Blank After for the field on the Output-Format Specifications form. After this field is put out, it is reset and the indicator assigned to zero or blank is set on. This allows two input field indicators to be on for the same field. The zero or blank indicator can be on with either the plus or minus indicator.

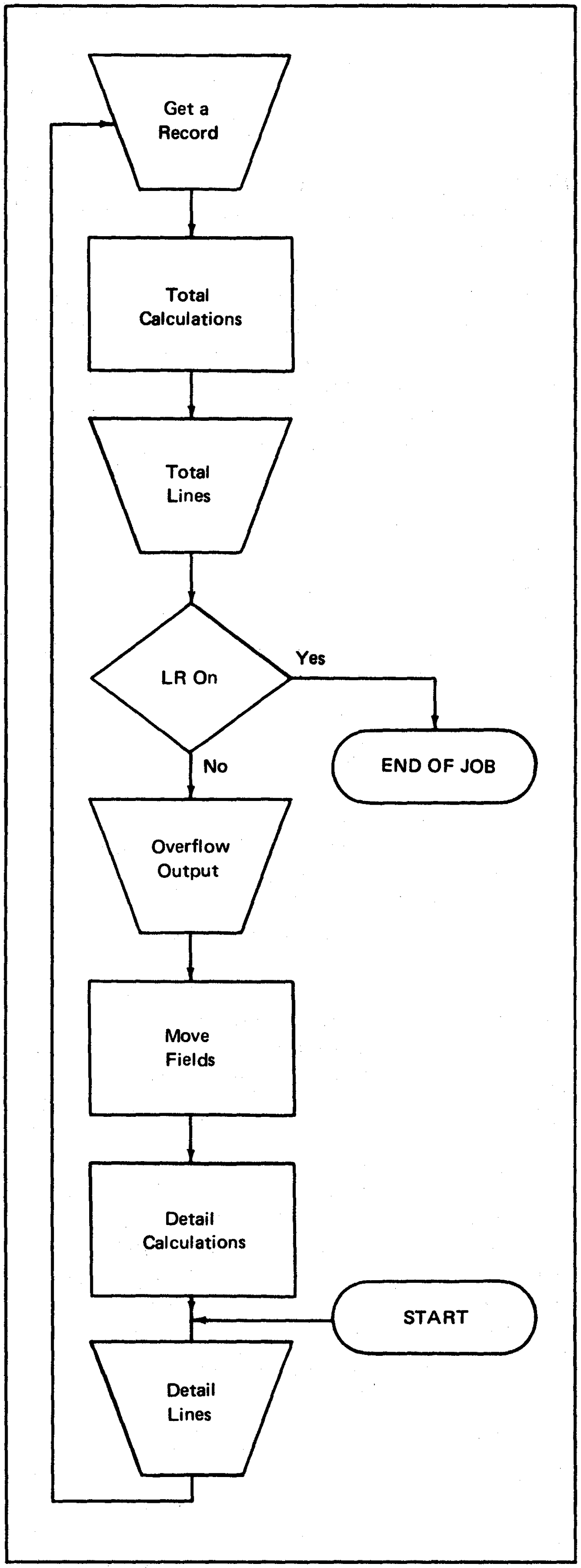


Figure 126. General Logic Cycle

Resulting Indicators (01-99, L1-L9, LR, H0-H9, OA-OG, OV)

Resulting indicators are entered in positions 54-59 of the Calculation Specifications form. Resulting indicators are set on if the specified condition exists when the calculation is executed. The indicator remains on until the calculation is executed again and the specified condition does not exist. The indicator assigned to zero or blank is set on when Blank After is specified for the field on the Output-Format Specifications form. Resulting indicators can be used to indicate the following conditions:

- To determine whether the results of an arithmetic operation is plus, minus, or zero. When half adjustment is specified, the indicator refers to the results after half adjustment.
- To test the results of a compare operation to determine if Factor 1 is greater than (>), less than (<), or equal to (=) Factor 2.
- To define the type of zone tested on a TESTZ operation. The results of the test may be a C zone or & (plus), D zone or – (minus), or any other zone (zero).
- To define the type of LOKUP operation to be performed. The indicator is used to specify how the look up of a table is to be accomplished and to indicate if that element was found. An indicator entry in the high position causes the element next higher than the search argument to be retrieved. An indicator entry in the low position causes the element next lower than the search argument to be retrieved. The equal position indicator entry retrieves the element equal to the search argument. When the element defined by the indicators is found, the indicator is set on. When equal is specified with either high or low, the equal takes precedence.
- To define the indicators affected by a SETON or SETOF operation code. From one to three indicators may be specified. Figure 127 is a summary of conditions which cause resulting indicators to be set on.

Condition		Positions 54-55	Positions 56-57	Positions 58-59	Remarks
Arithmetic Operation	If Result Field is	Plus	Minus	Zero	The same indicator may be used in any two positions, 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 positions, but not in all three. If the same indicator is used in any two positions, the indicator will be set on if either condition exists.
Test Zone Operation	If left-most zone of the Result Field is	Note 1	Note 2	Note 3	
Lookup		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.
SETON/ SETOF Operation	May be one, two, or three indicators	Specified indicator	Specified indicator	Specified indicator	

Note 1: The indicator in positions 54-55 is set on when the character to be tested is one of the following: A-I, $\overset{+}{\underline{O}}$, or &.
Note 2: The indicator in positions 56-67 is set on when the character to be tested is one of the following: J-R, O, or —.
Note 3: If the zone is made up of any characters other than those mentioned in Notes 1 and 2, the indicator in positions 58-59 is on.

Figure 127. Conditions that Set Resulting Indicators ON

Conditioning Indicator (Any Indicator Except the 1P Indicator)

The conditioning indicators are used on the Output-Format Specifications form in positions 23-31 and on the Calculation Specifications form in positions 9-17. All of the RPG indicators (except 1P) can be used as conditioning indicators. The status (on or off) of an indicator is not changed by its use as a conditioning indicator. Up to three indicators can be specified on one line. The indicators specified are in a AND condition which means that all conditions must be met before the line is executed.

Halt Indicator (H0-H9)

There are ten halt indicators designated as H0-H9 which cause the program to halt before reading the next input record. H0-H9 can be set by the programmer. Halt indicators H1-H9 can be defined as field indicators, record identifying indicators, or resulting indicators. Halt indicator can be used as conditioning indicators on calculation or output specifications. If a halt indicator is not set off by a SETOF operation code, the program halts before the next record is read.

The H0 indicator is used by the RPG program itself. The program halts before the next input record is read unless the H0 indicator is set off. Appendix D contains a list of conditions that cause the H0 indicator to be set on.

If carriage overflow occurs at detail time, the remainder of the detail lines are printed. Then the overflow indicator is set 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 set off.

If a carriage overflow occurs at total time, the remainder of the total lines are printed. Then the overflow indicator is set 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 set off.

Control Level Indicators (L1-L9)

Control level indicators are used to recognize control breaks and to produce totals at desired levels. There are nine control level indicators, L1-L9.

The control level indicators are assigned to a field on the Input Specifications form. When a control level field is encountered by the object program, the data in the control field is compared with data in the same control field from the previous record. If these differ, a control break occurs and the control level indicator assigned to the field is set on. The control level indicator remains on for the total time and the first detail time, but is set off before the next record is read.

Control level indicators are ranked in order of importance. L1 designates the lowest control level and L9 the highest. When a certain control level indicator is set on by a control break, all control level indicators of lower rank are also set on. These lower level indicators can only be used if they have been defined on either the Input or Calculation Specifications form.

The control level indicators can also be used as general indicators or with SETON and SETOF. When used in this manner, the lower level indicators are not affected.

When all input files are at an end-of-file condition, the LR (last record) indicator comes on. The LR indicator forces all lower rank indicators (L1-L9) to be set on. If the LR indicator is used as a general indicator or with SETON, the L1-L9 indicators are not set on. If the LR indicator is set on during total calculation time, the program terminates after total output time.

The L0 indicator enables calculations at total time even though no control break has occurred. The L0 indicator is always on and can never be set off.

The L0 indicator makes it possible to perform calculation at total time based on the status of the record identifying indicator for the next record. The next record is read before total time and the identifying indicator is set on. The fields of this record are not moved to the input till after total time. During total time the last record's fields are available along with the next record's identifying indicator.

External Indicators (U1-U8)

External indicators U1-U8 are set prior to executing the program by a job control statement. The PARM= entry in the EXEC statement sets the external indicators. The PARM= entry is coded as follows:

```
PARM= 'U=nnnnnnnn'
```

where n is replaced by 0 for each indicator (U1 through U8, left to right) that is to be off, and by 1 for each indicator that is to be on. Therefore, to set U7 and U8 on, the PARM= entry would be:

```
PARM= 'U=00000011'
```

To set U1, U4, U5, and U8 on, the PARM= entry would be:

```
PARM= 'U=10011001'
```

The external indicators U1-U8 can be used in OS RPG for a special purpose in the field record relation positions (63-64) of the Input Specifications form. If an external indicator is specified in positions 63-64, the field is not processed when the indicator is off. If the indicator is on, the field is processed like a normal field. External indicators can also be used as conditioning indicators to modify calculation operations or output operations.

SUMMARY

Figure 128 is a summary of indicators; part 1 shows the valid entries for each type of indicator. The top part of the figure shows where the indicator is defined and the lower part shows where the indicator can be used.

The external indicator is set outside the RPG program. The internal indicators are set by the RPG program when certain conditions are satisfied.

The SETON and SETOF operation code may be used to change the status of all indicators except U1-U8, L0, MR, 1P, and 00.

Part 2 of Figure 128 shows the conditions that cause each type of indicator to be set on and set off.

Where Defined	01- 99	1P	H0	H1- H9	L0	L1- L9	LR	MR	OA, OG, OV	U1- U8
Control Level Input specifications 59-60						X				
External Indicator										X
Field Indicator Input specifications	X		X	X						
Internal Indicator		X	X		X		X	X		
Overflow Indicator File description specifications 33-34									X	
Record Identifier Input Specifications 19-20	X									
Resulting Indicator Calculation specifications 54-59	X	X	X	X	X	X	X	X	X Note 1	
Where Used	01- 99	1P	H0	H1- H9	L0	L1- L9	LR	MR	OA, OG, OV	U1- U8
Conditioning Indicator Calculation specifications 9-17	X		X	X	X	X	X	X	X	X
Control Level Calculation specifications 7-8					X	X	X			
Field Record Relation Input specifications 63-64 (Note 2)	X									X
Output Indicator Output specifications 23-31	X	X Note 3	X	X	X	X	X	X	X	X

Note 1: The overflow indicator must be defined on the File Description Specification form first.

Note 2: Only a record identifying indicator from a main or OR record can be used to condition a control or match field. L1-L9 cannot be used to condition a control or match field.

Note 3: Only allowed on heading and detail lines.

Figure 128. Indicator Summary (Part 1 of 2)

	Set On	Set Off
Control Level	When the value in a control field changes. All indicators of the lower levels are also turned on	At end of following detail cycle
External U1-U8	By job control statement	End of job
Field Indicator	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
Internal Indicators		
1P	At beginning of processing before any input records are read	Before the first record is read
H0	By error condition or by programmer	Job is canceled
L0	Always on	Never off
LR	After processing the last record of the last file	At the beginning of processing
MR	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
overflow	When print or space pass channel 12	After the following heading and detail lines are completed
Record Identifying	When specified record has been read and before total calculations are executed	Before the next record is read during the next processing cycle
Resulting	When the calculation is performed and the condition for which the indicator represents is met	The next time a calculation is performed for which the program specifies the indicator as a resulting indicator and the specified condition is not satisfied

Figure 128. Indicator Summary (Part 2 of 2)

Using Tables and Exit Routines in the Object Module

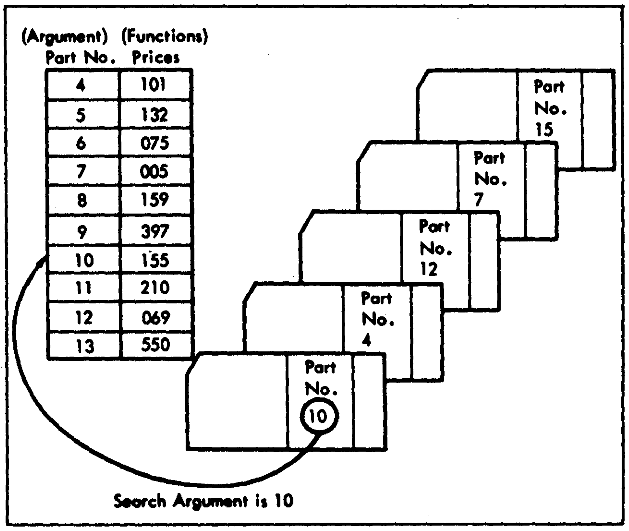
This section of the publication contains information on:

1. How to create and use tables.

2. How to transfer control from the RPG program to a subroutine coded by the user, and how to return to the RPG program.

USING TABLES IN THE OBJECT MODULE

RPG enables the programmer to use tables in the object module. A table is nothing more than a systematic arrangement of data which is used by the object program in much the same manner that a shipping clerk would use a rate table for obtaining freight rates. The clerk might scan the table for the desired city and then select the corresponding rate.



(Argument) Part No.	(Functions) Prices
4	101
5	132
6	075
7	005
8	159
9	397
10	155
11	210
12	069
13	550

Figure 129. Using a Table

53708

A table may consist of two parts: an argument and a function. In Figure 129 the table consists of part numbers (arguments) and prices (functions). 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 155 represents $1.55, in this example.) The number used to search the table is called the search argument. The card file in Figure 129 contains part numbers that have been placed in the table in a predefined sequence. 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. Tables are loaded into storage by the RPG object module before any files are processed.

All entries in a table will be one of these:

1. Arguments

2. Functions

3. Alternating arguments and functions

4. Alternating functions and arguments

Figure 130 shows these four possibilities.

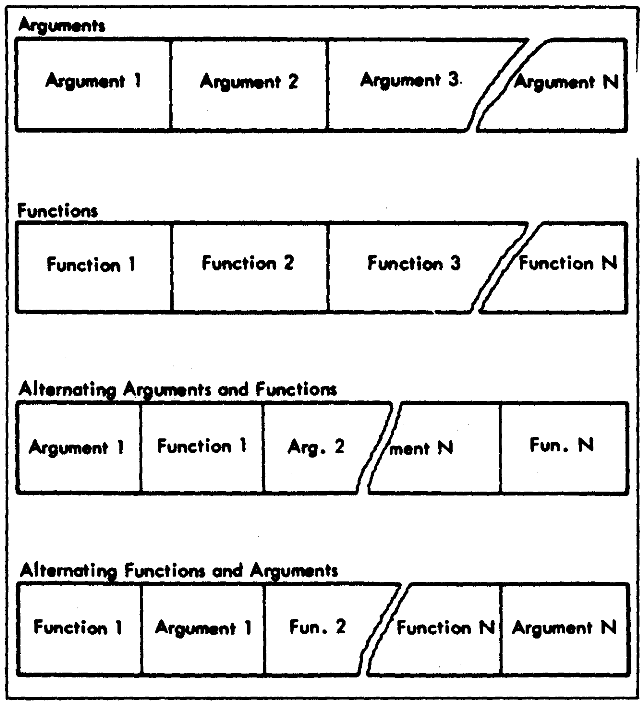


Figure 130. Four Types of Tables

53709

Rules for Forming Tables

1. Each unit of table data is called a table entry. That is, each argument is a table entry, and each function is a table entry.

2. The collection of all argument entries is assigned a name. The collection of all function entries is assigned a name.

   These table names must be unique, and must contain the letters TABnnn (nnn may be any alphameric entry). In Figure 134 the alternating table file called RATETABL is split into the collection of arguments (TABNUM) and the functions (TABRAT). RATETABL is the name of the file containing these two tables.

3. All tables may be loaded from the same device in the order they are specified on the Extension Specifications form. The tables will be loaded into storage before the object module is processed, and each line entry on the Extension Specifications form must have:

   a. A filename (positions 11-18). Multiple extension specifications for any table file are allowable.

   b. Entries in positions 27-45 if the table is only arguments or functions.

   c. Entries in positions 27-45 and 46-57 if the table consists of alternating arguments and functions.

Rules for Creating Records Containing Table Data

1. Each record must begin with the first table entry of that record in position 1.

2. All records must have the same number of table entries, except the last. In Figure 131, the first card in the table file has seven table entries. All subsequent card records must have seven table entries. For example, the second card could not contain six; the third could not contain eight.

3. All entries must be adjacent in every record. In Figure 132, the first entry begins in position 1 and the second entry begins in position 4. No blanks can be contained between the table entries.

4. All entries belonging to a table must have the same length. In Figure 132, 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. Each record must always begin with an argument, or each record must always begin with a function as shown in Figure 132.

6. When alternating tables are used, the table entries in each record must not be split. Function 3, for example, must be in the same record as argument 3. It is not permissible for a function to appear in a different record from its corresponding argument.

7. If a table consists of all arguments or all functions, an argument or a function must not be split. Assume that argument one, argument two, argument three, and argument four are contained in the first record. No part of argument four could overflow into the second record. Figure 131 illustrates the correct way to specify records containing arguments or functions.

8. The tables may be ascending, descending, or in no sequence. If the tables are not in sequence, only an equal search can be performed.

9. The records of a table must be on a sequentially organized file.

10. The table file to be loaded must contain the exact number of table entries as specified on the Extension Specifications form.

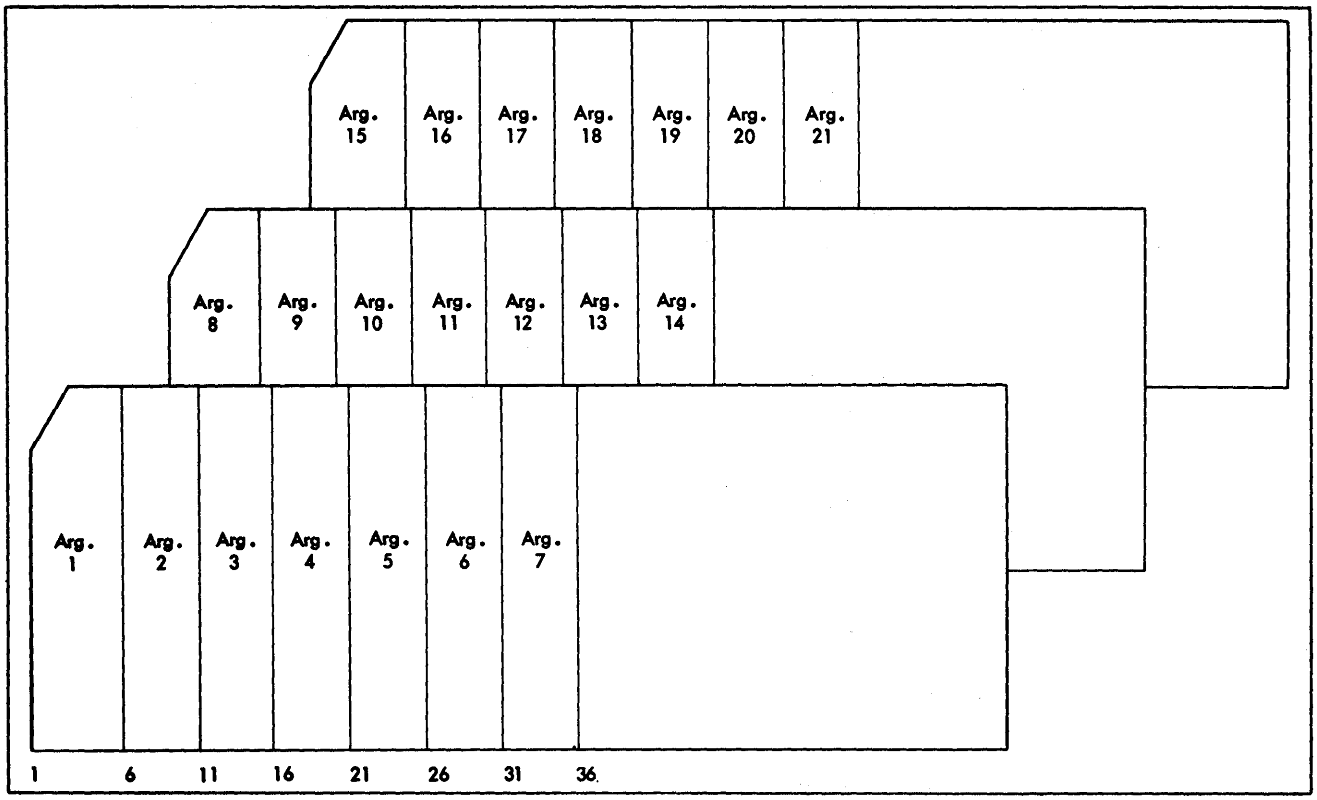


Figure 131. Table File Containing All Arguments

53710

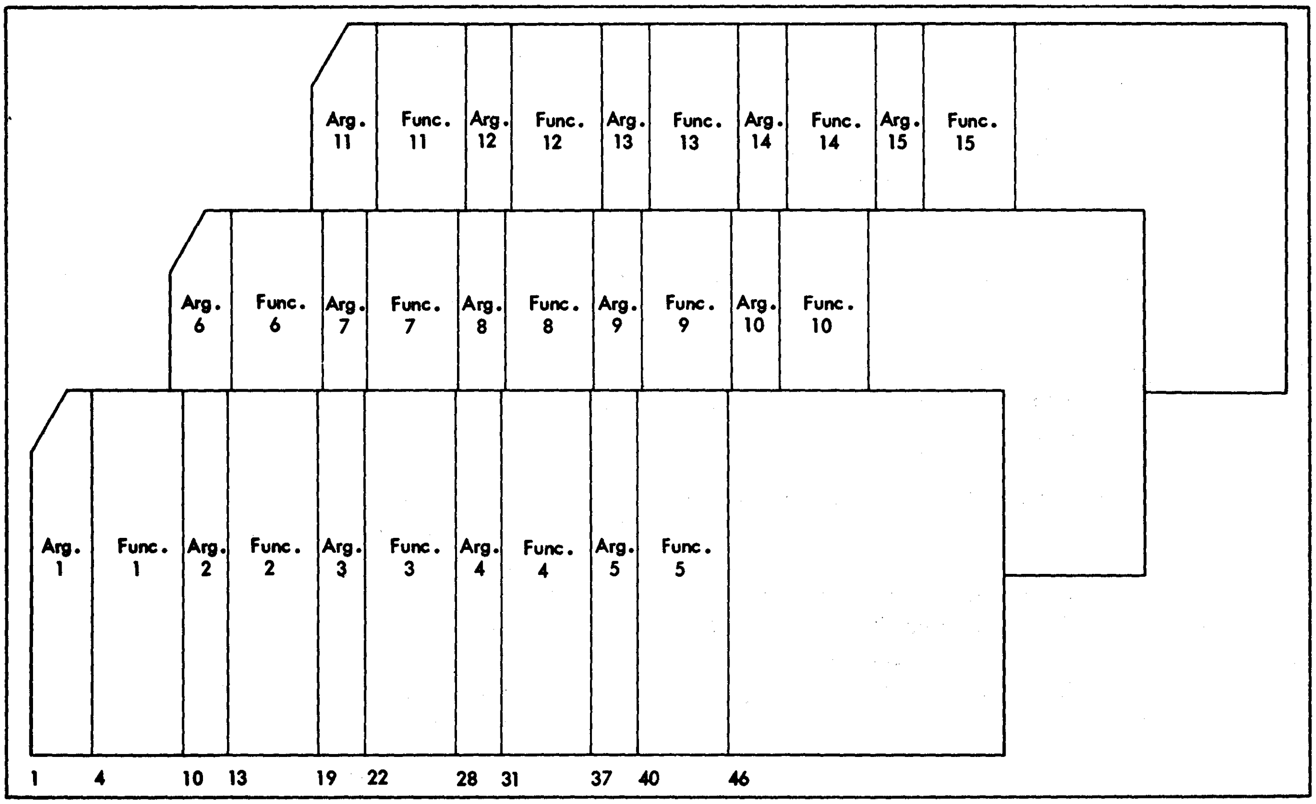


Figure 132. Table File Containing Arguments and Functions

53711

Retrieving Updated Tables

After a table has been updated, the table may be written out for later use.

On the File Description Specifications form, the programmer enters the specifications for the output file that will contain the updated table. The file must be defined as an output file.

On the Extension Specifications form, the programmer enters the name of this output file in To Filename. This entry is made on the same specification line that defines the table file.

The updated table file will be put onto the output file after the program has reached the end-of-job condition (LR) condition.

The name of the file need not be entered on the Output-Format Specifications form. If the updated table is to be put out on a printer, no automatic skip to a new page will be initiated by the RPG program.

The table written out has the same format as specified for the input table in positions 33-45 or 52-57 on the Extension Specifications form.

Methods of Processing Tables

The operation code LOKUP entered on the Calculation Specifications form causes a table lookup operation to be performed.

Factor 1 contains the search argument. The search argument may be a literal or a field name.

Note: The length of the data in the argument table (table argument) must be equal to the length of the search argument. The length includes the decimal positions.

Factor 2 contains the name of the table which contains the arguments.

The Result Field contains the name of the table from which an associated function is to be located. The Result Field may be left blank if the user wants to determine if an argument is present in the table, but does not require the corresponding function.

Resulting Indicators (positions 54-59) must always have an entry when the table lookup operation is performed. The presence of indicators in this specification indicates the type of lookup to be performed. The indicators are set on whenever the condition is satisfied.

The program may search for the table argument next higher than the search argument, it may search for the table argument next lower than the search argument, or it may search for the table argument equal to the search argument. An entry must be made in positions 54-59. Combinations of high-equal or low-equal searches may be specified by placing indicators in the appropriate two of the three fields (positions 54-59).

Performance of LOKUP

The lookup operation is performed in this way:

1. The object module takes the field name or literal in Factor 1 and searches the table indicated by Factor 2. The kind of lookup is determined by the entries in Resulting Indicators.

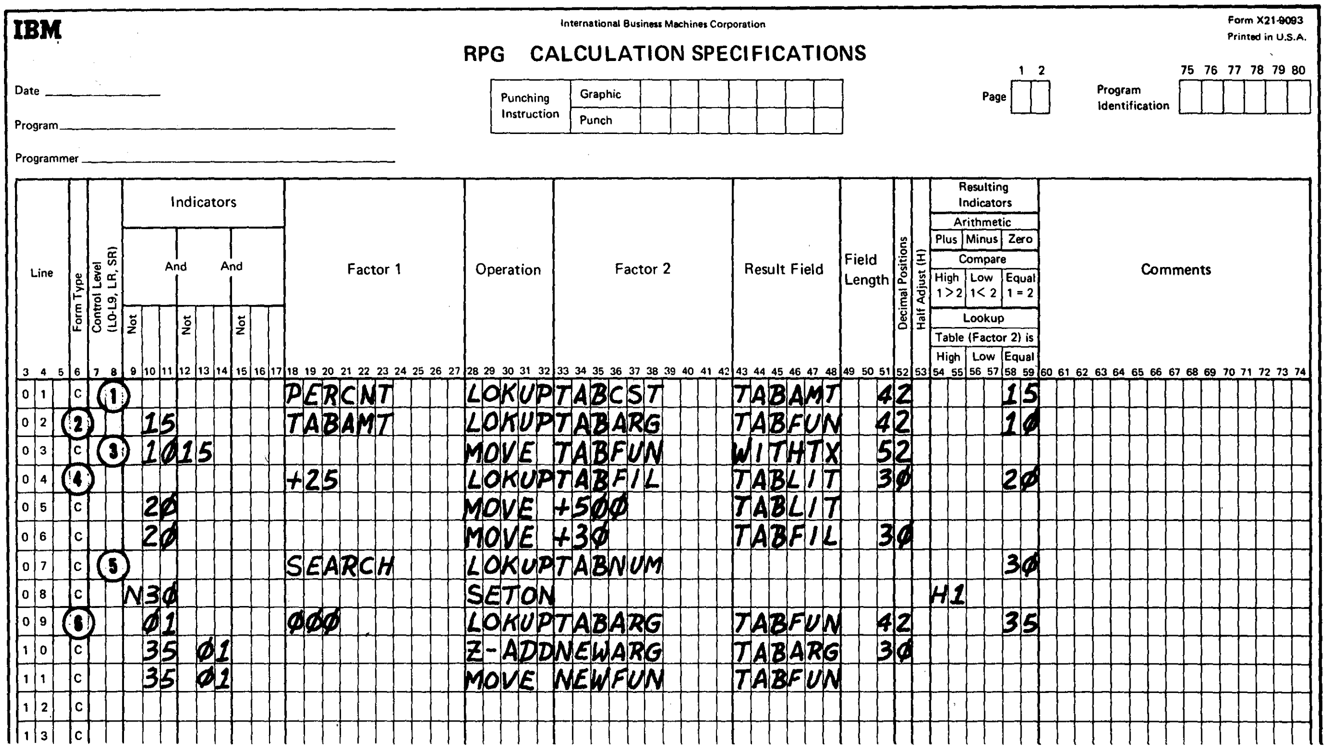
2. After the proper entry from the argument table has been found, the corresponding function from the function table indicated by the entry in Result Field is located and placed in the special hold area of the function table. If the proper table argument is not found, the indicators in positions 54-59 are not set on.

Using LOKUP Data Obtained

Other operations may be performed using the data just found by the table lookup operation. This data is stored in the special hold area within the function table and can be retrieved by merely using the name of the function table in either Factor 1 or Factor 2 of an operation.

Figure 133 illustrates several ways the data found by the operation may be used. The numbers on the figure correspond to this discussion.

1. Factor 1 contains the name of a field. The field PERCNT contains 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 (positions 58-59).



IBM International Business Machines Corporation

RPG CALCULATION SPECIFICATIONS

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

Date ______
Program ______
Programmer ______

Punching Instruction: Graphic / Punch

Page (1 2)

Program Identification (75 76 77 78 79 80)

Line (3-5)	Form Type (6)	Control Level (L0-L9, LR, SR) (7-8)	Indicators (9-11)	And (12-14)	And (15-17)	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 / Compare High 1>2 / Lookup Table (Factor 2) is High (54-55)	Minus / Low 1<2 / Low (56-57)	Zero / Equal 1=2 / Equal (58-59)	Comments (60-74)
01	C	1				PERCNT	LOKUP	TABCST	TABAMT	4	2				15	
02	2		15			TABAMT	LOKUP	TABARG	TABFUN	4	2				1Ø	
03	C	3	1Ø	15			MOVE	TABFUN	WITHTX	5	2					
04	4					+25	LOKUP	TABFIL	TABLIT	3	Ø				2Ø	
05	C		2Ø				MOVE	+5ØØ	TABLIT							
06	C		2Ø				MOVE	+3Ø	TABFIL	3	Ø					
07	C	5				SEARCH	LOKUP	TABNUM							3Ø	
08	C		N3Ø				SETON						H1			
09	6		Ø1			ØØØ	LOKUP	TABARG	TABFUN	4	2				35	
10	C		35	Ø1			Z-ADD	NEWARG	TABARG	3	Ø					
11	C		35	Ø1			MOVE	NEWFUN	TABFUN							
12	C															
13	C															

Figure 133. Using Table Files

2. The data found in TABAMT from the previous operation is used as the search argument in this example. (TABAMT is the name of the table; however, this name now refers to the special hold area of the table TABAMT.) The program searches for the value in the table TABARG that is equal to the search argument (positions 58-59).

3. In this example, the data obtained from the function table TABFUN from the previous lookup operation is moved to a field called WITHTX. It will be used for additional calculations.

4. In this example, the data found in the functions and argument tables is updated. The literal +25 is the search argument. The table TABFIL is searched for +25 (indicated by the entry in positions 58-59). A new entry for the corresponding function of +25 is entered in TABLIT and the special hold area. The new function is +500; the new argument is +30.

5. In example 5, a lookup with only one argument table sets on indicator 30 if SEARCH is equal to an argument in TABNUM. If 30 is not on (N30), H1 is set on by the SETON instruction.

6. This example illustrates the facility for adding to the table. In this example, the LOKUP operation is conditioned on indicator 01. (Indicator 01 is set on when the input file contains records with additional table information. Each record contains the two fields, NEWARG and NEWFUN.) To determine the first vacant argument, a field of zeros is used as the search argument. Zeros are used because the argument field is numeric. Blanks would be used if the argument field had been alphabetic. In loading the table, the vacant table entries must be filled with zeros or blanks for numeric or alphameric fields, respectively.

   If there is an equal compare, indicator 35 is set on. Since 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 corresponding new function (NEWFUN) is inserted in the TABFUN field.

Note: Whenever a field TABnnn appears in positions 32-37 in the output-format specification and Blank After is specified (B in position 39), the table value and hold area are updated to blanks or zero for alpha or numeric, respectively.

Example of Using Tables

Figures 134 and 135 illustrate an input data file, the way a table might appear, and the entries necessary on the RPG specification forms.

In this example, a card input file contains the number of hours worked by each employee (positions 42-44) and the employee's number (positions 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 salary rate has been found, it is multiplied by the hours worked by the employee. The result of this operation is the amount earned for each employee.

In this example, the table consists of alternating arguments and functions. The way the table data might appear is shown in Figure 134. The name of the file that contains the arguments and functions is RATETABL. The collection of arguments is called TABNUM (table number), and the collection of functions is called TABRAT (table rate).

Entries on the specification forms follow.

File Description Specifications Form

The two files are defined on the File Description Specifications form. The file containing the input card records is called TIMECARD. It is an input file (I in position 15); it is the primary file (P in position 16); and when the file is depleted, processing is terminated (E in position 17). The records in the file are in ascending order (A in position 18); they are fixed-length records (F in position 19). Each record has a block length of 80 (80 in positions 22-23), and each record is 80 characters long (80 in positions 26-27). This file is read in on the IBM 2501 Card Reader so the device :ode is READ01.

The table file is defined on the line below the card input file. The name of the file (RATETABL) is entered in Filename positions 7-14). It is an input file (I in position 15), and he records in the file are fixed-length (F in position 19). 'he file has a block length of 80, and each record is 80 haracters long. The E in position 39 indicates that addiional information about the file is coded on the Extenion Specifications form. This file is read in on the IBM 442 Card Read-Punch and, therefore, is assigned the evice code of READ42.

xtension Specifications Form

n the Extension Specifications form, the table file is furier defined. The name of the file is entered in From Fileime (positions 11-18). The collection of arguments (TABNUM) is entered in the first Table Name (positions 27-32). There are 8 arguments per record (positions 34-35), and there are 1500 arguments in the table (positions 36-39). Each table entry is five positions long (5 in position 42), and there are no decimal positions (0 in position 44). The table is in ascending order (A in position 45).

The collection of functions is described in positions 46-57. The name of the functions (TABRAT) is entered in the second Table Name (positions 46-51). Each entry in the table is four positions long (4 in position 54), and there are three decimal positions specified (3 in position 56).

Input Specifications Form

The input file (TIMECARD) is described on the Input Specifications form. The filename is entered in positions 7-14. The file is assigned a sequence of AA (positions 15-16), and record identifying indicator 01 is set on whenever an input record is present for processing. No record identification codes are specified because every record will be processed in the same way.

Lines 020 and 030 are used to describe the locations of the two input fields used by the program. The employee number is located in positions 1-5 of the input record, as specified by the entries in Field Location (positions 47 and 51), and the employee number is given the field name EMPNUM. The number of hours worked by the employee is found in positions 42-44 of the input record, as specified by the entries in Field Location. The name HRSWKD is assigned to the number of hours worked by each employee.

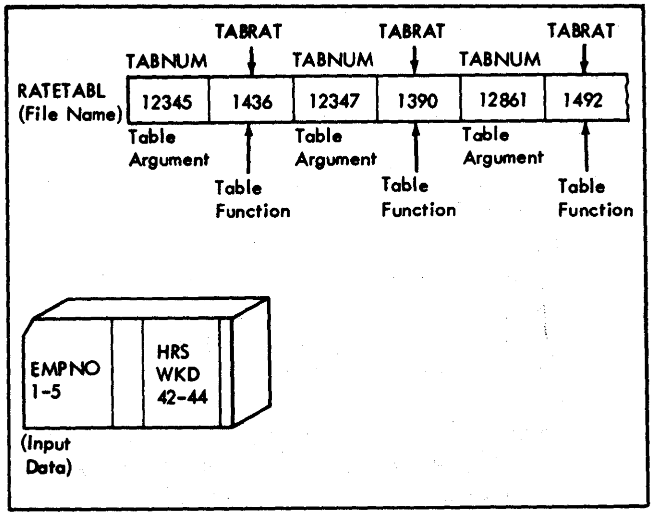


Figure 134. Using Alternating Arguments and Functions

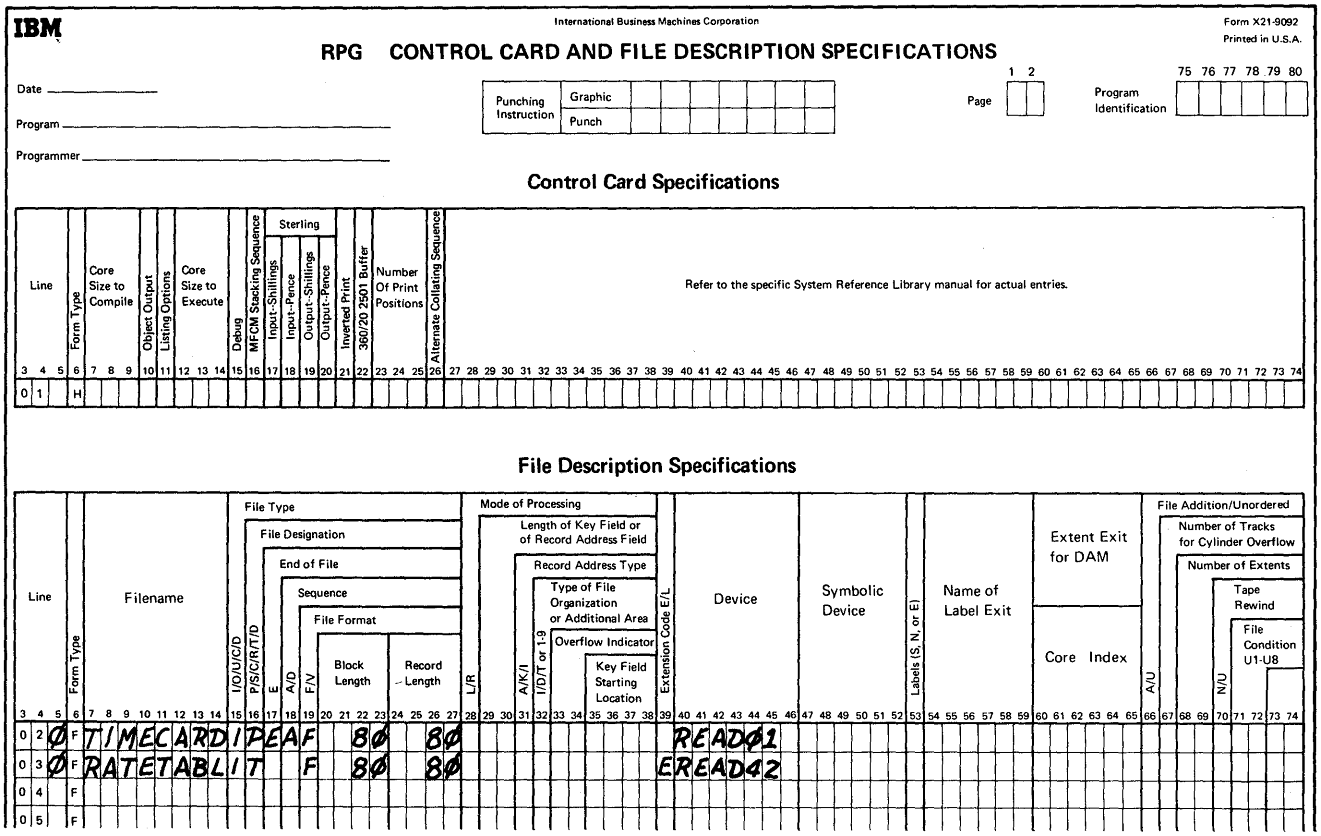
Calculation Specifications Form

Three calculation specifications are shown. On line 010, EMPNUM (employee number) is used as Factor 1. The employee number is the search argument. The operation code LOKUP which is coded in Operation (positions 28-32) causes the lookup operation to be performed. Factor 2 contains the name of the collection of arguments (TABNUM) which is searched. The Result Field contains the name of the collection of functions (TABRAT). Thus, this operation causes the employee number (EMPNUM) to be used as the search argument for the data contained in TABNUM. The result field is four positions long with three decimal positions. The 03 entered in positions 58-59 shows that indicator 03 will be set on when the search argument finds an entry in the argument table that is equal to the search argument.

The specifications on line 020 are performed when Indicator 03 in on. The rate for the employee (TABRAT) which has just been located is multiplied by the number of hours worked (HRSWKD), and the result is stored in EARNS which is five positions long and has two decimal positions. The answer is half adjusted.

If the search argument does not find an equal entry in the argument table (indicator 03 is not on), the specifications on line 030 are performed. Positions 9-11 contain the specification N03.

The literal +000.00 is then moved to the field EARNS, specifying that the employee does not have an entry in the table.



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

Program Identification 75 76 77 78 79 80

Control Card Specifications

Line (3-5)	Form Type (6)	Core Size to Compile (7-9)	Object Output (10)	Listing Options (11)	Core Size to Execute (12-14)	Debug (15)	MFCM Stacking Sequence (16)	Sterling: Input-Shillings (17)	Input-Pence (18)	Output-Shillings (19)	Output-Pence (20)	Inverted Print (21)	360/20 2501 Buffer (22)	Number Of Print Positions (23-25)	Alternate Collating Sequence (26)	27-74
01	H															Refer to the specific System Reference Library manual for actual entries.

File Description Specifications

Line (3-5)	Form Type (6)	Filename (7-14)	File Type (15) I/O/U/C/D	File Designation (16) P/S/C/R/T/D	End of File (17) E	Sequence (18) A/D	File Format (19) F/V	Block Length (20-23)	Record Length (24-27)	L/R (28)	Mode of Processing: Length of Key Field or of Record Address Field (29-30)	Record Address Type (31) A/K/I	Type of File Organization or Additional Area (32) I/D/T or 1-9	Overflow Indicator (33-34)	Key Field Starting Location (35-38)	Extension Code E/L (39)	Device (40-46)	Symbolic Device (47-52)	Labels (S, N, or E) (53)	Name of Label Exit (54-59)	Extent Exit for DAM / Core Index (60-65)	A/U (66)	File Addition/Unordered (67)	Number of Tracks for Cylinder Overflow (68-69)	Number of Extents (70) N/U	Tape Rewind (71-72)	File Condition U1-U8 (73-74)
02Ø	F	TIMECARD	I	P	E	A	F	8Ø	8Ø								READØ1										
03Ø	F	RATETABL	I	T			F	8Ø	8Ø							E	READ42										
04	F																										
05	F																										

Figure 135. Coding Forms for Table Example

EXIT TO A USER'S SUBROUTINE

By use of the EXIT operation code on the Calculation Specifications form, RPG provides the facility to transfer control from the RPG object program to some subroutine that has been coded independently. A subroutine might be a standard routine such as a state withholding tax.

The subroutine, written in the 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 the subroutine.
2. Execute the subroutine.
3. Reference fields and indicators defined in the RPG program (RLABL usage).
4. Reference fields defined in the user's subroutine (ULABL usage).
5. Return to the main program after the subroutine has been performed.

How to Code EXIT

On the Calculation Specifications form, the EXIT operation can be a conditional operation. When entries are placed in positions 7-8, 9-11, 12-14, or 15-17, the EXIT occurs when the designated conditions are satisfied. If no indicators are used, the EXIT occurs every time the detail calculations are performed. Positions 28-31 must contain the operation code EXIT and Factor 2 must contain the label of the user's subroutine. The subroutine name may be from 1-6 alphameric characters with the first character being alphabetic. Factor 1 is not used.

Position of EXIT in the Calculation Specifications

The following results will be obtained depending on the location of the EXIT code on the Calculation Specifications form.

Calculation Entry	*When the Exit Will Occur*
First detail	At the end of the data routine (after the data is extracted from the input record).
Last detail	Immediately before heading records are written.
First total	At the end of the input routine (after the record type has been determined and the control field break has been tested).
Last total	Immediately before the total records are written.

General Rules for Using EXIT

RPG provides the facility for the subroutine to test indicators and use tables and fields that have been defined in the RPG program. RPG also provides the facility for the RPG program to use fields that have been defined in the subroutine. These two facilities are provided by using the two operation codes RLABL and ULABL. RLABL and ULABL operations can be specified anywhere within the calculation specifications.

RLABL

If the user has defined a field or table in the RPG program and it is to be used in the subroutine to which the EXIT will occur, he must code:

1. RLABL in Operation.

2. The name of the field or table in Result Field.

3. The length of the field in Field Length.

4. The decimal indication in Decimal Positions.

The user may need to reference, in the subroutine, indicators that are used in the RPG portion of this program. To do this, the user must code:

1. RLABL in Operation.

2. INnn in Result Field. The nn represents the specific indicator that the user wants to test in the subroutine. Therefore, if MR was to be tested in the subroutine, he would code INMR in Result Field.

ULABL

If the user has defined a field in the subroutine and this field is to be used in this RPG program, he must code:

1. ULABL in Operation.

2. The name of the field in Result Field.

3. The length of the field in Field Length.

4. The decimal indication in Decimal Positions.

Any identifier declared ULABL in the RPG program must be declared ENTRY in the BAL subroutine which contains the field. When executing the subroutine, the user may have to use an indicator in the subroutine and later reference that indicator in the RPG program. This can only be accomplished by first defining the indicator in the RPG program and then defining it in a RLABL operation.

Use of Registers

The way in which registers are used by the programmer is strictly defined. These rules must be followed:

1. The using register that contains the entry address of the called subroutine is register 15.

2. When control of the program passes from the RPG program to the subroutine, the address of the RPG instruction to which the subroutine must return is stored in register 14.

3. The RPG instruction to which the subroutine returns is the instruction that follows the EXIT operation.

4. If registers are used within the subroutine, the contents of the registers the programmer intends to use must be stored before the subroutine is executed. When control passes to the subroutine, register 13 contains the address of a save area in the RPG program.

5. Before the subroutine transfers back to the RPG program, the registers must be restored to their original contents.

Using Indicators, Fields, and Tables in the EXIT Routine

Indicators

If, in the exit subroutine, the user sets on, sets off, or tests indicators, he must observe the following rules:

1. To set on an indicator, set the data located at INnn to hexadecimal FO.

2. To set off an indicator, set the data located in INnn to hexadecimal 00. (Indicator LO and 00 must never be set off.)

3. To test indicators:

   a. If on, the data in INnn will be hexadecimal FO.

   b. If off, the data in INnn will be hexadecimal 00.

Fields

If numeric data from the RPG object program is used in the subroutine, it will be in the packed decimal format. If numeric data from the subroutine is supplied to the RPG object program, it must be in the packed decimal format.

Tables

The subroutine may refer to a table which is defined in the RPG program. As each table is created in the program, a table linkage field is created for it. This field is a control field used by table operations. The format of this field is illustrated in Figure 137. Significant subfields are described below.

1. This subfield (1 byte) contains switches used by RPG.

2. This subfield (1 byte) contains the length minus 1 of each table entry. For numeric fields this subfield contains a number one less than the unpacked length although the actual table entry is in packed format.

3. This subfield (2 bytes) contains the number of entries in the table.

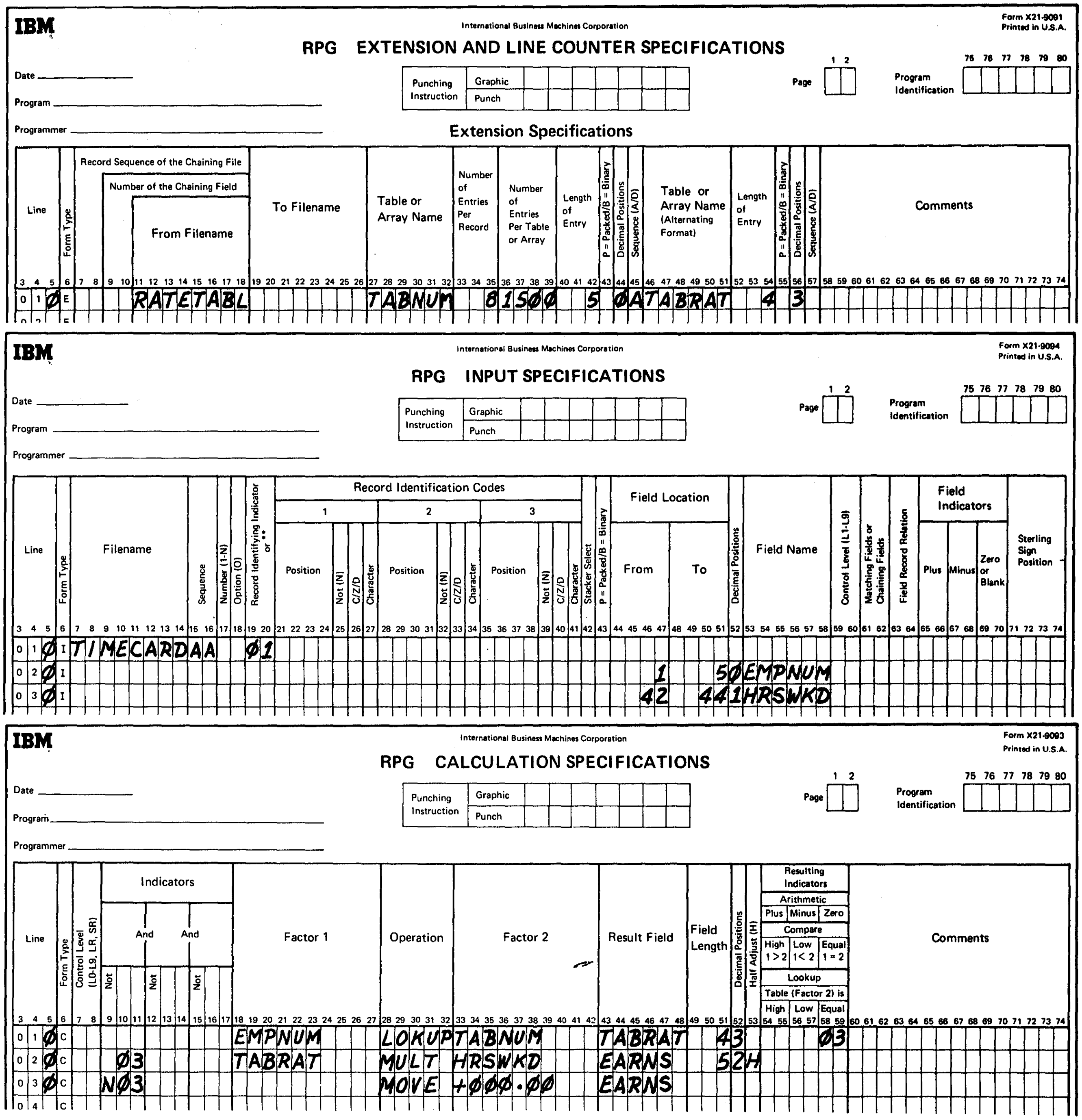
4. This subfield (4 bytes) contains the address of the beginning of the table.

5. This subfield (4 bytes) contains the address of the byte following the end of the table.

6. This subfield (4 bytes) is used by the RPG object program as a pointer to the selected table entry. For example, as the result of a LOKUP operation, this subfield contains the address of the corresponding function retrieved from the table when an equal is found in the argument table.

7. This subfield is used by the object modules as a transient work area. For example, as the result of a LOKUP operation, this subfield contains the data from the function retrieved from the table when an equal is found in the argument table. (This subfield is word aligned and its length is equal to the length of a table entry.) This area is also called the special hold area.

The subroutine can use the data retrieved from a preceding LOKUP operation by simply referring to TABnnn (assuming that TABnnn has been defined by an RLABL operation). The effective address of any reference to TABnnn is the first byte of subfield 7. To access the table itself, the address contained in subfield 4 of the table linkage field for TABnnn must be used.



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 — Program Identification

Extension Specifications

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P = Packed/B = Binary	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P = Packed/B = Binary	Decimal Positions	Sequence (A/D)	Comments
01Ø	E			RATETABL		TABNUM	8	15ØØ	5		Ø	A	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 — Program Identification

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes	Stacker Select	P = Packed/B = Binary	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Minus	Zero or Blank	Sterling Sign Position
01Ø	I	TIMECARD	AA			Ø1														
02Ø	I									1	5	Ø	EMPNUM							
03Ø	I									42	44	1	HRSWKD							

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

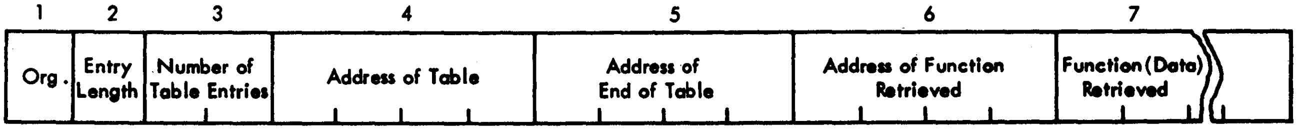
RPG CALCULATION SPECIFICATIONS

Date ______ Program ______ Programmer ______

Punching Instruction: Graphic / Punch — Page — Program Identification

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators	And	And	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	High 1>2 / Plus	Low 1<2 / Minus	Equal 1=2 / Zero	Comments
01Ø	C					EMPNUM	LOKUP	TABNUM	TABRAT	4	3				Ø3	
02Ø	C		Ø3			TABRAT	MULT	HRSWKD	EARNS	5	2	H				
03Ø	C		NØ3				MOVE	+ØØØ.ØØ	EARNS							
04	C															

Figure 136. Coding Forms for Table Example



1	2	3	4	5	6	7
Org.	Entry Length	Number of Table Entries	Address of Table	Address of End of Table	Address of Function Retrieved	Function (Data) Retrieved

Figure 137. Format of Table Linkage Field

53712

Example of EXIT to a Subroutine

Figure 138 shows the coding steps necessary to implement the exit subroutine.

1. An input file of the name INPUT sets on record identifying indicator 01 if an X is in position 80.

2. If the field AMOUNT is zero or blank, field indicator 02 is set on.

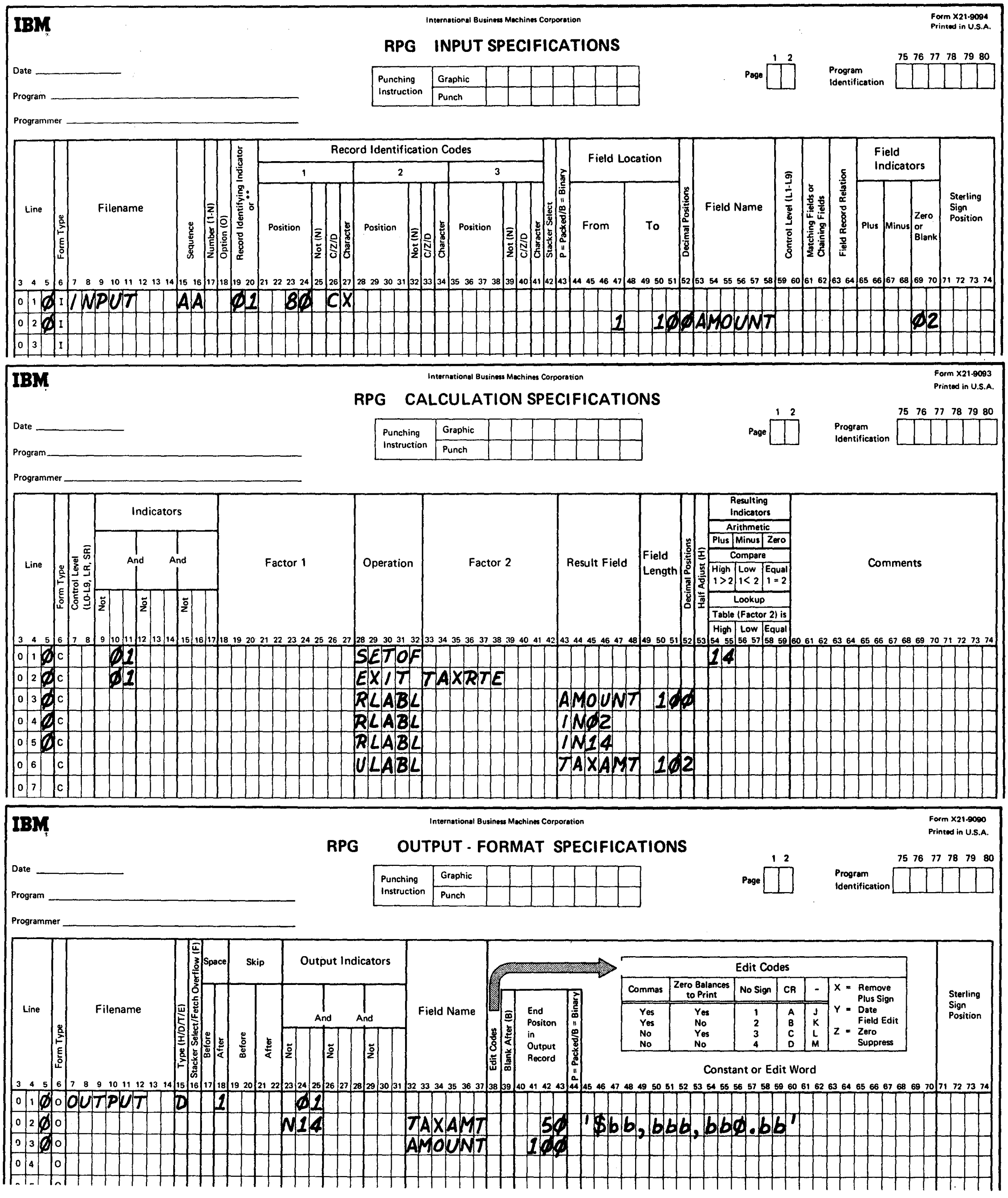
3. The operation SETOF defines (and sets off) indicator 14 for the RPG program so that it can be subsequently defined for use in the subroutine. (This is performed only if indicator 01 is on.)

4. Whenever indicator 01 is on, the calculation specifications entry EXIT causes the program to exit to the user's subroutine called TAXRTE.

5. Within the subroutine, the user wants three things: the AMOUNT field and the indicators 02 and 14. The RLABL operations enable the subroutine to reference the AMOUNT field, to test indicator 02, and to use indicator 14 in the subroutine.

6. In the subroutine, assume there is a field (TAXAMT) that the user wants to use on the output-format specifications. If the field TAXAMT in the subroutine is blank, the subroutine sets on indicator 14. TAXAMT is referenced in the output specifications, but it will not be printed if indicator 14 is on. If the user chooses, he can use it later in the calculation specifications. The entry ULABL enabels the field TAXAMT to be referenced by the RPG program.

7. On the Output-Format Specifications form, TAXAMT is treated as a field.



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 — Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator	Position (1)	Not (N)	C/Z/D	Character	Field Location From	To	Decimal Positions	Field Name	Field Indicators Plus	Minus	Zero or Blank
01Ø	I	INPUT	AA			Ø1	8Ø		C	X							
02Ø	I										1	1Ø	Ø	AMOUNT			Ø2
03	I																

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	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators High (Plus)	Low (Minus)	Equal (Zero)	Comments
01Ø	C		Ø1		SETOF						14			
02Ø	C		Ø1		EXIT	TAXRTE								
03Ø	C				RLABL		AMOUNT	1Ø	Ø					
04Ø	C				RLABL		INØ2							
05Ø	C				RLABL		IN14							
06	C				ULABL		TAXAMT	1Ø	2					
07	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 1 2 — Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Type (H/D/T/E)	Space After	Output Indicators	Field Name	End Position in Output Record	Constant or Edit Word
01Ø	O	OUTPUT	D	1	Ø1			
02Ø	O				N14	TAXAMT	5Ø	'$bb,bbb,bbØ.bb'
03Ø	O					AMOUNT	1ØØ	
04	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 138. Exit to a Subroutine

The data set organizations, indexed-sequential and direct, require the use of a direct access device. The sequential data set organization, when update is used, also requires a direct access device. A direct access device, other than that specified in RPG, may be specified through the OS job control language for the object program if the physical block size is compatible with the device specified. For example, if you have a 2314 defined on the File Description Specifications form, it is possible to use a 2311 if the track capacity, in bytes, for the 2311 is equal to or greater than the record length of the 2314.

A device, different from that specified in the RPG source program, may be assigned in the object program for sequential input/output if the device is functionally compatible. Spacing, skipping, and stacker select result in the output of a control character preceding each record when a printer or punch file is assigned to tape or disk in the object program.

Each file in an RPG program has a DCB (data control block) associated with it. Entries on the File Description Specifications form determine what type of DCB is created. The DCB describes the file to the OS system. In order to assign a different device (different than that specified in the RPG source program) in the object program, it may be necessary to alter the DCB. By using job control language you can modify the following fields of the DCB.

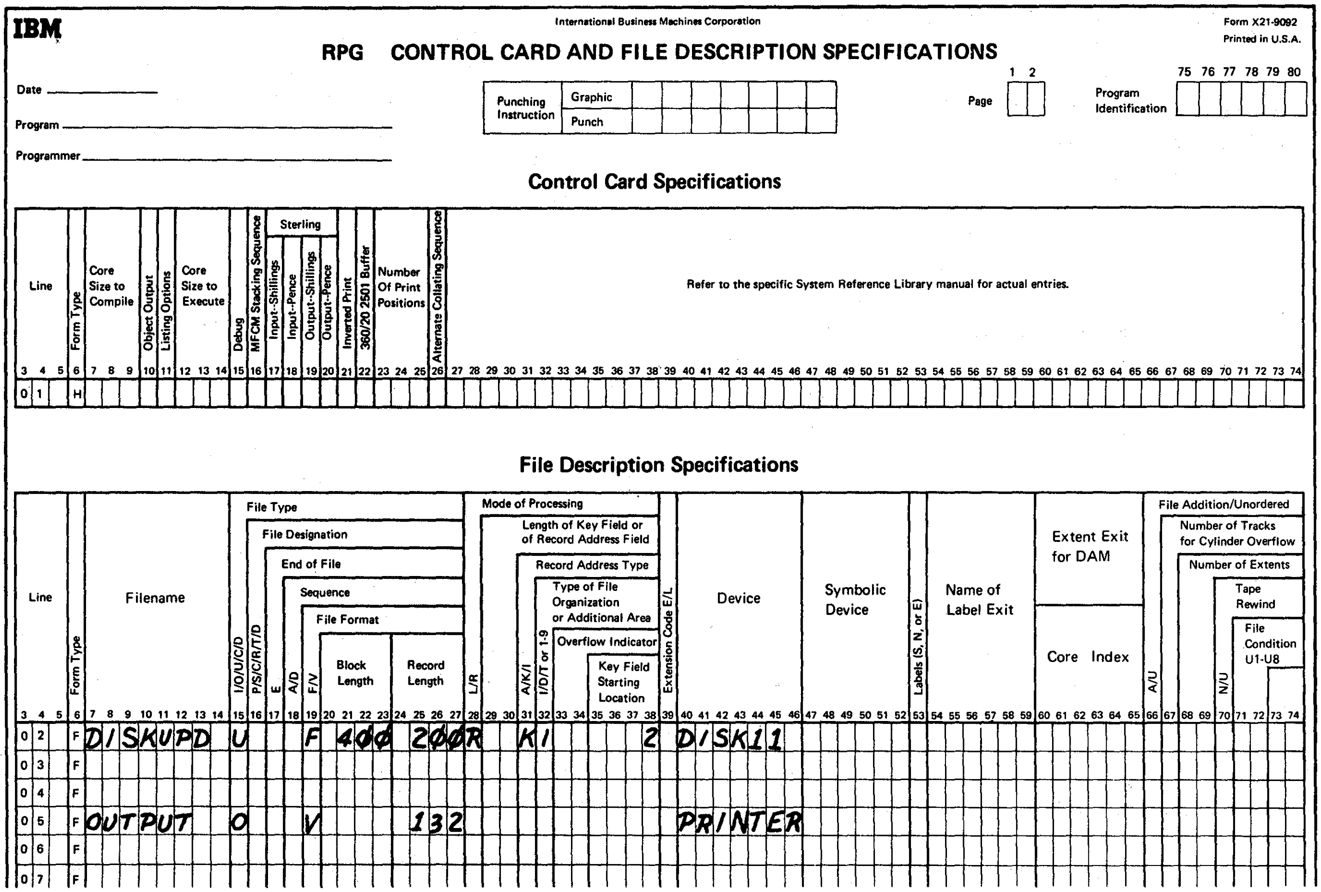
- BFALN (buffer alignment)
- BUFL (buffer length)
- BUFNO (buffer number)
- DEVD (device type)

Example 1

This example (Figure 139) specifies a disk update file on the IBM 2311 Disk Storage Drive. At object time the actual device specified through JCL could be an IBM 2314 Direct Access Storage Facility.

Example 2

This example (Figure 139) specifies a variable unblocked output file on a printer. At object time a tape, for example, could be specified as the actual output device. In this case, the maximum physical block size written would be 141 characters. The first four characters would contain the block length, the next four characters the record length, and the next character a one byte carriage control character.



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); Program Identification (75 76 77 78 79 80)

Control Card Specifications

Line (3-5)	Form Type (6)	Columns 7-74
0 1	H	Refer to the specific System Reference Library manual for actual entries.

File Description Specifications

Line	Form Type	Filename	File Type (I/O/U/C/D)	File Designation	End of File	Sequence	File Format (F/V)	Block Length	Record Length	Mode of Processing (L/R)	Length of Key Field or of Record Address Field	Record Address Type (A/K/I)	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 / Core Index	File Addition/Unordered (A/U)	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	File Condition U1-U8
0 2	F	DISKUPD	U				F	4ØØ	2ØØ	R		K	I		2		DISK11									
0 3	F																									
0 4	F																									
0 5	F	OUTPUT	O				V		132								PRINTER									
0 6	F																									
0 7	F																									

Figure 139. Changing Device Assignments

This section of the manual is for readers not familiar with disk storage operations. It contains a general description of data organization and retrieval and defines some of the terms you may encounter in related literature.

INTRODUCTION AND TERMINOLOGY

The distinction between two concepts is important: file organization and file processing.

File Organization is the method of arranging data records on a direct access storage device. A file is organized during the development stages of the application.

File Processing is the method of retrieving data records from the file.

To achieve the most efficient use of the System/360 components, carefully consider the relationship between how a file is organized and how to retrieve records from it. This is particularly important when designing data files for storage in a direct access storage device, such as the IBM 2311.

The method of organization best suited to a particular file of disk records depends upon many factors. These factors must be analyzed for each file in each particular application. Frequently, you can use more than one method of processing on the same file. For example, records within a file might be processed at random during an updating run and sequentially during a billing run.

Logical File vs Physical Unit

It is important to distinguish between a logical file and the physical unit used to store the file. A logical file is a group of related data records, such as payroll file.

A physical unit for storage of data records could be an IBM 2400 Series Magnetic Tape Unit, an IBM 2311 Disk Storage Drive, or an IBM 2501 Card Reader.

A logical file may occupy part of a disk storage drive, an entire disk storage drive, or more than one disk storage drive. The location of the logical file in disk storage is defined by its lowest and highest addresses. This area is the extent area. One logical file can occupy more than one extent area. The extent areas doe not have to be adjoining.

Data Files

Data files are recorded on such media as paper, cards, tape, or disk packs. Data files consist of a number of individual records that range from a few records up to thousands or millions of records.

Record

A record can be defined as a collection of information consisting of alphameric and/or nonalphameric characters related to a common identifier. The common identifier is known as a record's control field or key. Usually, one of the fields within a record identifies that record. For example, man number could be the key or identifier for a payroll record.

The size or length of records varies from file to file, and can be from eighteen characters to 4,000 characters.

A single record usually includes one or more logical data fields. A data field is a sequence of one or more characters which is treated as a processing unit of information. An individual data field is normally identified by its location within a record.

The logical structure of records and of fields within records is important in high-speed recording media such as magnetic tape and disk. This logical structure is strongly affected by whether a record is of fixed or variable length.

Fixed-Length Records

In fixed-length record files, all records are allocated the same number of character storage positions. Identical data fields are present in every record whether they are used or not. The control field (key) is usually the first field present in a record.

In many applications, fixed-length records would make inefficient use of file storage space. For example, a fixed record length of 850 positions would waste storage and processing time if the average record length is 230 positions and the minimum length is only 100 positions.

Situations such as this require the development of space-saving techniques based on varying the number of storage positions allocated to data records.

Variable-Length Records

Completely variable-length records are sometimes developed for more efficient use of storage. In this approach, the data portion of the record may be of any length, but the key (control field) size is constant. A record-length character count field in each record shows the length of a variable-length record.

Blocking Records

The length of individual data records varies with the type of data and the application that requires such data. The format of a data record is significant to the efficient use of the various storage media available on the System/360. One important element in the design of data records involves blocking and deblocking. Input/output units (storage media) are relatively inefficient when used to record short blocks of information. To increase the efficiency of input/output units, data records are assembled into blocks of records whose sizes are convenient and efficient for processing.

Each physical record on either tape or disk requires inter-record gaps. These gaps are blank areas used to distinguish beginning and ending points of a record. If records are blocked before loading onto a tape or disk, many of these gaps can be eliminated. Variable-length blocks are permitted in IBM System/360 Operating System RPG. The length of a variable-length block is indicated by a block-length character counter field present in each block. (See Variable-Length Records in this section.)

The operating system handles the blocking and deblocking of records so the user need only determine the most efficient blocking factor for his particular data file and equipment specifications. The system also creates and maintains the block-length and record-length count fields; no programming for these facilities is required by the RPG programmer.

In the operating system, only the input records for indexed-sequentially or sequentially organized files can be blocked.

FILE ORGANIZATION

Data records should be organized and stored to facilitate subsequent processing. The three types of file organization are sequential, indexed-sequential, and direct.

Sequential Organization

The logical sequence of records in this file depends upon a significant key (control field) appearing in the records. To establish a sequentially organized file, sort and then store the records in key sequence. This allows for records with successively higher or lower keys (control numbers) to have successively higher physical address numbers. Cards and tape files are always organized in this serial manner and usually are considered as one continuous string of records.

Indexed-Sequential Organization

In this type of file organization also, the sequence of records depends upon a key (control) field. The records are stored sequentially in the file. This variation of file organization differs from sequential organization in two ways:

1. The records may be retrieved from the file sequentially or in a random sequence.
2. Only records with transaction activity need be retrieved.

These differences occur because indexed-sequential organization uses index tables which indicate to the program the general location of the records. Thus, the program does not have to step through the file, record after record, to locate a specific record.

The index tables (prepared and maintained by the operating system) are analogous to the index card file in a library.

Sequential Processing

Sequential Files

In sequential processing of a sequentially organized file, every record in a file is examined, and each successive record in the physical file is processed in order. For example, in a card file, the card records are processed in the order that the cards are fed into the system. The fourteenth card in the file could not be processed until after the thirteenth card had been processed.

Indexed-Sequential Files

Sequential processing of an indexed-sequentially organized file has two variations.

1. An entire logical file is processed. For example, the physical unit consists of payroll records in cylinders 0 to 42 and inventory records in cylinders 43 to 99. Only the logical (payroll) file might be processed.

2. Only a segment of a file is processed. For example, a payroll file is to be updated with new pay increases. The payroll file is in sequence by department, and each week the 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 in a card file that contains the limits of the file to be processed. One such card record might indicate that the records for departments 26-41, are to be updated, another the records for departments 76-80, etc.

Random Processing

In random processing the sequence of processing has no relationship to the sequence in which the data is stored in the file. The data file could be organized in either a direct or an indexed-sequential order. This processing is sometimes called direct.

Indexed-Sequential Files

To find a random record in an indexed-sequential file, an index, or series of indexes is first scanned to localize the area of search by determining the track that contains the record. The index is a sequential list of the key records (of the data) with corresponding track addresses. The entire track is then scanned to find the individual record to be processed. This kind of processing is referred to as processing in a random sequence with record keys.

This type of processing is analogous to directing someone to a house location. "The Martin family lives on Harrison Street" (a track address) "and their house number is 4216" (a key).

For example, if the library user knows the name of the book or the author he can look in the index card file and find the location of the book in the book files. This might be an address (catalog number) of 426.25. He would then go to the book shelves, and (if it was his first time in the library) start at the first row of the book files and proceed through the rows until he found the shelf that contained 426.25. Usually, each row contains a sign to indicate the beginning and ending numbers of all books in that particular row. Thus, as he proceeded through the rows, he would compare 426.25 with the numbers posted on each row. Assume that one row was labeled 300.88-550.00. He would then search that row for the shelf that contained the book. The shelves (like the rows) might also contain number ranges to indicate their contents. In this case, he would scan the shelf numbers until he found something like 342.00-440.96. Then he would look at individual book numbers on that shelf until he found 426.25.

The RPG program uses index tables in much the same way to locate records organized in an indexed-sequential file.

Direct Organization

In direct file organization, the records are generally not stored in the sequence of their keys (control numbers). A randomizing formula converts the record key to a numerical address (physical address) of the storage device. The record is stored at the physical address developed by the randomizing formula. In effect, a file of records will be scattered throughout an entire disk file.

RPG does not provide for creating a file with direct organization. Creation of a file with direct organization must be accomplished by the programmer using the input/output macro facilities of the assembler programming system. During the processing of the object program, the user has the ability to exit to a subroutine to perform a randomizing routine upon the input data records. RPG cannot process duplicate records. (Duplicate records occur when two different control fields convert to the same physical address.)

FILE PROCESSING

For the three methods of file organization (sequential, indexed-sequential, and direct) there are three methods of file processing:

1. Sequential processing of sequentially organized files.

2. Sequential processing of indexed-sequentially organized files.

3. Random processing of indexed-sequentially organized files and directly organized files.

Direct Files

To find a random record in a direct file, compute the track address by the same randomizing formula used to load the file of records. You can make direct access to the record. Index tables are not required. This kind of processing is called processing in a random sequence and it can be done using keys or record identification (ID). The record identification indicates only the location of the record on the track. For example, the 2nd, 12th, 18th, etc. record on the track. The program makes no comparison of key (control field) data when a record number is provided.

This type of processing can also be compared to directing someone to a house location. "The Martin family lives on Harrison Street", (a track address) " and their house is the 5th house from the beginning of the street" (the 5th is the record identification).

If random processing is performed with key field only, the user supplies track address and record key field. Starting with this address the program searches the track for the record with the corresponding record key.

FILE PROCESSING IN RPG

The preceding information in this section provided a general introduction to disk storage concepts for the IBM System/360 Operating System. The material that follows is a summary of file processing methods for the RPG program.

RPG generated programs process the following input file organizations:

1. Sequential

2. Indexed-sequential

3. Direct

RPG will create only sequential and indexed-sequential output files.

Sequential Organization

The records on the file will be made available for processing in the same order in which they are located on the medium. The file might be contained in cards, on magnetic tape, or on a DASD. The entire file is processed, beginning with the first record and continuing until the file is depleted.

The end-of-file condition is determined as the last card of the file is read. In the case of DASD, the extent of the file is obtained from the operating system.

The records may be fixed or variable length and blocked or unblocked.

Indexed-Sequential Organization

An indexed-sequential file can only be on a direct access storage device (such as disk).

RPG rpocesses indexed-sequential files in three ways:

1. Sequentially by processing the entire file.

2. Sequentially by processing a segment of the file between the given limits.

3. Randomly by processing records on the file in a random order.

The records may be fixed-length and blocked or unblocked.

Processing the Entire File Sequentially

The generated program obtains the limits of the file from the operating system; the entire file is processed sequentially by record key in ascending or descending record key sequence.

Processing Part of the File Sequentially

If only a part of the entire file is to be processed, the generated program must be supplied with both the low and high keys that describe which part of the file will be processed.

An auxiliary file is used to supply these limits. This file is called a record address file (RA file). The RA file does not contain data to be processed. It contains the record keys (in this case the limits) of the data records which will be processed.

The object module obtains the limits to be processed from a record contained in the RA file and then processes all the data between those limits. The object module then reads another record in the RA file, and the procedure is continued until the RA file is depleted.

Processing the File Randomly

An indexed-sequential file may be processed randomly by supplying an RA file or a chaining file. Instead of supplying the limits (as in the case of sequential processing), the RA file or chaining file contains the record key of each record of the file to be processed.

Entry Combinations for Indexed-Sequential Processing

Figure 140 illustrates the combinations of entries necessary on the RPG coding forms for each type of indexed-sequential processing. Note that these entry combinations vary because of the location of the key, and whether the records are blocked or unblocked.

Direct Organization (Random)

A file with direct organization must reside on a direct access storage device. A direct file is always processed randomly. Records are retrieved from this file by using a relative track address and either a record key or record identification. The records must be fixed-length and unblocked.

A direct file is processed randomly by supplying a record address file or a chaining file.

The RPG program or an external subroutine must provide the necessary steps to convert the data fields contained in the RA file or chaining file to the relative track address and record key or record ID to be retrieved.

The format of the relative track address created by the subroutine must be in the form TTR. (Refer to *IBM System/360 Operating System, Supervisor and Data Management Services*, Form C28-6646.)

	File Type	Mode of Processing	Length of Key Field	Record Address Type	Type of Organization
Where Entered	FDS (15)	FDS (28)	FDS (29-30)	FDS (31)	FDS (32)
Allowable Entries	I/O/U/C	L/R/b	bb, 01-99	K/I	I/D/T
CREATE (LOAD)					
Unblocked					
Key in Data	O	blank	Required	K	I
Key not in Data	O	blank	Required	K	I
Blocked					
Key in Data	O	blank	Required	K	I
RETRIEVE					
Unblocked					
Key in Data	I	L/R/b	Required	K	I
Key not in Data	I	L/R/b	Required	K	I
Blocked					
Key in Data	I	L/R/b	Required	K	I
UPDATE					
Unblocked					
Key in Data	U	L/R/b	Required	K	I
Key not in Data	U	L/R/b	Required	K	I
Blocked					
Key in Data	U	L/R/b	Required	K	I
ADD with Input or Update					
Unblocked					
Key in Data	I/U	R	Required	K	I
Key not in Data	I/U	R	Required	K	I
Blocked					
Key in Data	I/U	R	Required	K	I
ADD					
Unblocked					
Key in Data	O	R	Required	K	I
Key not in Data	O	R	Required	K	I
Blocked					
Key in Data	O	R	Required	K	I

Figure 140. Entry Combinations for Indexed-Sequential Processing (Part 1 of 3)

	Key Field Starting Location	File Addition	Field Location (Key Location in Input)	Indexed Sequential Add	End Position in Output Record	Number of Tracks for Cylinder Overflow
Where Entered	FDS (35-38) blank or	FDS (66)	IS (44-51)	OS (16-18)	OS (40-43)	FDS (67)
Allowable Entries	0-4000 (key length)	A	Knnn-Knnn	ADD	Knnn	n
CREATE (LOAD)						
Unblocked						
Key in Data	Required	N/A	N/A	N/A	N/A	Optional
Key not in Data	N/A	N/A	N/A	N/A	Required	Optional
Blocked						
Key in Data	Required	N/A	N/A	N/A	N/A	Optional
RETRIEVE						
Unblocked						
Key in Data	Required	N/A	N/A	N/A	N/A	N/A
Key not in Data	N/A	N/A	Optional	N/A	N/A	N/A
Blocked						
Key in Data	Required	N/A	N/A	N/A	N/A	N/A
UPDATE						
Unblocked						
Key in Data	Required	N/A	N/A	N/A	N/A	N/A
Key not in Data	N/A	N/A	Optional	N/A	N/A	N/A
Blocked						
Key in Data	Required	N/A	N/A	N/A	N/A	N/A
ADD with Input or Update						
Unblocked						
Key in Data	Required	Required	N/A	Required	N/A	N/A
Key not in Data	N/A	Required	Optional	Required	Required	N/A
Blocked						
Key in Data	Required	Required	N/A	Required	N/A	N/A
ADD						
Unblocked						
Key in Data	Required	Required	N/A	Required	N/A	N/A
Key not in Data	N/A	Required	N/A	Required	Required	N/A
Blocked						
Key in Data	Required	Required	N/A	Required	N/A	N/A

Figure 140. Entry Combinations for Indexed-Sequential Processing (Part 2 of 3)

Legend

FDS	=	File description specifications
IS	=	Input specifications
OS	=	Output-format specifications
Knnn	=	The letter K followed by a 3-digit numeric entry
n	=	A 1-digit numeric entry
N/A	=	Not allowed

Figure 140. Entry Combinations for Indexed-Sequential Processing (Part 3 of 3)

This section provides and correlates information concerning compile time and object time requirements for using the indexed-sequential access method (ISAM). This includes a description of indexed-sequential file organization, examples of DD statement parameters, and references to corresponding RPG coding examples.

The RPG object program uses QISAM when creating, sequentially processing or updating an indexed-sequential file. BISAM is used when additions or other random processing are specified.

The reader should have some knowledge of job control and data management under the operating system, as job control statement descriptions and job control statement streams contained in this section should not be assumed to be complete enough to initiate and control the processing of RPG jobs under the operating system.

FILE ORGANIZATION

An indexed-sequential file is one in which records are arranged on the tracks of a direct access storage device in a sequence determined by keys. Each key is one or more contiguous control fields (for example, a part number) and is specified by the RPG source program. For blocked records, these fields are defined as part of each data record. Unblocked records may have keys defined as for blocked records or in a separate key area that is not part of each data record, but is associated with it. Figures 141, 142, and 143 illustrate these formats as they are recorded on a direct access storage device.

Figure 141 shows blocked records with the key always being in each logical data record. The block key is created by data management (QISAM or BISAM) and is equal to LOGICAL RCD N KEY.

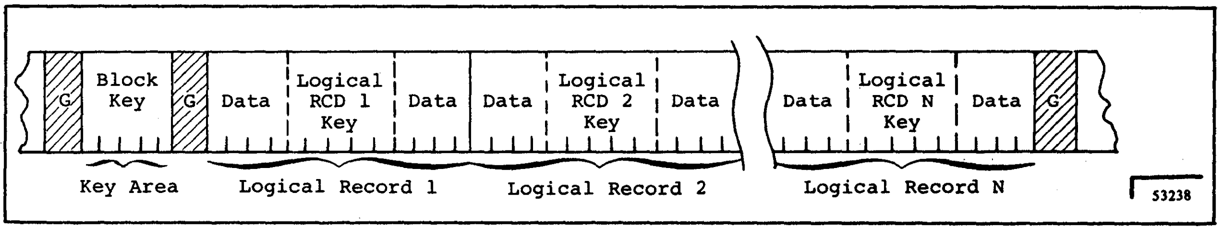


Figure 141. Blocked Records, Key in Data

Figure 142 shows unblocked records with the key in each logical data record. The key is defined within the logical data record and is also automatically placed in the key area by RPG.

In Figure 143 with unblocked records, the key is defined only within the key area by the RPG source program specifications. This is described under *File Description Specifications Form* (Key Field Starting Location, positions 35-38) *Input Specifications Form* (From and To Field Locations, positions 44-51), and *Output-Format Specifications Form* (End Position in Output Record, positions 40-43).

Specifications Form (From and To Field Locations, positions 44-51), and *Output-Format Specifications Form* (End Position in Output Record, positions 40-43).

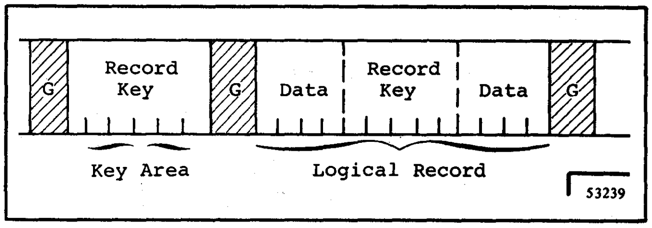


Figure 142. Unblocked Records, Key in Data

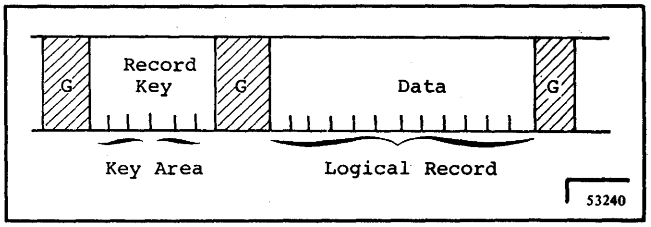


Figure 143. Unblocked Records, Key Not in Data

Records are written or read by the use of indexes. These indexes provide flexibility in that the programmer can:

- Write and later read or update logical records in a sequenial, ascending order (using QISAM) based on the collating sequence of the keys. This is done in a manner similar to that for sequential organization.

- Read or update statement logical records in a random manner (using BISAM). This method is somewhat slower per record than reading according to a collating sequence since a search for pointers in indexes is required for the retrieval of each record.

- Insert new logical records at any point within the file (using BISAM). Using the indexes, the system locates the proper position for the new record and makes all necessary adjustments so that the order of the records according to the keys, is maintained.

Indexes

There are two basic types of indexes, track indexes and cylinder indexes. There is one track index for each cylinder in the prime area (see *Indexed Sequential File Areas* in this section for a description of prime area). The track index is written on the first track of the cylinder that it indexes. An entry in the track index contains the identification of a specific track in the cylinder and the highest key on that track.

There is one cylinder index for each file in which prime area data takes up more than one cylinder. The cylinder index contains one entry for each cylinder in the prime area, each entry pointing to the track index in a particular cylinder.

If the file is being read randomly, the system locates the given record by its key after a search of a cylinder index and the track index within the indicated cylinder. If the file is being read sequentially starting with the first recore, no index search is necessary.

Indexed-Sequential File Areas

The programmer specifies the structure of an indexed file and space to be a''ocated for it in the DD statement for the file when the file is created. In some instances, more than one DD statement is required. (These DD statements are described in *Using the DD Statements – Single Volume Files* in this section). The space being allocated must be divided into one, two, or three areas, depending on the needs of the programmer. These areas are prime area, index area, and overflow area. The overflow area is optional.

Prime Area is the area in which data records are written when the file is created or reorganized. These records are in a sequence determined by the record keys. The track indexes also use a portion of the reserved prime area. To reserve prime area space so that new logical records may be inserted without forcing records into an overflow area (described below), dummy records may be written when the file is being created. The prime area may span multiple volumes and may consist of several noncontiguous areas.

Index Area contains the cylinder indexes and, if requested, master indexes (described later) for the file. This area exists for any file that has a prime area on more than one cylinder. Spcae for this area will be allocated separately from the prime area if specifically requested. The index area must be contained within one volume, but that volume need not be the same device type as the prime area volume. If not specifically requested, the index area will automatically be constructed in the independent overflow area, or, if none, in the prime area.

Overflow Area is the area in which space is allocated for records forced from their original (prime) tracks by the insertion of new records. The fact that some records are stored in these areas, physically out of sequence, does not change the ability of QISAM to read the file in a logical sequence. An overflow area need not be specified if records are either not going to be added to the file or sufficient space has been reserved by writing delete records in the prime area.

There are three ways in which space for an overflow area may be allocated:

1. Space may be requested for an independent overflow area using the dsname (OVFLOW) DD statement, either on the same volume or on a separate volume of the same device type as that of the prime area.

2. Tracks on each cylinder can be reserved to hold the overflow of that cylinder (cylinder overflow option).

3. If the prime area is not filled when the file is created, the remaining space will be designated as an independent overflow area even though it is not requested directly.

Additional information about indexed-sequential file structure is contained in the publication *IBM System/360 Operating System: Supervisor and Data Management Services,* Form C28-6646.

PROCESSING INDEXED-SEQUENTIAL FILES WITH QISAM AND BISAM

Figures 144 and 145 show the DD statement parameters that may be used with QISAM output files and QISAM and BISAM input and update files. Note that the SUBALLOC and SPLIT parameters are not applicable for any of these files.

A separate set of DD statements is required for each of the following operations on the same file:

1. Creating a QISAM file.
2. Opening it as a QISAM input/output or input file.
3. Opening it as a BISAM file.

The same filename is used each time in the DSNAME parameter.

Parameter	Applicable Subparameters
ddname	ddname used only for first DD statement of each file
DSNAME[1]	{dsname {&name [(INDEX) (PRIME) (OVFLOW)]
Device	Direct access required
UNIT	DEFER not permitted
SEP AFF	Restricted, see. "Job Control Procedures"
VOLUME	Volume sequence number sub-parameter not applicable
LABEL	SL
SPACE	CYL,[,,CONTIG]ABSTR
SUBALLOC	Not applicable
SPLIT	Not applicable
DISP	NEW[2] [,KEEP ,PASS ,DELETE]

DCB[3]	Required:
	DSORG=IS
	Optional:
	BUFNO=xxx
	OPTCD=[W][M][Y][I][R]

[1] If more than one DD statement is used, elements must be in this order.
[2] MOD not meaningful. CATLG allowed only if all areas are allocated with a single DD statement.
[3] The DCB parameter should be the same for each DD statement.

53241

Figure 144. DD Statement Parameters Applicable to QISAM Output Files

Processing with QISAM

Creating Indexed-Sequential Files

The structure of an indexed file and the space to be allocated is specified in the DD statement(s). The space can be subdivided and allocated in several different ways, the allocation specified on the DD statement(s) must be suffient for all the areas used.

Parameter	Applicable Subparameters
ddname	ddname used only for first DD statement of each file
DSNAME[1]	dsname
Device	Direct access required
UNIT[2]	Applicable subparameter
SEP AFF	Restricted, see Job Control Language publication
VOLUME	Applicable subparameters
LABEL	SL
SPACE	Not applicable
SUBALLOC	Not applicable
SPLIT	Not applicable
DISP	OLD[3] [,PASS ,KEEP ,DELETE]
DCB	Required: DSORG=IS Optional: BUFNO=xxx (not allowed for BISAM)

[1] Element subparameter must not be used.
[2] Not needed if file is cataloged.
[3] CATLG, UNCATLG not permitted.

53242

Figure 145. DD Statement Parameters Applicable to QISAM and BISAM Input, Output, and Update Files

QISAM DD Statement Requirements

The special parameter requirements for DD statements that define new indexed-sequential files are discussed below. The discussion is oriented to indexed-sequential files on one volume. Many of the parameters used for creating multi-volume files are not discussed here. For more detailed information about parameters for both single and multi-volume files, see the publication *IBM System/360 Operating System: Job Control Language,* Form C28-6539.

ddname (name field)

The name field of the first or only DD statement defining the indexed-sequential file can contain the symbolic identification ddname or procstep, ddname. Succeeding DD statements for the file must not be named.

DSNAME

This parameter must be specified and is coded as:

$$\text{DSNAME=}\begin{Bmatrix}\text{dsname}\\ \text{\&name}\end{Bmatrix}\quad[\,(\text{element})\,]$$

The first subparameter, dsname or &name, must be the same in all the DD statements defining one data set. The element subparameter, INDEX, PRIME, or OVFLOW, indicates the type of area defined by the DD statement. If more than one DD statement is used to define a file, the order in which the statements should be placed in the input stream is:

```
DD   DSNAME=dsname (INDEX)
DD   DSNAME=dsname (PRIME)
DD   DSNAME=dsname (OVFLOW)
```

Deviation from this order results in abnormal termination of the job. If the element subparameter is omitted, PRIME is assumed. Note that an indexed-sequential file cannot be specified by statements containing only index and overflow elements.

SPACE

This parameter specifies the space to be allocated for each of the separate areas on the device and must be included. Only cylinder (CYL) or absolute track (ABSTR) requests are permitted, and with ABSTR the designated tracks must encompass an integral number of cylinders. All the DD statements defining one indexed-sequential file must specify the same subparameter, either CYL or ABSTR. When all the DD statements specify CYL, all must also specify or omit CONTIG, depending on whether the space allocated is to be contiguous or noncontiguous. The directory or index quantity subparameter of the SPACE parameter is used to request the number of cylinders to be allocated for an index area embedded within the prime area.

SPLIT

This parameter should never be specified for a QISAM file, either for sharing a cylinder with QISAM files or for sharing it with a QISAM file and another type of file.

DISP

This parameter is written as it would be for any new file that cannot be cataloged. The CATLG subparameter must not be specified unless only one DD statement is used to allocate the file space. (See *Cataloging Files* in this section for additional information about cataloging indexed-sequential files.)

DCB

This parameter must be specified for each DD statement and is coded as:

```
DCB=(DSORG=IS
          [,BUFNO=integer]
          [,OPTCD=[L] [Y] [I] [R] [W]
          [M,NTM=integer] ]
```

DSORG=IS

The DSORG=IS subparamter is required and indicates that the organization of the file is indexed-sequential. The DCB subparameters of all the DD statements defining one file must not conflict. For example, if the OPTCD=Y subparameter appears in the first DD statement, the subsequent DD statements should also contain OPTCD-Y.

BUFNO=number of buffers

This subparameter is used to specify the number of buffers to be assigned to the file. The maximum number is 255; however, the maximum number allowed for an installation may differ and is established at system generation time.

OPTCD=options

This subparameter is used to tell the system that certain additional facilities are to be provided for this file. Any combination of the following options can be specified for the OPTCD subparameter. If more than one option is specified, the options are written as a character string (that is, without intervening blanks or commas). Note that if certain of these options are used, an additional subparameter must also be specified as indicated.

- OPTCD=L: This option requests that the control program delete marked records. Marked records will be deleted when space for new records is required.
- OPTCD=Y: This option requests that a cylinder overflow area be created. It specifies that a certain number of tracks on each cylinder are to be reserved to contain any overflow records from other tracks on that cylinder.

  The number of tracks reserved on each cylinder for overflow is specified on the File Description Specifications form.
- OPTCD=I: This option requests that an independent overflow area be reserved. It is used in conjunction with DSNAME=dsname (OVFLOW) parameter in the DD statement used to allocate the independent area.
- OPTCD=M: This option requests that a master index be created (see *Additional Facilities* in this section for a discussion of master indexes). Another DCB subparameter NTM=xx, must also be written. It specifies the maximum number of tracks to be contained in the cylinder index before a higher level index is created. The maximum value that can be specified is 99.
- OPTCD=W: This option requests the system to perform a write validity check. The operating system supplies OPTCD=W for the 2321 direct access device.
- OPTCD=R: This option requests reorganization criteria feedback as described in *Additional Facilities, Reorganizing Files* in this section.

  The following is an example of how the OPTCD subparameter can be used:

  DCB=(DSORG=IS,OPTCD=M,NTM=20)

  This example requests that a master index be created when the cylinder index exceeds 20 tracks.

Using the DD Statements -- Single Volume Files

The following examples all refer to files that can be contained on one volume. More details about DD statements, including information on multi-volume file allocation, can be found in the publication *IBM System/360 Operating System: Job Control Language,* Form C28-6539.

All three areas for an indexed-sequential file can be contained on a single volume if they are small enough. If this is the case and the programmer elects to allow the system to subdivide storage into the prime and index areas when the file is created he need only code the following DD statement:

```
//ddname DD   DSNAME=dsname (PRIME),          X
//            SPACE=(CYL,(no. of              X
//            cylinders)),UNIT=unit,          X
//            VOLUME=SER=serial,              X
//            DCB=DSORG=IS,...)
```

This DD statement will produce a prime area with the index area occupying the last cylinder(s) of the space in the prime area. If any tracks are left over on the last cylinder after the index area, they are used as the overflow area; if no tracks are left over, an overflow area does not exist.

If the programmer definitely wants an independent overflow area, he must provide the following:

```
//ddname  DD   DSNAME=dsname(PRIME),          X
//             SPACE=(CYL,(no. of             X
//             cylinders)).UNIT=unit,         X
//             VOLUME=SER=serial,             X
//             DCB=(DSORG=IS,. . .)
//        DD   DSNAME=dsname(OVFLOW),         X
//             SPACE=(CYL,(no. of             X
//             cylinders)),UNIT=unit,         X
//             VOLUME=SER=serial,             X
//             DCB=(DSORG=IS,. . .)
```

These DD statements will produce a prime area and a separate overflow area with the index area at the end of the overflow area. All three areas reside on the same volume.

Note: When more than one DD statement is used, only the first can be named. The others must not have a data definition name (ddname) but all must have the same data set name (dsname).

If the programmer desires more control in the placement of the index area, he can subdivide storage before the data set is created by providing another DD statement as follows:

```
//ddname  DD   DSNAME=dsname(INDEX),            X
//             SPACE=(CYL,(no. of               X
//             cylinders)).UNIT=unit,           X
//             VOLUME=SER=serial,               X
//             DCB=(DSORG=IS,. . .)
//        DD   DSNAME=dsname(PRIME),            X
//             SPACE=(CYL,(no. of               X
//             cylinders)),UNIT=unit,           X
//             VOLUME=SER=serial,               X
//             DCB=(DSORG=IS,. . .)
```

These DD statements will produce two separate areas: index and prime. Each area is on the same volume.

If, along with more control of his index, the programmer wishes an independent overflow area, he can specify a third DD statement as described above. The order will then be:

```
//ddname  DD   DSNAME=dsname(INDEX),. . .
//        DD   DSNAME=dsname(PRIME),. . .
//        DD   DSNAME=dsname(OVFLOW),. . .
```

These DD statements will produce three separate areas: index, prime, and overflow.

Note that the OPTCD subparameter of the DCB parameter in each of the DD statements must specify an independent overflow area (OPTCD=I). All three areas reside on the same volume if so specified in the VOLUME parameter.

Note 1: The prime and index overflow areas must always be on the same unit type.

Note 2: The order of the DSNAME parameter elements in all of the examples above must be followed when placing the DD statements into the input stream, or an abnormal termination of the job will result.

The example in Figure 146 defines a new indexed-sequential file that consists of three separate areas. All three areas reside on the same volume. The volume is on an IBM 2311 Disk Storage Drive.

Cataloging Files

An indexed-sequential file can be cataloged if:

- All the areas of the file are allocated with a single DD statement. Such a file is cataloged in the usual manner by specifying the DISP parameter in the DD statement:

  DISP=(NEW,CATLG)

- The areas are allocated with more than one DD statement, but all volumes are on the same type of device. Such a file is cataloged using the IEHPROGM utility program (see the publication *IBM System/360 Operating System: Utilities,* Form C28-6586).

An indexed-sequential file that is being created cannot be cataloged if its areas are on different device types. An existing indexed-sequential file cannot be cataloged through the specification of the CATLG subparameter of the DISP parameter in the DD statement.

Note: The DD statement or statements defining a new or existing indexed-sequential file can appear in cataloged procedures.

```
//FILE     DD   DSNAME=ISM(INDEX),UNIT=2311,SPACE=(CYL,(1)),    X
//              VOLUME=SER=111111,DCB=(DSORG=IS,...)
//         DD   DSNAME=ISM(PRIME),UNIT=2311,SPACE=(CYL,(5)),    X
//              VOLUME=SER=111111,DCB=(DSORG=IS,...)
//         DD   DSNAME=ISM(OVFLOW),UNIT=2311,SPACE=(CYL,(1)),   X
//              VOLUME=SER=111111,DCB=(DSORG=IS,...)
```

53237

Figure 146. Example of DD Statements for New Indexed-Sequential Files

Calculating Space Requirements

To determine the number of cylinders required for an indexed-sequential file, the programmer must consider the number of records that will fit on a cylinder, the number of records that will be processed, and the amount of space required for indexes and overflow areas. In making the computations, additional space is also required for device overhead. Detailed information can be found in the publication *IBM System/360 Operating System: Supervisor and Data Management Services,* Form C28-6646.

The discussion that follows is primarily concerned with indexed-sequential files that can be contained on a single volume. Additional information about processing existing indexed-sequential files, including multi-volume files, can be found in the publication *IBM System/360 Operating System: Job Control Language,* Form C28-6539.

Parameter Requirements

In the DD statement(s) indicating an existing indexed-sequential file, the following differences and requirements should be noted:

DCB

The DSORG=IS subparameter must be specified, whereas the BUFNO subparameter is optional. The OPTCD field must not be specified again. Any OPTCD subparameter facilities that were specified when the file was created are in effect as long as the data set exists. For example, if the programmer specified the write validity check option (OPTCD=W) when he created the file, the option is still in effect at the time of any subsequent WRITE statement. The BLKSIZE subparameter must not be specified.

DSNAME

This parameter is written DSNAME=dsname. The element subparameters (INDEX, PRIME, OVFLOW) must not be written.

DISP

The first subparameter must be OLD. The second subparameter cannot be CATLG or UNCATLG (see *Cataloging Files* in this section for more information on cataloging indexed-sequential files).

For further information about QISAM parameters, see *QISAM DD Statement Requirements* in this section.

Using the DD Statement -- Single-Volume Files

Only one DD statement is needed to specify an existing file if all of the areas are on one volume. The following is an example of a DD statement that can be used when processing a single-volume QISAM file.

```
//ddname  DD    DSNAME=dsname,DCB=                X
//              (DSORG=IS,. . .),                 X
//              UNIT=unit name, DISP=OLD          X
```

Additional Facilities: Master Index

QISAM provides a master index facility to avoid inefficient serial searches of large cylinder indexes. The master index provides an index to the cylinder index. The programmer can specify with the DCB parameter in his DD statement(s) (see *QISAM DD Statement Requirements* in this section) that a master index be built if the size of a cylinder index exceeds a certain number of tracks. Each entry in the master index points to a track of the cylinder index. If the size of the master index exceeds the number of tracks specified in the NTM parameter of the DD statement, the master index is automatically indexed by a higher level master index. Three such higher level master indexes can be constructed.

Using RPG to Create an Indexed-Sequential File

The section *Creating Indexed-Sequential Files* and Figure 147 describe the RPG specifications and entries needed to create indexed-sequential files.

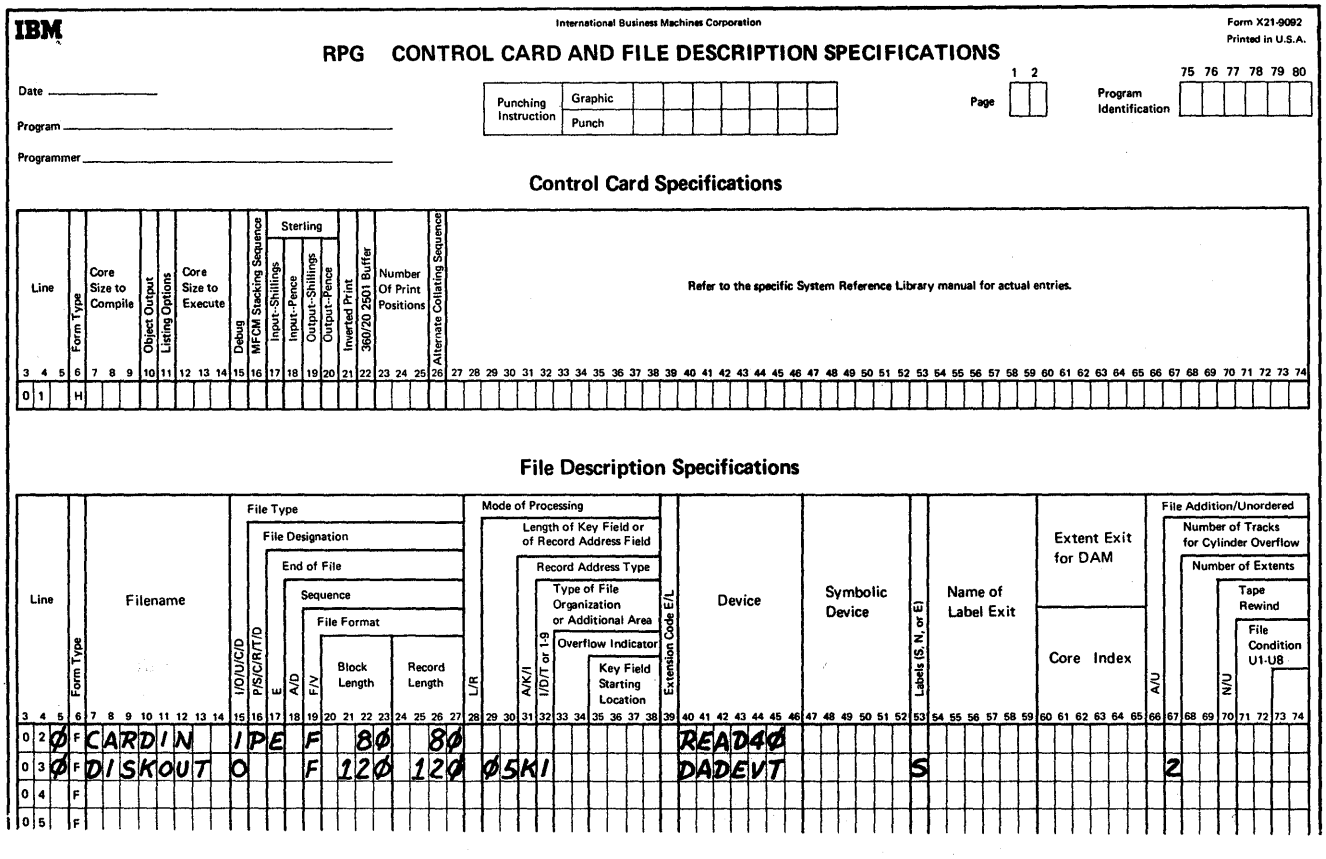
Processing with QISAM: Reading or Updating Indexed Sequential Files

The QISAM file processing technique can be used to read or update an existing indexed-sequential file. Adding a record to an already existing file, however, can only be done with BISAM (see *Processing with BISAM* in this section).

When the QISAM file processing technique is used to read an input file, one logical record at a time is read in an ascending order determined by the record keys. Deleted records are not made available. If there are records in the overflow area, this order will not correspond exactly to the physical order of the records in the file. The file must have been created using QISAM.

When the QISAM file processing technique is used to update a file, RPG permits updating in place or deletion of a logical record. A logical record is updated in place by reading, updating, and rewriting it during one cycle of the RPG object program. Alteration of record length or insertion of new records is not permitted. A logical record is marked for deletion by moving a delete character into the first character position of the record and then conditioning the record to be written on output. (See this section *Additional Facilities, Delete Option.)* Records in the file that contain this deletion code are not made available.

Further details about DD statements for existing single-volume and multi-volume QISAM files can be found in the publication *IBM System/360 Operating System: Job Control Language,* Form C28-6539.



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)

Program Identification (75 76 77 78 79 80)

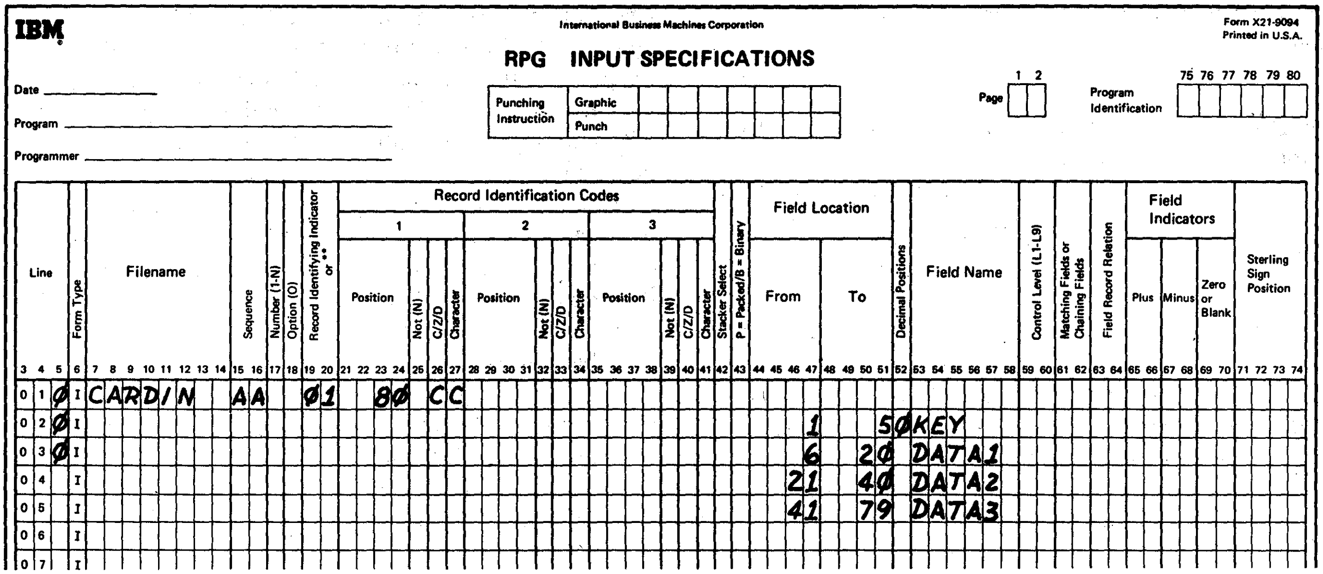
Control Card Specifications

Line (3-5)	Form Type (6)	Core Size to Compile (7-9)	Object Output (10)	Listing Options (11)	Core Size to Execute (12-14)	Debug (15)	MFCM Stacking Sequence (16)	Sterling: Input-Shillings (17)	Input-Pence (18)	Output-Shillings (19)	Output-Pence (20)	Inverted Print (21)	360/20 2501 Buffer (22)	Number Of Print Positions (23-25)	Alternate Collating Sequence (26)	Refer to the specific System Reference Library manual for actual entries. (27-74)
01	H															

File Description Specifications

Line (3-5)	Form Type (6)	Filename (7-14)	File Type: I/O/U/C/D (15)	File Designation: P/S/C/R/T/D (16)	End of File: E (17)	Sequence: A/D (18)	File Format: F/V (19)	Block Length (20-23)	Record Length (24-27)	L/R (28)	Mode of Processing: Length of Key Field or of Record Address Field (29-30)	Record Address Type: A/K/I (31)	Type of File Organization or Additional Area: I/D/T or 1-9 (32)	Overflow Indicator (33-34)	Key Field Starting Location (35-38)	Extension Code E/L (39)	Device (40-46)	Symbolic Device (47-52)	Labels (S, N, or E) (53)	Name of Label Exit (54-59)	Extent Exit for DAM / Core Index (60-65)	File Addition/Unordered: A/U (66)	Number of Tracks for Cylinder Overflow (67)	Number of Extents (68-69)	Tape Rewind: N/U (70)	File Condition U1-U8 (71-72)	(73-74)
02ϕ	F	CARDIN	I	P	E		F	8ϕ	8ϕ								READ4ϕ										
03ϕ	F	DISKOUT	O				F	12ϕ	12ϕ		ϕ5	K	I				DADEVT		S				2				
04	F																										
05	F																										

Figure 147. Indexed-Sequential Load (Part 1 of 2)



IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

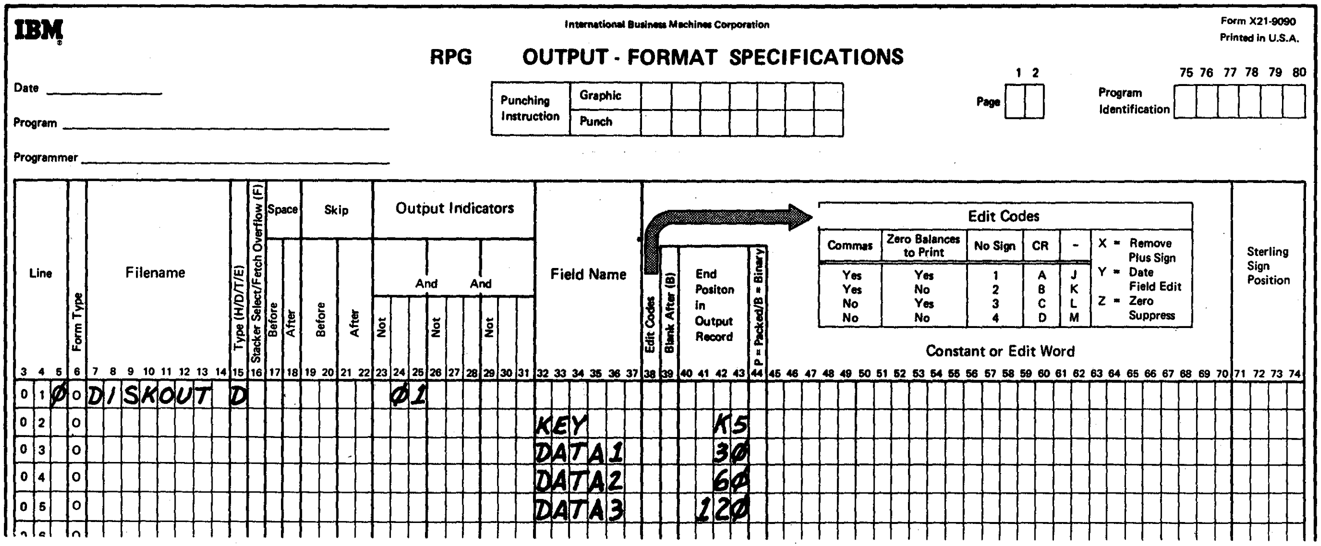
Date
Program
Programmer

Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line (3-5)	Form Type (6)	Filename (7-14)	Sequence (15-16)	Number (1-N) (17)	Option (O) (18)	Record Identifying Indicator or ** (19-20)	Record Identification Codes 1: Position (21-24)	Not (N) (25)	C/Z/D (26)	Character (27)	Field Location From (44-47)	Field Location To (48-51)	Decimal Positions (52)	Field Name (53-58)
01Ø	I	CARDIN	AA			Ø1	8Ø		C	C				
02Ø	I										1	5	Ø	KEY
03Ø	I										6	2Ø		DATA1
04	I										21	4Ø		DATA2
05	I										41	79		DATA3
06	I													
07	I													



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 1 2

Program Identification 75 76 77 78 79 80

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 (3-5)	Form Type (6)	Filename (7-14)	Type (H/D/T/E) (15)	Output Indicators (23-25)	Field Name (32-37)	End Position in Output Record (40-43)
01Ø	O	DISKOUT	D	Ø1		
02	O				KEY	K5
03	O				DATA1	3Ø
04	O				DATA2	6Ø
05	O				DATA3	12Ø
06	O					

Figure 147. Indexed-Sequential Load (Part 2 of 2)

Using RPG to Process and Update Files Sequentially

The section *Processing Indexed-Sequential Files* and Figure 148 give examples of the RPG specifications and entries needed for file processing.

Processing with BISAM: Reading, Writing, And Rewriting Indexed-Sequential Files

The BISAM file processing technique is used for the direct retrieval of any logical record by its key, the direct update in place of any logical record, and the direct insertion of new logical records into an indexed sequential file. Only files created by the use of QISAM can be referred to by BISAM.

Because of their complementary use of the indexed-sequential file organization, much of the data given for QISAM in *Processing with QISAM* applies to BISAM. The main difference between using QISAM and BISAM is that the latter supports random operations and no particular sequencing of keys is required when processing.

The addition of new records may be combined only with other random processing of a particular file in a given job step.

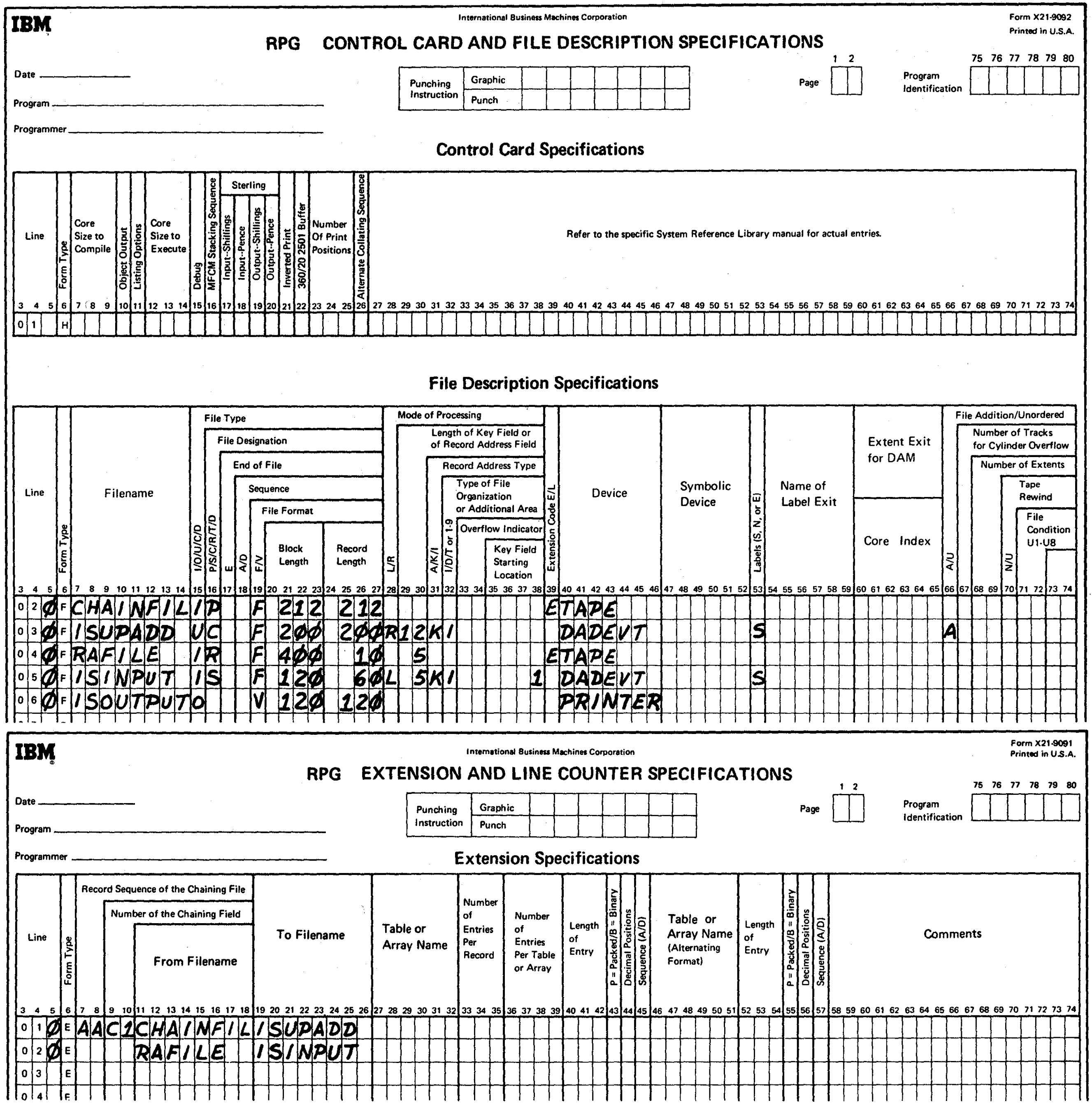
Using RPG to Process and Add to Files Randomly

The sections *Processing Indexed-Sequential Files* and *Adding to Indexed-Sequential Files* along with Figures 148 and 149 illustrate the RPG specifications and entries needed. For input or updating, the programmer must supply a key matching that of the record to be read. The keys of added records are supplied through the same methods available when creating files.

Additional Facilities: Reorganizing Files

As new records are added to an indexed-sequential file, chains of records may be created in the overflow area if one exists. The access time for retrieving records in an overflow area is greater than that required for retrieving records in the prime area. I/O performance is, therefore, sharply reduced when many overflow records develop. For this reason, an indexed-sequential file can be reorganized as soon as the need becomes evident. The system maintains a set of statistics to assist the programmer when reorganization is desired. These statistics are maintained as fields of the file's data control block. The location of the data control block can be passed to a called assembler langauge program which can then test its fields. If these statistics are desired, the OPTCD subparameter of the DCB parameter must have included OPTCD=R in each of the DD statements when the file was created. Additional information is contained in the publication *IBM System/360 Operating System: Supervisor and Data Management Services,* Form C28-6646.

Delete Option: In order to keep the number of records in the overflow area to a minimum, and to eliminate unnecessary records, an existing record may be marked for deletion. This is done by moving a byte of all 1's into the first character position of the record. One method of creating a byte of all 1's is to define it in an assembly language subroutine link edited to the RPG object program. The record is not physically deleted unless it is forced off its prime track by the insertion of a new record, or if the file is reorganized (see preceeding section). The deleted record may be replaced (using BISAM) by a record with an identical key without an error being indicated. RPG will not read a record marked for deletion, whether the record has been physically deleted or not (this is true only for sequential processing).



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 — Program Identification 75 76 77 78 79 80

Control Card Specifications

Line	Form Type	Core Size to Compile	Object Output	Listing Options	Core Size to Execute	Debug	MFCM Stacking Sequence	Input-Shillings	Input-Pence	Output-Shillings	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.
0 1	H															

File Description Specifications

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V	Block Length	Record Length	L/R	Length of Key Field or of Record Address Field	Record Address Type (A/K/I)	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	Core Index	A/U
0 2 Ø	F	CHAINFIL	I	P			F	212	212							E	TAPE					
0 3 Ø	F	ISUPADD	U	C			F	200	200	R	12	K	I				DADEVT		S			A
0 4 Ø	F	RAFILE	I	R			F	400	10		5					E	TAPE					
0 5 Ø	F	ISINPUT	I	S			F	120	60	L	5	K	I		1		DADEVT		S			
0 6 Ø	F	ISOUTPUT	O				V	120	120								PRINTER					

Mode of Processing — Extent Exit for DAM — File Addition/Unordered — Number of Tracks for Cylinder Overflow — Number of Extents — Tape Rewind — File Condition U1-U8 — N/U

IBM International Business Machines Corporation Form X21-9091 Printed in U.S.A.

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

Date ____ Program ____ Programmer ____

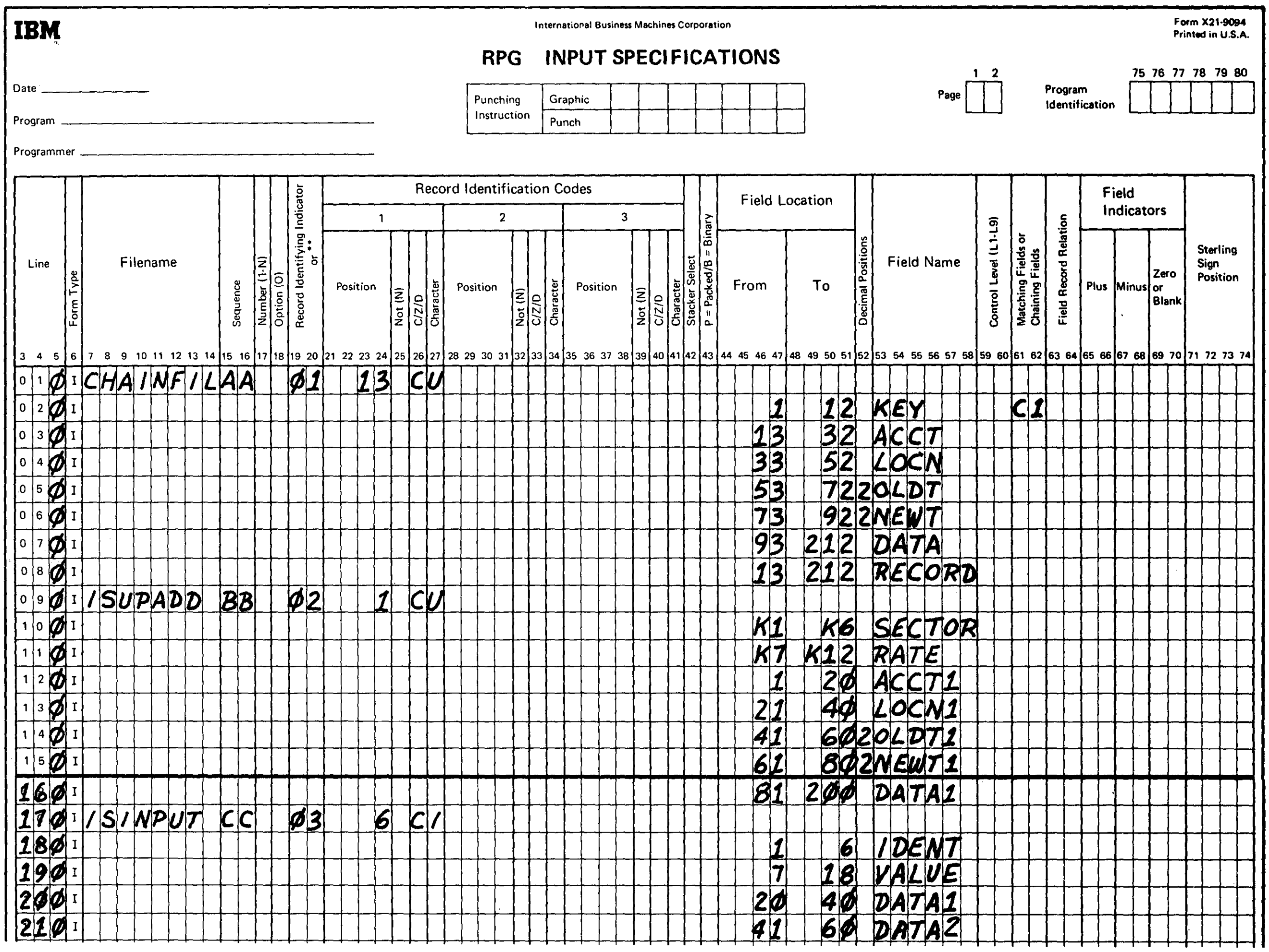
Punching Instruction: Graphic / Punch

Page 1 2 — Program Identification 75 76 77 78 79 80

Extension Specifications

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P = Packed/B = Binary	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P = Packed/B = Binary	Decimal Positions	Sequence (A/D)	Comments
0 1 Ø	E	AA	C1	CHAINFIL	ISUPADD													
0 2 Ø	E			RAFILE	ISINPUT													
0 3	E																	
0 4	E																	

Figure 148. Processing ISAM Files Using Chaining and RA Files (Part 1 of 3)



IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date

Program

Programmer

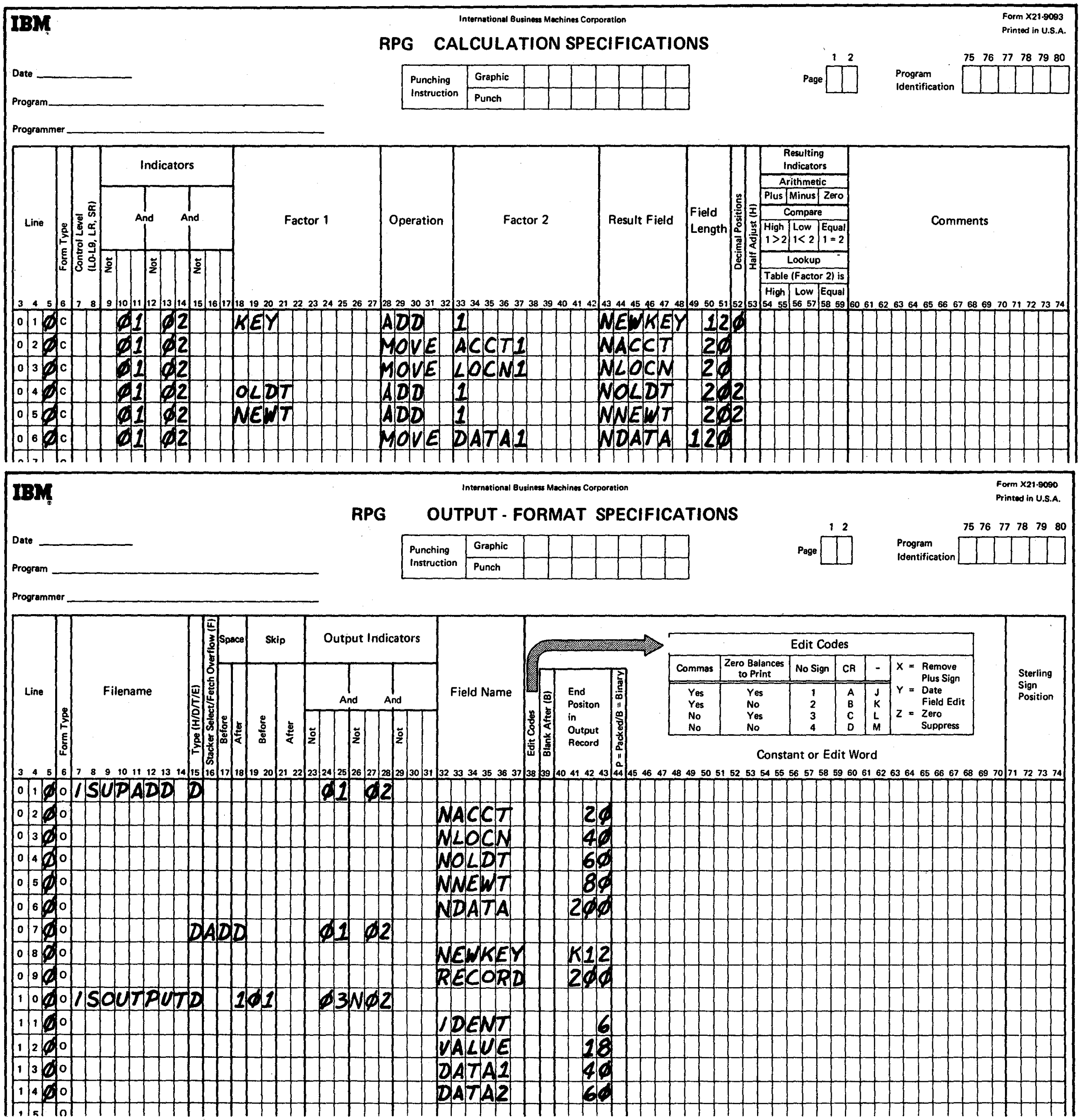
Punching Instruction	Graphic							
	Punch							

Page 1 2 □□

Program Identification 75 76 77 78 79 80 □□□□□□

Line (3-5)	Form Type (6)	Filename (7-14)	Sequence (15-16)	Number (1-N) (17)	Option (O) (18)	Record Identifying Indicator or ** (19-20)	Record Identification Codes 1: Position (21-24)	Not (N) (25)	C/Z/D (26)	Character (27)	2: Position (28-31)	Not (N) (32)	C/Z/D (33)	Character (34)	3: Position (35-38)	Not (N) (39)	C/Z/D (40)	Character (41)	Stacker Select (42)	P = Packed/B = Binary (43)	Field Location From (44-47)	Field Location To (48-51)	Decimal Positions (52)	Field Name (53-58)	Control Level (L1-L9) (59-60)	Matching Fields or Chaining Fields (61-62)	Field Record Relation (63-64)	Field Indicators Plus (65-66)	Minus (67-68)	Zero or Blank (69-70)	Sterling Sign Position (71-74)
01Ø	I	CHAINFIL	AA			Ø1	13		C	U																					
02Ø	I																				1	12		KEY		C1					
03Ø	I																				13	32		ACCT							
04Ø	I																				33	52		LOCN							
05Ø	I																				53	72	2	OLDT							
06Ø	I																				73	92	2	NEWT							
07Ø	I																				93	212		DATA							
08Ø	I																				13	212		RECORD							
09Ø	I	/SUPADD	BB			Ø2	1		C	U																					
10Ø	I																				K1	K6		SECTOR							
11Ø	I																				K7	K12		RATE							
12Ø	I																				1	2Ø		ACCT1							
13Ø	I																				21	4Ø		LOCN1							
14Ø	I																				41	6Ø	2	OLDT1							
15Ø	I																				61	8Ø	2	NEWT1							
16Ø	I																				81	2ØØ		DATA1							
17Ø	I	/SINPUT	CC			Ø3	6		C	/																					
18Ø	I																				1	6		IDENT							
19Ø	I																				7	18		VALUE							
2ØØ	I																				2Ø	4Ø		DATA1							
21Ø	I																				41	6Ø		DATA2							

Figure 148. Processing ISAM Files Using Chaining and RA Files (Part 2 of 3)



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	And	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators	Comments
01Ø	C		Ø1	Ø2		KEY	ADD	1	NEWKEY	12	Ø			
02Ø	C		Ø1	Ø2			MOVE	ACCT1	NACCT	2Ø				
03Ø	C		Ø1	Ø2			MOVE	LOCN1	NLOCN	2Ø				
04Ø	C		Ø1	Ø2		OLDT	ADD	1	NOLDT	2Ø	2			
05Ø	C		Ø1	Ø2		NEWT	ADD	1	NNEWT	2Ø	2			
06Ø	C		Ø1	Ø2			MOVE	DATA1	NDATA	12Ø				

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 1 2 Program Identification 75 76 77 78 79 80

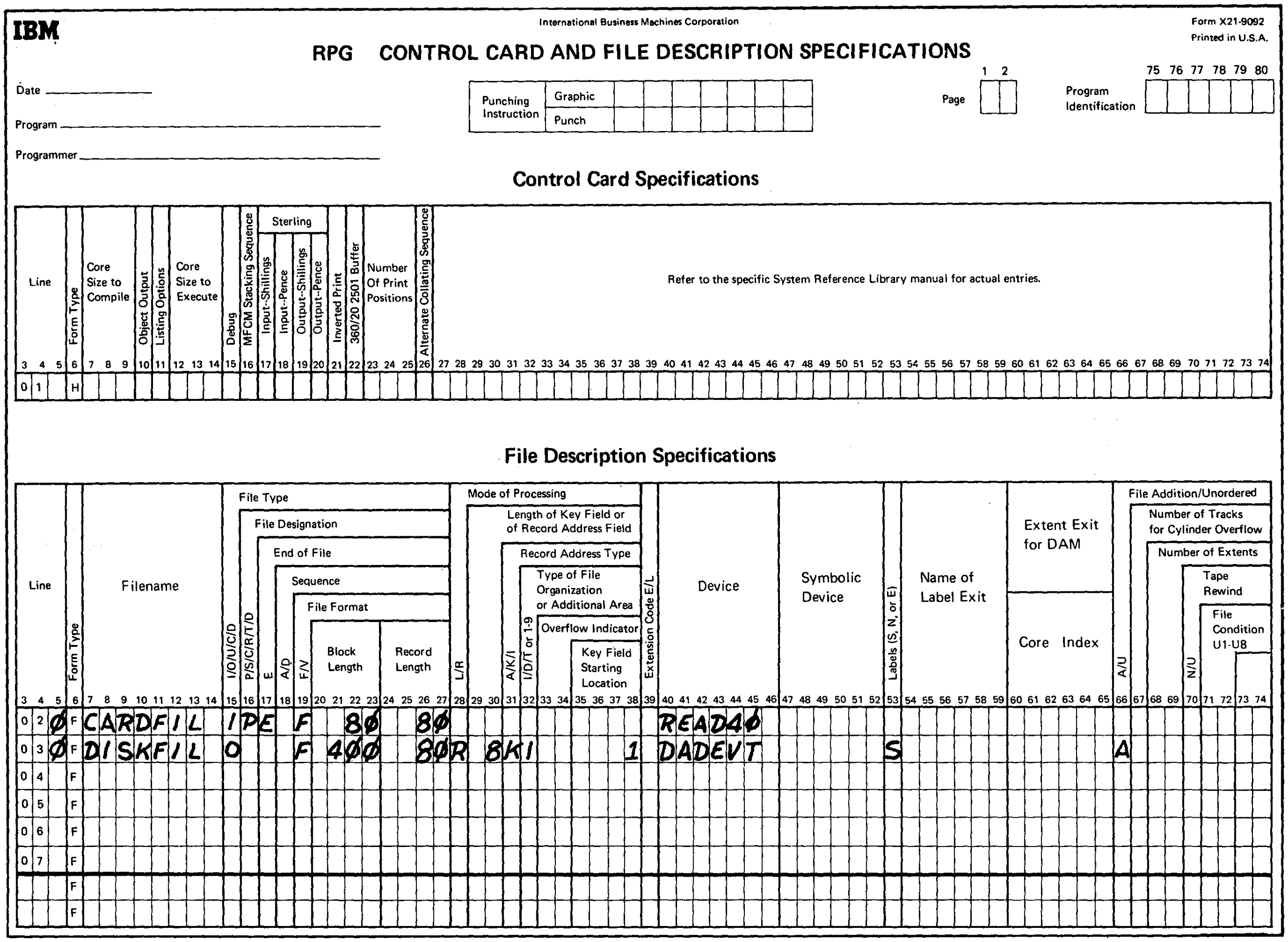
Edit Codes

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/Fetch Overflow (F)	Space	Skip	Output Indicators	And	And	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	P = Packed/B = Binary	Constant or Edit Word	Sterling Sign Position
01Ø	O	ISUPADD	D				Ø1	Ø2								
02Ø	O									NACCT			2Ø			
03Ø	O									NLOCN			4Ø			
04Ø	O									NOLDT			6Ø			
05Ø	O									NNEWT			8Ø			
06Ø	O									NDATA			2ØØ			
07Ø	O		DADD				Ø1	Ø2								
08Ø	O									NEWKEY			K12			
09Ø	O									RECORD			2ØØ			
10Ø	O	ISOUTPUT	D		1	Ø1	Ø3	NØ2								
11Ø	O									IDENT			6			
12Ø	O									VALUE			18			
13Ø	O									DATA1			4Ø			
14Ø	O									DATA2			6Ø			

Figure 148. Processing ISAM Files Using Chaining and RA Files (Part 3 of 3)



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

Program Identification 75 76 77 78 79 80

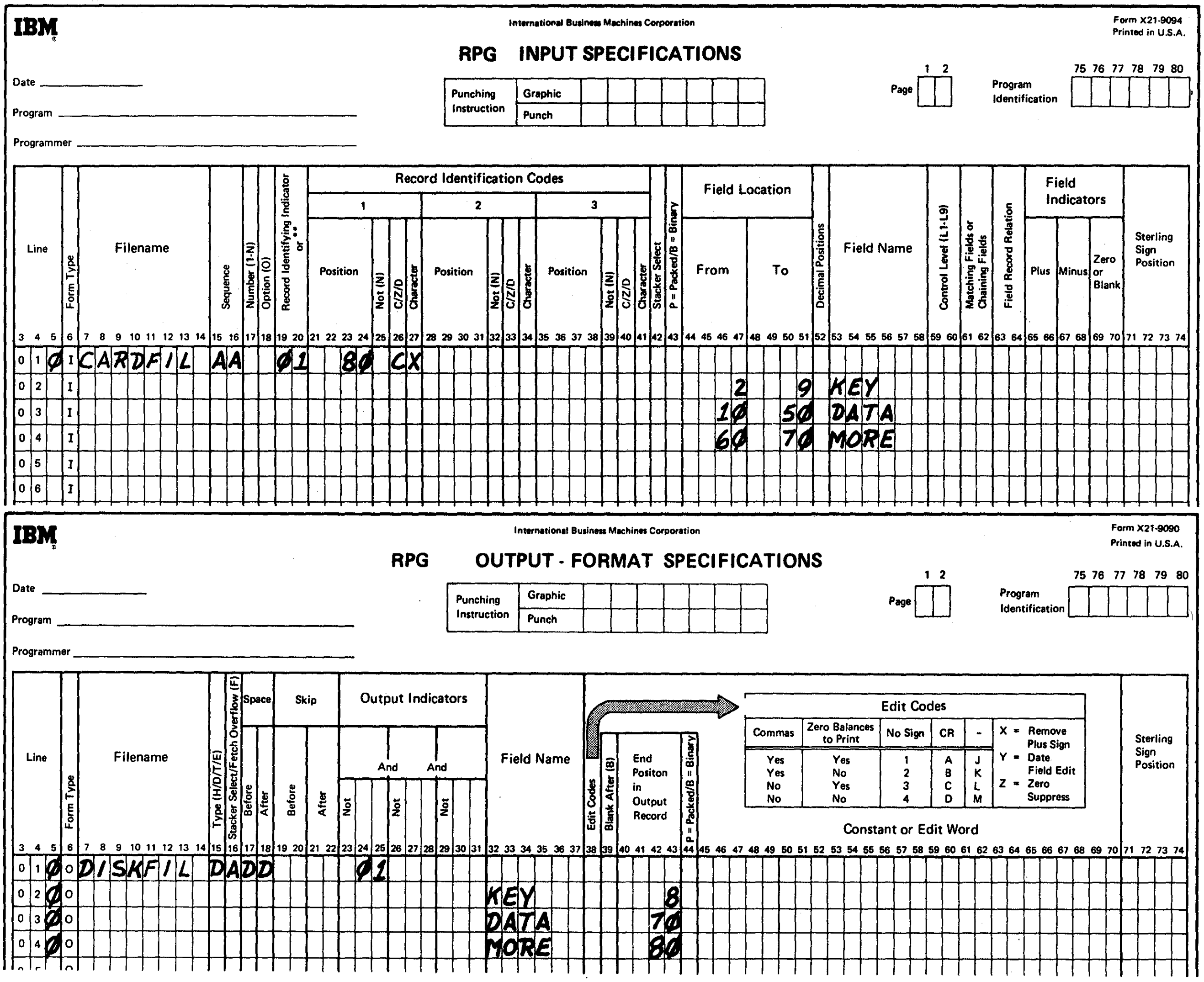
Control Card Specifications

Line (3-5)	Form Type (6)	Core Size to Compile (7-9)	Object Output (10)	Listing Options (11)	Core Size to Execute (12-14)	Debug (15)	MFCM Stacking Sequence (16)	Sterling: Input-Shillings (17)	Input-Pence (18)	Output-Shillings (19)	Output-Pence (20)	Inverted Print (21)	360/20 2501 Buffer (22)	Number Of Print Positions (23-25)	Alternate Collating Sequence (26)	27-74
0 1	H															Refer to the specific System Reference Library manual for actual entries.

File Description Specifications

Line (3-5)	Form Type (6)	Filename (7-14)	File Type: I/O/U/C/D (15)	File Designation: P/S/C/R/T/D (16)	End of File: E (17)	Sequence: A/D (18)	File Format: F/V (19)	Block Length (20-23)	Record Length (24-27)	Mode of Processing: L/R (28)	Length of Key Field or of Record Address Field (29-30)	Record Address Type: A/K/I (31)	Type of File Organization or Additional Area: I/D/T or 1-9 (32)	Overflow Indicator (33-34)	Key Field Starting Location (35-38)	Extension Code E/L (39)	Device (40-46)	Symbolic Device (47-52)	Labels (S, N, or E) (53)	Name of Label Exit (54-59)	Extent Exit for DAM / Core Index (60-65)	File Addition/Unordered: A/U (66)	Number of Tracks for Cylinder Overflow (67-68)	Number of Extents (69)	Tape Rewind: N/U (70)	File Condition U1-U8 (71-72)	73-74
0 2 Ø	F	CARDFIL	I	P	E		F	8Ø	8Ø								READ4Ø										
0 3 Ø	F	DISKFIL	O				F	4ØØ	8Ø	R	8	K	I		1		DADEVT		S			A					
0 4	F																										
0 5	F																										
0 6	F																										
0 7	F																										
	F																										
	F																										

Figure 149. Indexed-Sequential Add (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 (1 2)

Program Identification (75 76 77 78 79 80)

Line (3-5)	Form Type (6)	Filename (7-14)	Sequence (15-16)	Number (1-N) (17)	Option (O) (18)	Record Identifying Indicator or ** (19-20)	Record Identification Codes 1: Position (21-24)	Not (N) (25)	C/Z/D (26)	Character (27)	Record Identification Codes 2: Position (28-31)	Not (N) (32)	C/Z/D (33)	Character (34)	Record Identification Codes 3: Position (35-38)	Not (N) (39)	C/Z/D (40)	Character (41)	Stacker Select (42)	P = Packed/B = Binary (43)	Field Location From (44-47)	Field Location To (48-51)	Decimal Positions (52)	Field Name (53-58)	Control Level (L1-L9) (59-60)	Matching Fields or Chaining Fields (61-62)	Field Record Relation (63-64)	Field Indicators Plus (65-66)	Minus (67-68)	Zero or Blank (69-70)	Sterling Sign Position (71-74)
0 1 Ø	I	CARDFIL	AA			Ø1	8Ø		C	X																					
0 2	I																				2	9		KEY							
0 3	I																				1Ø	5Ø		DATA							
0 4	I																				6Ø	7Ø		MORE							
0 5	I																														
0 6	I																														

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 (1 2)

Program Identification (75 76 77 78 79 80)

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

Line (3-5)	Form Type (6)	Filename (7-14)	Type (H/D/T/E) (15)	Stacker Select/Fetch Overflow (F) (16)	Space Before (17)	Space After (18)	Skip Before (19-20)	Skip After (21-22)	Output Indicators Not (23)	(24-25)	And Not (26)	(27-28)	And Not (29)	(30-31)	Field Name (32-37)	Edit Codes (38)	Blank After (B) (39)	End Positon in Output Record (40-43)	P = Packed/B = Binary (44)	Constant or Edit Word (45-70)	Sterling Sign Position (71-74)
0 1 Ø	O	DISKFIL	D	A	D	D				Ø1											
0 2 Ø	O														KEY			8			
0 3 Ø	O														DATA			7Ø			
0 4 Ø	O														MORE			8Ø			
0 5	O																				

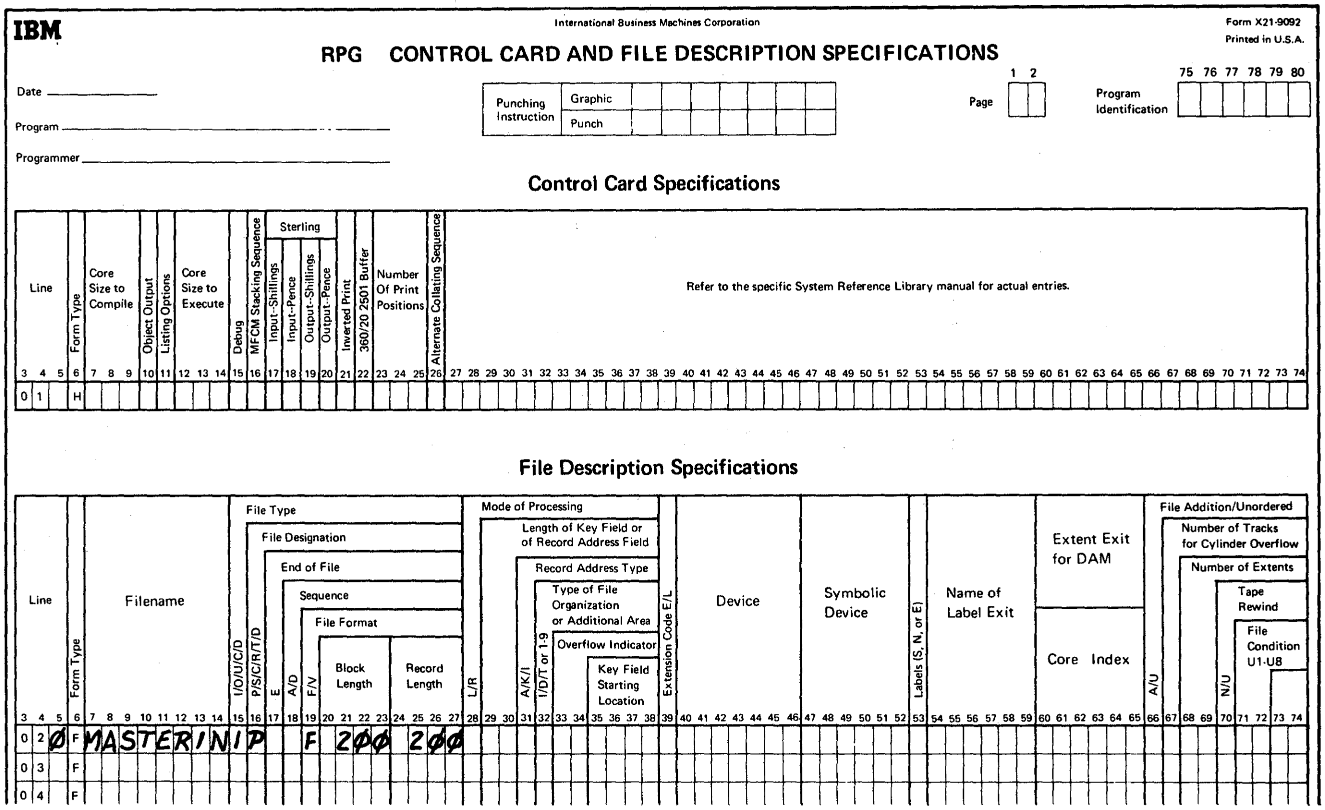
Figure 149. Indexed-Sequential Add (Part 2 of 2)

In this section, the methods of retrieving records from one input file are discussed. Three types of file organization can be processed: Sequential, indexed-sequential, and direct.

If there is only one input file, it must be designated as the primary file by entering a P in position 16 of the File Description Specifications form.

Sequential File

Figure 150 shows the coding on the File Description Specifications form necessary to process a sequential file. The name of the file is MASTERIN. The records are fixed-length, and the block length is 200. Each record is 200 characters long. A blank in position 32 indicates that the file is a 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 1 2

Program Identification 75 76 77 78 79 80

Control Card Specifications

Line (3–5) | Form Type (6) | Core Size to Compile (7–9) | Object Output (10) | Listing Options (11) | Core Size to Execute (12–14) | Debug (15) | MFCM Stacking Sequence (16) | Sterling: Input-Shillings (17), Input-Pence (18), Output-Shillings (19), Output-Pence (20) | Inverted Print (21) | 360/20 2501 Buffer (22) | Number Of Print Positions (23–25) | Alternate Collating Sequence (26) | 27–74: Refer to the specific System Reference Library manual for actual entries.

0 1	H

File Description Specifications

Line (3–5) | Form Type (6) | Filename (7–14) | File Type: I/O/U/C/D (15) | File Designation: P/S/C/R/T/D (16) | End of File: E (17) | Sequence: A/D (18) | File Format: F/V (19) | Block Length (20–23) | Record Length (24–27) | Mode of Processing: L/R (28) | Length of Key Field or of Record Address Field (29–30) | Record Address Type: A/K/I (31) | Type of File Organization or Additional Area: I/D/T or 1-9 (32) | Overflow Indicator (33–34) | Key Field Starting Location (35–38) | Extension Code E/L (39) | Device (40–46) | Symbolic Device (47–52) | Labels (S, N, or E) (53) | Name of Label Exit (54–59) | Extent Exit for DAM / Core Index (60–65) | File Addition/Unordered: A/U (66) | Number of Tracks for Cylinder Overflow (67–68) | Number of Extents (69) | Tape Rewind: N/U (70) | File Condition U1-U8 (71–72) | 73–74

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V	Block Length	Record Length
0 2 Ø	F	MASTERIN	I	P			F	2ØØ	2ØØ
0 3	F								
0 4	F								

Figure 150. Coding a Sequential File

Processing An Indexed-Sequential File Between Limits

If only a part of an indexed-sequential file is to be processed, the object program must be supplied with both the low and high keys that describe which part of the file will be processed. Mode of Processing (position 28) of the File Description Specifications form must contain L.

The object program obtains the limits to be processed from a record contained in an RA file and then processes all data between those limits. The object program then reads another record in the RA file. The procedure is continued until the RA file is depleted.

In Figure 151 the first card of the record address file shows that the object module is to process from Bell to Dennis. The second card shows that the program is to process from Dixon to Howard. The third card shows that the third part of the file to be processed ranges from Keith to Paige. Thus, the data records that the program processes are contained with these limits. In this example, the record keys are customer names.

The record address file in this example is nothing more than a file which supplies limits of the file to be processed. Rules for forming record address files are contained in *Creating Record Address Files* in this section.

Figure 152 illustrates the coding required on the File Description and Extension Specifications forms when limits of an indexed-sequential file are to be processed. In Figure 152, on the File Description Specifications form DISKIN is the name of the input file. The records are fixed-length, and the block length is 150. Each record is 150 characters long. Limits of the file are to be processed as indicated by the L in position 28. The K in position 31 indicates that the record key will be used to obtain the records from the file.

The record address file (RAFLIMIT) is also an input file. The R in position 16 indicates that it is an RA file. It is fixed length; it has a block length of 80, and each record is 80 characters in length. The length of each record address field is 6 characters.

Because the Extension Specifications form is required to relate the RA file to the DASD file, an E is entered in Extension Code (position 39). On the Extension Specifications form the coding illustrates that RAFLIMIT is to provide the addresses for DISKIN.

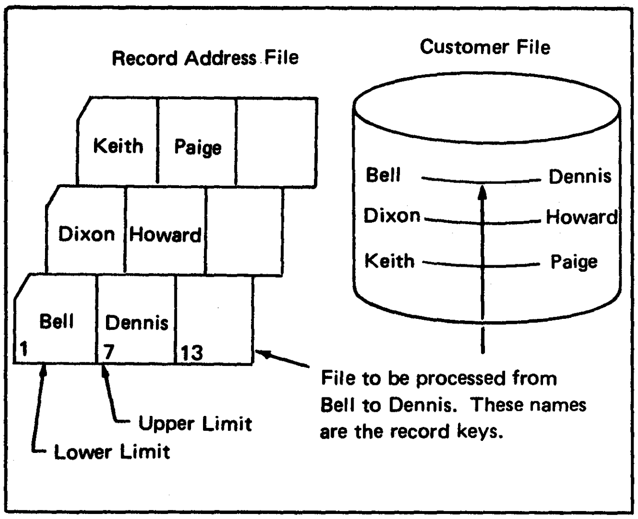
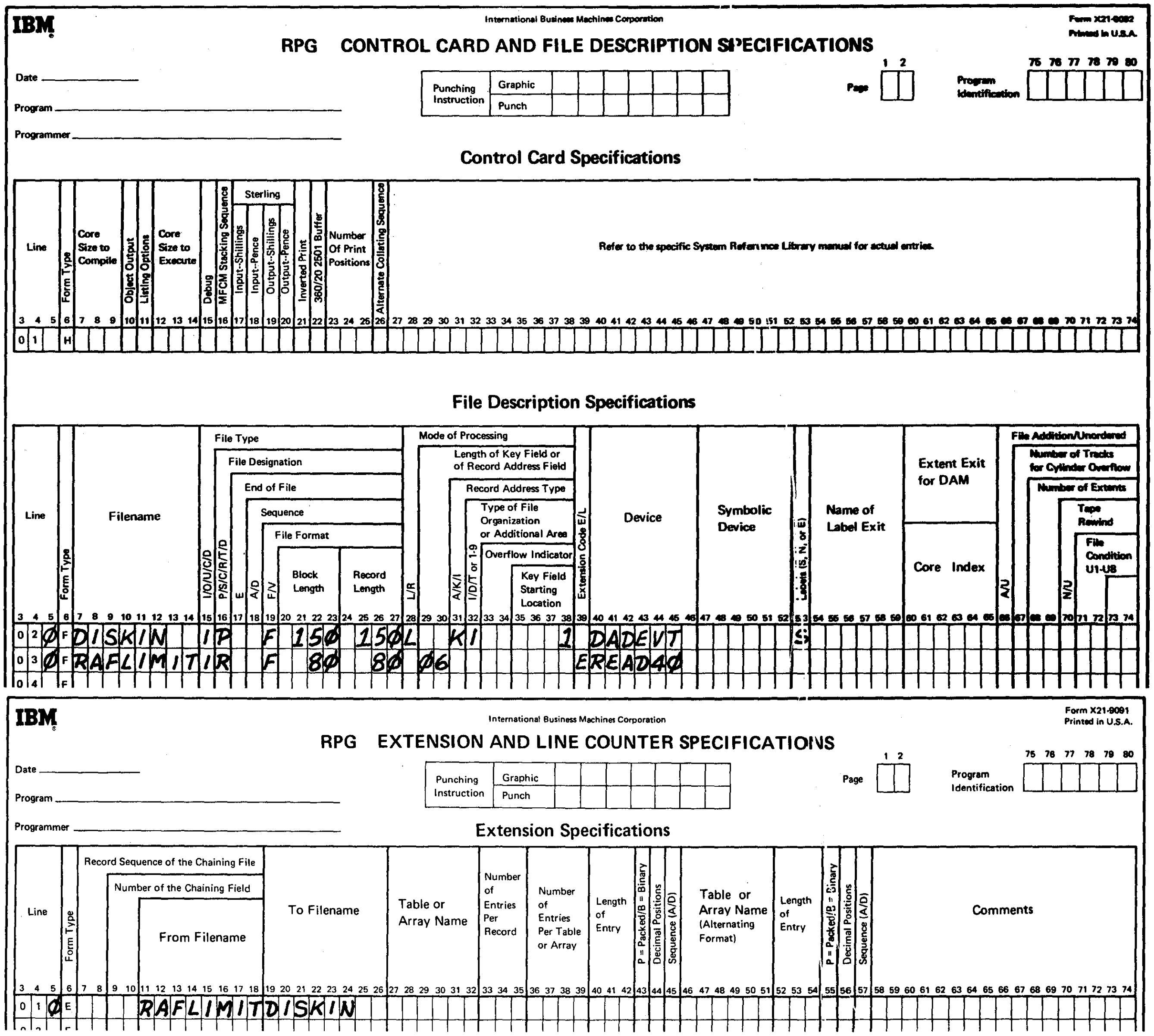


Figure 151. Contents of the Record Address File



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 1 2 — Program Identification 75 76 77 78 79 80

Control Card Specifications

Refer to the specific System Reference Library manual for actual entries.

Line	Form Type
01	H

File Description Specifications

Line	Form Type	Filename (7–14)	I/O/U/C/D (15)	P/S/C/R/T/D (16)	F/V (19)	Block Length (20–23)	Record Length (24–27)	L/R (28)	Length of Key Field or of Record Address Field (29–30)	A/K/I (31)	I/D/T or 1-9 (32)	Key Field Starting Location (35–38)	Extension Code E/L (39)	Device (40–46)	Labels (S, N, or E) (53)
02Ø	F	DISKIN	I	P	F	15Ø	15Ø	L		K	I	1		DADEVT	S
03Ø	F	RAFLIMIT	I	R	F	8Ø	8Ø		Ø6				E	READ4Ø	
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 1 2 — Program Identification 75 76 77 78 79 80

Extension Specifications

Line	Form Type	From Filename (11–18)	To Filename (19–26)
01Ø	E	RAFLIMIT	DISKIN

Figure 152. Processing a Part of the Indexed-Sequential File

Random Processing of an Indexed-Sequential File

An indexed-sequential file may be processed randomly by supplying an RA file. Instead of supplying the limits (as in the case of sequential processing), the RA file contains the record key of each record of the file to be processed.

Figures 153 and 154 illustrate the specifications for this type of organization. In this example, the first record from the record address file is read. The first RA file entry ADAMS is used to obtain the data record whose record key is ADAMS. The record is retrieved and is processed. The next entry, CABOT, is then used to find the next data record.

Processing Files With Direct Organization

A file with direct organization must reside on a direct access storage device. Records are retrieved from this file by using relative track address and record key or record identification.

Direct organization is specified by a D in position 32 of the File Description Specifications form. The mode of processing is random, and an R in position 28 indicates random processing. If the record key is used to obtain the records, enter a K in position 31. If a record ID is used, enter an I in position 31.

When an RPG program processes an input file of this organization, an RA file must be supplied. The necessary steps must be supplied which convert the data fields contained in the RA file to the relative track address and the record key or record ID to be retrieved. The conversion routine depends, of course, on the way a particular data field can be converted to the track address of the record.

The conversion routine is unique, according to the needs of a particular installation. It may be nothing more than supplying the information without any calculations. To retrieve records from the file, some data fields must be converted to produce the relative track address of the record.

Supplying Data to be Converted

One entry in the RA file is supplied for each record to be retrieved. The data supplied by the RA file should be such that the relative track address and the record key or record identification can be derived. Because the RA file is not described on the input specifications, the entries in the RA file will be made available consecutively in a field called CONTD. This field is always alphameric, and it has the record length as specified in positions 29-30 of the File Description Specifications form.

Note: The field CONTD is always predefined by RPG as an RPG Label (RLABL).

Associating a Particular Conversion Routine with RA File

The extension specifications describing the RA file contain a field name in positions 27-32. This name is used as Factor 1 on the calculation specificatons with the first specification of the conversion routine.

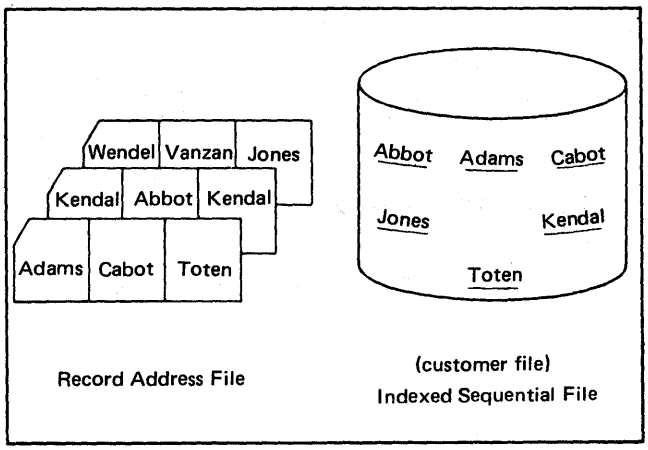
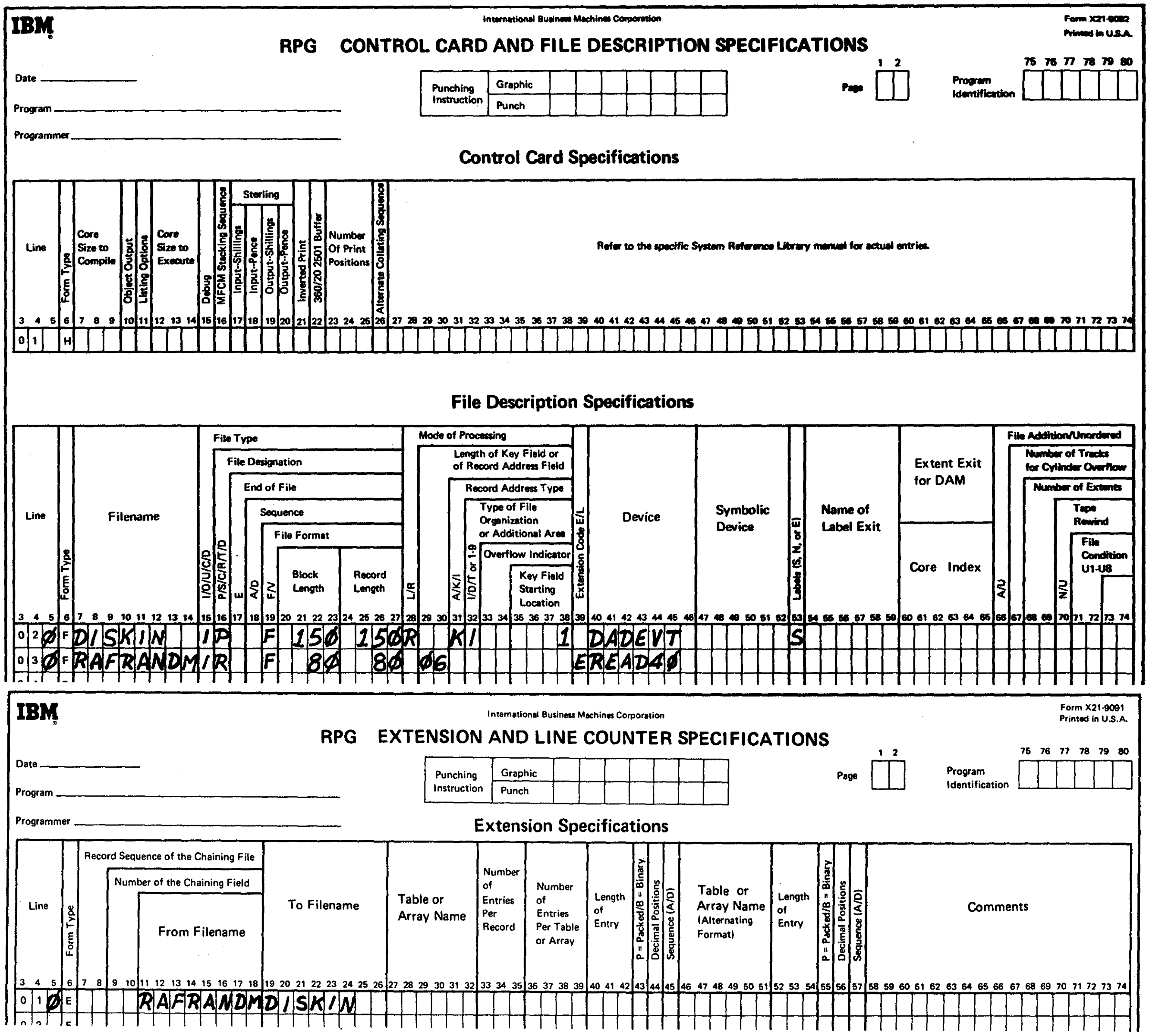


Figure 153. Randomly Processing an Indexed-Sequential File



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 (1 2) — Program Identification (75 76 77 78 79 80)

Control Card Specifications

Refer to the specific System Reference Library manual for actual entries.

Line (3-5)	Form Type (6)
01	H

File Description Specifications

Line (3-5)	Form Type (6)	Filename (7-14)	File Type I/O/U/C/D (15)	File Designation P/S/C/R/T/D (16)	File Format F/V (19)	Block Length (20-23)	Record Length (24-27)	L/R (28)	Length of Key Field or of Record Address Field (29-30)	Record Address Type A/K/I (31)	Type of File Organization or Additional Area I/D/T or 1-9 (32)	Key Field Starting Location (35-38)	Extension Code E/L (39)	Device (40-46)	Labels (S, N, or E) (53)
02∅	F	DISKIN	I	P	F	15∅	15∅	R		K	I	1		DADEVT	S
03∅	F	RAFRANDM	I	R	F	8∅	8∅		∅6				E	READ4∅	

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 (1 2) — Program Identification (75 76 77 78 79 80)

Extension Specifications

Line (3-5)	Form Type (6)	Record Sequence of the Chaining File (7-8)	Number of the Chaining Field (9-10)	From Filename (11-18)	To Filename (19-26)	Table or Array Name (27-32)	Number of Entries Per Record (33-35)	Number of Entries Per Table or Array (36-39)	Length of Entry (40-42)	P = Packed/B = Binary (43)	Decimal Positions (44)	Sequence (A/D) (45)	Table or Array Name (Alternating Format) (46-51)	Length of Entry (52-54)	P = Packed/B = Binary (55)	Decimal Positions (56)	Sequence (A/D) (57)	Comments (58-74)
01∅	E			RAFRANDM	DISKIN													

Figure 154. Processing an Indexed-Sequential File Randomly, Using an RA File

Methods for Specifying a Conversion Routine

Two methods of specifying a conversion routine to RPG are available:

1. The conversion routine is written on the Calculation Specifications form.

2. The conversion routine can be written as an independent routine that must be combined with the generated object program.

The first specification of the conversion routine defines the type of conversion by means of the operation codes RPGCV or EXTCV. If the conversion routine is coded on the Calculation Specifications form, RPGCV is specified. If the conversion routine is external to the RPG language, EXTCV is specified.

External Conversion Routine

If the conversion routine is external to the RPG language, Factor 2 must contain the name (or label) of the user-supplied routine. This external conversion routine must follow the same conventions which are specified for the EXIT routine. See *Using Tables and Exit Routines in the Object Module, Use of Registers* in this publication.

The Result Field must contain the name (or label) of the field in the subroutine which will contain the track address to be retrieved. This is the result of the conversion.

Defining the Key or ID Field

Regardless of whether an RPG or an external conversion routine is specified, if record key retrieval is used (rather than record ID), the next calculation specification entry must define the field that will contain the record key.

The operation code is KEYCV, (a K in position 31 of the File Description Specifications form). The Result Field must contain the name (or label) of the field which will contain the actual record key used to locate the record.

If record ID retrieval is used, an operation code entry is not required. However, the File Description Specifications form must contain an I in position 31.

In either key or ID retrieval, the Result Field of EXTCV or RPGCV must be a relative track address in the form TTR.

Conversion Operation Codes

The operation codes which follow are used in conjunction with conversion routines. If there are several conversion routines in the program, these codes are repeated.

1. RPGCV. This operation code indicates that the conversion routine is coded on the RPG Calculation Specifications form.

2. ERPGC. This entry terminates the RPG conversion step entries that have been coded on the Calculation Specifications form.

3. EXTCV. This entry indicates that the conversion routine is supplied by the user in a separate subroutine which is external to the RPG language.

4. KEYCV. This entry declares that the field specified in Result Field will contain the record key to be used with the relative track address. This entry must follow the RPGCV or EXTCV entry.

In the example shown in Figures 155 and 156 the data supplied in the RA file is both the record key and the data to be converted. The conversion routine shows how this field is then separated into two elements.

The field CONTD contains the 14-character field from the RA file. The first nine characters contain the customer name which is used as the key. The remaining five characters contain a code for calculating the track address of the customer's record. Line 04 of the Calculation Specifications form (Figure 156) moves the first part of CONTD to the key field (KEYFLD) and line 05 moves the remaining part to the work field (WORKFD). The alternating table on TABFIL is used to convert this 5-character code to the 3-character relative track address that is moved to the field TRKADR.

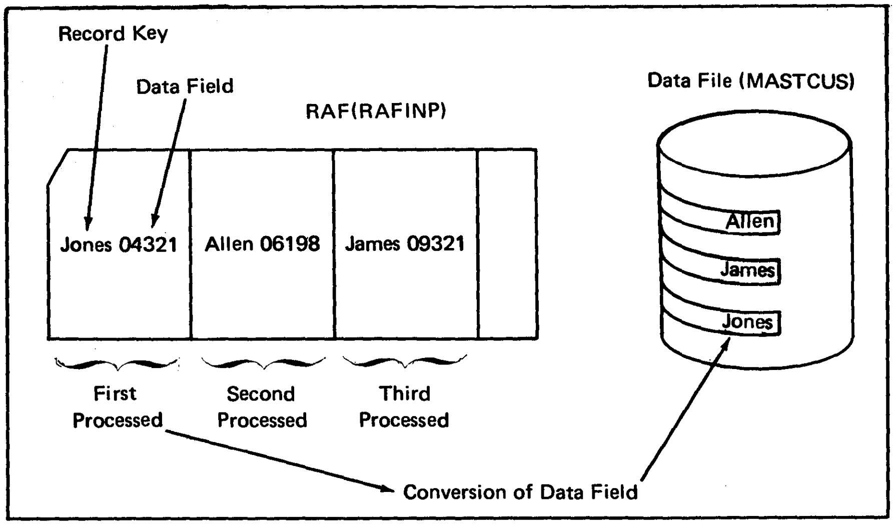
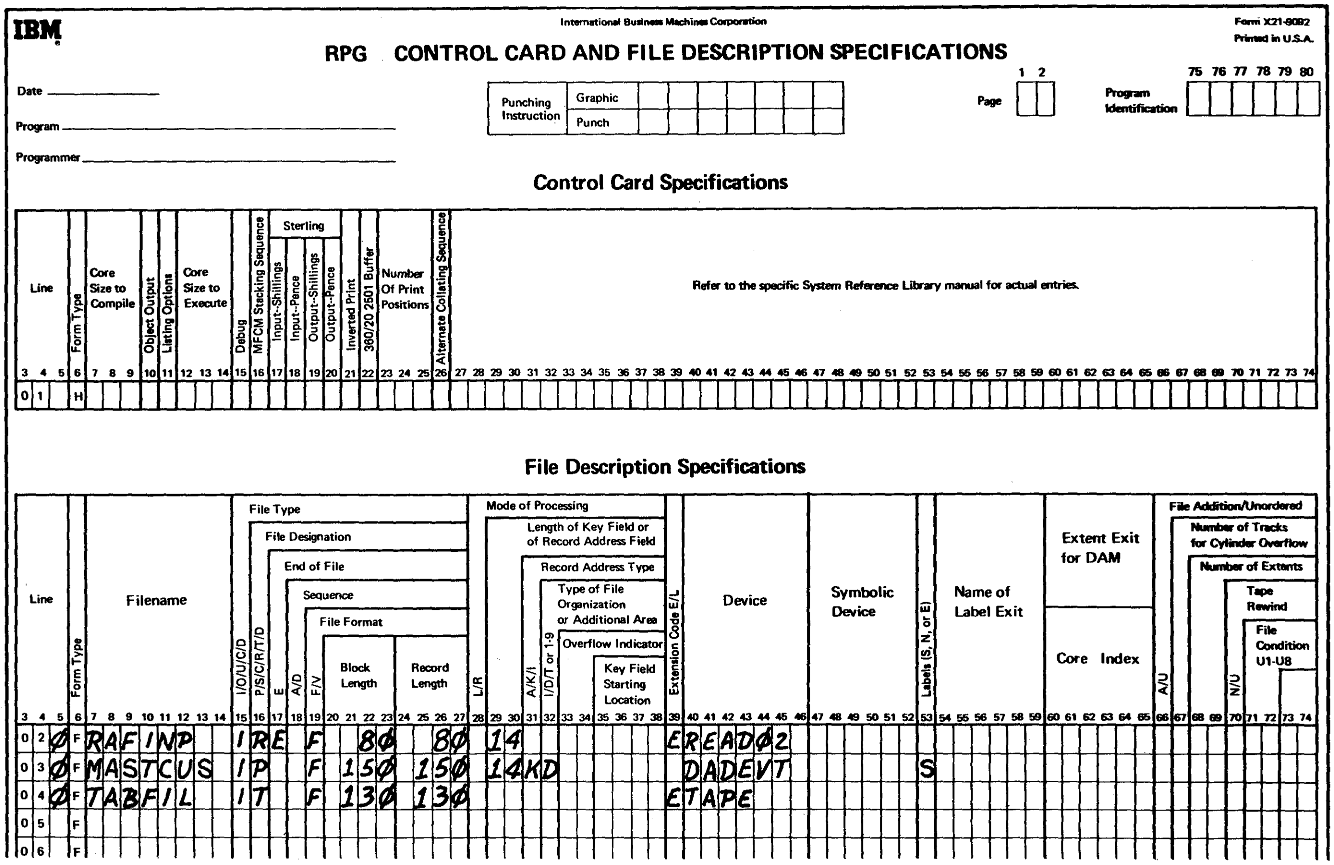


Figure 155. Conversion of a Record Address File



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 1 2

Program Identification 75 76 77 78 79 80

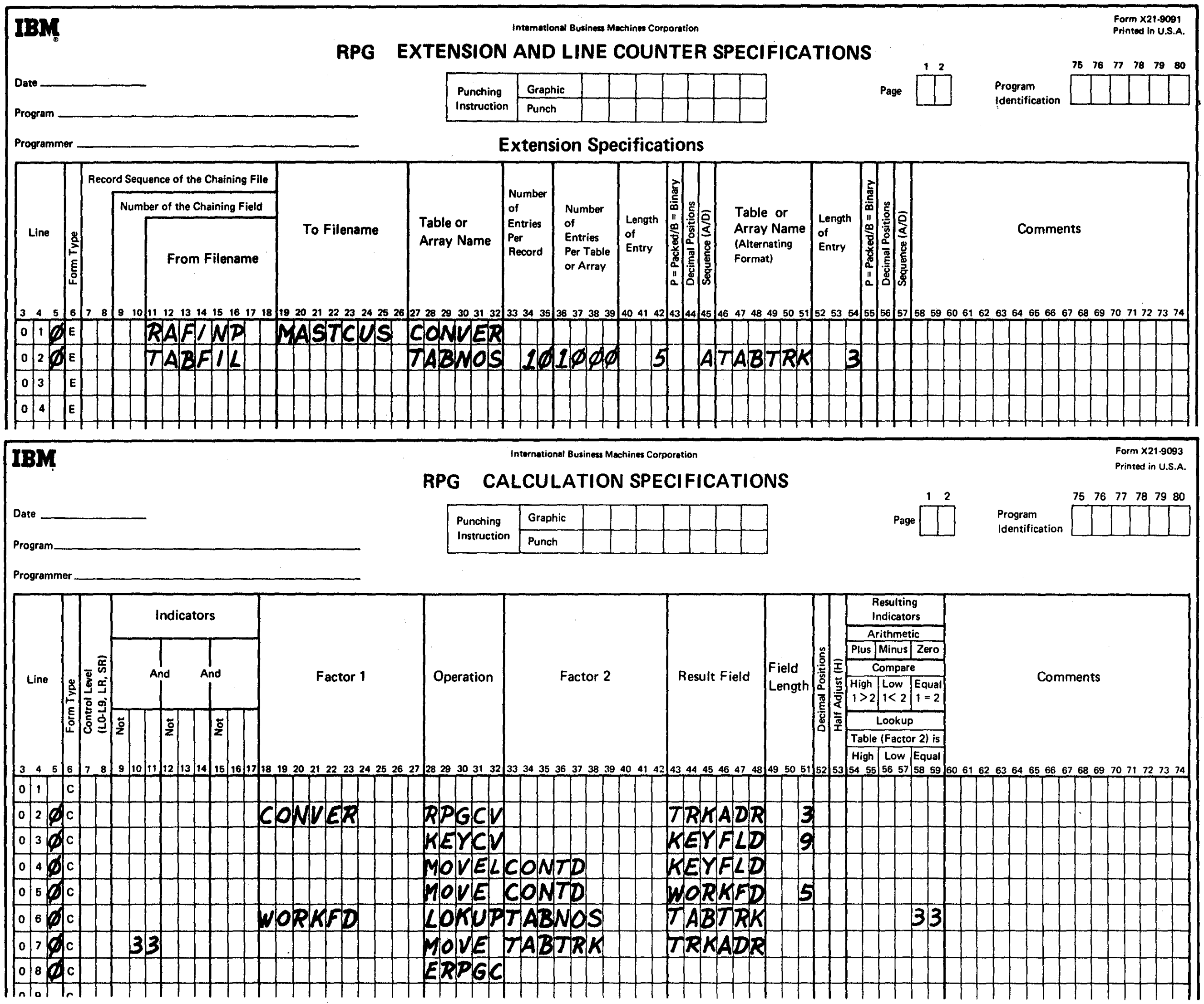
Control Card Specifications

Line (3-5)	Form Type (6)	Core Size to Compile (7-9)	Object Output (10)	Listing Options (11)	Core Size to Execute (12-14)	Debug (15)	MFCM Stacking Sequence (16)	Sterling: Input-Shillings (17)	Input-Pence (18)	Output-Shillings (19)	Output-Pence (20)	Inverted Print (21)	360/20 2501 Buffer (22)	Number Of Print Positions (23-25)	Alternate Collating Sequence (26)	Refer to the specific System Reference Library manual for actual entries. (27-74)
01	H															

File Description Specifications

Line (3-5)	Form Type (6)	Filename (7-14)	File Type: I/O/U/C/D (15)	File Designation: P/S/C/R/T/D (16)	End of File: E (17)	Sequence: A/D (18)	File Format: F/V (19)	Block Length (20-23)	Record Length (24-27)	L/R (28)	Mode of Processing: Length of Key Field or of Record Address Field (29-30)	Record Address Type: A/K/I (31)	Type of File Organization or Additional Area: I/D/T or 1-9 (32)	Overflow Indicator (33-34)	Key Field Starting Location (35-38)	Extension Code E/L (39)	Device (40-46)	Symbolic Device (47-52)	Labels (S, N, or E) (53)	Name of Label Exit (54-59)	Extent Exit for DAM / Core Index (60-65)	A/U (66)	File Addition/Unordered: Number of Tracks for Cylinder Overflow (67-68)	Number of Extents (69)	N/U (70)	Tape Rewind (71-72)	File Condition U1-U8 (73-74)
02Ø	F	RAFINP	I	R	E		F	8Ø	8Ø		14					E	READØ2										
03Ø	F	MASTCUS	I	P			F	15Ø	15Ø		14	K	D				DADEVT		S								
04Ø	F	TABFIL	I	T			F	13Ø	13Ø							E	TAPE										
05	F																										
06	F																										

Figure 156. Specifying Conversion on the Calculation Form (Part.1 of 2)



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 1 2 — Program Identification 75 76 77 78 79 80

Extension Specifications

Line (3-5) | Form Type (6) | Record Sequence of the Chaining File (7-8) | Number of the Chaining Field (9-10) | From Filename (11-18) | To Filename (19-26) | Table or Array Name (27-32) | Number of Entries Per Record (33-35) | Number of Entries Per Table or Array (36-39) | Length of Entry (40-42) | P = Packed/B = Binary (43) | Decimal Positions (44) | Sequence (A/D) (45) | Table or Array Name (Alternating Format) (46-51) | Length of Entry (52-54) | P = Packed/B = Binary (55) | Decimal Positions (56) | Sequence (A/D) (57) | Comments (58-74)

```
3456789012345678901234567890123456789012345678901234
01ØE    RAF/NP  MASTCUS CONVER
02ØE    TABFIL          TABNOS 1Ø1ØØØ  5  ATABTRK  3
03 E
04 E
```

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 (3-5) | Form Type (6) | Control Level (L0-L9, LR, SR) (7-8) | Indicators: Not (9-11), And Not (12-14), And Not (15-17) | 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 Minus Zero; Compare High 1>2 Low 1<2 Equal 1=2; Lookup Table (Factor 2) is High (54-55) Low (56-57) Equal (58-59) | Comments (60-74)

```
345678901234567890123456789012345678901234567890123456789
01 C
02ØC           CONVER    RPGCV          TRKADR  3
03ØC                     KEYCV          KEYFLD  9
04ØC                     MOVELCONTD     KEYFLD
05ØC                     MOVE CONTD     WORKFD  5
06ØC           WORKFD    LOKUPTABNOS    TABTRK          33
07ØC     33              MOVE TABTRK    TRKADR
08ØC                     ERPGC
09 C
```

Figure 156. Specifying Conversion on the Calculation Form (Part 2 of 2)

Figure 157 shows how the calculation specifications would be coded if the conversion routine were external to the RPG language.

Creating Record Address Files (RA File)

A record address file is one of the ways by which the necessary information to retrieve records from non-sequential files is supplied to the RPG program. Two types of record address files may be used:

1. For random processing of a file with indexed-sequential organization or of a file with direct organization.

2. For sequential processing between limits of a file with indexed-sequential organization.

Only one RA file may be specified for an RPG program. An RA file is processed sequentially, and it must be on a file with sequential organization. An RA file is described on the File Description and the Extension Specifications forms, but it is not described on the Input Specifications form.

Random Processing of Indexed-Sequential or Direct Organization

These rules must be followed when creating an RA file for random processing.

1. For indexed-sequential organization, the record address field contains the record key. For a direct organization, each entry in the RA file must consist of a field to be converted to the track address and to either the record key or the record ID.

2. The record addresses must begin in position 1 of the record and continue without blank spaces between the record address fields.

3. The length of the field must be the same for all records. The numeric fields must always be unpacked.

4. The number of field entries in a record may vary. A blank field, which is equal in length to the record address field, causes the RPG program to read the next record in the RA file.

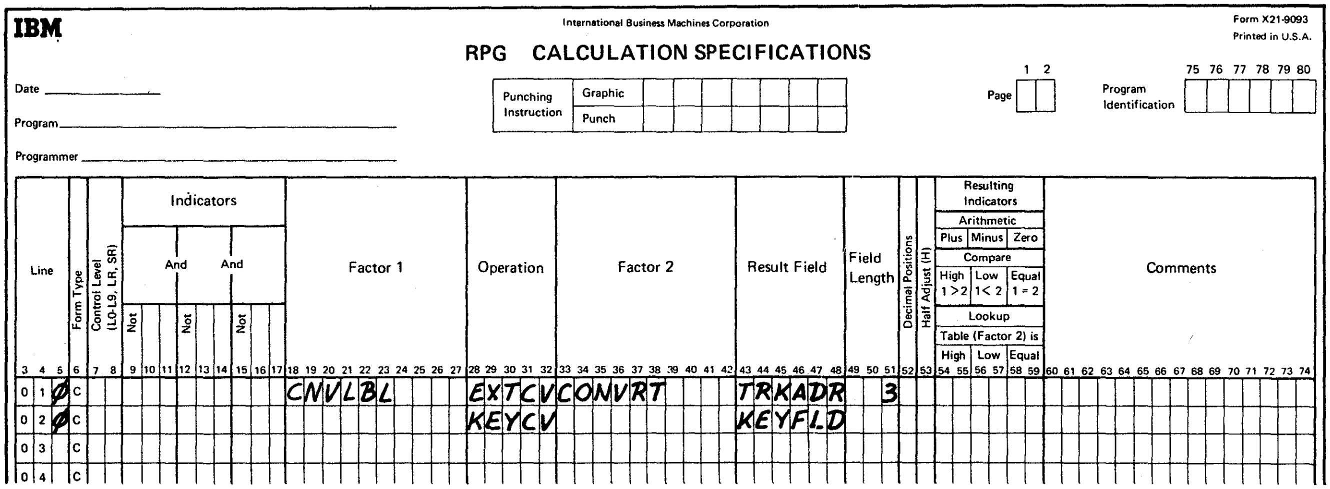
Processing Limits of an Indexed-Sequential Organization

When an RA file is used to indicate what limits of a file (with indexed-sequential organization) are to be processed, the following rules must be observed:

1. Only two record address entries can be in each record.

2. The record address entry must begin in position 1 of the record. The first entry indicates the low limit of the file to be processed. The second entry indicates the upper limit of the file. The program processes from the lower limit to the upper limit.

3. The second entry of the record must begin in the position immediately following the first entry. No blank spaces are allowed.



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 (3-5)	Form Type (6)	Control Level (L0-L9, LR, SR) (7-8)	Indicators: Not / And Not / And Not (9-17)	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 Minus Zero / Compare High 1>2 Low 1<2 Equal 1=2 / Lookup Table (Factor 2) is High Low Equal (54-59)	Comments (60-74)
01Ø	C			CNVLBL	EXTCV	CONVRT	TRKADR	3				
02Ø	C				KEYCV		KEYFLD					
03	C											
04	C											

Figure 157. Specifying a Conversion Routine Which is Not on the Calculation Form

Multiple input files may be processed using either the matching record technique or chaining. These two concepts are discussed here.

SEQUENTIAL PROCESSING OF MULTIPLE INPUT FILES (MATCHING)

The Matching Fields entry, (entered in positions 61-62 on the Input Specifications form) determines which records of a secondary file are to be processed. Both primary and secondary files are processed sequentially. Processing always begins with the primary file. The first record of each secondary file is then read in the same order as the file is specified in the input specifications (refer to rule 4). The following rules apply to the Matching Fields entry.

1. There can be three matching field specification entries (M1, M2, and M3) per record.

2. The locations of the matching fields within a record type of a file must remain fixed.

3. When there is more than one record type in an input file, the locations of the matching fields in the various types need not be the same.

4. Not all the record types in a file must have a matching field. When processing begins on any file with no matching fields specified, that entire file is read before a following secondary file is processed.

5. If M1, M2, and M3 are specified in the primary file, M1, M2, and M3 must be specified in the secondary file. Incorrect results will be obtained if the same number of matching fields is not specified. For example, Figure 158 shows three record types. Record type AA has two matching fields, and record type BB has two matching fields. If record type BB has three fields, incorrect results would be obtained. Record type CC has no matching fields (refer to rule 4).

The following example may be used to illustrate how the Matching Fields specification is used in conjunction with primary and secondary files. Assume that two files (in sequential organization) are used as shown in Figure 159. The primary file has records which contain heading and rate information. The secondary file contains detailed information which supplements the primary file.

The input specifications required to match these records are shown in Figure 160.

Specifications M1 and M2 cause each detail record to be compared against the primary file's record that has just been read. The fields DIVSON and DETDIV in both files are identified by M2; the field department (DEPT and DETDEP) in both files is identified by M1.

Matching Record Indicator

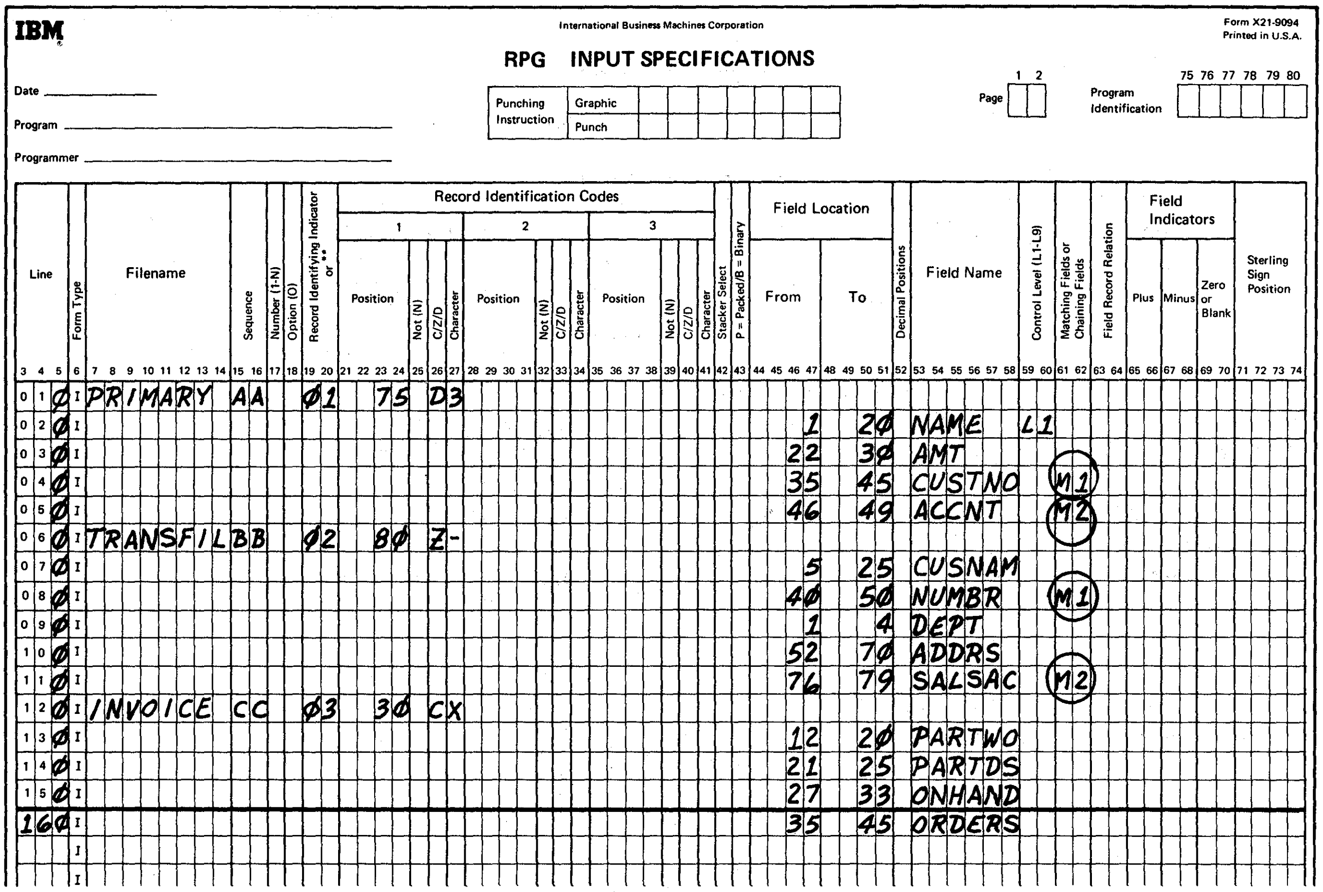
The matching field entries of M1 and M2 (and also M3) have an associated internal indicator MR (matching record). This indicator, which is similar to a record identifying indicator, is used to control functions specified on the Calculation and Output-Format Specifications forms.

The MR indicator is set on when a record of a secondary file matches a record of the primary file. It remains on during the complete processing of the record, and it is set off when all total calculations and printing that may be caused for this record are completed.

If, as in Figure 159, a detail record does not have a matching primary record (card 035) or if a master record does not have a matching detail record (card 025), the MR indicator is not set on.

This indicator can be used on the calculation specifications to prevent calculations upon a detail record contained in the secondary file. It could also be used on the output specifications to select unmatched detail cards.

The Matching Fields specification can be used even though not all of the record types in the file contain the fields used for matching. When these record types 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. These records are processed immediately after any total operations whose conditions are satisfied.



IBM

International Business Machines Corporation

RPG INPUT SPECIFICATIONS

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

Date

Program

Programmer

Punching Instruction	Graphic							
	Punch							

Page 1 2

Program Identification 75 76 77 78 79 80

							Record Identification Codes														Field Location							Field Indicators			
							1				2				3																
Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Position	Not (N)	C/Z/D	Character	Position	Not (N)	C/Z/D	Character	Position	Not (N)	C/Z/D	Character	Stacker Select	P = Packed/B = Binary	From	To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	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	PRIMARY	AA			Ø1	75		D	3																					
0 2 Ø	I																				1	2Ø		NAME	L1						
0 3 Ø	I																				22	3Ø		AMT							
0 4 Ø	I																				35	45		CUSTNO		M1					
0 5 Ø	I																				46	49		ACCNT		M2					
0 6 Ø	I	TRANSFIL	BB			Ø2	8Ø		Z	-																					
0 7 Ø	I																				5	25		CUSNAM							
0 8 Ø	I																				4Ø	5Ø		NUMBR		M1					
0 9 Ø	I																				1	4		DEPT							
1 0 Ø	I																				52	7Ø		ADDRS							
1 1 Ø	I																				76	79		SALSAC		M2					
1 2 Ø	I	INVOICE	CC			Ø3	3Ø		C	X																					
1 3 Ø	I																				12	2Ø		PARTWO							
1 4 Ø	I																				21	25		PARTDS							
1 5 Ø	I																				27	33		ONHAND							
16Ø	I																				35	45		ORDERS							
	I																														
	I																														

Figure 158. Matching Fields Entries

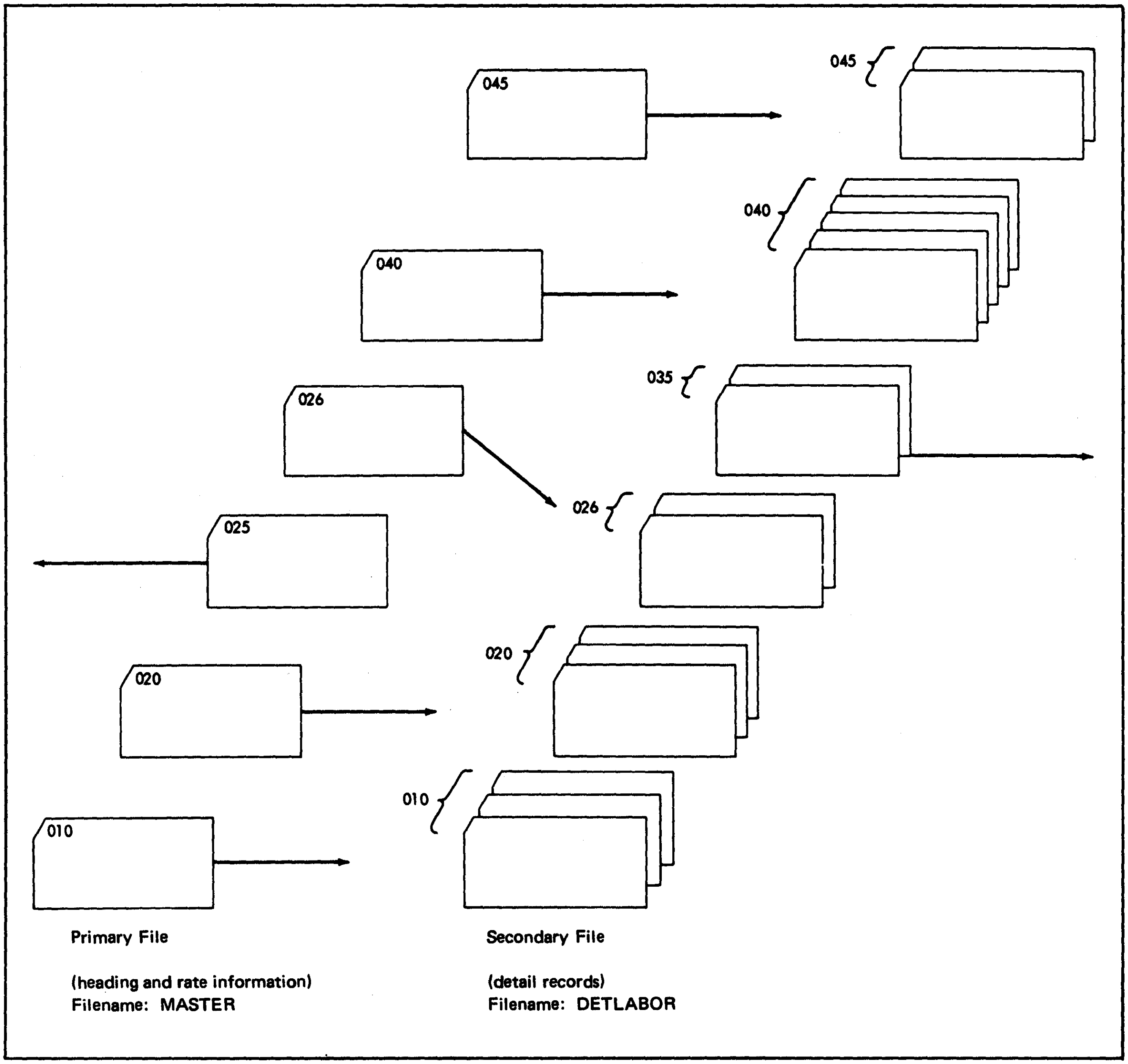
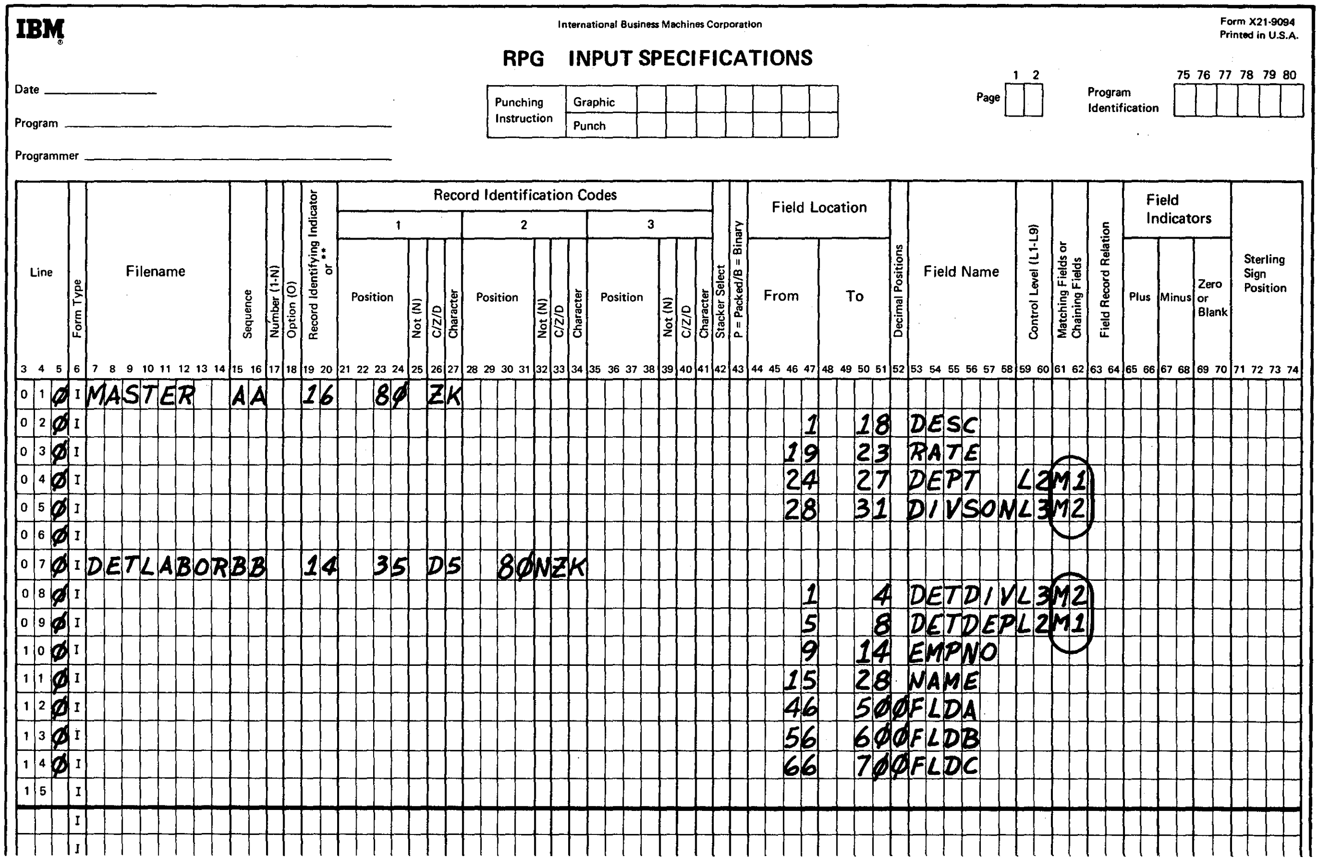


Figure 159. Two Input Files with Matching Records



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)

Program Identification (75 76 77 78 79 80)

							Record Identification Codes													Field Location							Field Indicators			
							1				2				3																
Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Position	Not (N)	C/Z/D	Character	Position	Not (N)	C/Z/D	Character	Position	Not (N)	C/Z/D	Character	Stacker Select	P = Packed/B = Binary	From	To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Plus	Minus	Zero or Blank	Sterling Sign Position
01Ø	I	MASTER	AA			16	8Ø		Z	K																					
02Ø	I																				1	18		DESC							
03Ø	I																				19	23		RATE							
04Ø	I																				24	27		DEPT	L2	M1					
05Ø	I																				28	31		DIVSON	L3	M2					
06Ø	I																														
07Ø	I	DETLABOR	BB			14	35		D	5	8Ø	N	Z	K																	
08Ø	I																				1	4		DETDIV	L3	M2					
09Ø	I																				5	8		DETDEP	L2	M1					
10Ø	I																				9	14		EMPNO							
11Ø	I																				15	28		NAME							
12Ø	I																				46	5Ø	Ø	FLDA							
13Ø	I																				56	6Ø	Ø	FLDB							
14Ø	I																				66	7Ø	Ø	FLDC							
15	I																														
	I																														
	I																														

Figure 160. Specifying Matching Fields

Matching Fields Specified as Numeric or Alphameric

Numeric

A matching record field specified as numeric by an entry in Decimal Positions but in unpacked format (blank in Packed Field) is packed before the comparison of matching fields is made. The sign of the packed field is plus with a C regardless of the sign of the input field.

Alphameric

If a field used as a matching record field has been specified as an alphameric field (blank decimal positions), no zone bits are removed from the record before the comparison is made.

Sequence of Assigning Matching Fields

One, two, or three fields can be matched in one operation. However, if more than one field is matched, the designations of M3, M2, and M1 must be assigned in the same sequence in which the fields are to be arranged for matching. M3 is assigned to the highest-order field, M2 to the next lower-order field, and M1 to the lowest-order field.

For example, in Figure 160, only two fields are matched. M1 is assigned to DEPT which is the low-order matching field, and M2 is assigned DIVSON, which is the high-order matching field.

The position in the record of the fields to be matched does not have to be the same in both files. For example, in Figure 160, the field DETDIV is located in positions 1-4 of the detail records, and the field DIVSON is located in positions 28-31 of the rate-header records.

Order of Processing Matched Records

Figure 161 illustrates a primary and a secondary file. The records in the two files are processed according to four possibilities:

1. Whenever there is a matching primary record.
2. Whenever there is a matching secondary record.
3. Whenever there is an unmatched primary record.
4. Whenever there is an unmatched secondary record.

In Figure 161, indicator 01 is set on whenever there is a primary record. Indicator 02 is set on whenever there is a secondary record. Thus,

1. A matching primary record is coded 01 and MR.
2. A matching secondary record is coded 02 and MR.
3. An unmatched primary record is coded 01 and NMR.
4. An unmatched secondary record is coded 02 and NMR.

The order in which the primary file and the secondary file are processed is shown below. *Sample Programs, Sample Program Two* uses the matching field specifications.

Primary File		*Secondary File*	
Matching Field in the *Record*	*Processed*	Matching Field in the *Record*	*Processed*
001	1st	001	3rd
001	2nd	001	4th
002	5th	002	6th
004	9th	003	7th
004	10th	003	8th
006	12th	005	11th

If more than one secondary file is specified, the secondary files will be processed in the sequence they are specified in the Input Specifications form.

Note: Input Specifications for each file must be in the same sequence as specified on the File Description Specifications form.

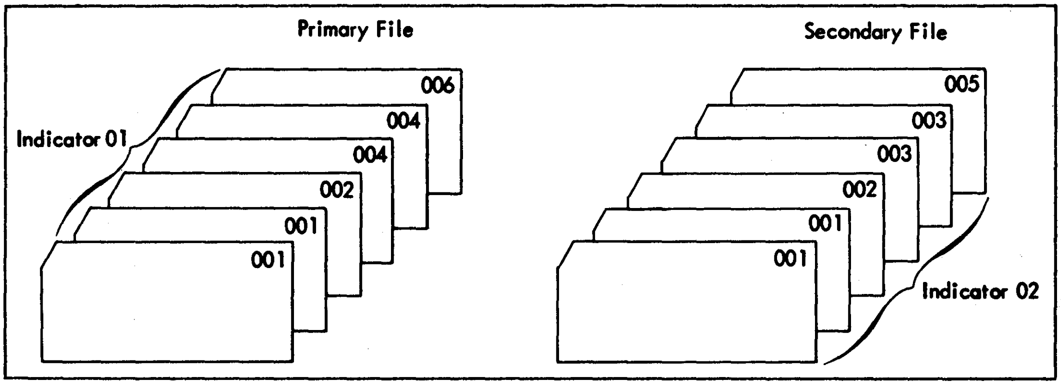


Figure 161. Order of Processing Matched Records

RANDOMLY PROCESSING MULTIPLE INPUT FILES (CHAINING)

To understand chaining, assume that an input file, as shown in Figure 162, contains information about transactions made with several customers. The card file contains the customer's number, but it does not contain his name, address, or balance. Another file called MASTCUS (master customer file) contains information about each customer. The RPG object program has the ability to 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 record key in each disk record contains the customer's numbers. The field in the transaction file, which contains the customer's number, can be used to chain the files together. The object program takes the customer's number and locates the record with the same record key. The additional information, such as the customer's name, address, balance, etc., is associated with each record key; it is immediately available for processing.

The field which links or chains a record of one file to a record in another file is called a chaining field. The transaction file (CARDIN) is called the chaining file. The master customer file (MASTCUS) is the chained file because it is linked to the transaction file.

Up to nine chaining fields may be specified. The chaining fields can be located in one or more files. The chaining fields are designated by entering C1 through C9 in positions 61-62 of the Input Specifications form.

Note: There is no specific relationship between levels C1-C9 other than specifying the nine possibilities for chaining fields.

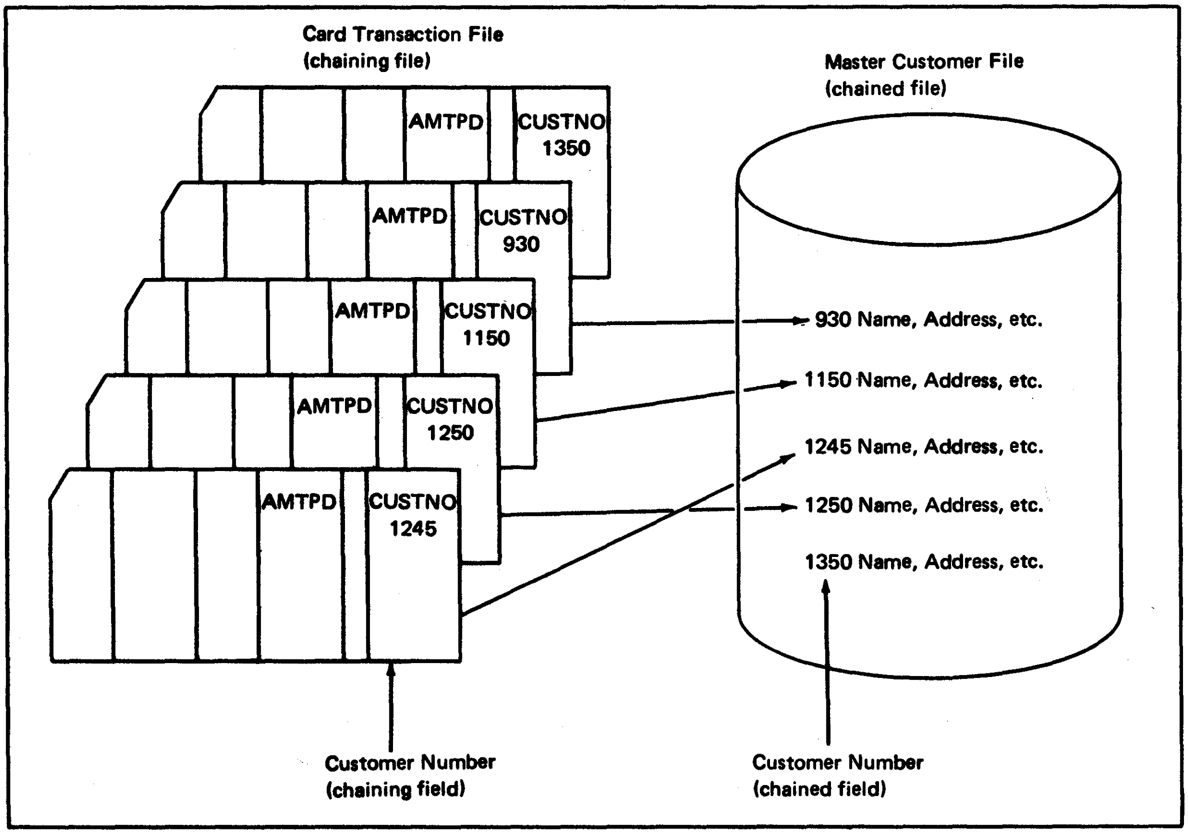


Figure 162. Using Chaining to Process an Indexed-Sequential File

Chaining Example

Figure 163 illustrates coding used for chaining the the transaction file CARDIN to the customer file MASTCUS.

File Description Specifications

On the File Description Specifications form, the card file is considered the primary file. When CARDIN is depleted, the program goes to the end-of-job routine (E in position 17).

Extension Specifications

The Extension Specifications form contains the record sequence of the CARDIN file and the number of the chaining field.

Input Specifications

On the Input Specifications form, the two files are defined. CUSTNO is the field which chains the files.

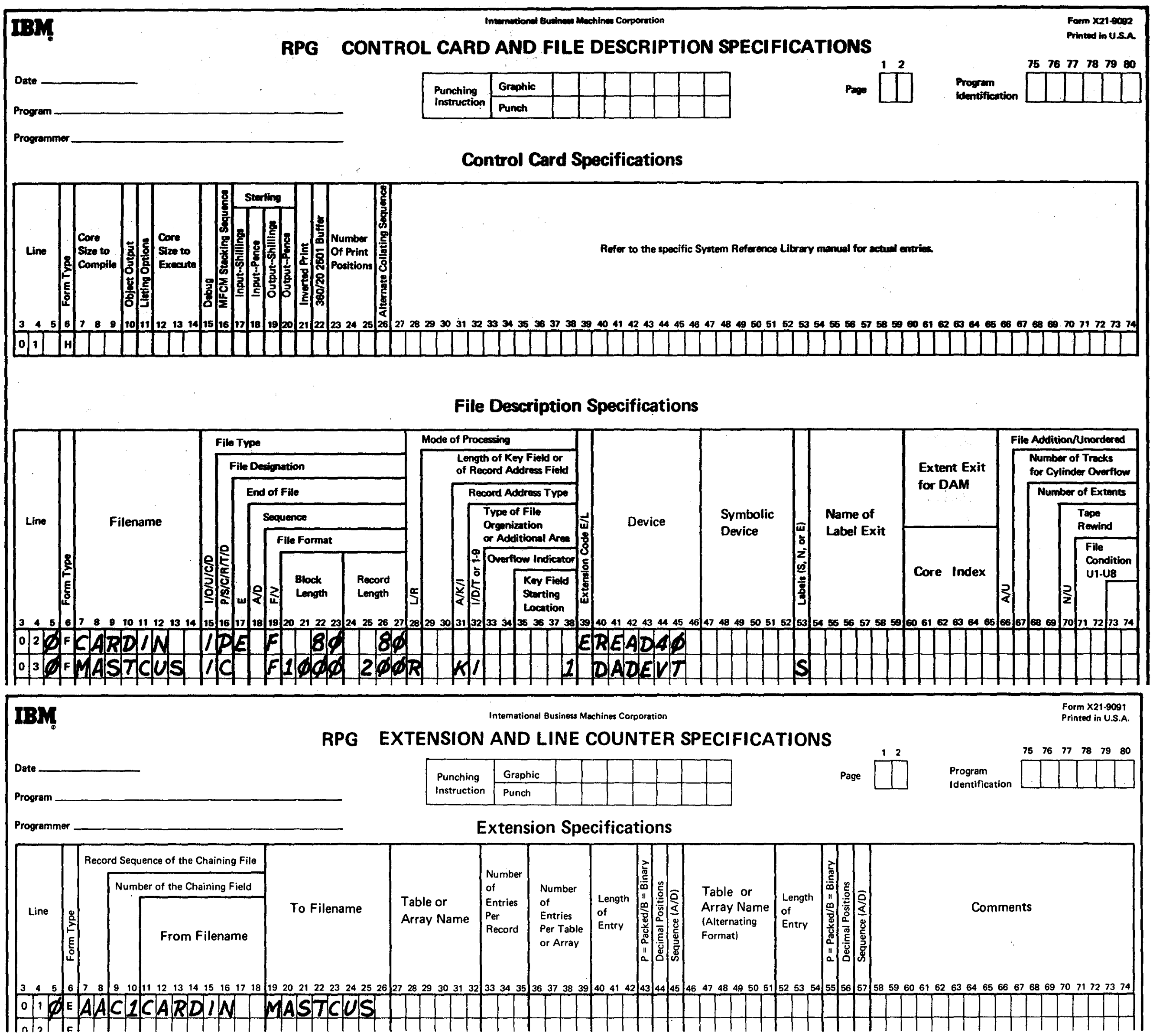
Note: Numeric chaining fields can only be used if the record keys of an indexed-sequential file are in packed format (see *Input Specifications Form, Packed* and *Output-Format Specifications Form, Packed Field*).

Calculation Specifications

When chaining is used, the following situation may occur. A chaining record (for example, JONES) has no corresponding chained record. (JONES is not present in the chained file.) Such a situation may require special action by the program.

Indicators 01 and 02 are both on when a chaining record has a corresponding chained record. In this case 01 represents the chaining record, and 02 represents the chained record. When 01 and 02 are both on, AMPTD is subtracted from BALANC. However, if there is no corresponding record in the chained file (01N02), halt indicator H1 is set on. Thus, the possibility of not having a corresponding record in the chained file is accounted for, and processing will terminate when this situation occurs.

Note: This example illustrates the only time that two record identifying indicators (representing two different record types)can be on at the same time.



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 — Program Identification 75 76 77 78 79 80

Control Card Specifications

Line | Form Type | Core Size to Compile | Object Output | Listing Options | Core Size to Execute | Debug | MFCM Stacking Sequence | Sterling: Input-Shillings, Input-Pence, Output-Shillings, 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: I/O/U/C/D | File Designation: P/S/C/R/T/D | End of File: E | Sequence: A/D | File Format: F/V | Block Length | Record Length | Mode of Processing: L/R | Length of Key Field or of Record Address Field | Record Address Type: A/K/I | 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 | Core Index | A/U | File Addition/Unordered | Number of Tracks for Cylinder Overflow | Number of Extents | Tape Rewind: N/U | File Condition U1-U8

```
02ØFCARDIN  IPE F  8Ø  8Ø           EREAD4Ø
03ØFMASTCUS IC  F1ØØØ 2ØØR  KI     1 DADEVT       S
```

IBM International Business Machines Corporation Form X21-9091 Printed in U.S.A.

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

Date ______ Program ______ Programmer ______

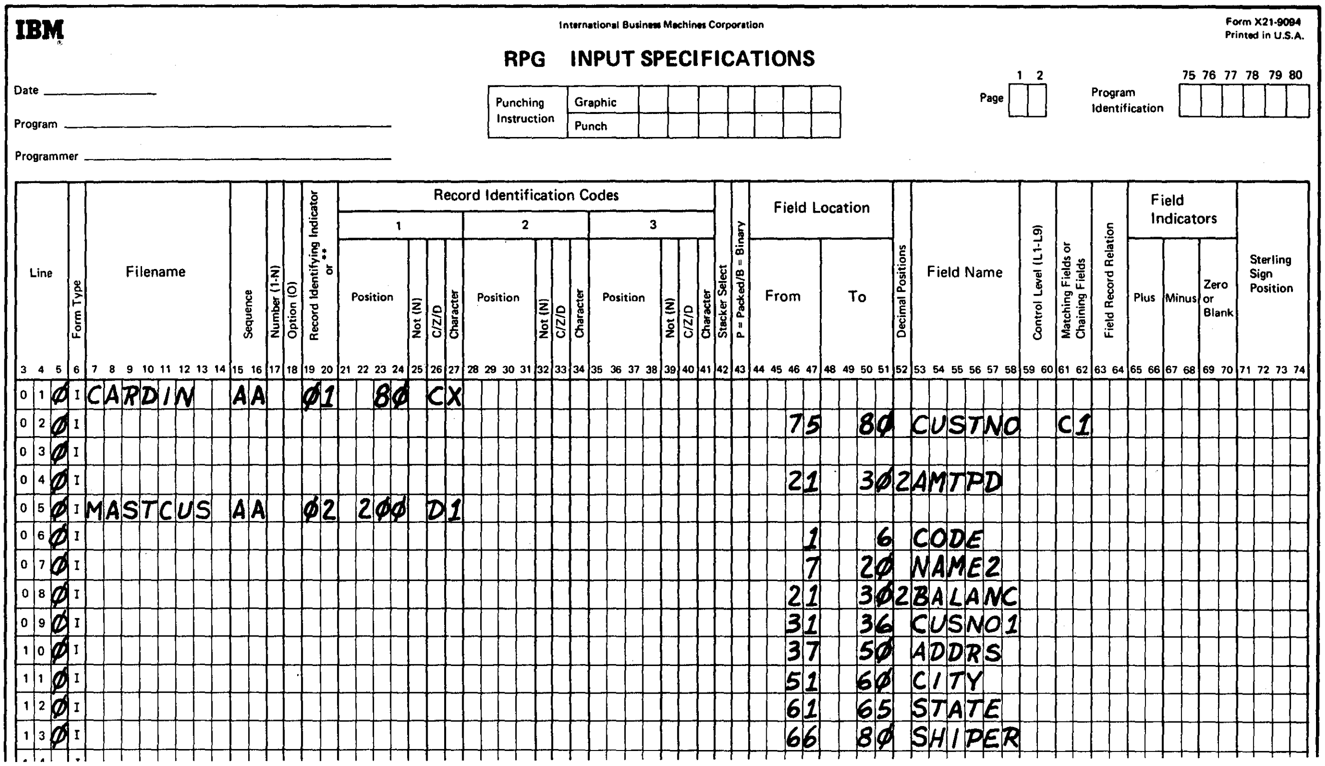
Punching Instruction: Graphic / Punch — Page 1 2 — Program Identification 75 76 77 78 79 80

Extension Specifications

Line | Form Type | Record Sequence of the Chaining File | Number of the Chaining Field | From Filename | To Filename | Table or Array Name | Number of Entries Per Record | Number of Entries Per Table or Array | Length of Entry | P = Packed/B = Binary | Decimal Positions | Sequence (A/D) | Table or Array Name (Alternating Format) | Length of Entry | P = Packed/B = Binary | Decimal Positions | Sequence (A/D) | Comments

```
01ØEAAC1CARDIN  MASTCUS
02 E
```

Figure 163. Specifying Chaining (Part 1 of 2)



IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date ______

Program ______

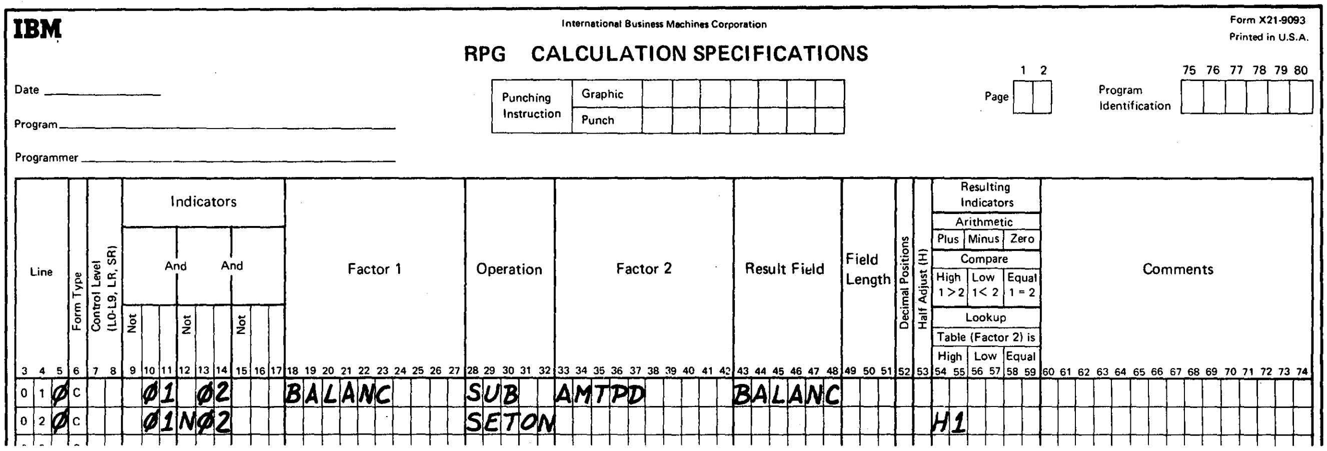
Programmer ______

Punching Instruction: Graphic | Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	Not (N)	C/Z/D	Character	2: Position	Not (N)	C/Z/D	Character	3: Position	Not (N)	C/Z/D	Character	Stacker Select	P = Packed/B = Binary	Field Location: From	To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank	Sterling Sign Position
01Ø	I	CARDIN	AA			Ø1	8Ø		C	X																					
02Ø	I																				75	8Ø		CUSTNO		C1					
03Ø	I																														
04Ø	I																				21	3Ø	2	AMTPD							
05Ø	I	MASTCUS	AA			Ø2	2ØØ		D	1																					
06Ø	I																				1	6		CODE							
07Ø	I																				7	2Ø		NAME2							
08Ø	I																				21	3Ø	2	BALANC							
09Ø	I																				31	36		CUSNO1							
10Ø	I																				37	5Ø		ADDRS							
11Ø	I																				51	6Ø		CITY							
12Ø	I																				61	65		STATE							
13Ø	I																				66	8Ø		SHIPER							



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: Not		And: Not		And: Not		Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Plus / High 1>2	Minus / Low 1<2	Zero / Equal 1=2	Comments
01Ø	C			Ø1		Ø2			BALANC	SUB	AMTPD	BALANC							
02Ø	C			Ø1	N	Ø2				SETON						H1			

Figure 163. Specifying Chaining (Part 2 of 2)

Additional Uses of Chaining

Figures 164 and 165 show two additional uses of chaining. In Figure 164, the card file contains two fields which are both used as chaining fields (salesman number and customer number). The field containing salesman number is chained to a file that is organized by salesman number. The field containing the customer's number is used to chain to the customer file.

In Figure 165, the card file is chained to the customer file. Within each customer record is a field which may be used to chain to another file. In this case, each record in the customer file contains an account number. The account number is used to chain to the account file.

Split Chaining Fields

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, and the fields are then used as one chaining field. The fields are placed in the same sequence as they are defined on the Input Specifications form.

The first field defined on the Input Specifications form is placed in the leftmost position, and the last field is placed in the rightmost position.

Conversion of Chaining Fields for Direct File Organization

The data to be converted will be located in the fields designated as the chaining fields (Figure 166). The data field must be such that the record key or record ID can be derived from the conversion. The rules for conversion of a chaining field are the same as those for an RA file.

Figure 167 and 168 illustrate the conversion routine which converts the data contained in a chaining field (Figure 166) to the relative track address and record key of the record.

In this example, the conversion routine is coded on the Calculations Specifications form. On the File Description Specifications form, the two input files are described. TRANSREC is the primary file. MASTCUS is the DASD file which has direct organization and is processed randomly. Since in direct organization the key is separate from the data record, no entry is made in the Key Field Starting Location (positions 35-38). The conversion routine supplies the track address and the record key.

On the Extension Specifications form, the record sequence of the chaining file (AA in positions 7 and 8) is specified.

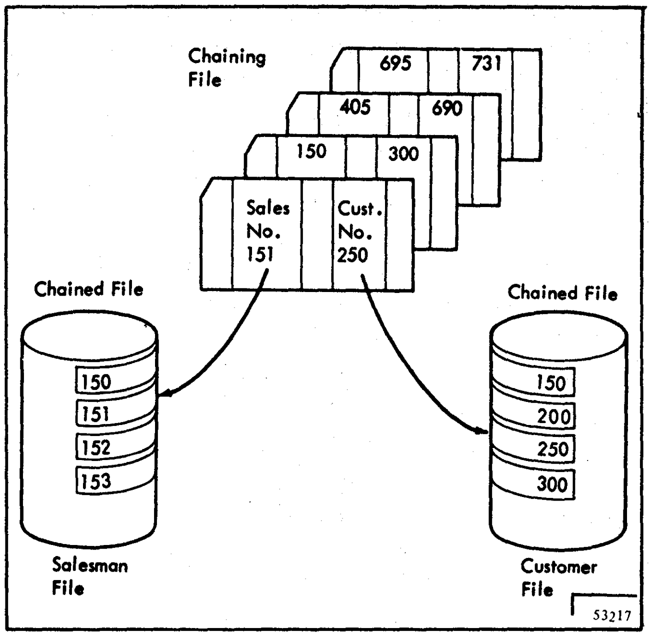


Figure 164. Chaining to Two Files

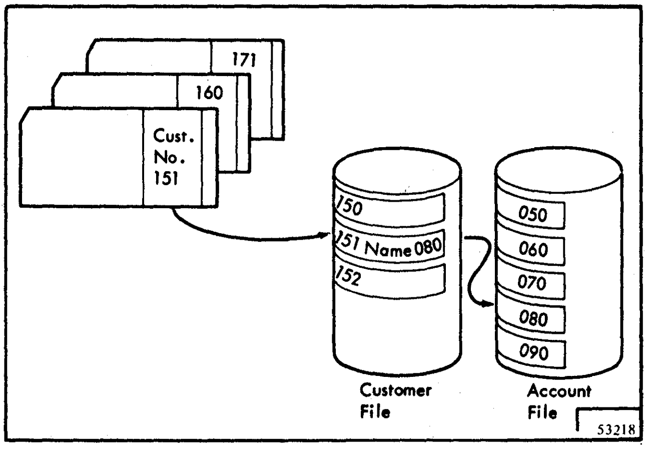


Figure 165. Using a Chained File as a Chaining File

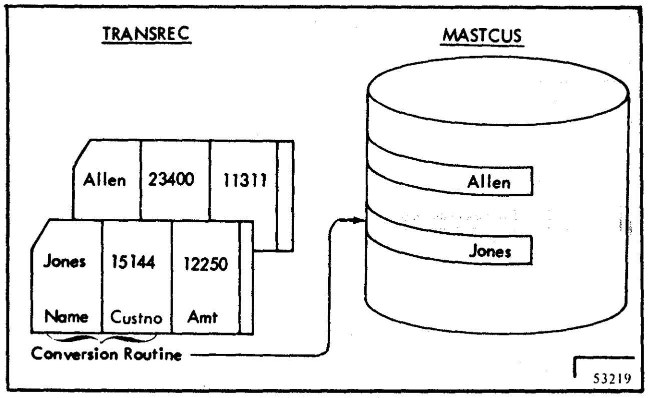
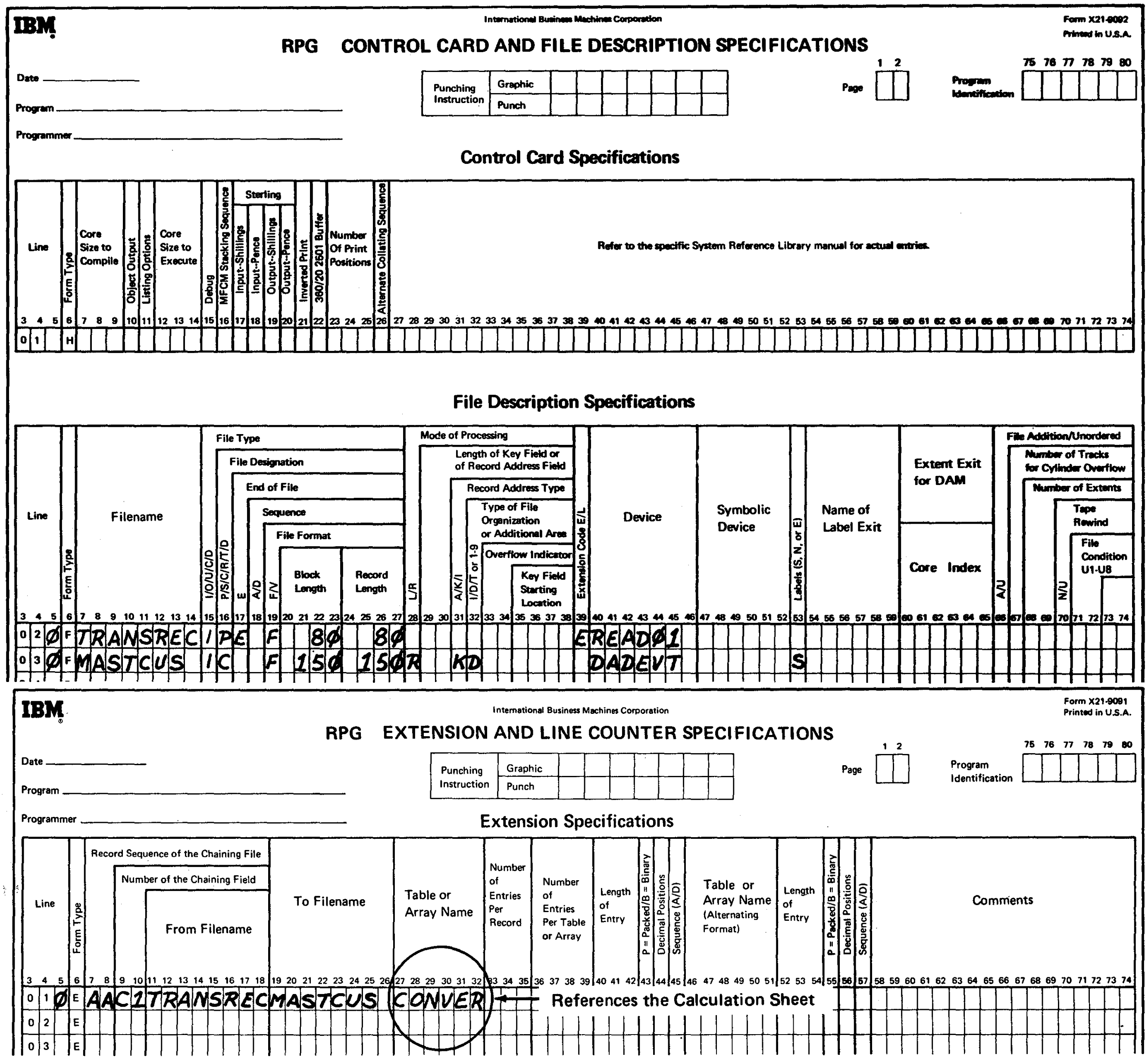


Figure 166. Specifying a Random Disk Address by Converting an Input Record (Chaining File) Position.

The number of the chaining field is C1. Both AA and C1 are taken from the Input Specifications form. TRANSREC relates to MASTCUS, so these filenames are entered in positions 11-18 and 19-26, respectively. The entry CONVER in positions 27-32 indicates that this label, when entered in Factor 1 of the Calculation Specifications form, specifies the conversion routine.

Two fields of the primary file, NAME and CUSNUM, are designated on the Input Specifications form as the chaining fields to be converted.

The conversion routine is coded on the Calculation Specifications form. The result field of the RPGCV entry (line 010) defines the field which contains the relative track address. The customer number, CUSNUM, is converted to the relative track address (lines 040-080). The entry KEYCV defines the name of the field which contains the record key. In this example, the record key contains the customer's name which is obtained directly from the input data.



IBM International Business Machines Corporation Form X21-9092 Printed in U.S.A.

RPG CONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS

Control Card Specifications

Line	Form Type
01	H

File Description Specifications

Line	Form Type	Filename	File Type	File Designation	End of File	File Format	Block Length	Record Length	Mode of Processing	Record Address Type	Type of File Organization	Extension Code	Device	Labels
02Ø	F	TRANSREC	I	P	E	F	8Ø	8Ø				E	READØ1	
03Ø	F	MASTCUS	I	C		F	15Ø	15Ø	R	K	D		DADEVT	S

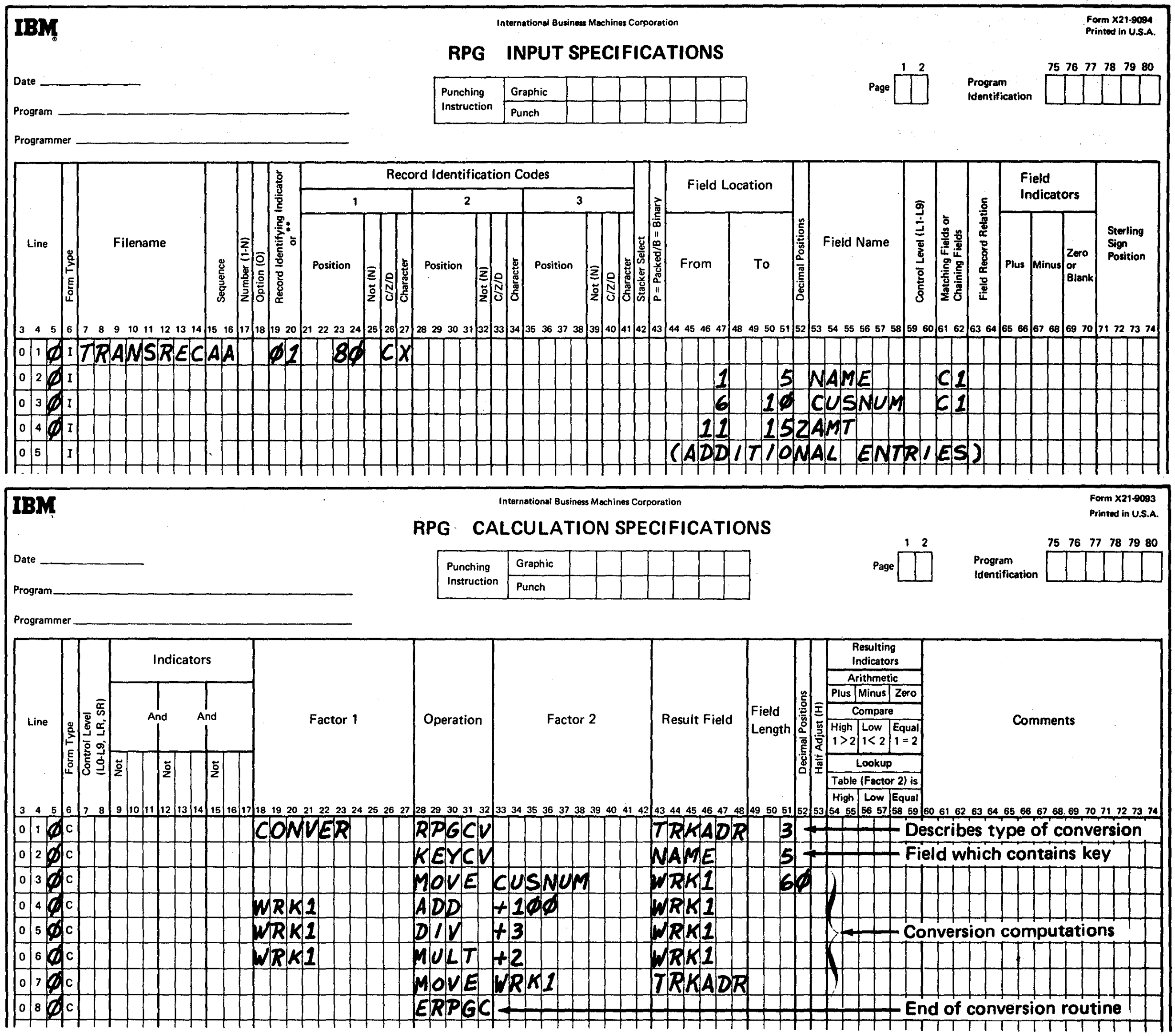
IBM International Business Machines Corporation Form X21-9091 Printed in U.S.A.

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

Extension Specifications

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table or Array Name
01Ø	E	AA	C1	TRANSREC	MASTCUS	CONVER
02	E					
03	E					

Figure 167. Conversion of a Chaining Field



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

RPG INPUT SPECIFICATIONS

Date

Program

Programmer

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Position	Not (N)	C/Z/D	Character	From	To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields
01Ø	I	TRANSREC	AA			Ø1	8Ø		C	X						
02Ø	I										1	5		NAME		C1
03Ø	I										6	1Ø		CUSNUM		C1
04Ø	I										11	15	2	AMT		
05	I										(ADDITIONAL ENTRIES)					

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

RPG CALCULATION SPECIFICATIONS

Date

Program

Programmer

Line	Form Type	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Comments
01Ø	C	CONVER	RPGCV		TRKADR	3		Describes type of conversion
02Ø	C		KEYCV		NAME	5		Field which contains key
03Ø	C		MOVE	CUSNUM	WRK1	6	Ø	Conversion computations
04Ø	C	WRK1	ADD	+1ØØ	WRK1			
05Ø	C	WRK1	DIV	+3	WRK1			
06Ø	C	WRK1	MULT	+2	WRK1			
07Ø	C		MOVE	WRK1	TRKADR			
08Ø	C		ERPGC					End of conversion routine

Figure 168. Conversion of a Chaining Field

Summary of Multiple File Processing

RPG processes multiple files two ways:

1. Sequentially by using the matching record technique.
2. Randomly by using the chaining technique.

Figure 169 shows the processing possibilities for files which have sequential, indexed-sequential, or direct organization.

The numbers 1, 2, and 3 refer to the major subjects listed in the following text. The letters A, B, and C refer to the subgroups.

		A	B	C
	To From	Sequential Organization	Indexed-Sequential Organization (DASD)	Direct Organization (DASD)
1	Sequential Organization	Sequential Processing (matching)	Sequential Processing (matching) or Random Processing (chaining)	Random Processing (chaining)
2	Indexed-Sequential Organization *	□ Sequential Processing (matching)	Sequential Processing □ (matching) or Random Processing (chaining)	Random Processing (chaining)
3	Direct (DASD) Organization +	—	Random Processing (chaining)	Random Processing (chaining)

* A record address file may be used to supply the limits (in the case of sequential processing) or the actual record keys (in the case of random processing).

□ The From File must be specified as sequential.

+ A record address file is converted to supply the record locations on the DASD.

Figure 169. Combined Negative and Positive Record Keys

Files with Sequential Organization

1. A file with sequential organization is processed sequentially and controls the processing of records in:

   a. Another sequential organization. Both files are processed sequentially, using matching records to govern processing.

   b. An indexed-sequential organization. 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.

   c. A direct organization. A direct organization is processed randomly, under control of the sequential file. The sequential file contains chaining fields which are converted to the relative track addresses of the records on the direct file.

Files with Indexed-Sequential Organization

1. A file with indexed-sequential organization may be processed sequentially or randomly, and it controls processing of records in:

   a. A sequential organization. The records in both files will be processed sequentially, using matching records to control processing.

   b. Another indexed-sequential organization. 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.

   c. A direct organization. Chaining fields in the indexed file are converted to supply the record locations in the direct file which is processed. The direct file is processed randomly.

Files with Direct Organization

1. A file with direct organization is processed randomly and controls processing of records in:

   a. An indexed-sequential organization. The indexed file is processed randomly, and chaining fields in the direct file control processing of the indexed file.

      In both cases, the direct organization is processed randomly.

   b. Another direct organization. Chaining fields contained within the direct file are converted to provide the location of the records in another direct organization. The records in both files are processed randomly.

Updating a DASD File

An RPG program may perform update processing of a DASD file. The file may be of sequential, indexed-sequential, or direct organization. The fields of records contained in the file may be changed; however, the size of the records may not be changed. Only records existing in the file may be processed. When an update file (U in position 15 on the File Description Specifications form) is processed, only the fields to be updated must be entered on the Output-Format form. Although the entire record is to be retained, only the affected fields are entered on the Output-Format form.

Creating Indexed-Sequential Files

Figure 147 illustrates the coding used for loading the indexed-sequential file DIKSOUT.

File Description Specifications: The input file CARDIN is the primary file. When CARDIN is exhausted, the end of job routine is entered. The file DISKOUT is to be loaded (0 in position 15, blank in position 66) on an IBM direct access storage device. The key area associated with each of the unblocked output records is five bytes in length (entry in positions 29-30). The keys are not within the data records (no entry in positions 35-38). Two tracks (position 67) of each cylinder allocated for data records will be reserved to contain overflow data records in the event the file is expanded.

Input Specifications: The fields to be used from the cards read from CARDIN are defined.

Output-Format Specifications: The information read from CARDIN will be placed in the file DISKOUT. The field KEY will be placed within the key area associated with the data record (K5 in positions 40-43) and will not be written as part of the data record.

Note: The key field(s) for an indexed-sequential output record must always be defined on the Output-Format Specifications form. The key always consists of one or more fields within each logical record in a blocked file. The key may be defined in this same way for an unblocked file, or it may be created in a key area separate from the logical record (see the section *RPG User's Guide for ISAM Processing* for a complete discussion of record formats).

Negative Keys

The use of negative keys in an indexed-sequential file is not recommended. Creation of a file and file processing are more difficult when negative keys are used. An indexed-sequential file's records are arranged so that the key values are in ascending collating sequence, and they must be placed into the file in this order during file creation (that is, a record with key equal to –1 must precede a record with key equal to –2). If both negative and positive keys are used in a file, the orderings in Figure 170 must be used. These key orderings are required because the indexed-sequential access method (ISAM) uses a logical compare to check for proper ascending key sequencing.

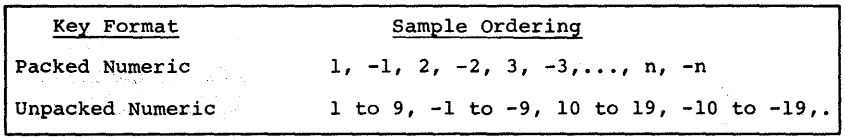
Adding Records to Indexed-Sequential Files

Figure 149 illustrates the coding which is necessary to add records to an indexed-sequential file.

File Description Specifications: The input file CARDFIL is the primary file. The output file DISKFIL will add records to an existing indexed-sequential file (0 in position 15, A in position 66) which consists of five 80-byte records per block with an 8-byte key field beginning at location one within the record (entries in positions 29-30 and 35-38). The file is on a direct access storage device.

Input Specifications: The input card file CARDFIL is described in Figure 149.

Output-Format Specifications: Each input record will cause an 80-byte record to be added, in proper key sequence, to the file DISKFIL. All reblocking of records and structuring of overflow records and areas will be handled by the RPG object program and the OS access method.



Key Format	Sample Ordering
Packed Numeric	1, -1, 2, -2, 3, -3,..., n, -n
Unpacked Numeric	1 to 9, -1 to -9, 10 to 19, -10 to -19,.

53236

Figure 170. Processing Multiple Input Files

Important points to be remembered when creating or adding to an indexed-sequential file include:

1. Records used when creating a file must be in ascending sequence (negative keys are considered to be larger than positive keys). Records to be added may have keys in random order.

2. Cylinder overflow tracks (File Description Specifications, position 67) must be specified for all load functions (between 0 and 8 for 2311; 0 and 9 for 2314/2321).

3. For unblocked files, if no key location is specified on the File Description Specifications form (positions 35-38), the key area must be defined on the Output-Format Specifications form (positions 40-43). The numeric entry specifying the end positions of a field within the key area is preceded by a K. If some portion of the key area is undefined, it will be made blank.

Processing Indexed-Sequential Files

Figure 148 illustrates coding which is used for updating and adding to a file using chaining, and coding which is used for reading a file using an RA file.

File Description Specifications: CHAINFIL is a primary tape file containing records which are used to supply keys for chaining and reading from the file ISUPADD. ISUPADD is an unblocked indexed-sequential update file. It is contained on a direct access device and is processed randomly. Records will be read from ISUPADD when a chaining key is supplied from CHAINFIL. Two types of output will be made to ISUPADD: records may be read, updated, and returned to the file (U in position 15) and/or new records may be added to the file (A inposition 66). ISUPADD's 12-position record keys are not within the logical data records (no entry in positions 35-38).

RA file supplies two 5-position keys in each of its records to define low and high limits between which records are read from ISINPUT. ISINPUT is a blocked indexed-sequential file with 5-position keys located at the start of each logical record (1 in positions 35-38). This file is on a direct access device and records will be read from it sequentially between limits.

Extension Specifications: The field identified as C1 in each of CHAINFIL's records will be used as a key to chain to a record in ISUPADD with the same key. RA file will supply records containing low and high limit keys. Records will be read sequentially from ISINPUT starting with the record having a key equal to the low limit key and ending with the record having a key equal to the high limit key.

Input Specifications: The CHAINFIL field KEY is used to chain to ISUPADD so that a record with the same key may be read (C1 in positions 61-62). For each ISUPADD record, the fields of the key area are defined as SECTOR and RATE. These fields are not within the logical data records (K entries in positions 44-51).

Calculation Specifications: The first calculation specification builds the key (NEWKEY) for the record that will be added to the ISUPADD file on the output cycle. The remainder of the calculation specifications show the building of the fields that will be placed in the updated record of the ISUPADD file. All calculations are conditioned on indicators 01 and 02, both of which will be on when a successful chain is accomplished for CHAINFIL to ISUPADD.

Note: If a no-record-found condition occurs when chaining to an indexed-sequential file, the H0 indicator is set on and processing is terminated at the end of the cycle. To continue processing, H0 must be set off by the calculation specifications.

Output-Format Specifications: Two types of output occur for ISUPADD on every output cycle for which the conditioning indicators are on: old records are updated and new records are added when indicators 01 and 02 are on. Key area fields may only be specified for added records (K in positions 40-43).

Note: The following example shows how to code a program so that one record in an indexed-sequential file is being updated and a new record is being added during the same object program cycle. In this instance, the object program operates most efficiently if the output specifications for the updated record preceded those for the added record, as shown by this example.

RPG enables the programmer to branch from detail calculations to total calculations causing the generated program to repeat certain specifications. This repetition is called a detail-to-total loop. Essentially, this gives the programmer the facility to put out one or more total lines for each input record. Each total line may vary in content according to program specifications and may be put out during a detail-to-total loop, depending on its output indicators.

Certain restrictions are imposed on the use of the detail-to-total branch by the flow of logic used by RPG. These restrictions should be understood by the programmer before using the detail-to-total branch. A detailed example is provided by Figures 171, 172, and 173.

Data Movement From Input Area To Field Locations

Each time a detail-to-total loop is executed, data items are moved from the input area to their respective field locations. If an input field is to be used as a result field and changed by calculations during a detail-to-total loop, the changes in the Result Field will be overlaid when the input fields are reinitialized to their input area values. Therefore, a new field should be defined and the input item moved to this new field, which can be then used as a Result Field. These results can thus be protected.

An indicator can be used to allow the move from an input field to a new field only once, usually during the normal flow of logic before a detail-to-total branch. This indicator can be used to prevent the new field from being overlaid with the reinitialized input field.

Suppression of Normal Activity

Before performing a detail-to-total branch, total calculations, total lines, detail calculations, and detail lines may have to be suppressed to get the desired results. This may be done by conditioning lines and calculations with indicators. One such suppression might be to allow data to be moved from an input field to a protected field only once.

Branching

The branch from detail-to-total calculations is accomplished by a GOTO specification in detail calculations and a TAG in total calculations. Branching is controlled by conditioning the GOTO specification on an indicator. This indicator is set on during detail calculations to initiate the detail-to-total branch. The results of a total calculation performed during the detail-to-total loop may be used to set off this indicator and thus discontinue the loop.

The indicators used to control a detail-to-total branch must be specified with care. These indicators must be specified so that they initiate the detail-to-total branch; but when the desired results of the detail-to-total loop have been obtained, these indicators must inhibit the branch from further initiation. The generated program must resume normal logic flow and read the next input record. Failure to control the detail-to-total loop in this manner may result in an infinite repetition of the detail-to-total branch.

Output

The sequence of output lines produced by the generated program is altered by the use of the detail-to-total branch:

1. Total lines produced during a detail-to-total loop are put out first.

2. Detail lines conditioned for output at detail time follow.

3. Finally, all normal total lines (not conditioned as part of the detail-to-total loop) are put out according to their indicators.

4. If the overflow indicator is set on while printing total lines in a detail-to-total loop, all lines conditioned on the overflow indicator will be put out before that indicator is set off.

Example

In the following example (Figures 171, 172, and 173) seven fields are read on input. They are: NAME, ADDRES, LOCATE, MONTH, INITIL, NUMBER, and INCRMT. The desired output is a set of mailing labels, addressed NAME, ADDRES, and LOCATE. These labels are dated. The first label bears the data MONTH, INITIL, where INITIL is the day of the month. The day of the month on subsequent labels is incremented by INCRMT. The field NUMBER determines how many labels are produced. The following numbered steps refer to circled numbers on Figures 171, 172, and 173.

1. Heading lines are printed, the first input record is read, and its type determined.

2. Total calculations are suppressed as they are conditioned on indicator 77, which is on only during a detail-to-total loop in this example.

3. Data is moved from the input area to input fields.

4. Input data is moved from the input fields to new fields and protected by indicator 77. (The move is performed only when indicator 77 is off, which is true only during normal detail time.)

5. Results are formed in the new fields and protected by indicator 77.

6. Indicator 77 is set on, conditioning the detail-to-total branch and protecting the new fields.

7. A detail time test is made to decide whether to perform the detail-to-total branch or discontinue the loop.

8. When the result of the test is low, indicator 88 is set on and a detail-to-total branch is performed.

9. Total time calculations are changing one of the fields used in the test performed in step 7. In this manner, the number of times the loop is performed is controlled.

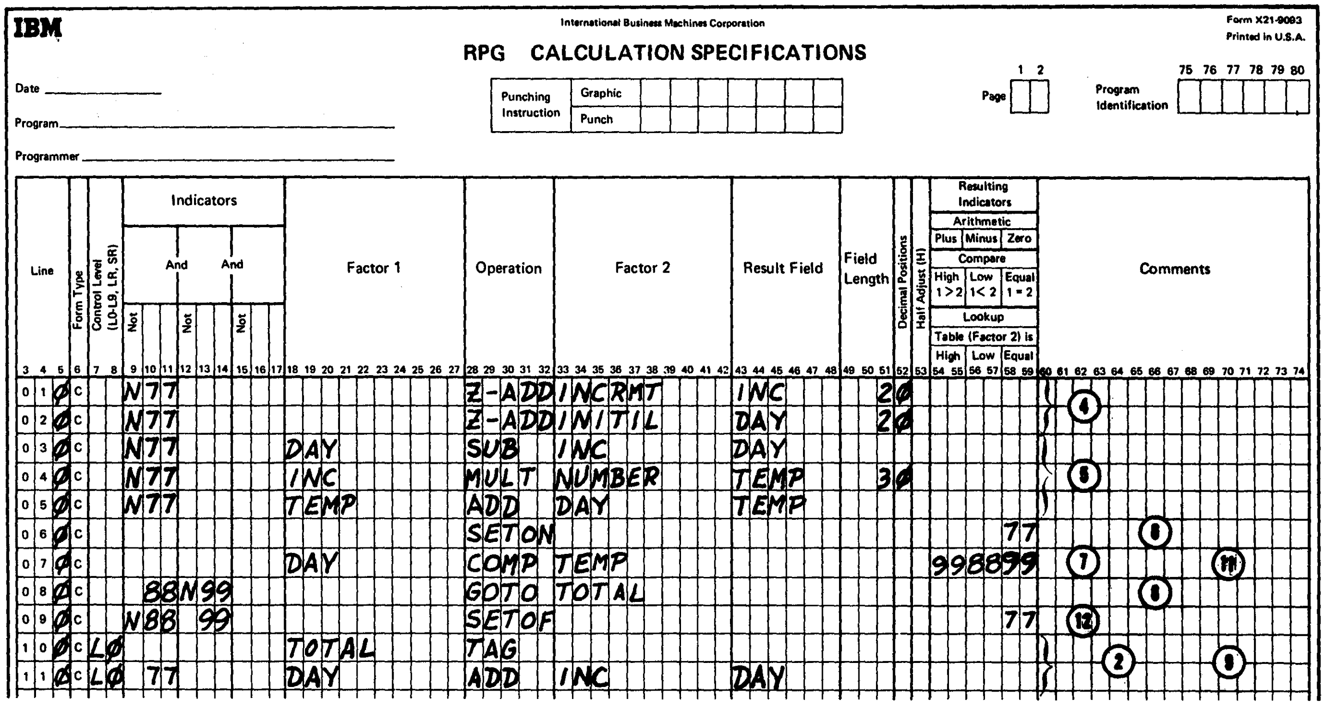
10. Total lines are printed in conjunction with the detail-to-total loop.

11. Step 7 is repeated.

12. When the results for this detail record have been obtained, the settings for indicators 88 and 99 are reversed, indicator 77 is set off and the detail-to-total loop is discontinued.

13. Detail time output lines are printed and the next input record is read.

14. Normal total lines are printed.



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 (3-5)	Form Type (6)	Control Level (L0-L9, LR, SR) (7-8)	Indicators: Not (9)	(10-11)	And: Not (12)	(13-14)	And: Not (15)	(16-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field (43-48)	Field Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Arithmetic Plus / Compare High 1>2 / Lookup Table (Factor 2) is High (54-55)	Minus / Low 1<2 / Low (56-57)	Zero / Equal 1=2 / Equal (58-59)	Comments (60-74)
01Ø	C		N	77						Z-ADD	INCRMT	INC	2	Ø					
02Ø	C		N	77						Z-ADD	INITIL	DAY	2	Ø					4
03Ø	C		N	77					DAY	SUB	INC	DAY							
04Ø	C		N	77					INC	MULT	NUMBER	TEMP	3	Ø					5
05Ø	C		N	77					TEMP	ADD	DAY	TEMP							
06Ø	C									SETON								77	6
07Ø	C								DAY	COMP	TEMP					99	88	99	7 11
08Ø	C			88	N	99				GOTO	TOTAL								8
09Ø	C		N	88		99				SETOF								77	12
10Ø	C	LØ							TOTAL	TAG									
11Ø	C	LØ		77					DAY	ADD	INC	DAY							2 9

Figure 171. Coding for Detail-to-Total Branch Example (Part 1 of 2)

IBM

International Business Machines Corporation

Form X21-9090
Printed in U.S.A.

RPG OUTPUT - FORMAT SPECIFICATIONS

Date ____________

Program ____________

Programmer ____________

Punching Instruction	Graphic	
	Punch	

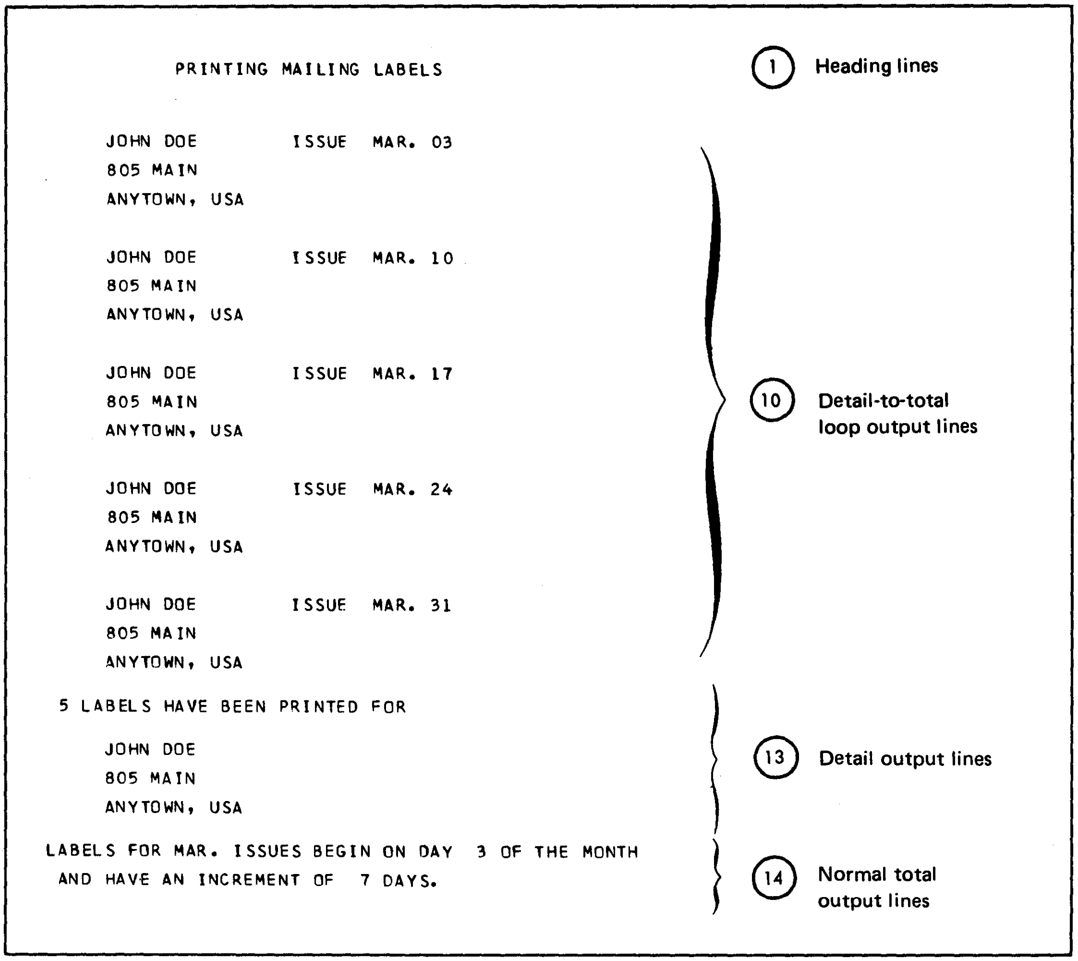
Page (1 2) □□

Program Identification (75 76 77 78 79 80) □□□□□□

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

Line (3 4 5)	Form Type (6)	Filename (7–14)	Type (H/D/T/E) (15)	Stacker Select/Fetch Overflow (F) (16)	Space Before (17)	Space After (18)	Skip Before (19–20)	Skip After (21–22)	Output Indicators (23–31)	Field Name (32–37)	Edit Codes (38)	Blank After (B) (39)	End Positon in Output Record (40–43)	P = Packed/B = Binary (44)	Constant or Edit Word (45–70)	Sterling Sign Position (71–74)
01Ø	O	OUTPUT	H		1	2	Ø1		1P							
02Ø	O	OR							OF							1
03Ø	O												43		'PRINTING MAILING LABELS'	
04Ø	O		D		2	2			Ø1							
05Ø	O									NUMBER	Z		11			
06Ø	O												36		'LABELS HAVE BEEN PRINTED'	
07Ø	O												41		'FOR '	
08Ø	O		D		1	1			Ø1							
09Ø	O									NAME			25			13
10Ø	O		D		1	1			Ø1							
11Ø	O									ADDRES			25			
12Ø	O		D		1	1			Ø1							
13Ø	O									LOCATE			26			
14Ø	O		T		2	1			LRN77							
15Ø	O												19		'LABELS FOR	
16Ø	O									MONTH			23			
17Ø	O												44		'. ISSUES BEGIN ON DAY'	
18Ø	O									INITIL	Z		47			
19Ø	O												6Ø		'OF THE MONTH'	14
2ØØ	O		T		1	1			LRN77							
21Ø	O												34		'AND HAVE AN INCREMENT OF'	
22Ø	O									INCRMT	Z		37			
23Ø	O												43		'DAYS.'	
24Ø	O		T		3	1			77							
25Ø	O									NAME			25			
26Ø	O												36		'ISSUE '	
27Ø	O									MONTH			4Ø			
28Ø	O												41		'.'	
29Ø	O									DAY			44			10
30Ø	O		T		1	1			77							
31Ø	O									ADDRES			25			
32Ø	O		T		1	1			77							
33Ø	O									LOCATE			26			
14	O															
15	O															
	O															
	O															
	O															
	O															
	O															

Figure 171. Coding for Detail-to-Total Branch Example (Part 1 of 2)



```
             PRINTING MAILING LABELS

      JOHN DOE          ISSUE  MAR. 03
      805 MAIN
      ANYTOWN, USA

      JOHN DOE          ISSUE  MAR. 10
      805 MAIN
      ANYTOWN, USA

      JOHN DOE          ISSUE  MAR. 17
      805 MAIN
      ANYTOWN, USA

      JOHN DOE          ISSUE  MAR. 24
      805 MAIN
      ANYTOWN, USA

      JOHN DOE          ISSUE  MAR. 31
      805 MAIN
      ANYTOWN, USA
  5 LABELS HAVE BEEN PRINTED FOR
      JOHN DOE
      805 MAIN
      ANYTOWN, USA
 LABELS FOR MAR. ISSUES BEGIN ON DAY  3 OF THE MONTH
  AND HAVE AN INCREMENT OF  7 DAYS.
```

1 Heading lines
10 Detail-to-total loop output lines
13 Detail output lines
14 Normal total output lines

Figure 172. Output from Detail-to-Total Branch Example

53715

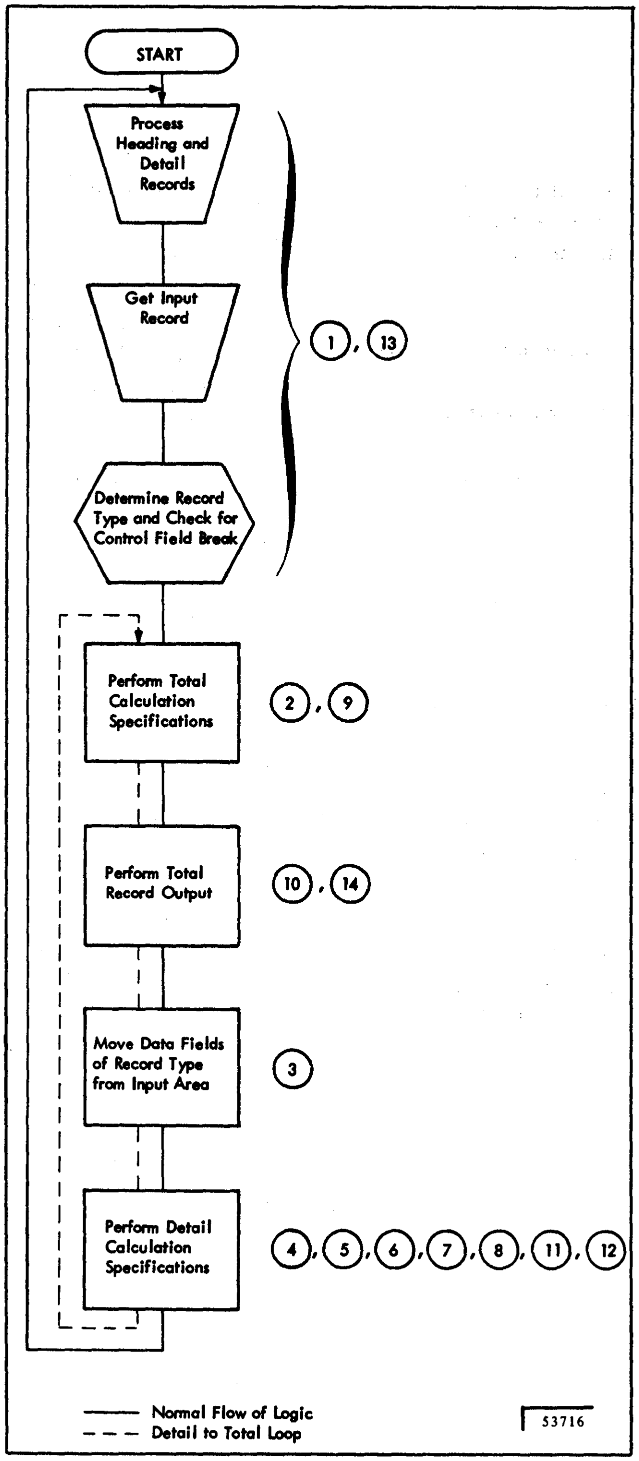

START
Process Heading and Detail Records
Get Input Record
Determine Record Type and Check for Control Field Break
1, 13
Perform Total Calculation Specifications
2, 9
Perform Total Record Output
10, 14
Move Data Fields of Record Type from Input Area
3
Perform Detail Calculation Specifications
4, 5, 6, 7, 8, 11, 12
Normal Flow of Logic
Detail to Total Loop
53716

Figure 173. Logic Flow for Detail-to-Total Branch Example

The capabilities of OS RPG can be significantly increased by the use of user prepared subroutines. The example in this section shows a subroutine that provides the following capabilities.

- 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.

Figure 174 is an assembler listing of an example of this subroutine. Once compiled, it should work in any RPG program. The subroutine requires 194 bytes of core storage.

The user must accurately follow one or the other of the examples shown in this section (Figures 175 and 177) and use the appropnate 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.

   - 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 three-position numeric field with zero decimals.

   - IVAL must be used to describe the length of IBIG. Normally a Z-ADD statement will place a value in IVAL.

   - IN85 (indicator 85) must be used to cause INDX to either:

     a. If on, move into IBIG (change value).

     b. If off, move out of IBIG (extract a field from IBIG).

   - IN86 (indicator 86) must be used to cause INDX to either:

     a. If on, begin move with value in ICNT (specific byte location within IBIG).

     b. If off, multiply ILTH x ICNT and use the result as the beginning byte (index to the Nth field).

   - INH9 indicates an error can occur within INDX. It is the result of requesting a character which is outside of IBIG. This can occur through a multiplication of ILTH and ICNT resulting in a number greater than the length of IBIG. In this case, halt indicator H9 is set on. Unless it is set off by the programmer, the program will terminate at the end of the cycle.

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 occured, the subroutine would be working with a field that was outside of IBIG. Rather than allow this to occur, the subroutine will set on H9. 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.

The user must provide RLABL statements which provide for communication between the RPG program and the BAL subroutine. 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: Additional information on subroutines may be found in the section *Using Tables and Exit Routines in the Object Module, Exit To a User's Subroutine.*

4. 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 statement relative to each other is important.

5. Performing arithmetic in IBIG: It is possible, and may be desirable, to use IBIG as a series of counters. IBIG is limited to 256 bytes, 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.

6. Examples of uses of INDX: The following is a discussion of some typical uses of INDX and some examples (Figures 175-178).

   a. Example 1 – Extracting fields sequentially from IBIG (Figure 175).

      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 06 of the calculation specifications shows a COMP to six being made (number of fields).

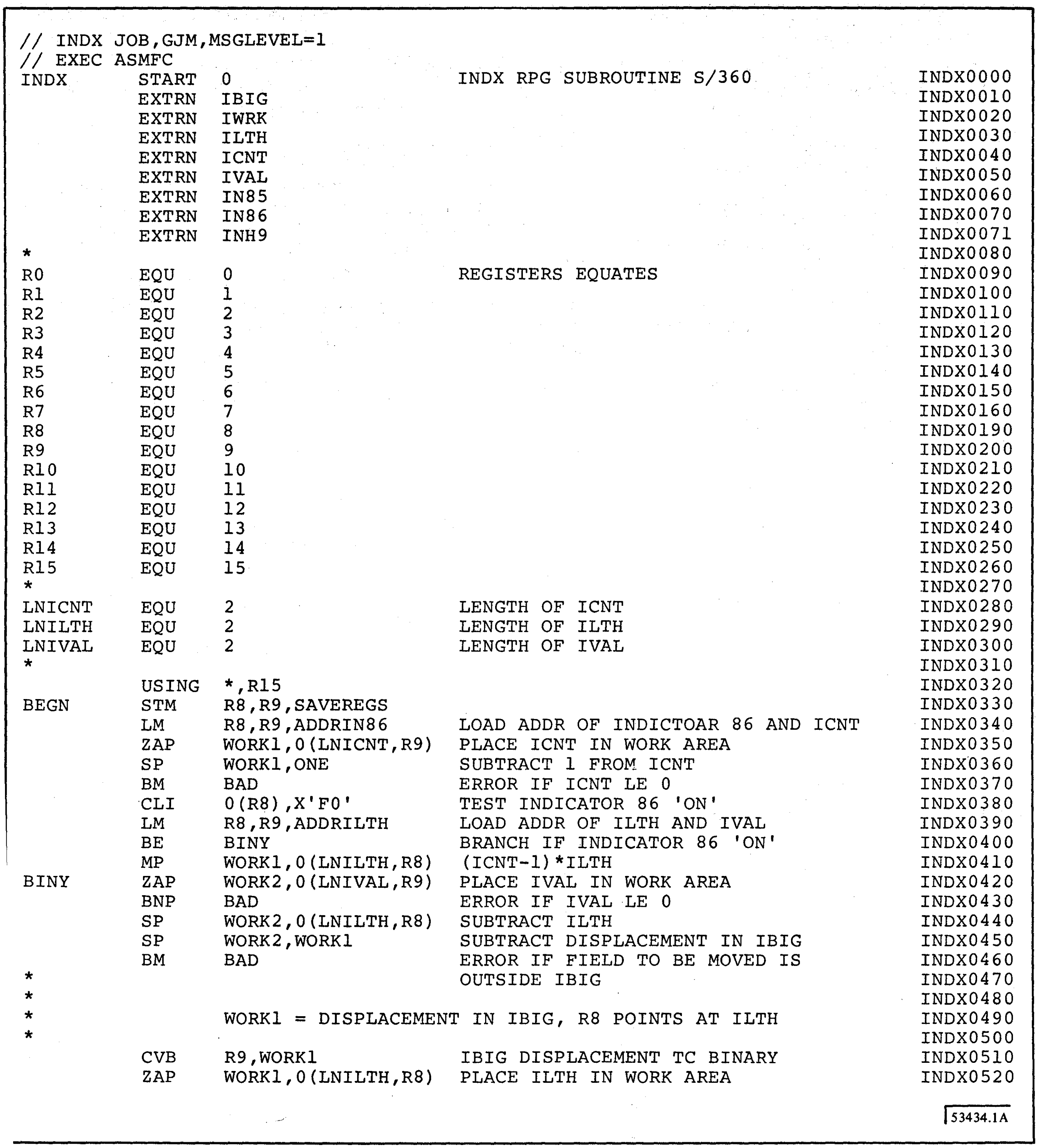
      Indicator 82 is used to start the extraction again. Looping and exception output is used to allow printing of each field extracted.

   b. Example 2 – Moving a variable length field to a variable location in IBIG (Figure 177).

      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.

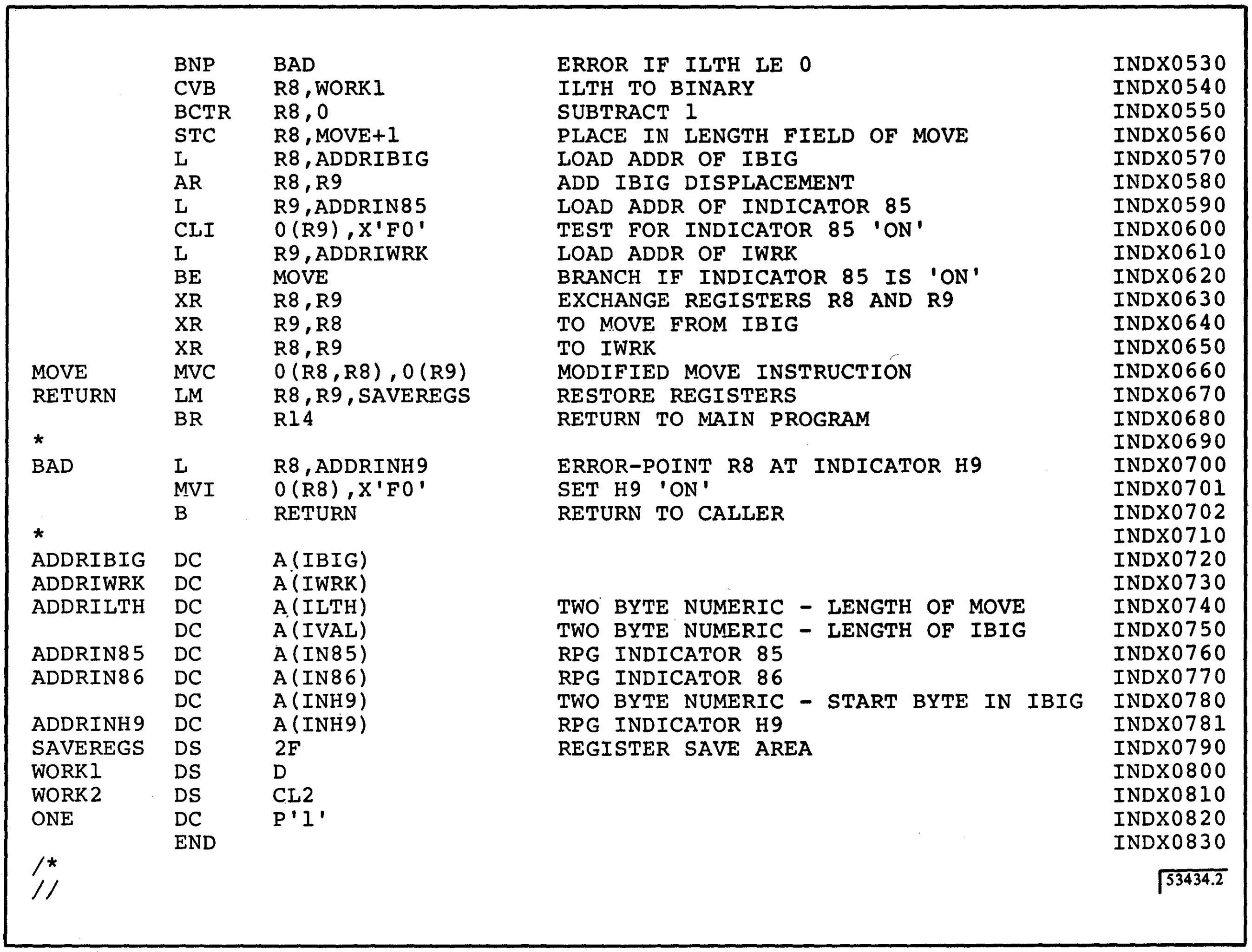
      Variable values are shown for ICNT and ILTH.



```
// INDX JOB,GJM,MSGLEVEL=1
// EXEC ASMFC
INDX      START  0                     INDX RPG SUBROUTINE S/360          INDX0000
          EXTRN  IBIG                                                     INDX0010
          EXTRN  IWRK                                                     INDX0020
          EXTRN  ILTH                                                     INDX0030
          EXTRN  ICNT                                                     INDX0040
          EXTRN  IVAL                                                     INDX0050
          EXTRN  IN85                                                     INDX0060
          EXTRN  IN86                                                     INDX0070
          EXTRN  INH9                                                     INDX0071
*                                                                         INDX0080
R0        EQU    0                     REGISTERS EQUATES                  INDX0090
R1        EQU    1                                                        INDX0100
R2        EQU    2                                                        INDX0110
R3        EQU    3                                                        INDX0120
R4        EQU    4                                                        INDX0130
R5        EQU    5                                                        INDX0140
R6        EQU    6                                                        INDX0150
R7        EQU    7                                                        INDX0160
R8        EQU    8                                                        INDX0190
R9        EQU    9                                                        INDX0200
R10       EQU    10                                                       INDX0210
R11       EQU    11                                                       INDX0220
R12       EQU    12                                                       INDX0230
R13       EQU    13                                                       INDX0240
R14       EQU    14                                                       INDX0250
R15       EQU    15                                                       INDX0260
*                                                                         INDX0270
LNICNT    EQU    2                     LENGTH OF ICNT                     INDX0280
LNILTH    EQU    2                     LENGTH OF ILTH                     INDX0290
LNIVAL    EQU    2                     LENGTH OF IVAL                     INDX0300
*                                                                         INDX0310
          USING  *,R15                                                    INDX0320
BEGN      STM    R8,R9,SAVEREGS                                           INDX0330
          LM     R8,R9,ADDRIN86        LOAD ADDR OF INDICTOAR 86 AND ICNT INDX0340
          ZAP    WORK1,0(LNICNT,R9)    PLACE ICNT IN WORK AREA            INDX0350
          SP     WORK1,ONE             SUBTRACT 1 FROM ICNT               INDX0360
          BM     BAD                   ERROR IF ICNT LE 0                 INDX0370
          CLI    0(R8),X'F0'           TEST INDICATOR 86 'ON'             INDX0380
          LM     R8,R9,ADDRILTH        LOAD ADDR OF ILTH AND IVAL         INDX0390
          BE     BINY                  BRANCH IF INDICATOR 86 'ON'        INDX0400
          MP     WORK1,0(LNILTH,R8)    (ICNT-1)*ILTH                      INDX0410
BINY      ZAP    WORK2,0(LNIVAL,R9)    PLACE IVAL IN WORK AREA            INDX0420
          BNP    BAD                   ERROR IF IVAL LE 0                 INDX0430
          SP     WORK2,0(LNILTH,R8)    SUBTRACT ILTH                      INDX0440
          SP     WORK2,WORK1           SUBTRACT DISPLACEMENT IN IBIG      INDX0450
          BM     BAD                   ERROR IF FIELD TO BE MOVED IS      INDX0460
*                                      OUTSIDE IBIG                       INDX0470
*                                                                         INDX0480
*                WORK1 = DISPLACEMENT IN IBIG, R8 POINTS AT ILTH          INDX0490
*                                                                         INDX0500
          CVB    R9,WORK1              IBIG DISPLACEMENT TC BINARY        INDX0510
          ZAP    WORK1,0(LNILTH,R8)    PLACE ILTH IN WORK AREA            INDX0520
```

53434.1A

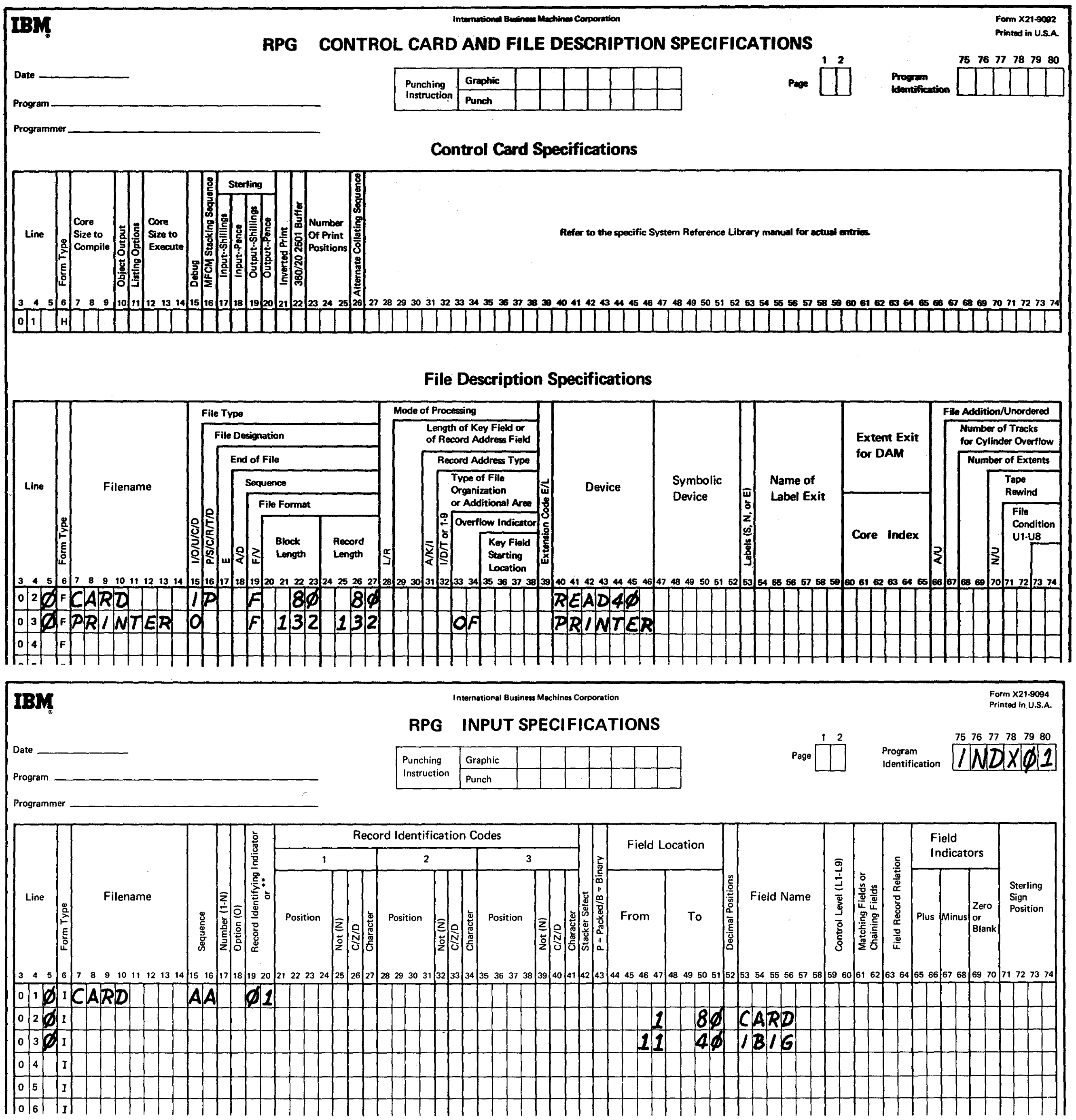
'igure 174. Indexing Subroutine Listing (Part 1 of 2)



```
         BNP   BAD                 ERROR IF ILTH LE 0                   INDX0530
         CVB   R8,WORK1            ILTH TO BINARY                       INDX0540
         BCTR  R8,0                SUBTRACT 1                           INDX0550
         STC   R8,MOVE+1           PLACE IN LENGTH FIELD OF MOVE        INDX0560
         L     R8,ADDRIBIG         LOAD ADDR OF IBIG                    INDX0570
         AR    R8,R9               ADD IBIG DISPLACEMENT                INDX0580
         L     R9,ADDRIN85         LOAD ADDR OF INDICATOR 85            INDX0590
         CLI   0(R9),X'F0'         TEST FOR INDICATOR 85 'ON'           INDX0600
         L     R9,ADDRIWRK         LOAD ADDR OF IWRK                    INDX0610
         BE    MOVE                BRANCH IF INDICATOR 85 IS 'ON'       INDX0620
         XR    R8,R9               EXCHANGE REGISTERS R8 AND R9         INDX0630
         XR    R9,R8               TO MOVE FROM IBIG                    INDX0640
         XR    R8,R9               TO IWRK                              INDX0650
MOVE     MVC   0(R8,R8),0(R9)      MODIFIED MOVE INSTRUCTION            INDX0660
RETURN   LM    R8,R9,SAVEREGS      RESTORE REGISTERS                    INDX0670
         BR    R14                 RETURN TO MAIN PROGRAM               INDX0680
*                                                                       INDX0690
BAD      L     R8,ADDRINH9         ERROR-POINT R8 AT INDICATOR H9       INDX0700
         MVI   0(R8),X'F0'         SET H9 'ON'                          INDX0701
         B     RETURN              RETURN TO CALLER                     INDX0702
*                                                                       INDX0710
ADDRIBIG DC    A(IBIG)                                                  INDX0720
ADDRIWRK DC    A(IWRK)                                                  INDX0730
ADDRILTH DC    A(ILTH)             TWO BYTE NUMERIC - LENGTH OF MOVE    INDX0740
         DC    A(IVAL)             TWO BYTE NUMERIC - LENGTH OF IBIG    INDX0750
ADDRIN85 DC    A(IN85)             RPG INDICATOR 85                     INDX0760
ADDRIN86 DC    A(IN86)             RPG INDICATOR 86                     INDX0770
         DC    A(INH9)             TWO BYTE NUMERIC - START BYTE IN IBIG INDX0780
ADDRINH9 DC    A(INH9)             RPG INDICATOR H9                     INDX0781
SAVEREGS DS    2F                  REGISTER SAVE AREA                   INDX0790
WORK1    DS    D                                                        INDX0800
WORK2    DS    CL2                                                      INDX0810
ONE      DC    P'1'                                                     INDX0820
         END                                                            INDX0830
/*
//
```

53434.2

Figure 174. Indexing Subroutine Listing (Part 2 of 2)



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 Program Identification 75 76 77 78 79 80

Control Card Specifications

Line	Form Type	Core Size to Compile	Object Output	Listing Options	Core Size to Execute	Debug	MFCM Stacking Sequence	Input-Shillings	Input-Pence	Output-Shillings	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	File Type I/O/U/C/D	File Designation P/S/C/R/T/D	End of File E	Sequence A/D	File Format F/V	Block Length	Record Length	L/R	Length of Key Field or of Record Address Field	Record Address Type A/K/I	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 / Core Index	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind N/U	File Condition U1-U8
02Ø	F	CARD	I	P			F	8Ø	8Ø								READ4Ø									
03Ø	F	PRINTER	O				F	132	132					OF			PRINTER									
04	F																									

IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

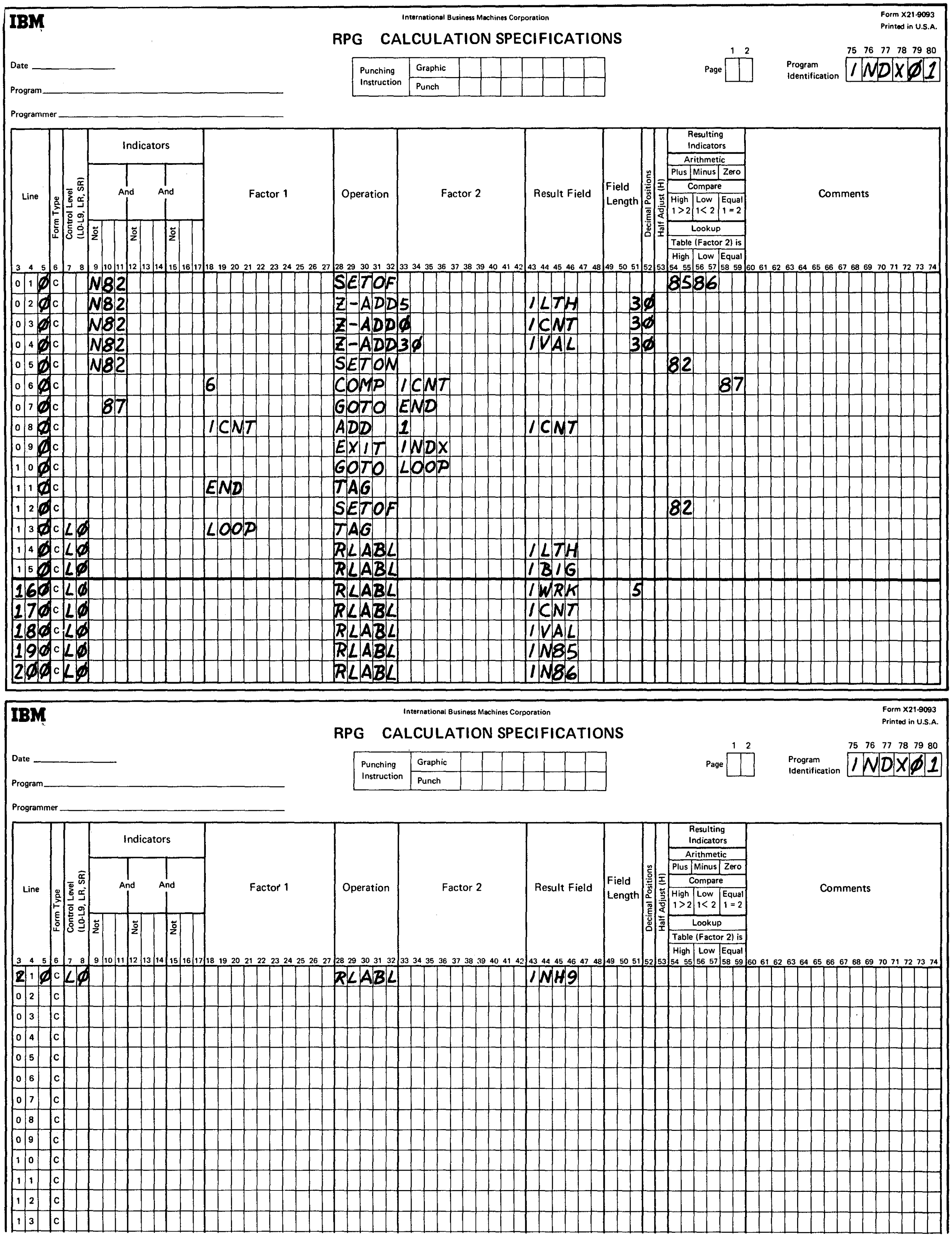
Date ______ Program ______ Programmer ______

Punching Instruction: Graphic / Punch

Page 1 2 Program Identification 75 76 77 78 79 80: INDXØ1

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	Not (N)	C/Z/D	Character	2: Position	Not (N)	C/Z/D	Character	3: Position	Not (N)	C/Z/D	Character	Stacker Select	P = Packed/B = Binary	Field Location From	To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Minus	Zero or Blank	Sterling Sign Position
01Ø	I	CARD	AA			Ø1																									
02Ø	I																				1	8Ø		CARD							
03Ø	I																				11	4Ø		IBIG							
04	I																														
05	I																														
06	I																														

Figure 175. Extracting Fields Sequentially from IBIG (Part 1 of 3)



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

RPG CALCULATION SPECIFICATIONS

Date
Program
Programmer

Punching Instruction: Graphic / Punch
Page
Program Identification (75–80): INDXØ1

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Plus / High 1>2	Minus / Low 1<2	Zero / Equal 1=2	Comments
01Ø	C		N82		SETOF						85	86		
02Ø	C		N82		Z-ADD	5	ILTH	3	Ø					
03Ø	C		N82		Z-ADD	Ø	ICNT	3	Ø					
04Ø	C		N82		Z-ADD	3Ø	IVAL	3	Ø					
05Ø	C		N82		SETON						82			
06Ø	C			6	COMP	ICNT							87	
07Ø	C		87		GOTO	END								
08Ø	C			ICNT	ADD	1	ICNT							
09Ø	C				EXIT	INDX								
10Ø	C				GOTO	LOOP								
11Ø	C			END	TAG									
12Ø	C				SETOF						82			
13Ø	C	LØ		LOOP	TAG									
14Ø	C	LØ			RLABL		ILTH							
15Ø	C	LØ			RLABL		IBIG							
16Ø	C	LØ			RLABL		IWRK	5						
17Ø	C	LØ			RLABL		ICNT							
18Ø	C	LØ			RLABL		IVAL							
19Ø	C	LØ			RLABL		IN85							
2ØØ	C	LØ			RLABL		IN86							

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

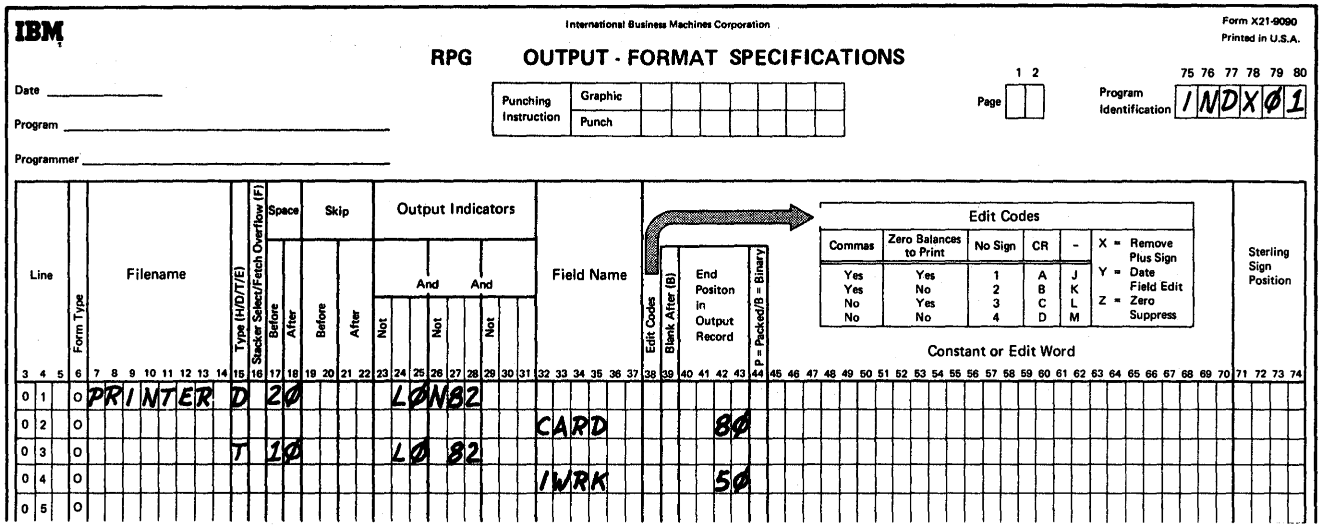
RPG CALCULATION SPECIFICATIONS

Date
Program
Programmer

Punching Instruction: Graphic / Punch
Page
Program Identification (75–80): INDXØ1

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Plus / High 1>2	Minus / Low 1<2	Zero / Equal 1=2	Comments
21Ø	C	LØ			RLABL		INH9							
02	C													
03	C													
04	C													
05	C													
06	C													
07	C													
08	C													
09	C													
10	C													
11	C													
12	C													
13	C													

Figure 175. Extracting Fields Sequentially from IBIG (Part 2 of 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 1 2 □□ — Program Identification 75 76 77 78 79 80: INDXØ1

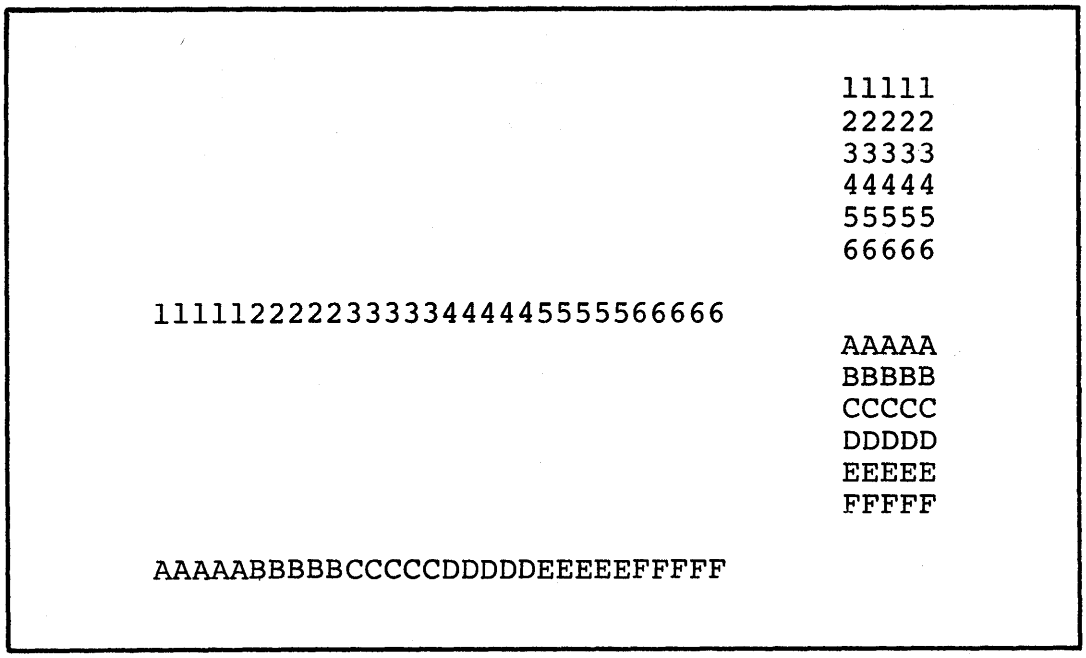
Edit Codes

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

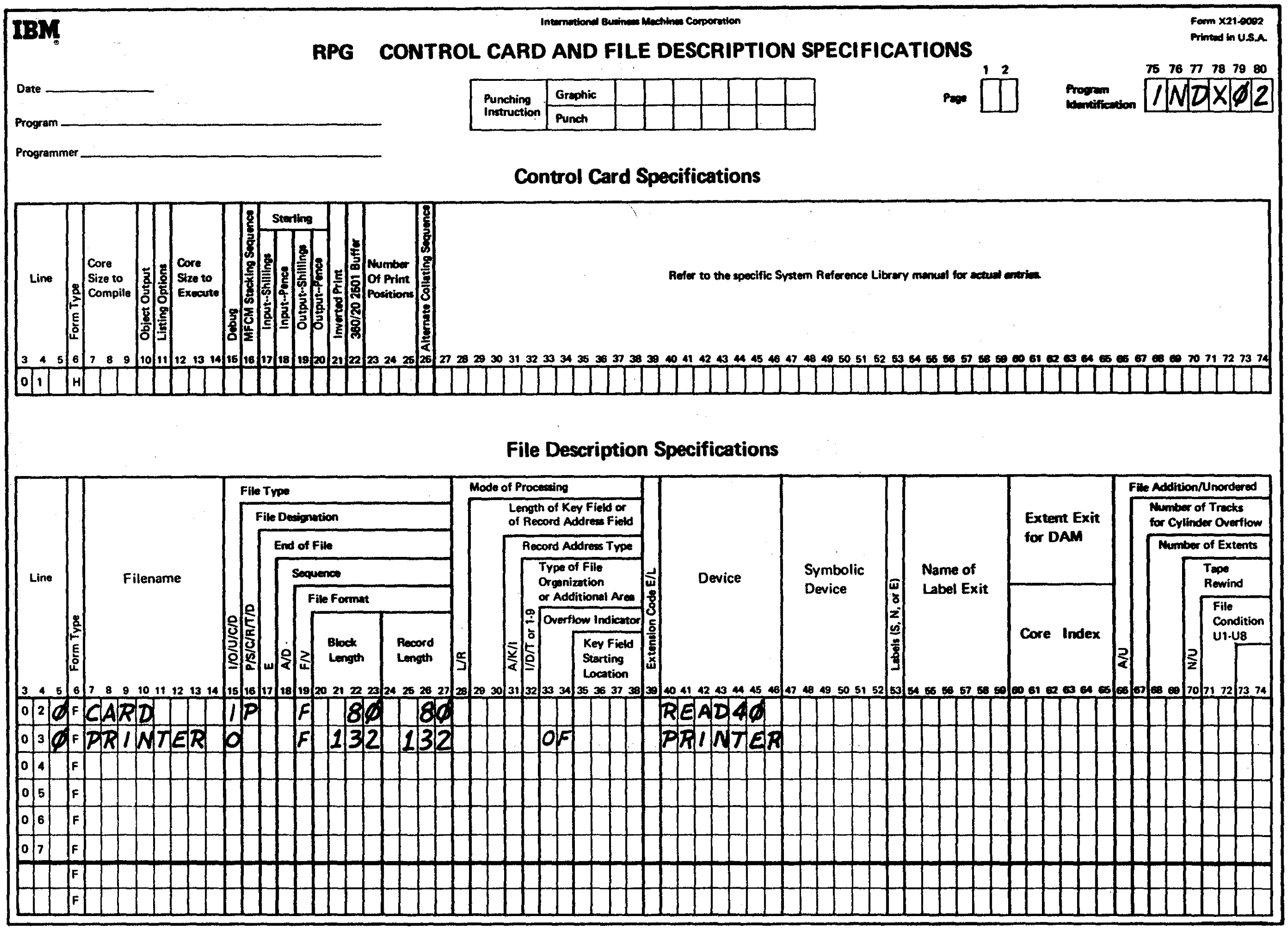
Line (3-5)	Form Type (6)	Filename (7-14)	Type (H/D/T/E) (15)	Stacker Select/Fetch Overflow (F) (16)	Space Before (17)	Space After (18)	Skip Before (19-20)	Skip After (21-22)	Output Indicators (23-31)	Field Name (32-37)	Edit Codes (38)	Blank After (B) (39)	End Positon in Output Record (40-43)	P = Packed/B = Binary (44)	Constant or Edit Word (45-70)	Sterling Sign Position (71-74)
01	O	PRINTER	D		2	Ø			LØN82							
02	O									CARD			8Ø			
03	O		T		1	Ø			LØ 82							
04	O									IWRK			5Ø			
05	O															

Figure 175. Extracting Fields Sequentially from IBIG (Part 3 of 3)



```
                                   11111
                                   22222
                                   33333
                                   44444
                                   55555
                                   66666

111112222233333444445555566666
                                   AAAAA
                                   BBBBB
                                   CCCCC
                                   DDDDD
                                   EEEEE
                                   FFFFF

AAAAABBBBBCCCCCDDDDDEEEEEFFFFF
```

Figure 176. Sample Output of INDX01



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

Program Identification 75 76 77 78 79 80: I N D X Ø 2

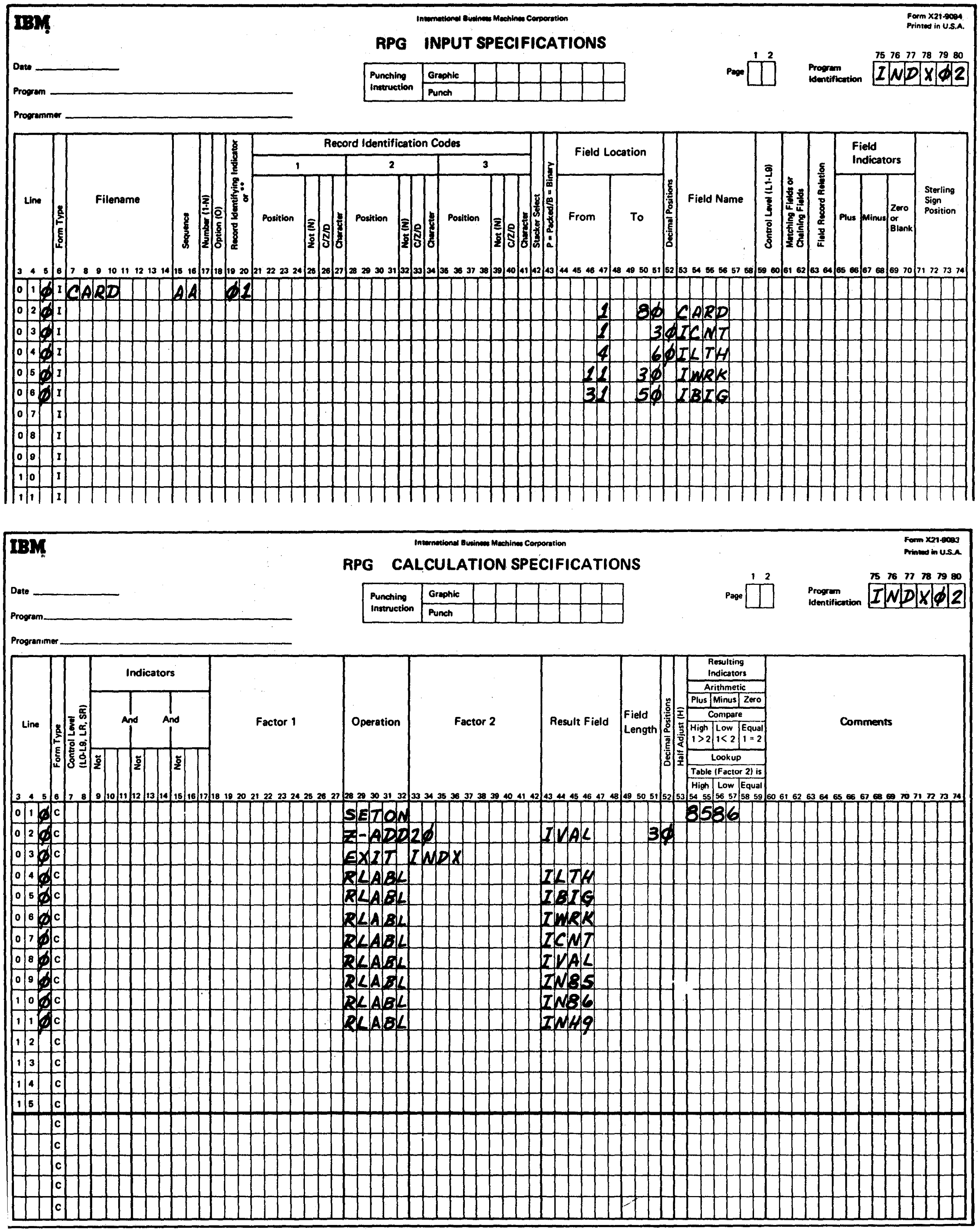
Control Card Specifications

Line (3-5)	Form Type (6)	Core Size to Compile (7-9)	Object Output (10)	Listing Options (11)	Core Size to Execute (12-14)	Debug (15)	MFCM Stacking Sequence (16)	Sterling: Input-Shillings (17)	Input-Pence (18)	Output-Shillings (19)	Output-Pence (20)	Inverted Print (21)	360/20 2501 Buffer (22)	Number Of Print Positions (23-25)	Alternate Collating Sequence (26)	27-74
01	H															Refer to the specific System Reference Library manual for actual entries.

File Description Specifications

Line (3-5)	Form Type (6)	Filename (7-14)	File Type: I/O/U/C/D (15)	File Designation: P/S/C/R/T/D (16)	End of File: E (17)	Sequence: A/D (18)	File Format: F/V (19)	Block Length (20-23)	Record Length (24-27)	Mode of Processing: L/R (28)	Length of Key Field or of Record Address Field (29-30)	Record Address Type: A/K/I (31)	Type of File Organization or Additional Area: I/D/T or 1-9 (32)	Overflow Indicator (33-34)	Key Field Starting Location (35-38)	Extension Code E/L (39)	Device (40-46)	Symbolic Device (47-52)	Labels (S, N, or E) (53)	Name of Label Exit (54-59)	Extent Exit for DAM / Core Index (60-65)	A/U (66)	File Addition/Unordered (67)	Number of Tracks for Cylinder Overflow (68-69)	Number of Extents: N/U (70)	Tape Rewind (71-72)	File Condition U1-U8 (73-74)
02Ø	F	CARD	I	P			F	8Ø	8Ø								READ4Ø										
03Ø	F	PRINTER	O				F	132	132					OF			PRINTER										
04	F																										
05	F																										
06	F																										
07	F																										
	F																										
	F																										

Figure 177. Moving a Variable Field to a Variable Location in IBIG (Part 1 of 3)



IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Program Identification (75–80): INDXØ2

Line	Form Type	Filename	Sequence	Record Identifying Indicator	Field Location From	Field Location To	Decimal Positions	Field Name
01Ø	I	CARD	AA	Ø1				
02Ø	I				1	8Ø		CARD
03Ø	I				1	3	Ø	ICNT
04Ø	I				4	6	Ø	ILTH
05Ø	I				11	3Ø		IWRK
06Ø	I				31	5Ø		IBIG
07	I							
08	I							
09	I							
10	I							
11	I							

IBM

International Business Machines Corporation

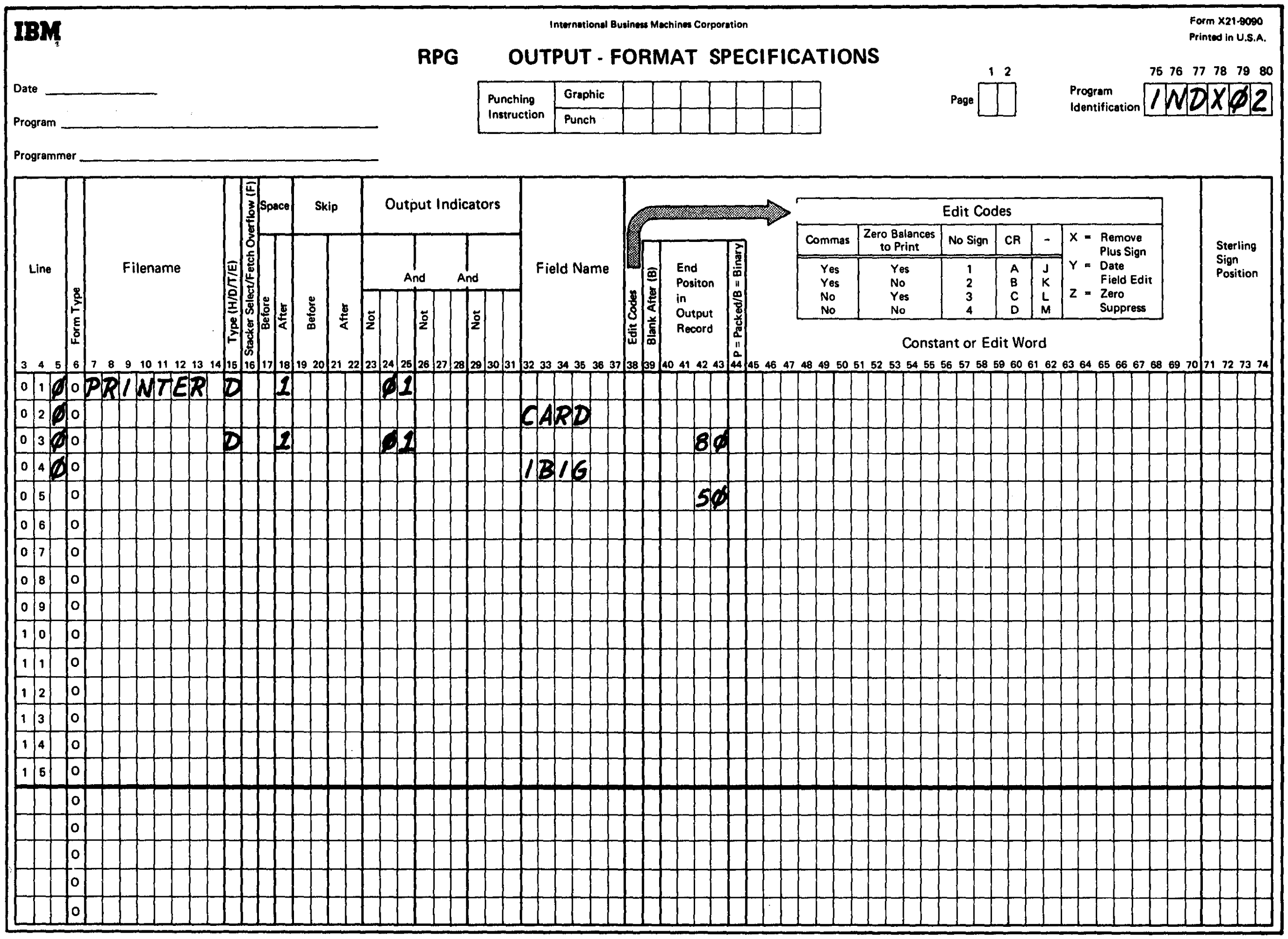
Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Program Identification (75–80): INDXØ2

Line	Form Type	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Resulting Indicators High 54 55	Resulting Indicators Low 56 57
01Ø	C		SETON					85	86
02Ø	C		Z-ADD	2Ø	IVAL	3	Ø		
03Ø	C		EXIT	INDX					
04Ø	C		RLABL		ILTH				
05Ø	C		RLABL		IBIG				
06Ø	C		RLABL		IWRK				
07Ø	C		RLABL		ICNT				
08Ø	C		RLABL		IVAL				
09Ø	C		RLABL		IN85				
10Ø	C		RLABL		IN86				
11Ø	C		RLABL		INH9				
12	C								
13	C								
14	C								
15	C								

Figure 177. Moving a Variable Field to a Variable Location in IBIG (Part 2 of 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 (1 2) □□

Program Identification (75 76 77 78 79 80): I N D X Ø 2

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 (3-5)	Form Type (6)	Filename (7-14)	Type (H/D/T/E) (15)	Space After (18)	Output Indicators (23-25)	Field Name (32-37)	End Position in Output Record (40-43)
01Ø	O	PRINTER	D	1	Ø1		
02Ø	O					CARD	
03Ø	O		D	1	Ø1		8Ø
04Ø	O					IBIG	
05	O						5Ø
06	O						
07	O						
08	O						
09	O						
10	O						
11	O						
12	O						
13	O						
14	O						
15	O						
	O						
	O						
	O						
	O						
	O						

Figure 177. Moving a Variable Field to a Variable Location in IBIG (Part 3 of 3)

```
001001        X                   12345678901234567890
                                  X2345678901234567890
001020        XX                  12345678901234567890
                                  XX
001020        XXXXXXXXXXXXXXXXXXXX12345678901234567890
                                  XXXXXXXXXXXXXXXXXXXX
005002        XX                  12345678901234567890
                                  1234XX78901234567890
009003        XXX                 12345678901234567890
                                  12345678XXX234567890
015005        XXXXX               12345678901234567890
                                  12345678901234XXXXX0
```

Figure 178. Sample Output of INDX02

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 specifications forms and in the control card. The File Description, Extension, and Calculation Specifications forms are not affected.

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 nine, positions.

Note 2: BSI or IBM input files of one program must use the same code combination throughout.

Input Specifications

The placement of the sign must be specified in positions 71-74. Enter an S in position 74 if the sign is in the normal place. If the pence field has decimal positions, the normal place for the sign is in the rightmost decimal position of the pence field. If the pence field has no decimal positions, the normal position 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 position 52 to indicate the number of required decimal positions.

Note 2: It is not permissible to use the same name 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 placement of the sign for sterling output fields must be specified in positions 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 Which is 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 position 74 of the Output-Format Specifications form.

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.

Note 2: The pounds field consists of at least one and no more than 9 positions. Zero suppression on the pounds field may be obtained by placing Z in position 38 of the specification form. The sign will not be suppressed since its position depends on the presence or absence of decimal pence.

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.

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, 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 and 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 239,999,999,999.999. This converts to a field containing nine pound positions, which is the maximum allowed.

Note: In order to avoid the possible loss of the high-order digit, fields that are read in as IBM or BSI sterling always contain one more position when put out as IBM or BSI sterling. For example, the five-position sterling input field 9919+ (IBM format) converts to the five-position pence field 23999. Since the RPG compiler in putting out this field must allow for a five-position pence field containing up to 99999 pence (which converts to 416133, IBM format), the field on output will be six positions long.

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 sterling fields: IBM or BSI. Positions 17-20 of the Control Card Specifications form indicate either the IBM or BSI formats. The formats for IBM and BSI are shown here.

	IBM Format	BSI Format
Shillings	(two positions) 00-19 for 0-19 shillings	(one position) 0-9 shillings by a 0-9 punch, 10 shillings by a 12-punch, 11-19 shillings by an A-I punch.
Pence	0-9 pence by a 0-9 punch, 10 pence by an 11-punch, 11 pence by a 12-punch. (one position)	0-9 pence by a 0-9 punch, 10 pence by a 12-punch, 11 pence by an 11-punch. (one position)

SAMPLE PROGRAM ONE

The input file in the first program consists of punched cards. Each card in the file contains a data record that includes from one to eighty characters of information. Each data record represents a purchase made from the reporting firm by a customer. The types of information and the card columns in which each appears are shown in Figure 179.

The labels assigned to the fields into which the object program will place the information are:

Field	*Label*
Customer name	NAME
Invoice data — month	MONTH
Invoice data — day	DAY
Invoice number	INVNO
Customer number	CUSTNO
Customer location	STATE
Customer location	CITY
Invoice amount	INVAMT

Figure 180 is an output listing of the sample program.

To produce an object module capable of writing the report shown in Figure 181, the programmer must prepare a source program as shown in Figure 182. The entries in the RPG specifications form are described here.

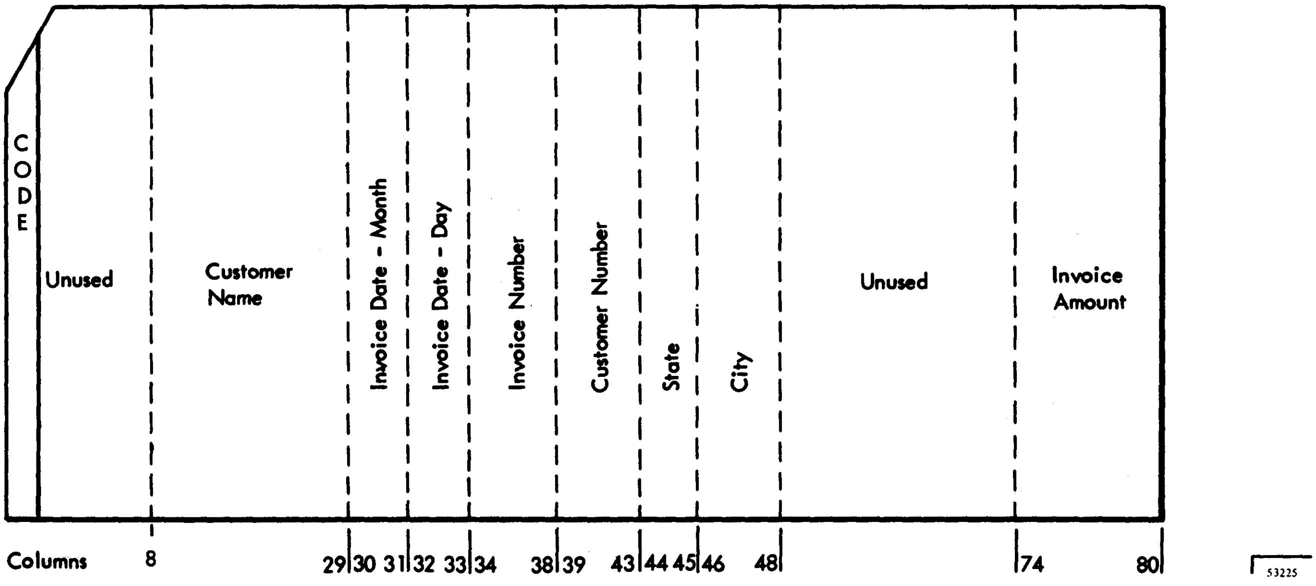
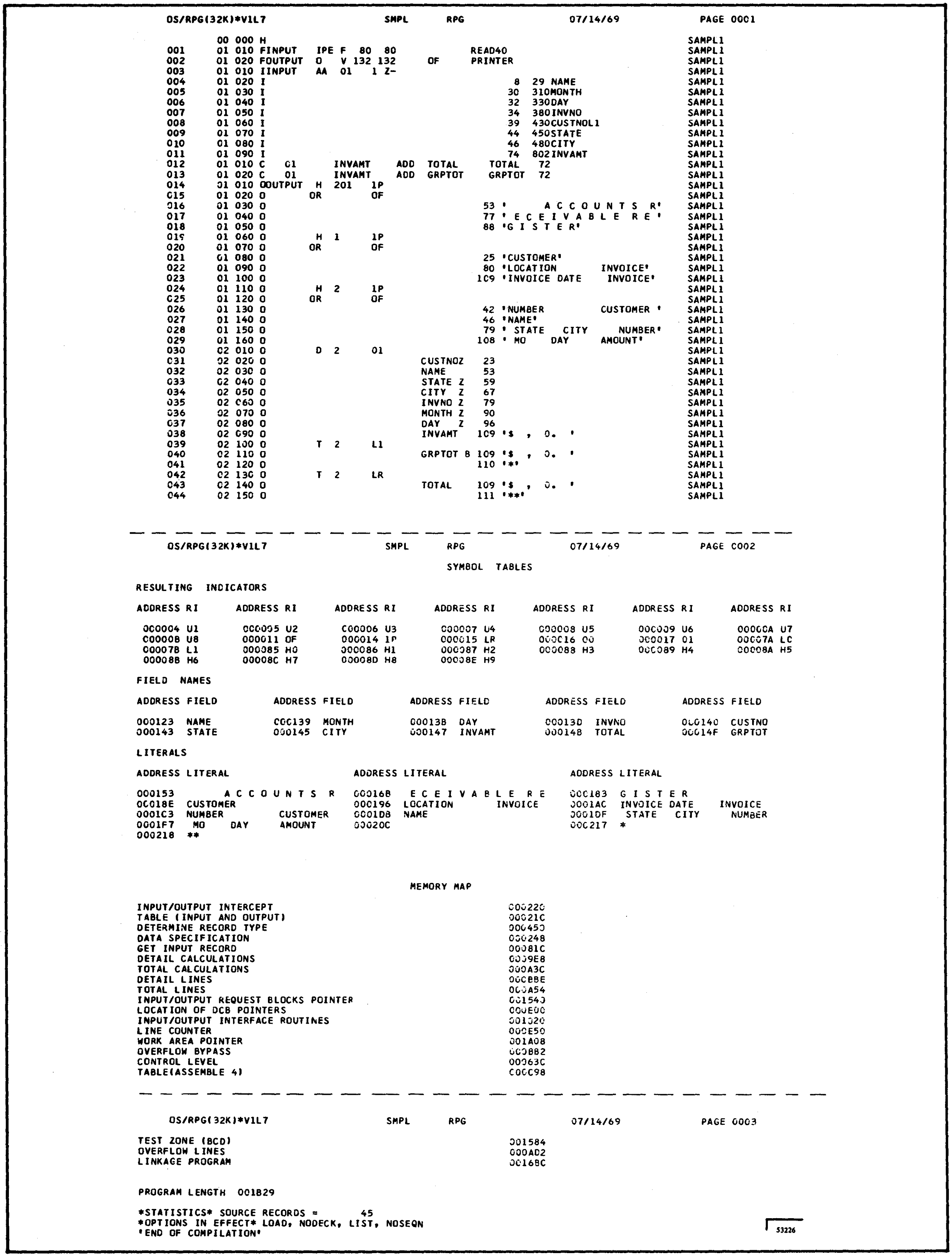


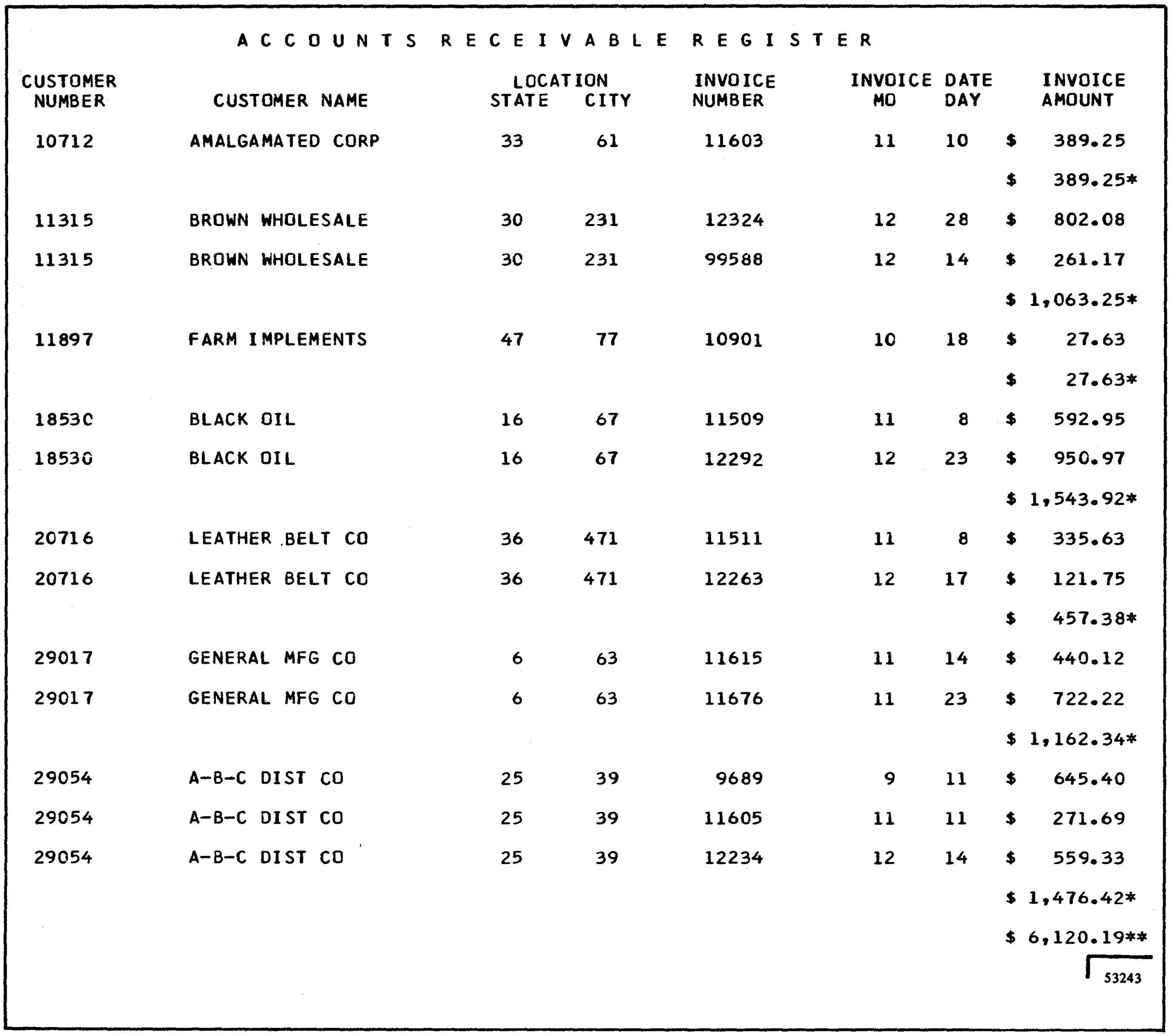
Figure 179. Input File-Card Format



```
     OS/RPG(32K)*V1L7                 SMPL      RPG                 07/14/69           PAGE 0001

              00 000 H                                                                SAMPL1
     001      01 010 FINPUT   IPE F  80  80           READ40                          SAMPL1
     002      01 020 FOUTPUT  O   V 132 132     OF    PRINTER                         SAMPL1
     003      01 010 IINPUT   AA  01   1 Z-                                           SAMPL1
     004      01 020 I                                        8  29 NAME              SAMPL1
     005      01 030 I                                       30  310MONTH             SAMPL1
     006      01 040 I                                       32  330DAY               SAMPL1
     007      01 050 I                                       34  380INVNO             SAMPL1
     008      01 060 I                                       39  430CUSTNOL1          SAMPL1
     009      01 070 I                                       44  450STATE             SAMPL1
     010      01 080 I                                       46  480CITY              SAMPL1
     011      01 090 I                                       74  802INVAMT            SAMPL1
     012      01 010 C    C1       INVAMT    ADD  TOTAL     TOTAL   72                SAMPL1
     013      01 020 C    01       INVAMT    ADD  GRPTOT    GRPTOT  72                SAMPL1
     014      01 010 OOUTPUT  H  201    1P                                            SAMPL1
     C15      01 020 O       OR         OF                                            SAMPL1
     016      01 030 O                                  53 '      A C C O U N T S  R' SAMPL1
     017      01 040 O                                  77 ' E C E I V A B L E  R E ' SAMPL1
     018      01 050 O                                  88 'G I S T E R'              SAMPL1
     019      01 060 O        H  1      1P                                            SAMPL1
     020      01 070 O       OR         OF                                            SAMPL1
     021      01 080 O                                  25 'CUSTOMER'                 SAMPL1
     022      01 090 O                                  80 'LOCATION       INVOICE'   SAMPL1
     023      01 100 O                                 1C9 'INVOICE DATE    INVOICE'  SAMPL1
     024      01 110 O        H  2      1P                                            SAMPL1
     C25      01 120 O       OR         OF                                            SAMPL1
     026      01 130 O                                  42 'NUMBER          CUSTOMER ' SAMPL1
     027      01 140 O                                  46 'NAME'                     SAMPL1
     028      01 150 O                                  79 ' STATE   CITY     NUMBER' SAMPL1
     029      01 160 O                                 108 ' MO    DAY       AMOUNT'  SAMPL1
     030      02 010 O        D  2      01                                            SAMPL1
     C31      02 020 O                        CUSTNOZ   23                            SAMPL1
     032      02 030 O                        NAME      53                            SAMPL1
     033      02 040 O                        STATE Z   59                            SAMPL1
     034      02 050 O                        CITY  Z   67                            SAMPL1
     035      02 C60 O                        INVNO Z   79                            SAMPL1
     036      02 070 O                        MONTH Z   90                            SAMPL1
     037      02 080 O                        DAY   Z   96                            SAMPL1
     038      02 C90 O                        INVAMT   1C9 '$  ,  0.  '               SAMPL1
     039      02 100 O        T  2      L1                                            SAMPL1
     040      02 110 O                        GRPTOT B 109 '$  ,  0.  '               SAMPL1
     041      02 120 O                                 110 '*'                        SAMPL1
     042      02 13C O        T  2      LR                                            SAMPL1
     043      02 140 O                        TOTAL    109 '$  ,  0.  '               SAMPL1
     044      02 150 O                                 111 '**'                       SAMPL1

 — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —

     OS/RPG(32K)*V1L7                 SMPL      RPG                 07/14/69           PAGE C002

                                                SYMBOL  TABLES

 RESULTING  INDICATORS

 ADDRESS RI      ADDRESS RI      ADDRESS RI      ADDRESS RI      ADDRESS RI      ADDRESS RI      ADDRESS RI

  0C0004 U1       0C0005 U2       C00006 U3       000007 U4       000008 U5       000009 U6       00000A U7
  C0000B U8       000011 OF       000014 1P       000015 LR       000C16 00       000017 01       00C07A LC
  C0007B L1       000085 H0       000086 H1       000087 H2       000088 H3       000089 H4       00008A H5
  00008B H6       00008C H7       00008D H8       00008E H9

 FIELD  NAMES

 ADDRESS FIELD          ADDRESS FIELD          ADDRESS FIELD          ADDRESS FIELD          ADDRESS FIELD

 000123  NAME           CCC139  MONTH          00013B  DAY            00013D  INVNO          000140  CUSTNO
 000143  STATE          000145  CITY           000147  INVAMT         000148  TOTAL          00014F  GRPTOT

 LITERALS

 ADDRESS LITERAL                       ADDRESS LITERAL                       ADDRESS LITERAL

 000153        A C C O U N T S  R      00016B   E C E I V A B L E  R E       00C183  G I S T E R
 0C018E  CUSTOMER                      000196  LOCATION        INVOICE       0001AC  INVOICE DATE    INVOICE
 0001C3  NUMBER          CUSTOMER      0001DB  NAME                          0001DF   STATE   CITY     NUMBER
 0001F7   MO     DAY     AMOUNT        00020C                                00C217  *
 000218  **


                                              MEMORY MAP

 INPUT/OUTPUT INTERCEPT                                        000220
 TABLE (INPUT AND OUTPUT)                                      00021C
 DETERMINE RECORD TYPE                                         000450
 DATA SPECIFICATION                                            000248
 GET INPUT RECORD                                              00081C
 DETAIL CALCULATIONS                                           0009E8
 TOTAL CALCULATIONS                                            000A3C
 DETAIL LINES                                                  000BBE
 TOTAL LINES                                                   000A54
 INPUT/OUTPUT REQUEST BLOCKS POINTER                           001540
 LOCATION OF DCB POINTERS                                      000E00
 INPUT/OUTPUT INTERFACE ROUTINES                               001020
 LINE COUNTER                                                  000E50
 WORK AREA POINTER                                             001A08
 OVERFLOW BYPASS                                               000B82
 CONTROL LEVEL                                                 00063C
 TABLE(ASSEMBLE 4)                                             C00C98
 — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —

     OS/RPG(32K)*V1L7                 SMPL      RPG                 07/14/69           PAGE 0003

 TEST ZONE (BCD)                                               001584
 OVERFLOW LINES                                                000AD2
 LINKAGE PROGRAM                                               00168C


 PROGRAM LENGTH  001B29

 *STATISTICS* SOURCE RECORDS =       45
 *OPTIONS IN EFFECT* LOAD, NODECK, LIST, NOSEQN
 'END OF COMPILATION'
```

53226

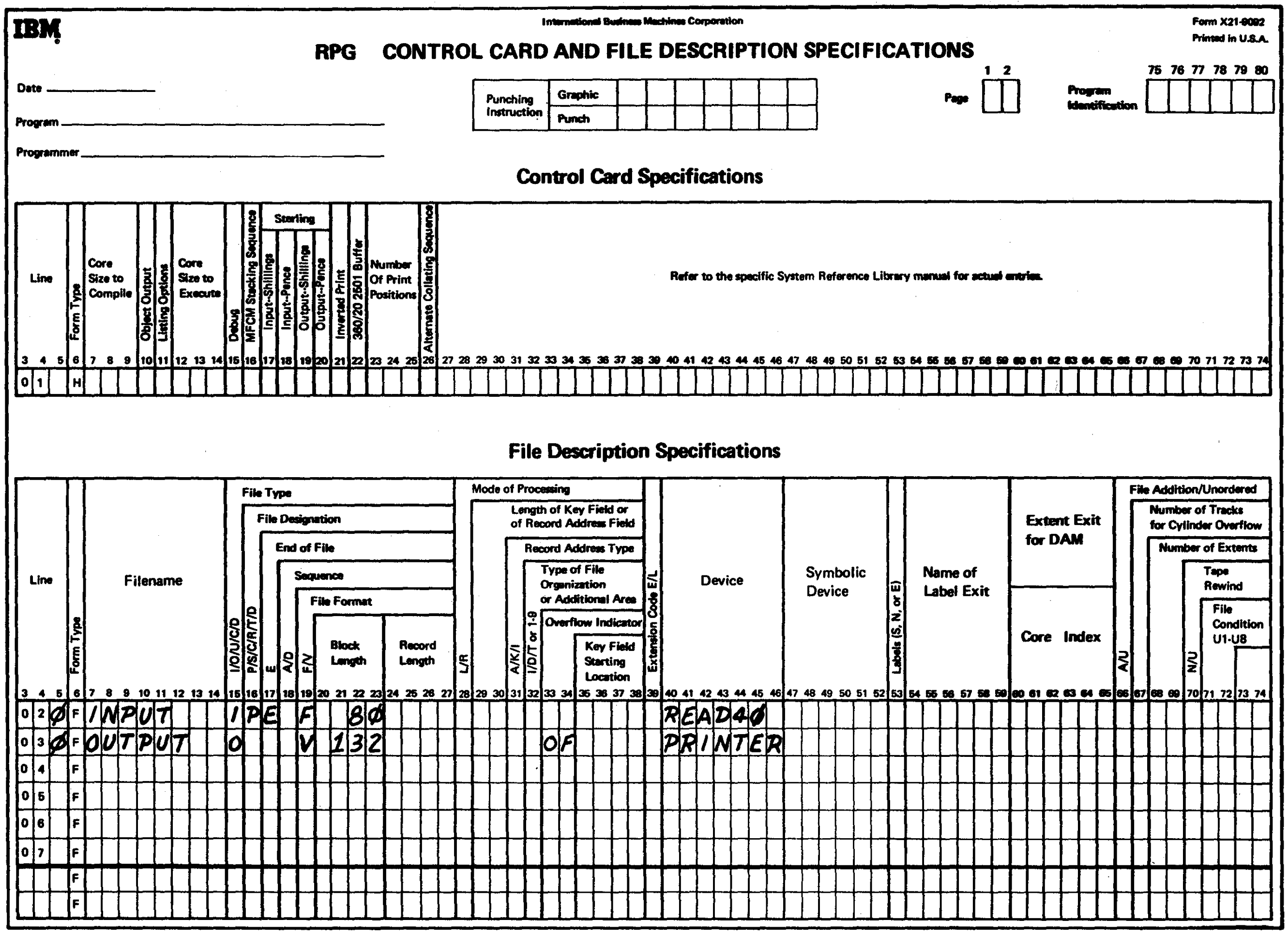
Figure 180. Output Listing



```
                    A C C O U N T S   R E C E I V A B L E   R E G I S T E R

CUSTOMER                                   LOCATION        INVOICE          INVOICE  DATE       INVOICE
 NUMBER         CUSTOMER NAME            STATE   CITY      NUMBER              MO     DAY       AMOUNT

 10712        AMALGAMATED CORP             33     61        11603              11      10   $     389.25

                                                                                            $     389.25*

 11315        BROWN WHOLESALE              30    231        12324              12      28   $     802.08

 11315        BROWN WHOLESALE              30    231        99588              12      14   $     261.17

                                                                                            $ 1,063.25*

 11897        FARM IMPLEMENTS              47     77        10901              10      18   $      27.63

                                                                                            $      27.63*

 18530        BLACK OIL                    16     67        11509              11       8   $     592.95

 18530        BLACK OIL                    16     67        12292              12      23   $     950.97

                                                                                            $ 1,543.92*

 20716        LEATHER BELT CO              36    471        11511              11       8   $     335.63

 20716        LEATHER BELT CO              36    471        12263              12      17   $     121.75

                                                                                            $     457.38*

 29017        GENERAL MFG CO                6     63        11615              11      14   $     440.12

 29017        GENERAL MFG CO                6     63        11676              11      23   $     722.22

                                                                                            $ 1,162.34*

 29054        A-B-C DIST CO                25     39         9689               9      11   $     645.40

 29054        A-B-C DIST CO                25     39        11605              11      11   $     271.69

 29054        A-B-C DIST CO                25     39        12234              12      14   $     559.33

                                                                                            $ 1,476.42*

                                                                                            $ 6,120.19**
```

53243

Figure 181. Printed Report



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)

Program Identification (75 76 77 78 79 80)

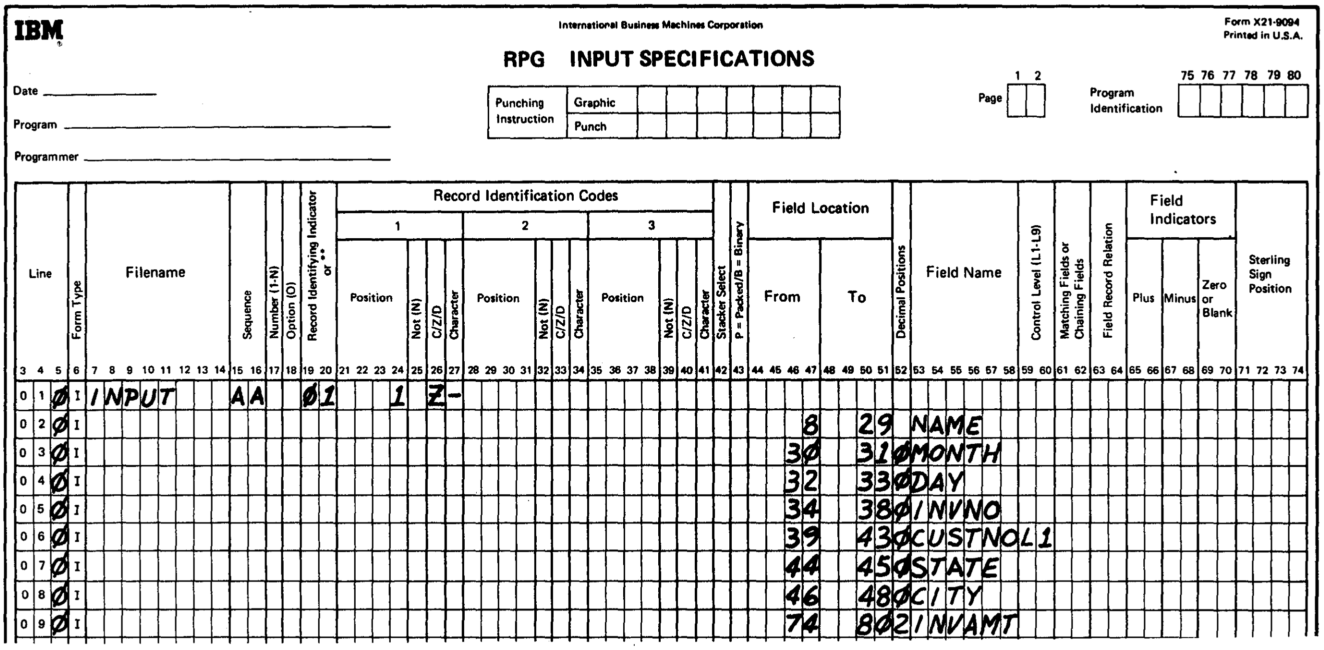
Control Card Specifications

Line (3-5)	Form Type (6)	Core Size to Compile (7-9)	Object Output (10)	Listing Options (11)	Core Size to Execute (12-14)	Debug (15)	MFCM Stacking Sequence (16)	Sterling: Input-Shillings (17)	Input-Pence (18)	Output-Shillings (19)	Output-Pence (20)	Inverted Print (21)	360/20 2501 Buffer (22)	Number Of Print Positions (23-25)	Alternate Collating Sequence (26)	27-74
01	H															Refer to the specific System Reference Library manual for actual entries.

File Description Specifications

Line (3-5)	Form Type (6)	Filename (7-14)	File Type: I/O/U/C/D (15)	File Designation: P/S/C/R/T/D (16)	End of File: E (17)	Sequence: A/D (18)	File Format: F/V (19)	Block Length (20-23)	Record Length (24-27)	Mode of Processing: L/R (28)	Length of Key Field or of Record Address Field (29-30)	Record Address Type: A/K/I (31)	Type of File Organization or Additional Area: I/D/T or 1-9 (32)	Overflow Indicator (33-34)	Key Field Starting Location (35-38)	Extension Code E/L (39)	Device (40-46)	Symbolic Device (47-52)	Labels (S, N, or E) (53)	Name of Label Exit (54-59)	Extent Exit for DAM / Core Index (60-65)	File Addition/Unordered: A/U (66)	Number of Tracks for Cylinder Overflow (67)	Number of Extents (68-69)	Tape Rewind: N/U (70)	File Condition U1-U8 (71-72)	73-74
02Ø	F	INPUT	I	P	E		F	8Ø									READ4Ø										
03Ø	F	OUTPUT	O				V	132						OF			PRINTER										
04	F																										
05	F																										
06	F																										
07	F																										
	F																										
	F																										

Figure 182. RPG Specification Forms-Program One (Part 1 of 3)



IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

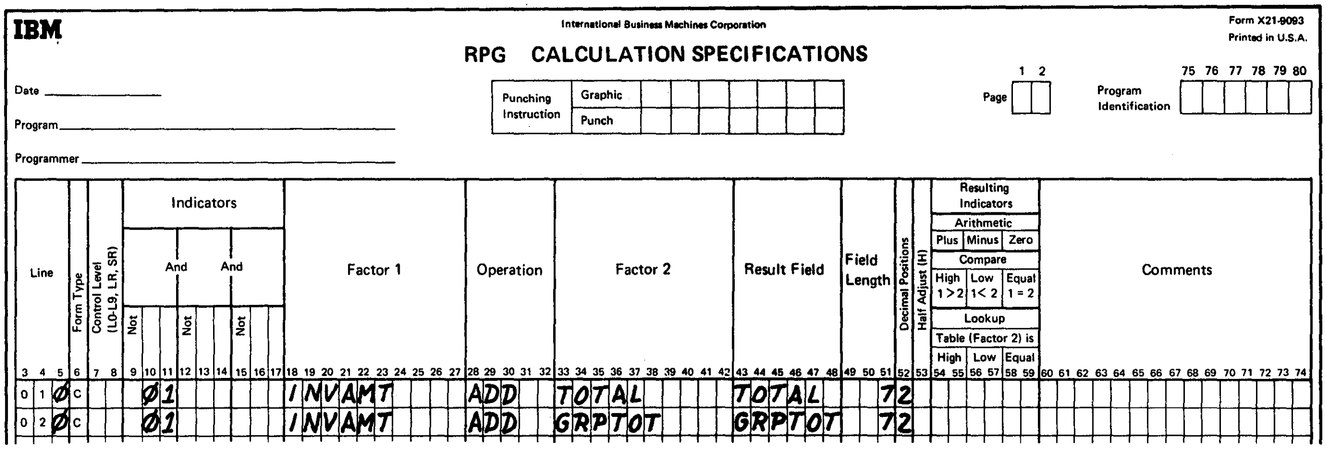
Date ____
Program ____
Programmer ____

Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1 Position	Not (N)	C/Z/D	Character	Field Location From	To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Minus	Zero or Blank	Sterling Sign Position
0 1 Ø	I	INPUT	AA			Ø1	1		Z	-											
0 2 Ø	I										8	29		NAME							
0 3 Ø	I										3Ø	31	Ø	MONTH							
0 4 Ø	I										32	33	Ø	DAY							
0 5 Ø	I										34	38	Ø	INVNO							
0 6 Ø	I										39	43	Ø	CUSTNO	L1						
0 7 Ø	I										44	45	Ø	STATE							
0 8 Ø	I										46	48	Ø	CITY							
0 9 Ø	I										74	8Ø	2	INVAMT							



IBM

International Business Machines Corporation

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

RPG CALCULATION SPECIFICATIONS

Date ____
Program ____
Programmer ____

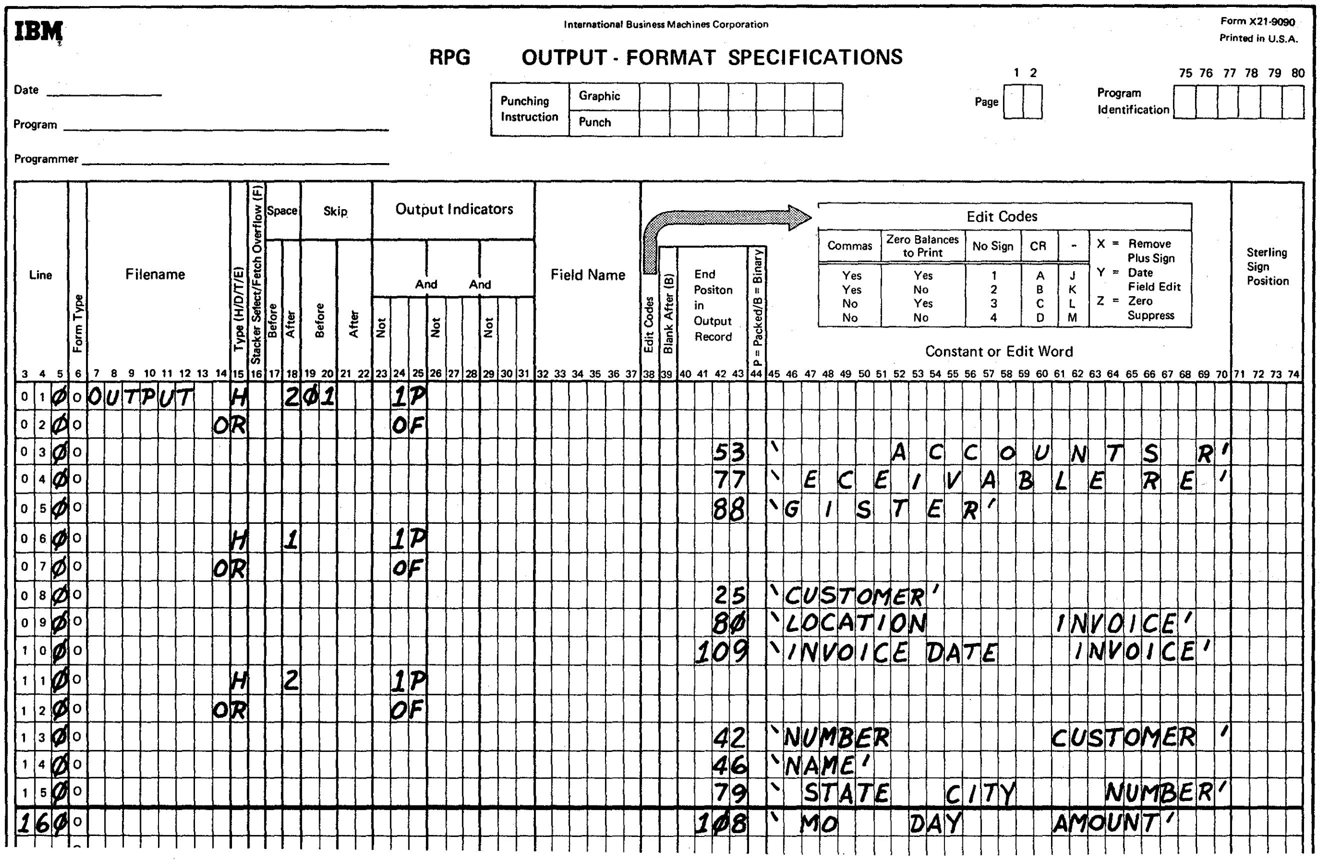
Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators	And	And	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators Plus / High / High	Minus / Low / Low	Zero / Equal / Equal	Comments
0 1 Ø	C		Ø1			INVAMT	ADD	TOTAL	TOTAL	7	2					
0 2 Ø	C		Ø1			INVAMT	ADD	GRPTOT	GRPTOT	7	2					

Figure 182. RPG Specification Forms-Program one (Part 2 of 3)



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

RPG OUTPUT - FORMAT SPECIFICATIONS

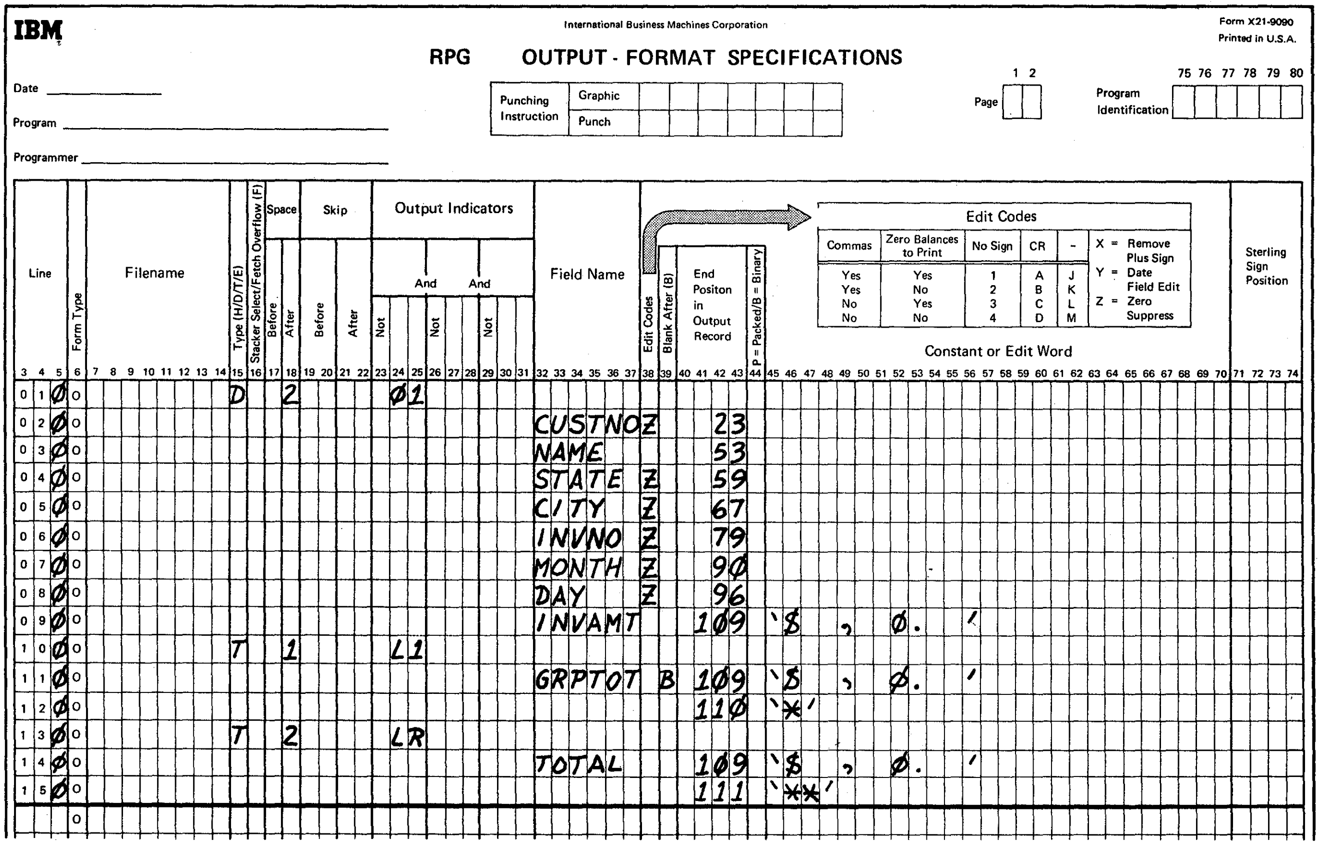
Date ______ Program ______ Programmer ______

Punching Instruction: Graphic / Punch — Page 1 2 — Program Identification 75 76 77 78 79 80

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

Line | Form Type | Filename | Type (H/D/T/E) | Stacker Select/Fetch Overflow (F) | Space Before After | Skip Before After | Output Indicators Not And Not And Not | Field Name | Edit Codes | Blank After (B) | End Positon in Output Record | P = Packed/B = Binary | Constant or Edit Word | Sterling Sign Position

```
01ØOOUTPUT  H  2Ø1   1P
02ØO       OR        OF
03ØO                                   53 '      A C C O U N T S  R'
04ØO                                   77 ' E C E I V A B L E  R E '
05ØO                                   88 'G I S T E R'
06ØO        H  1     1P
07ØO       OR        OF
08ØO                                   25 'CUSTOMER'
09ØO                                   8Ø 'LOCATION           INVOICE'
10ØO                                  109 'INVOICE DATE         INVOICE'
11ØO        H  2     1P
12ØO       OR        OF
13ØO                                   42 'NUMBER            CUSTOMER  '
14ØO                                   46 'NAME'
15ØO                                   79 ' STATE     CITY        NUMBER'
16ØO                                  1Ø8 ' MO      DAY         AMOUNT'
```



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 1 2 — Program Identification 75 76 77 78 79 80

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

Line | Form Type | Filename | Type (H/D/T/E) | Stacker Select/Fetch Overflow (F) | Space Before After | Skip Before After | Output Indicators Not And Not And Not | Field Name | Edit Codes | Blank After (B) | End Positon in Output Record | P = Packed/B = Binary | Constant or Edit Word | Sterling Sign Position

```
01ØO        D  2     Ø1
02ØO                         CUSTNOZ   23
03ØO                         NAME      53
04ØO                         STATE Z   59
05ØO                         CITY  Z   67
06ØO                         INVNO Z   79
07ØO                         MONTH Z   9Ø
08ØO                         DAY   Z   96
09ØO                         INVAMT   1Ø9 '$  ,  Ø.   '
10ØO        T  1     L1
11ØO                         GRPTOT B 1Ø9 '$  ,  Ø.   '
12ØO                                  11Ø '*'
13ØO        T  2     LR
14ØO                         TOTAL    1Ø9 '$  ,  Ø.   '
15ØO                                  111 '**'
   O
```

Figure 182. RPG Specification Forms-Program one (Part 3 of 3)

File Description Specifications Form

Two files (input and output) are described on this form. The input file INPUT (positions 7-11) is the primary file (position 16). It causes the end-of-job condition when it is depleted (position 17). The input records are fixed in length (position 19); the block length is 80 (positions 22 and 23); and the records are 80 characters in length (positions 26 and 27).

The output file OUTPUT is also defined on the File Description Specifications form. The format is variable; the block length is 132 and the records are 132 characters in length. The entry OF in positions 33-34 indicates that the output file defined on the line is to cause the overflow condition.

Input Specifications Form

The input file has a sequence of AA and, if position 1 contains the zone of a minus, record identifying indicator 01 is set on (as indicated by the entries in 19-20, 24 and 26-27). The locations of the fields which contain the input data are defined in positions 44-51.

The names of the input fields are entered in positions 53-58. Whenever a new customer number is read in, control level 1 is set on (positions 59-60).

Calculation Specifications Form

The contents of the field TOTAL are added to the contents of the field INVAMT, and the result is stored in TOTAL.

The Result Field has a length of seven positions, with two positions reserved for the decimal portion. The field GRPTOT is added to INVAMT, and the result is stored in GRPTOT which is seven positions long and has two decimal positions.

Output-Format Specifications Form

OUTPUT, the name of the file to which the records defined on the line belong, is entered under Filename on the first line of the form. In position 15 the output types, H, D, and T are entered to designate the heading, detail, and total lines.

The first heading line, ACCOUNTS RECEIVABLE REGISTER, prints either on the first page (1P) or overflow conditions (OF). The OR entered in position 14 and 15 of the second line allows for printing on the first page or on overflow. The other heading lines also print on these conditions.

When output indicator 01 is on, the field entered in Field Name will print in the positions indicated in positions 40-43. Zero suppression occurs on CUSTNO, STATE, CITY, INVNO, MONTH, and DAY.

The total lines are to print whenever control fields L1 or LR are on. The group total GRPTOT prints when L1 is ON, and after it is printed, the contents of GRPTOT are blanked out. The final total is printed when the LR (last record) indicator is on.

SAMPLE PROGRAM TWO

This is similar to the previous program. In this example, however, two input files are used. The transaction file is a card file with fields as shown in Figure 179 of the previous program. Another input file (master customer file), which is on tape, contains information about the firm's customers (Figure 183). The fields contained in the two input files are illustrated in Figure 184.

The program is to process 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 185.

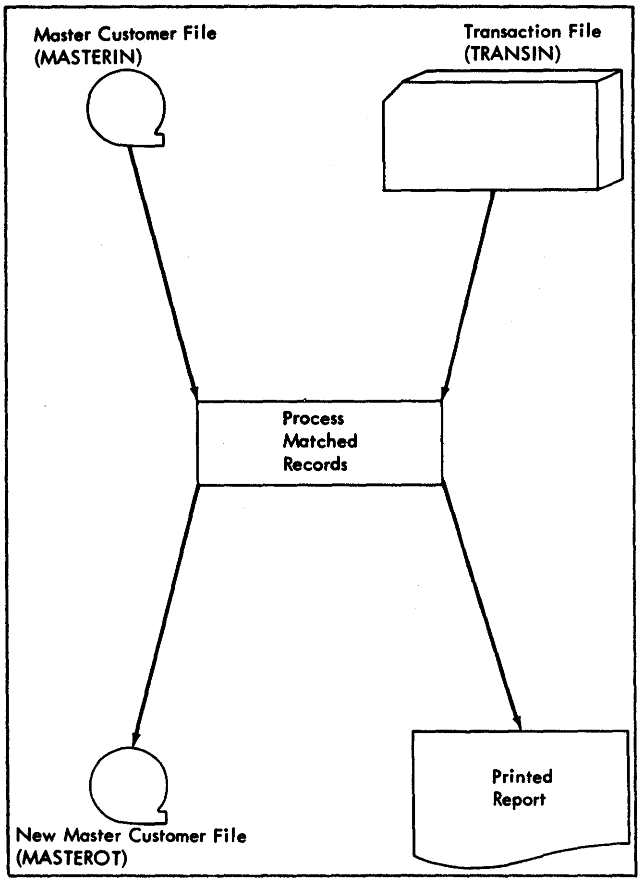


Figure 183. Sample Program Two

53717

Transaction File

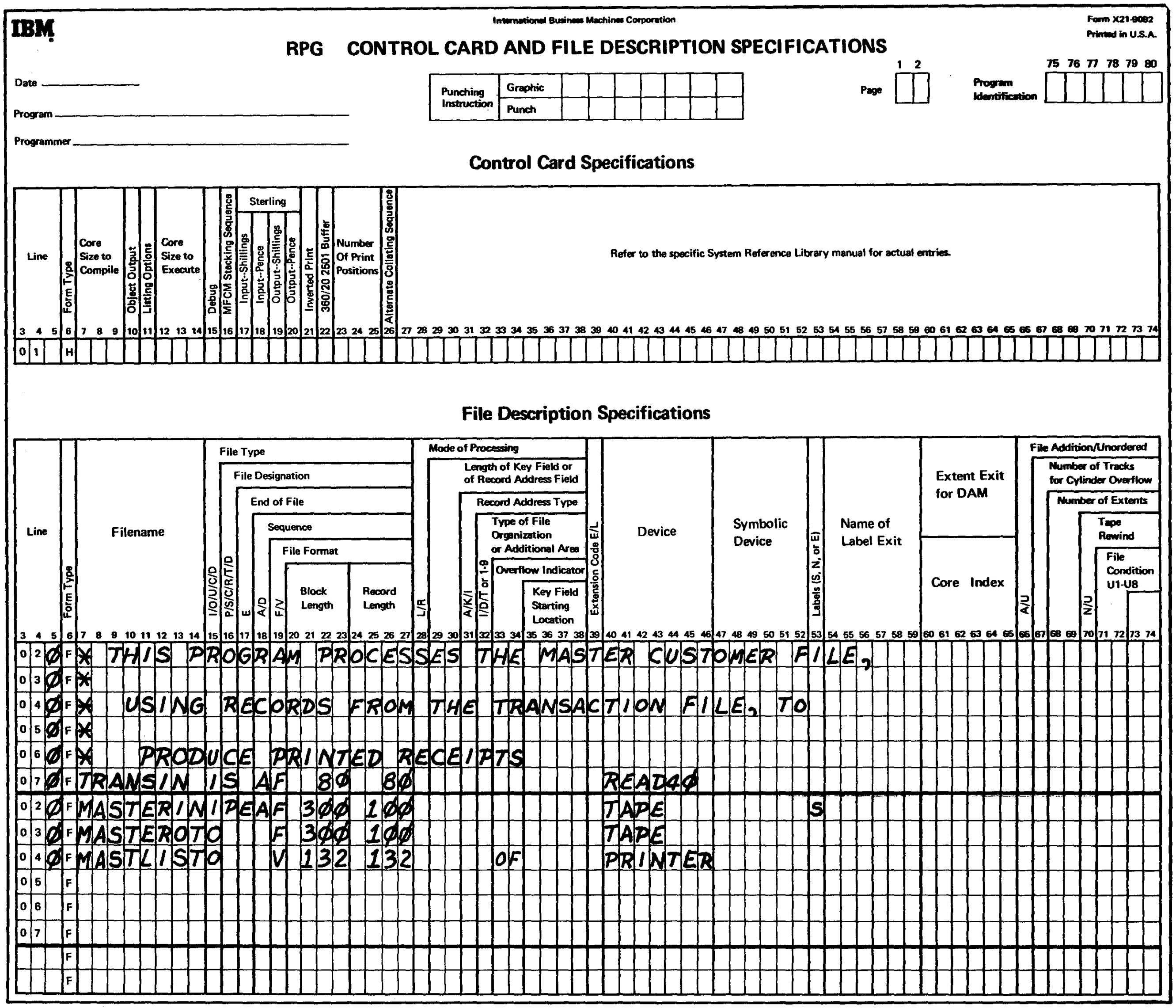
Field	Label	Card Columns
Code	Minus (-) Zone, or Plus (+) Zone	1
Customer Name	NAME	8-29
Invoice Date	MONTH	30-31
Invoice Date	DAY	32-33
Invoice Number	INVNO	34-38
Customer Number	CUSTNO	39-43
State	STATE	44-45
City	CITY	46-48
Invoice Amount / Amount Paid	AMT	74-80

Master Customer File

Field	Label	Location
Customer Number	CUSTNO	1-5
Customer Name	MASNAM	6-27
Street	MASTRT	28-46
City	MASCTY	47-57
State	MASTAT	58-62
Customer Balance	MASBAL	64-70
Date Of Last Payment	PAYDATE	71-76
Date Of Last Purchase	PAYPUR	77-82

Figure 184. Input Fields-Sample Program Two

53718



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

Program Identification 75 76 77 78 79 80

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 | Debug 15 | MFCM Stacking Sequence 16 | Sterling: Input-Shillings 17, Input-Pence 18, Output-Shillings 19, Output-Pence 20 | Inverted Print 21 | 360/20 2501 Buffer 22 | Number Of Print Positions 23 24 25 | Alternate Collating Sequence 26 | 27–74: Refer to the specific System Reference Library manual for actual entries.

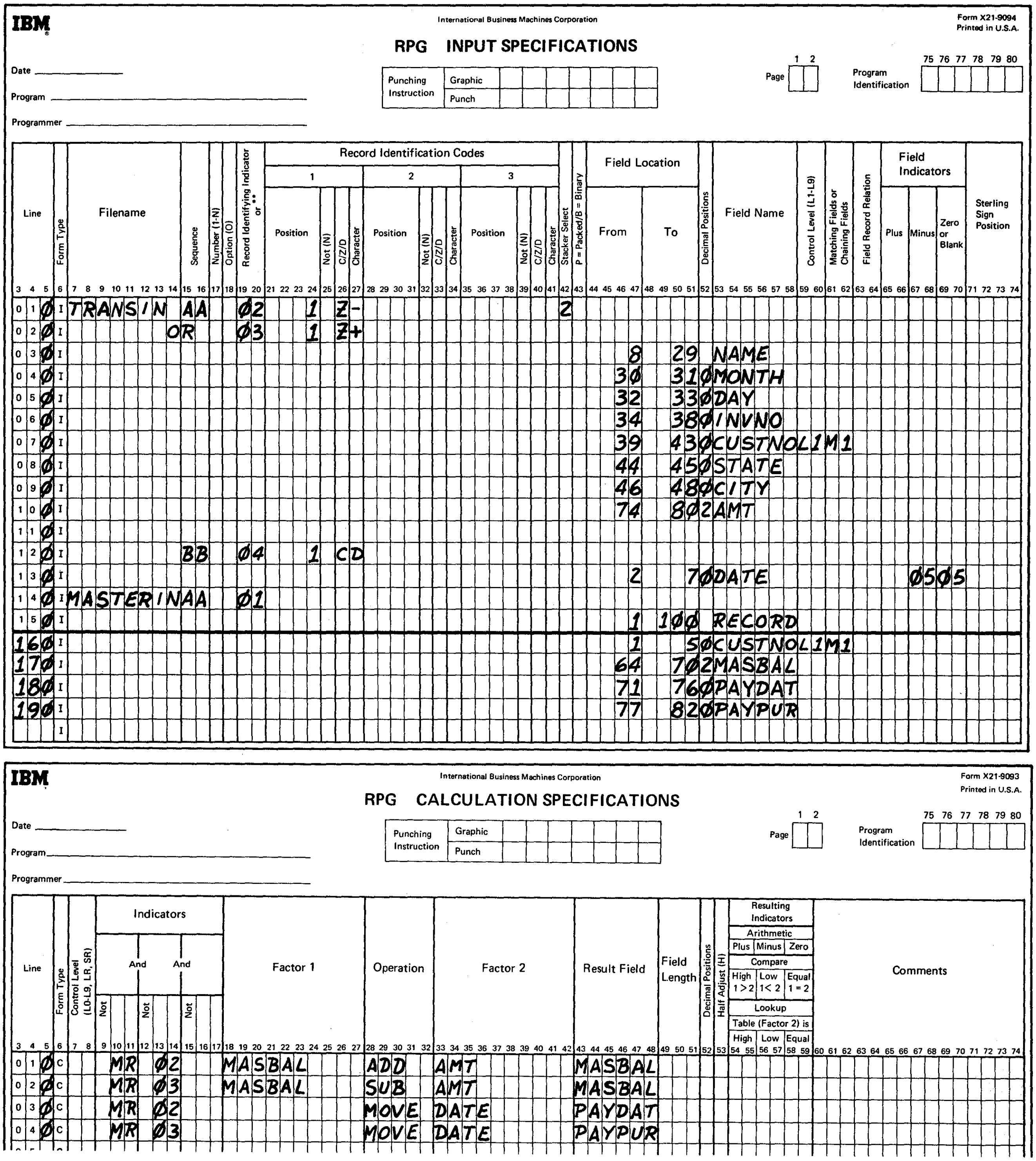
```
01H
```

File Description Specifications

Line 3 4 5 | Form Type 6 | Filename 7–14 | File Type: I/O/U/C/D 15 | File Designation: P/S/C/R/T/D 16 | End of File: E 17 | Sequence: A/D 18 | File Format: F/V 19 | Block Length 20–23 | Record Length 24–27 | Mode of Processing: L/R 28 | Length of Key Field or of Record Address Field 29 30 | Record Address Type: A/K/I 31 | Type of File Organization or Additional Area: I/D/T or 1-9 32 | Overflow Indicator 33 34 | Key Field Starting Location 35–38 | Extension Code E/L 39 | Device 40–46 | Symbolic Device 47–52 | Labels (S, N, or E) 53 | Name of Label Exit 54–59 | Extent Exit for DAM / Core Index 60–65 | File Addition/Unordered: A/U 66 | Number of Tracks for Cylinder Overflow 67 68 | Number of Extents 69 70 | Tape Rewind: R/U/N 70 | File Condition U1-U8 71 72 | 73 74

```
02ØF* THIS PROGRAM PROCESSES THE MASTER CUSTOMER FILE,
03ØF*
04ØF*   USING RECORDS FROM THE TRANSACTION FILE, TO
05ØF*
06ØF*    PRODUCE PRINTED RECEIPTS
07ØFTRANSIN IS AF  80  80            READ40
02ØFMASTERINIPEAF 300 100            TAPE         S
03ØFMASTEROTO   F 300 100            TAPE
04ØFMASTLISTO   V 132 132     OF     PRINTER
05 F
06 F
07 F
    F
    F
```

Figure 185. Specifications for Sample Program Two (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 1 2

Program Identification 75 76 77 78 79 80

Line (3-5)	Form Type (6)	Filename (7-14)	Sequence (15-16)	Number (1-N) (17)	Option (O) (18)	Record Identifying Indicator or ** (19-20)	Record Identification Codes 1: Position (21-24)	Not (N) (25)	C/Z/D (26)	Character (27)	2: Position (28-31)	Not (N) (32)	C/Z/D (33)	Character (34)	3: Position (35-38)	Not (N) (39)	C/Z/D (40)	Character (41)	Stacker Select (42)	P = Packed/B = Binary (43)	Field Location From (44-47)	To (48-51)	Decimal Positions (52)	Field Name (53-58)	Control Level (L1-L9) (59-60)	Matching Fields or Chaining Fields (61-62)	Field Record Relation (63-64)	Field Indicators Plus (65-66)	Minus (67-68)	Zero or Blank (69-70)	Sterling Sign Position (71-74)
01Ø	I	TRANSIN	AA			Ø2	1		Z	-									2												
02Ø	I	OR				Ø3	1		Z	+																					
03Ø	I																				8	29		NAME							
04Ø	I																				30	31	Ø	MONTH							
05Ø	I																				32	33	Ø	DAY							
06Ø	I																				34	38	Ø	INVNO							
07Ø	I																				39	43	Ø	CUSTNO	L1	M1					
08Ø	I																				44	45	Ø	STATE							
09Ø	I																				46	48	Ø	CITY							
10Ø	I																				74	80	2	AMT							
11Ø	I																														
12Ø	I		BB			Ø4	1		C	D																					
13Ø	I																				2	7	Ø	DATE					Ø5	Ø5	
14Ø	I	MASTERIN	AA			Ø1																									
15Ø	I																				1	1ØØ		RECORD							
16Ø	I																				1	5	Ø	CUSTNO	L1	M1					
17Ø	I																				64	70	2	MASBAL							
18Ø	I																				71	76	Ø	PAYDAT							
19Ø	I																				77	82	Ø	PAYPUR							
	I																														

IBM

International Business Machines Corporation

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

RPG CALCULATION SPECIFICATIONS

Date ____
Program ____
Programmer ____

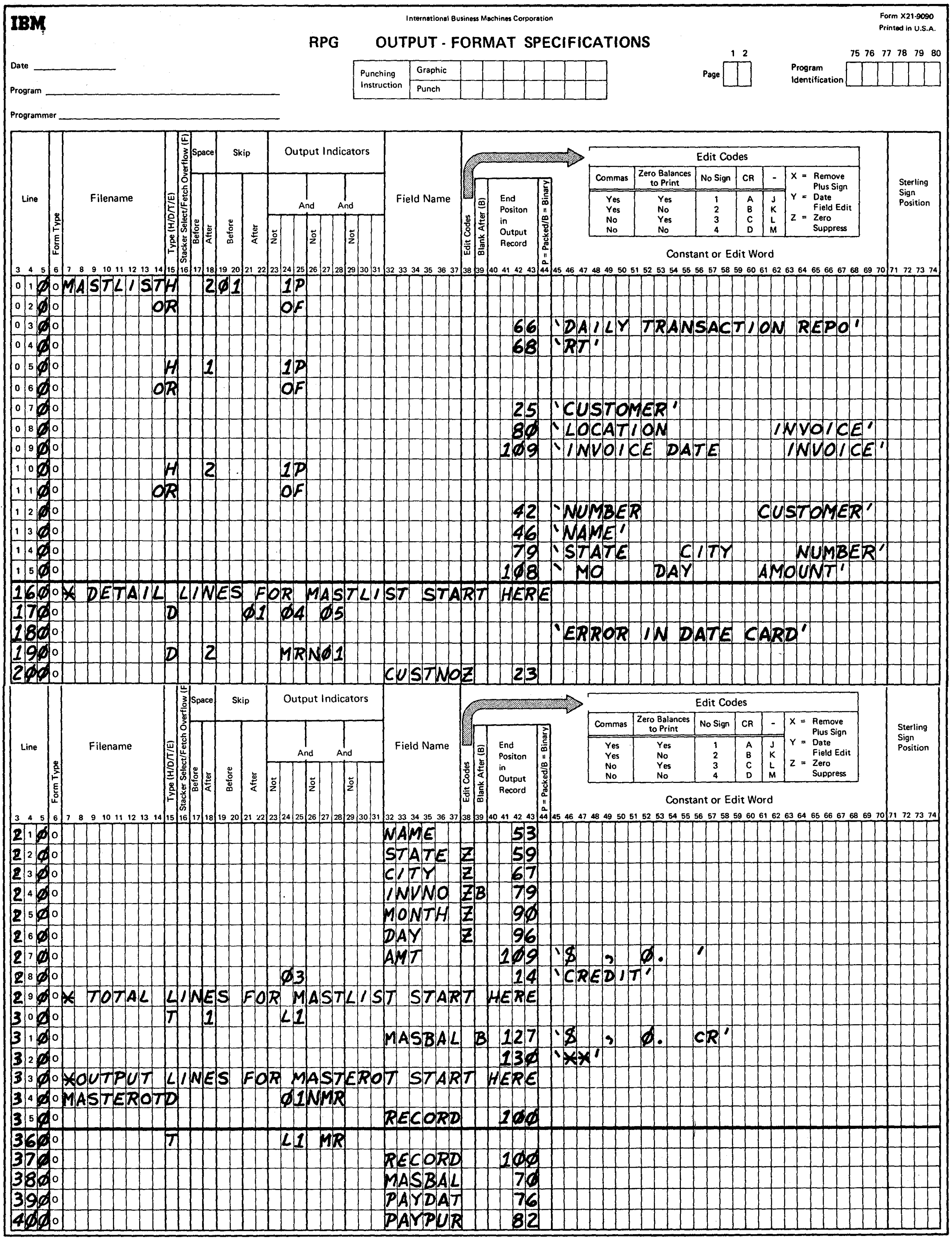
Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line (3-5)	Form Type (6)	Control Level (L0-L9, LR, SR) (7-8)	Indicators: Not (9)	(10-11)	And: Not (12)	(13-14)	And: Not (15)	(16-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field (43-48)	Field Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Plus / High 1>2 (54-55)	Minus / Low 1<2 (56-57)	Zero / Equal 1=2 (58-59)	Comments (60-74)
01Ø	C			MR		Ø2			MASBAL	ADD	AMT	MASBAL							
02Ø	C			MR		Ø3			MASBAL	SUB	AMT	MASBAL							
03Ø	C			MR		Ø2				MOVE	DATE	PAYDAT							
04Ø	C			MR		Ø3				MOVE	DATE	PAYPUR							

Figure 185. Specifications for Sample Program Two (Part 2 of 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 (1 2)

Program Identification (75 76 77 78 79 80)

Line (3-5) | Form Type (6) | Filename (7-14) | Type (H/D/T/E) (15) | Stacker Select/Fetch Overflow (F) (16) | Space: Before (17), After (18) | Skip: Before (19-20), After (21-22) | Output Indicators: Not (23) 24-25, And Not (26) 27-28, And Not (29) 30-31 | Field Name (32-37) | Edit Codes (38) | Blank After (B) (39) | End Positon in Output Record (40-43) | P = Packed/B = Binary (44) | Constant or Edit Word (45-70) | Sterling Sign Position (71-74)

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

```
01ØOMASTLISTH  2Ø1   1P
02ØO       OR        OF
03ØO                                   66 'DAILY TRANSACTION REPO'
04ØO                                   68 'RT'
05ØO        H  1     1P
06ØO       OR        OF
07ØO                                   25 'CUSTOMER'
08ØO                                   8Ø 'LOCATION        INVOICE'
09ØO                                  1Ø9 'INVOICE DATE     INVOICE'
10ØO        H  2     1P
11ØO       OR        OF
12ØO                                   42 'NUMBER         CUSTOMER'
13ØO                                   46 'NAME'
14ØO                                   79 'STATE    CITY     NUMBER'
15ØO                                  1Ø8 ' MO    DAY     AMOUNT'
16ØO* DETAIL LINES FOR MASTLIST START HERE
17ØO        D     Ø1 Ø4 Ø5
18ØO                                      'ERROR IN DATE CARD'
19ØO        D  2     MRNØ1
2ØØO                         CUSTNOZ   23
```

Line (3-5) | Form Type (6) | Filename (7-14) | Type (H/D/T/E) (15) | Stacker Select/Fetch Overflow (F) (16) | Space: Before (17), After (18) | Skip: Before (19-20), After (21-22) | Output Indicators: Not (23) 24-25, And Not (26) 27-28, And Not (29) 30-31 | Field Name (32-37) | Edit Codes (38) | Blank After (B) (39) | End Positon in Output Record (40-43) | P = Packed/B = Binary (44) | Constant or Edit Word (45-70) | Sterling Sign Position (71-74)

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

```
21ØO                         NAME      53
22ØO                         STATE Z   59
23ØO                         CITY  Z   67
24ØO                         INVNO ZB  79
25ØO                         MONTH Z   9Ø
26ØO                         DAY   Z   96
27ØO                         AMT      1Ø9 '$  ,  Ø.  '
28ØO                 Ø3                14 'CREDIT'
29ØO* TOTAL LINES FOR MASTLIST START HERE
3ØØO        T  1     L1
31ØO                         MASBAL B 127 '$  ,  Ø.  CR'
32ØO                                  13Ø '**'
33ØO*OUTPUT LINES FOR MASTEROT START HERE
34ØOMASTEROTD        Ø1NMR
35ØO                         RECORD   1ØØ
36ØO        T        L1 MR
37ØO                         RECORD   1ØØ
38ØO                         MASBAL    7Ø
39ØO                         PAYDAT    76
4ØØO                         PAYPUR    82
```

Figure 185. Specifications for Sample Program Two (Part 3 of 3)

File Description Specifications Form

The four files are defined on this form. The two input files TRANSIN and MASTERIN are defined, and the two output files MASTEROT, which is the updated master file, and MASTLIST, which is the printed report, are defined on the File Description Specifications form. TRANSIN is designated as the secondary file because it may not contain transactions involving all the customers on the master customer file. TRANSIN is ascending in order. It has fixed-length records which are 80 characters long. The block length is 80.

MASTERIN is the primary file. If the transaction file does not have a corresponding customer number (specified by the Matching Fields specification on the Input Specifications form), the master file is processed. Processing continues until all the records in the master customer file have been processed (indicated by the E in position 17). The input records contained in the master customer file are ascending in order, fixed in length, and have a block length of 300.

The file MASTLIST is the printed report. The format is variable. The length of the records can be 132. The overflow entry in positions 33-34 indicates that the overflow condition is to occur on the MASTLIST printer.

The output file MASTEROT is a tape file which contains the updated master customer records. The output records are fixed in length, and are 100 characters in length. The block length is 300.

Note: A blank in position 53 indicates that there are no labels in the output file.

Input Specifications Form

The two input files TRANSIN and MASTERIN are defined on the Input Specification form.

TRANSIN: The input records may be obtained from three types of cards.

Sequence AA has been assigned to two types. If card position 1 contains the zone of a minus, record identifying indicator 02 is set on. If card position 1 contains the zone of a plus, indicator 03 is set on. The cards that have a minus in position 1 are selected into the 2 pocket (position 42). The locations of the input records and their labels are defined in positions 46-58 of the form.

The field CUSTNO (customer number) has entries in positions 59-60 (Control Level) and positions 61-62 (Matching Fields) of the Input Specifications form. Whenever a new customer number is read in, control level 1 (L1) is set 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 customer number will be used to match another field (CUSTNO) in the MASTERIN file.

The first card in the transaction file is a date card. It is assigned sequence BB. Whenever position 1 contains a D, indicator 04 is set on. The date is contained in positions 2-7 of the card. Indicator 05 is set on whenever these positions are zero or minus.

MASTERIN: The tape 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 referenced on the Output-Format Specifications form. CUSTNO of the master file corresponds to CUSTNO in the transaction file. Whenever a new customer number is read in, L1 is set. The entry M1 indicates that the customer number in the master file will be matched with the customer number in the transaction file.

Calculation Specifications Form

Whenever the matching record indicator MR is on and indicator 02 is on, the contents of the field AMT are added to the MASBAL. The result is stored in MASBAL. The date is moved to the field PAYDAT.

Whenever the matching record indicator MR is on and indicator 03 is on, AMT is subtracted from MASBAL, and the result is stored in MASBAL. The date is moved to PAYPUR.

Output-Format Specifications Form

The output file MASTEROT is the updated tape 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 tape input record will be written out on tape. This condition results because there was 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 out on the updated tape file when no information is present in the transaction file.

If, however, L1 and MR are on, the input record is written out on tape. The entire record is written, but 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 master customer file, but the customer's name and address are retained.

The specifications for the printed report are listed under the name of the output file MASTLIST.

SAMPLE PROGRAM THREE

The third sample (Figures 186 & 187) illustrates some of the more complex capabilities of RPG.

1. Processing chained records on a direct access storage device (DASD).
2. Updating records on a DASD.
3. Multiple input and output files.
4. Creating exception records.
5. Providing for processing when a record in the chained file is missing.

CARDIN

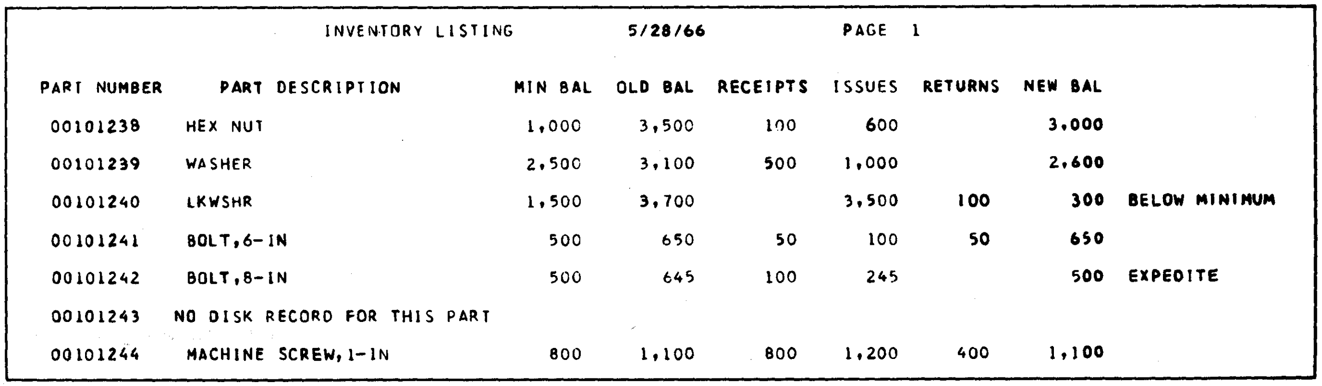
Field	Card Column
Date Card	
Code	1
Date	2-7
Not Used	8-80
Transaction Card	
Code	1
Part Number	2-9
Receipts	10-13
Issues	14-17
Returns	18-21
Not Used	22-80

CARDOUT

Field	Card Column
Code	1
Part Number	2-9
Description	10-36
Vendor Number	37-42
Date	43-48
Not Used	49-80

INVFIL

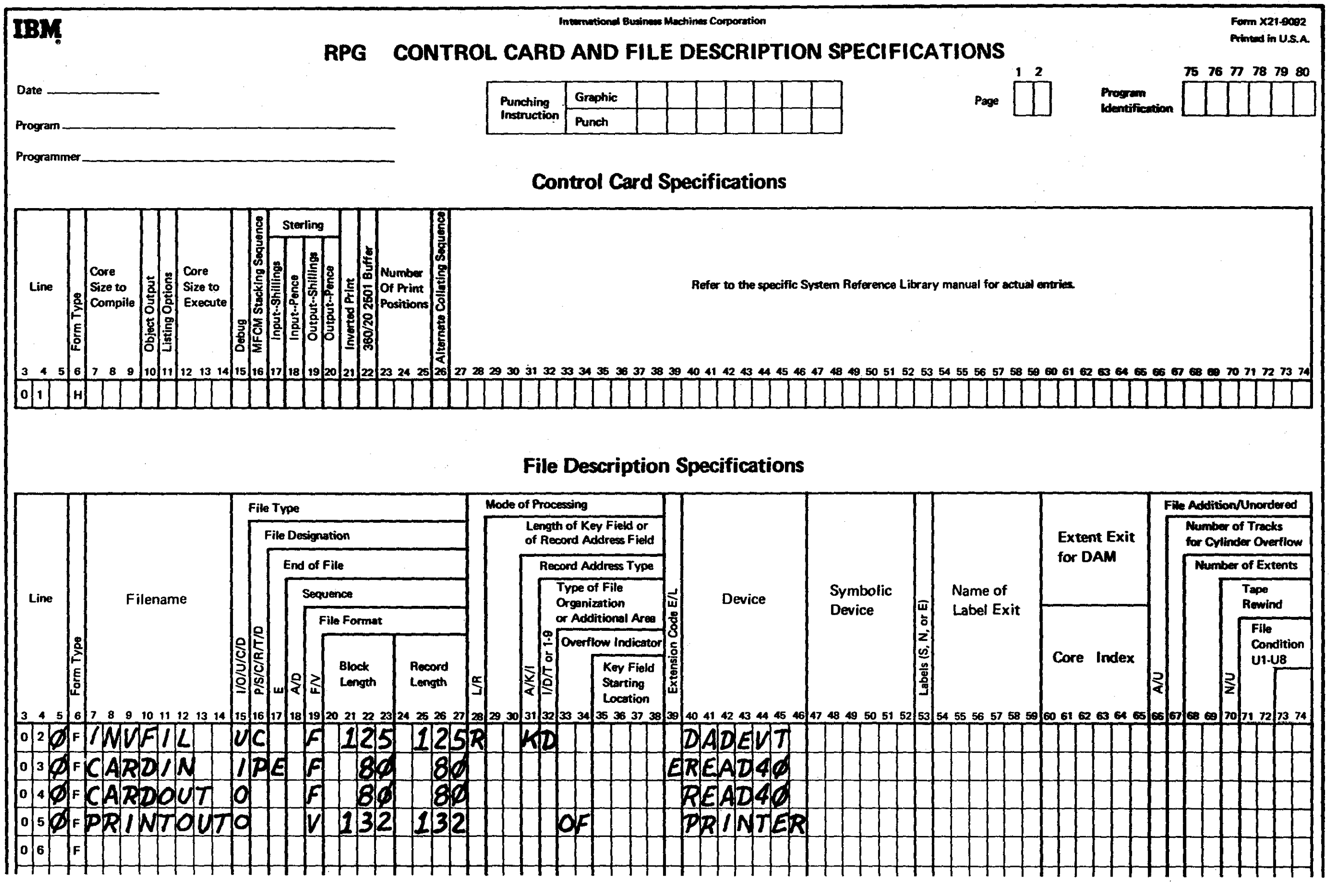
Field	Position
Code	1
Part Number	2-9
Description	10-36
Vendor Number	37-42
Not Specified	43-49
Minimum	50-53
On Hand	54-58
Not Specified	59-125



```
                        INVENTORY LISTING          5/28/66           PAGE  1

PART NUMBER       PART DESCRIPTION         MIN BAL   OLD BAL  RECEIPTS  ISSUES  RETURNS  NEW BAL

 00101238     HEX NUT                       1,000     3,500      100      600              3,000
 00101239     WASHER                        2,500     3,100      500    1,000              2,600
 00101240     LKWSHR                        1,500     3,700             3,500     100        300  BELOW MINIMUM
 00101241     BOLT,6-IN                       500       650       50      100      50        650
 00101242     BOLT,8-IN                       500       645      100      245                500  EXPEDITE
 00101243    NO DISK RECORD FOR THIS PART
 00101244     MACHINE SCREW,1-IN              800     1,100      800    1,200     400      1,100
```

Figure 186. Input and Output Formats for Sample Program Three

53719



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)

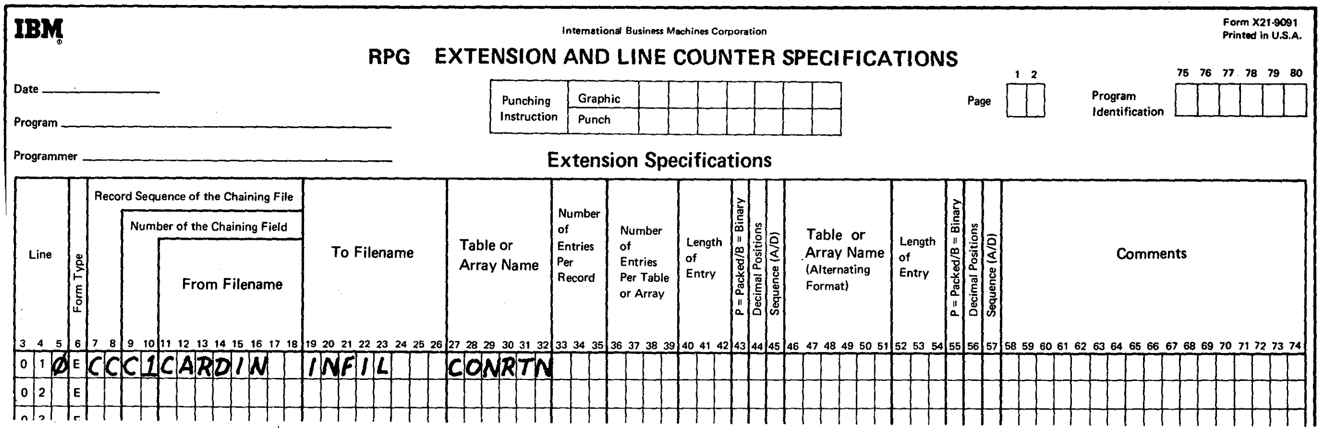
Program Identification (75 76 77 78 79 80)

Control Card Specifications

Line (3 4 5)	Form Type (6)	Core Size to Compile (7-9)	Object Output (10)	Listing Options (11)	Core Size to Execute (12-14)	Debug (15)	MFCM Stacking Sequence (16)	Sterling: Input-Shillings (17)	Input-Pence (18)	Output-Shillings (19)	Output-Pence (20)	Inverted Print (21)	360/20 2501 Buffer (22)	Number Of Print Positions (23-25)	Alternate Collating Sequence (26)	27-74: Refer to the specific System Reference Library manual for actual entries.
0 1	H															

File Description Specifications

Line (3 4 5)	Form Type (6)	Filename (7-14)	File Type I/O/U/C/D (15)	File Designation P/S/C/R/T/D (16)	End of File E (17)	Sequence A/D (18)	File Format F/V (19)	Block Length (20-23)	Record Length (24-27)	Mode of Processing L/R (28)	Length of Key Field or of Record Address Field (29-30)	Record Address Type A/K/I (31)	Type of File Organization or Additional Area I/D/T or 1-9 (32)	Overflow Indicator (33-34)	Key Field Starting Location (35-38)	Extension Code E/L (39)	Device (40-46)	Symbolic Device (47-52)	Labels (S, N, or E) (53)	Name of Label Exit (54-59)	Extent Exit for DAM / Core Index (60-65)	File Addition/Unordered A/U (66)	Number of Tracks for Cylinder Overflow (67-68)	Number of Extents (69)	Tape Rewind N/U (70)	File Condition U1-U8 (71-72)	73-74
0 2 Ø	F	INVFIL	U	C			F	125	125	R		K	D				DADEVT										
0 3 Ø	F	CARDIN	I	P	E		F	8Ø	8Ø							E	READ4Ø										
0 4 Ø	F	CARDOUT	O				F	8Ø	8Ø								READ4Ø										
0 5 Ø	F	PRINTOUT	O				V	132	132					OF			PRINTER										
0 6	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

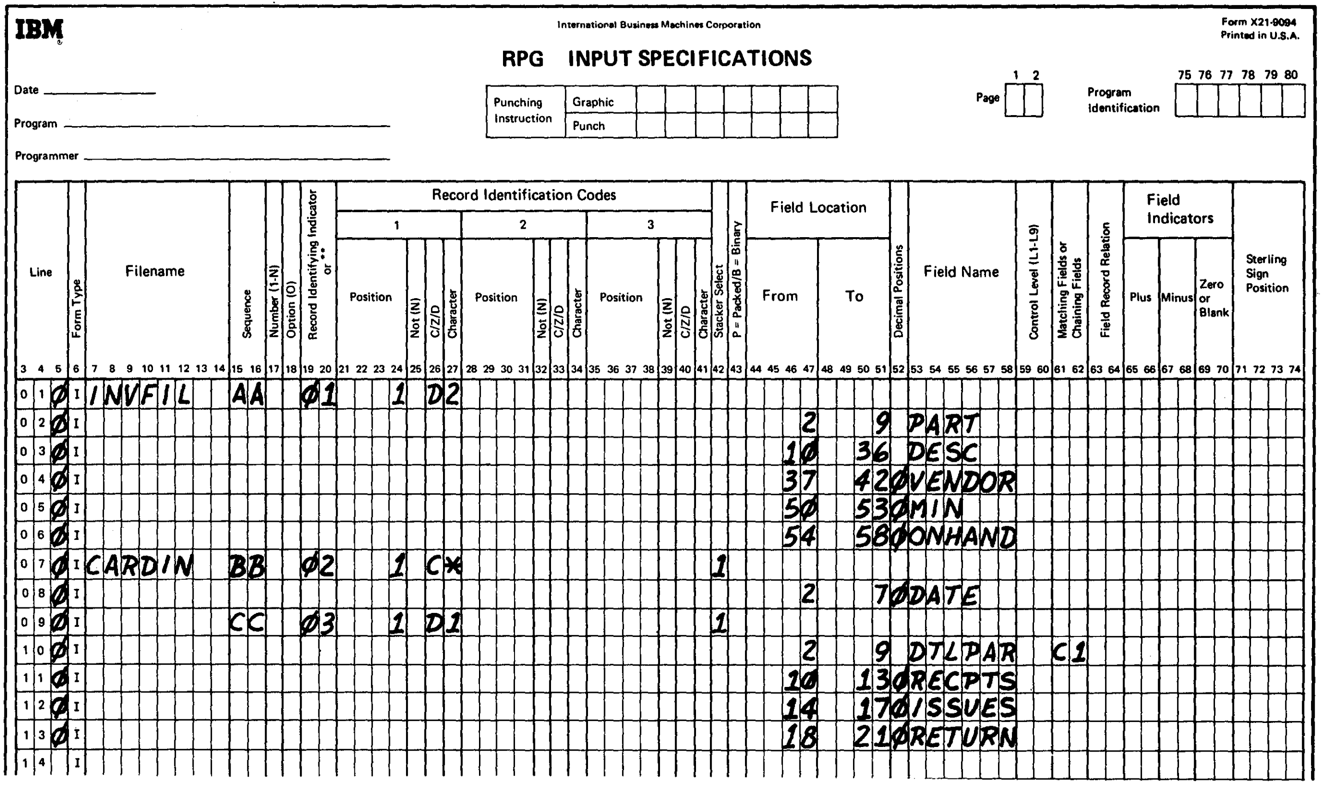
Page (1 2)

Program Identification (75 76 77 78 79 80)

Extension Specifications

Line (3 4 5)	Form Type (6)	Record Sequence of the Chaining File (7-8)	Number of the Chaining Field (9-10)	From Filename (11-18)	To Filename (19-26)	Table or Array Name (27-32)	Number of Entries Per Record (33-35)	Number of Entries Per Table or Array (36-39)	Length of Entry (40-42)	P = Packed/B = Binary (43)	Decimal Positions (44)	Sequence (A/D) (45)	Table or Array Name (Alternating Format) (46-51)	Length of Entry (52-54)	P = Packed/B = Binary (55)	Decimal Positions (56)	Sequence (A/D) (57)	Comments (58-74)
0 1 Ø	E	CC	C1	CARDIN	INFIL	CONRTN												
0 2	E																	
0 3	E																	

igure 187. Specifications for Sample Program Three (Part 1 of 5)



IBM

International Business Machines Corporation

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

RPG INPUT SPECIFICATIONS

Date ____________

Program ____________

Programmer ____________

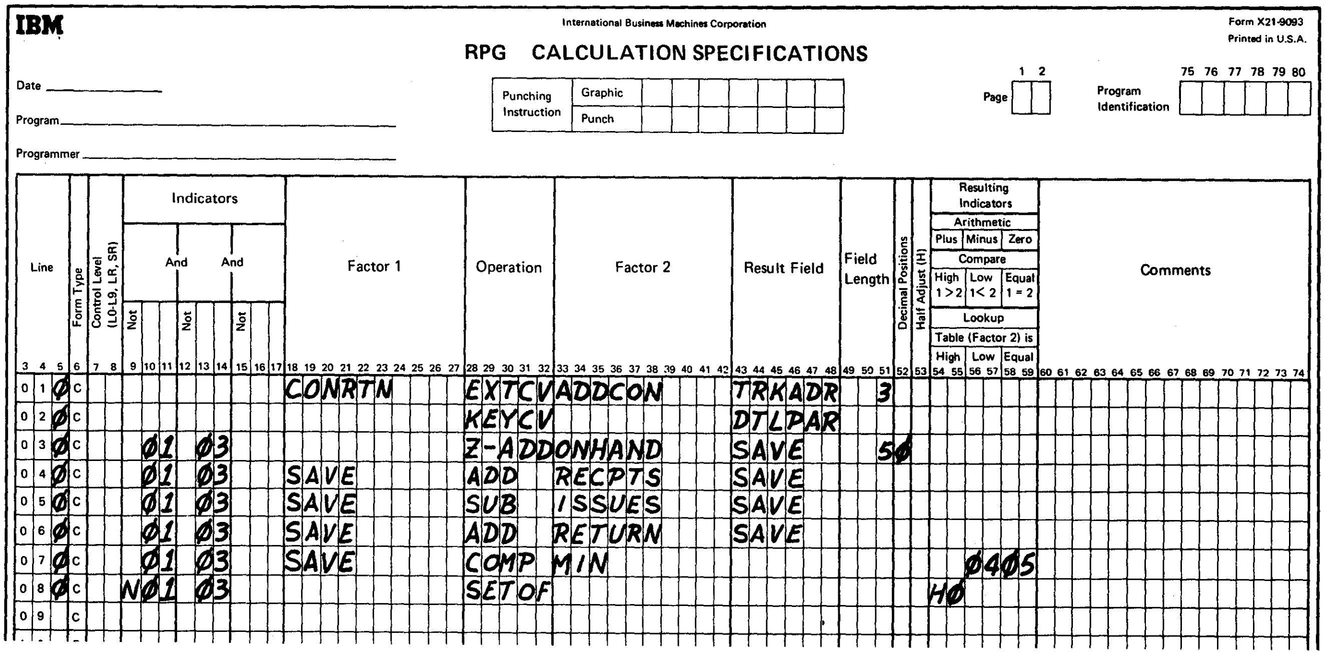
Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	1 Position	1 Not (N)	1 C/Z/D	1 Character	2 Position	2 Not (N)	2 C/Z/D	2 Character	3 Position	3 Not (N)	3 C/Z/D	3 Character	Stacker Select	P = Packed/B = Binary	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Field Indicators Minus	Field Indicators Zero or Blank	Sterling Sign Position
01ø	I	INVFIL	AA			ø1	1		D	2																					
02ø	I																				2	9		PART							
03ø	I																				1ø	36		DESC							
04ø	I																				37	42	ø	VENDOR							
05ø	I																				5ø	53	ø	MIN							
06ø	I																				54	58	ø	ONHAND							
07ø	I	CARDIN	BB			ø2	1		C	*									1												
08ø	I																				2	7	ø	DATE							
09ø	I		CC			ø3	1		D	1									1												
10ø	I																				2	9		DTLPAR		C1					
11ø	I																				1ø	13	ø	RECPTS							
12ø	I																				14	17	ø	ISSUES							
13ø	I																				18	21	ø	RETURN							
14	I																														

Figure 187. Specifications for Sample Program Three (Part 2 of 5)



IBM

International Business Machines Corporation

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

RPG CALCULATION SPECIFICATIONS

Date ____________

Program ____________

Programmer ____________

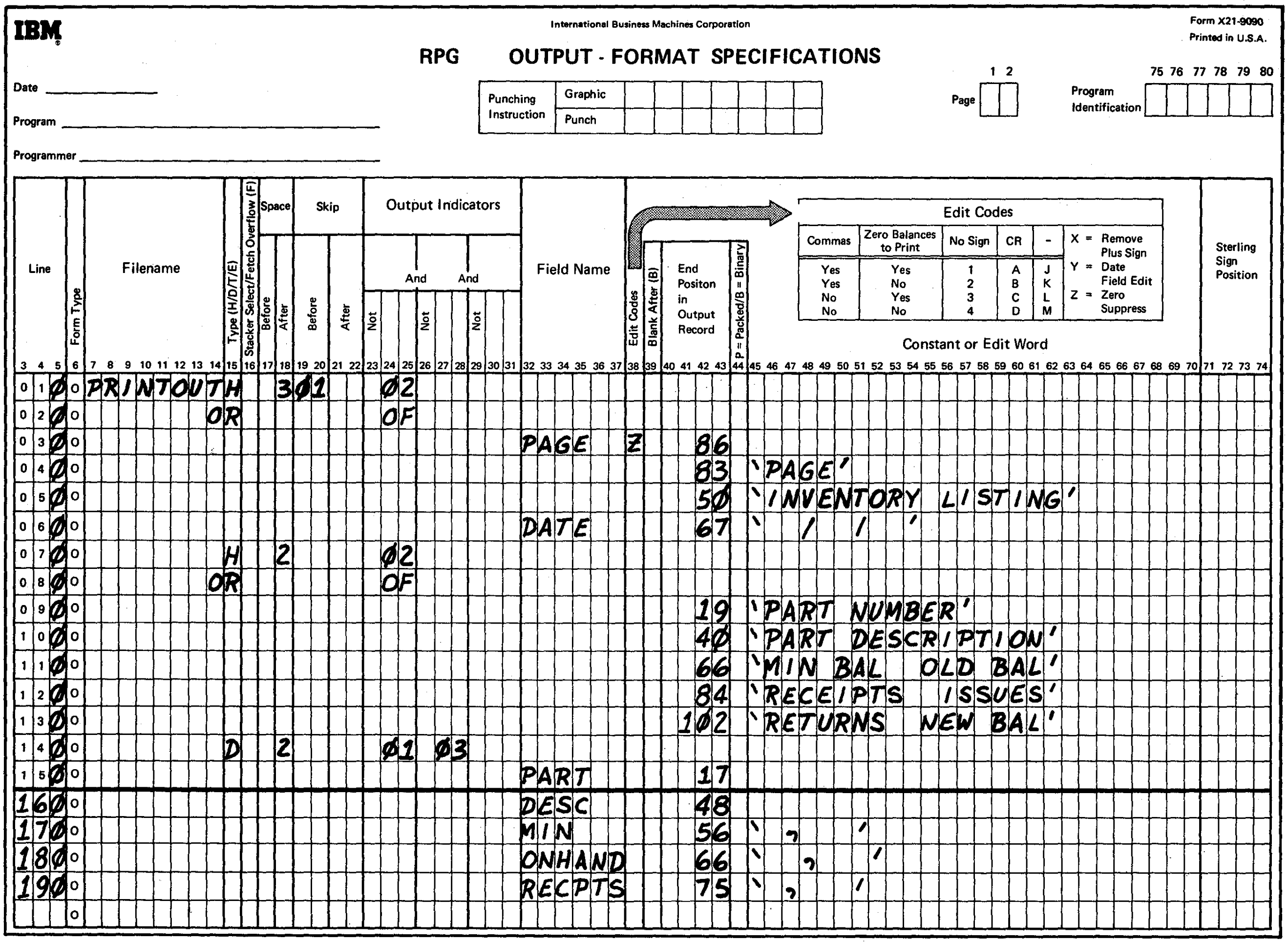
Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L0-L9, LR, SR)	Not	Indicators	And Not	And Indicators	And Not	And Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Plus / High 1>2	Minus / Low 1<2	Zero / Equal 1=2	Comments
01ø	C								CONRTN	EXTCV	ADDCON	TRKADR	3						
02ø	C									KEYCV		DTLPAR							
03ø	C			ø1		ø3				Z-ADD	ONHAND	SAVE	5	ø					
04ø	C			ø1		ø3			SAVE	ADD	RECPTS	SAVE							
05ø	C			ø1		ø3			SAVE	SUB	ISSUES	SAVE							
06ø	C			ø1		ø3			SAVE	ADD	RETURN	SAVE							
07ø	C			ø1		ø3			SAVE	COMP	MIN						ø4	ø5	
08ø	C		N	ø1		ø3				SETOF						Hø			
09	C																		

Figure 187. Specifications for Sample Program Three (Part 3 of 5)



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 (1 2) □□

Program Identification (75 76 77 78 79 80) □□□□□□

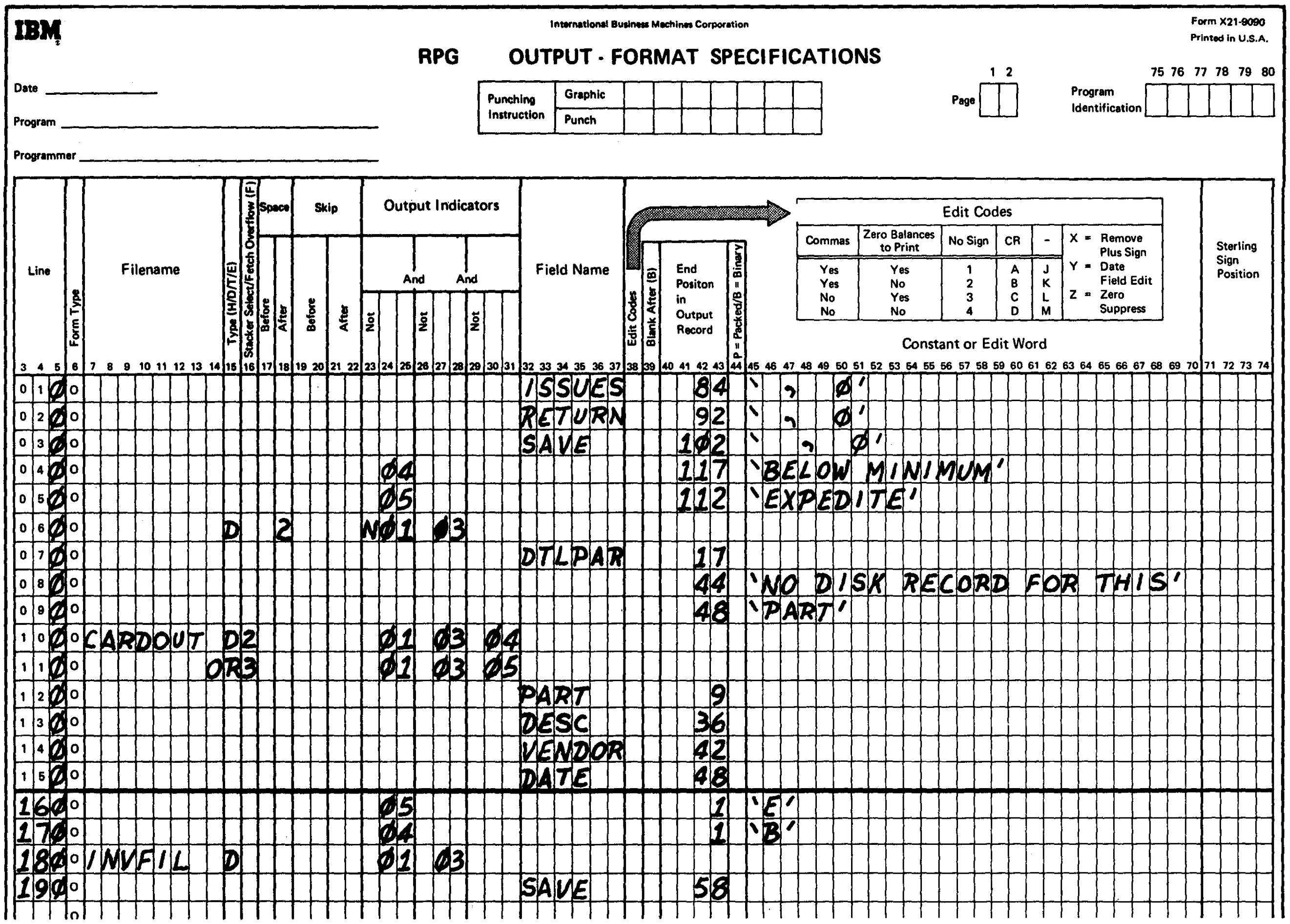
Edit Codes

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 (3-5)	Form Type (6)	Filename (7-14)	Type (H/D/T/E) (15)	Stacker Select/Fetch Overflow (F) (16)	Space Before (17)	Space After (18)	Skip Before (19-20)	Skip After (21-22)	Output Indicators (23-25)	And Output Indicators (26-28)	And Output Indicators (29-31)	Field Name (32-37)	Edit Codes (38)	Blank After (B) (39)	End Positon in Output Record (40-43)	P = Packed/B = Binary (44)	Constant or Edit Word (45-70)	Sterling Sign Position (71-74)
01Ø	O	PRINTOUT	H			3	Ø1		Ø2									
02Ø	O	OR							OF									
03Ø	O											PAGE	Z		86			
04Ø	O														83		'PAGE'	
05Ø	O														5Ø		'INVENTORY LISTING'	
06Ø	O											DATE			67		' / / '	
07Ø	O		H			2			Ø2									
08Ø	O	OR							OF									
09Ø	O														19		'PART NUMBER'	
10Ø	O														4Ø		'PART DESCRIPTION'	
11Ø	O														66		'MIN BAL OLD BAL'	
12Ø	O														84		'RECEIPTS ISSUES'	
13Ø	O														1Ø2		'RETURNS NEW BAL'	
14Ø	O		D			2			Ø1	Ø3								
15Ø	O											PART			17			
16Ø	O											DESC			48			
17Ø	O											MIN			56		' , '	
18Ø	O											ONHAND			66		' , '	
19Ø	O											RECPTS			75		' , '	
	O																	

Figure 187. Specifications for Sample Program Three (Part 4 of 5)



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 (1 2) □□

Program Identification (75 76 77 78 79 80) □□□□□□

Edit Codes

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

Line (3-5)	Form Type (6)	Filename (7-14)	Type (H/D/T/E) (15)	Stacker Select/Fetch Overflow (F) (16)	Space Before (17)	Space After (18)	Skip Before (19-20)	Skip After (21-22)	Output Indicators Not (23)	(24-25)	And Not (26)	(27-28)	And Not (29)	(30-31)	Field Name (32-37)	Edit Codes (38)	Blank After (B) (39)	End Position in Output Record (40-43)	P = Packed/B = Binary (44)	Constant or Edit Word (45-70)	Sterling Sign Position (71-74)
01Ø	O														ISSUES			84		' , Ø'	
02Ø	O														RETURN			92		' , Ø'	
03Ø	O														SAVE			1Ø2		' , Ø'	
04Ø	O									Ø4								117		'BELOW MINIMUM'	
05Ø	O									Ø5								112		'EXPEDITE'	
06Ø	O		D			2			N	Ø1		Ø3									
07Ø	O														DTLPAR			17			
08Ø	O																	44		'NO DISK RECORD FOR THIS'	
09Ø	O																	48		'PART'	
10Ø	O	CARDOUT	D	2						Ø1		Ø3		Ø4							
11Ø	O		OR	3						Ø1		Ø3		Ø5							
12Ø	O														PART			9			
13Ø	O														DESC			36			
14Ø	O														VENDOR			42			
15Ø	O														DATE			48			
16Ø	O									Ø5								1		'E'	
17Ø	O									Ø4								1		'B'	
18Ø	O	INVFIL	D							Ø1		Ø3									
19Ø	O														SAVE			58			
	O																				

Figure 187. Specifications for Sample Program Three (Part 5 of 5)

File Description Specifications Form

The primary input file, designated by a P in position 16, is on the card reader and is labeled CARDIN. This card file also acts as a transaction file for the updating of the inventory records contained on the file labeled INVFIL. The U in position 15 of the specification for INVFIL indicates that this DASD file is updated (used for both input and output). The file has direct organization indicated by the D in position 32, and is processed randomly as specified by the R in position 28.

The E in position 17 of the CARDIN file description entry directs the program to end execution when the last record of the CARDIN file has been processed. The E in position 39 indicates that an extension specification has been coded for the CARDIN file.

The output files are an exception card file labeled CARDOUT and the printed transaction report labeled PRINTOUT.

CARDIN and CARDOUT consist of 80-character records. INVFIL has a block length of 125 characters and a record length of 125 characters. Record length for PRINTOUT is variable, with a maximum of 132 characters.

Extension Specifications Form

The Extension Specifications form identifies CARDIN as the chaining file, indicating that record sequence CC contains the chaining field C1. INVFIL is the chained file and CONRTN is the RPG internal label of the specifications on the Calculation Specifications form for the external conversion routine that processes the chaining field to obtain the relative track address in the DASD file.

Input Specifications Form

The input card and DASD files are described on the Input Specifications form. Each field that is used is defined. The C1 in positions 61-62 of the transaction card file designates the field labeled DTLPAR as the chaining field. The conversion routine CONRTN, indicated on the Calculation Specifications form, makes the DTLPAR field available to the user's external conversion routine.

Calculation Specifications Form

Two sets of calculation specifications are described. After the label CONRTN, the operation code EXTCV is used to indicate that the conversion routine is to be performed. ADDCON is the symbolic address of the external conversion routine. TRKADR, the field that contains the relative track address in the DASD file, must have a length of three. The external routine can be a member of a library or in the form of an object module card deck. As a card deck it is included with the RPG object module as input to the linkage editor.

The operation code KEYCV states that the record key of the DASD record can be obtained directly from the field DTLPAR.

The calculations in lines 030-070 require the presence of corresponding chaining (CARDIN) and chained (INVFIL) records. The condition is indicated by the simultaneous setting of record identifying indicators 01 and 03. The ONHAND field from the INVFIL record is zero and added to 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 drops below the minimum, resulting indicator 04 is set on.

The entry on line 080 resets the H0 indicator which would otherwise terminate processing when a record in the chained file is missing. The printed output notes this error condition.

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). The OF indicator governs 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 the entries in lines 150 of Figure 187 (part 4) through 050 of Figure 187 (part 5) requires the presence of both the CARDIN and INVFIL records (record identifying indicators 01 and 03 are on). If resulting indicator 04 is on, the words BELOW MINIMUM indicate the stock violation. If indicator 05 is on, the message EXPEDITE is added.

The output line described in PRINTOUT entries 060-090 is the message printed when the error condition occurs of a CARDIN record and no corresponding INVFIL record. This condition is identified by record identifying indicator 01 being off when indicator 03 is on.

The exception file CARDOUT has a card punched for each transaction that results in a below-minimum of at-minimum stock level. These cards contain the part number and description, vendor number, date, and the E or B code for EXPEDITE or BELOW MINIMUM. Below minimum cards, identified by the simultaneous on settings of indicators 01, 03, and 04, are selected to stacker number P2. At-minimum cards, with indicators 01, 03, and 05 on, are selected to stacker number RP3.

Lines 180 and 190 provide for the updating and writing out 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 from the work area SAVE, and the record is written.

The IBM System/360 Operating System consists of a control program and processing programs. The control program supervises execution of all processing programs, such as the RPG compiler, and all problem programs, such as an RPG program. Therefore, to execute an RPG program the programmer must first communicate with the operating system. The medium of communication between the programmer and the operating system is the job control language.

Job control language statements define two units of work to the operating system: the job and the job step. A job consists of executing one or more job steps. For example, three job steps are involved to compile, link edit and execute an RPG program.

1. Translate the source program into an object module by executing the RPG compiler component of the operating system.

2. Process the object module to produce a load module by executing the linkage editor component of the operating system.

3. Execute the compiled and link edited load module.

JOB CONTROL LANGUAGE

The RPG programmer uses the job control statements shown in Figure 188 to compile, link edit, and execute programs.

These statements are discussed in this section as they are used to specify RPG job processing. A detailed explanation of each statement is given in *IBM System/360, Operating System, Job Control Language,* Form C28-6539.

Statement	Function
JOB	Indicates the beginning of a new job and describes that job
EXEC	Indicates a job step and describes that job step; indicates the cataloged procedure or load module to be executed
DD	Describes data sets and controls device and volume assignment
Delimiter	Separates data sets in the input stream from control statements; it appears after each data set in the input stream

Figure 188. Job Control Statements

Compiler Processing

The names for DD statements (ddnames) relate I/O statements in the compiler with data sets used by the compiler. These ddnames must be used for the compiler. When the system is generated, names for I/O device classes are also established and must be used by the programmer.

Compiler Name

The program name for the RPG compiler is IESRPG. If the compiler is to be executed without using the supplied cataloged procedures in a job step (see *Using Cataloged Procedures,* in this section), the EXEC statement parameter, PGM = IESRPG, must be used.

Compiler ddnames

The compiler can use 7 data sets. To establish communication between the compiler and the programmer, each data set is assigned a specific ddname. Each data set has a specific function and device requirement. Figure 189 lists the ddnames, functions, and device requirements for the data sets.

To compile an RPG source program, five of these data sets are necessary: SYSIN, SYSPRINT, SYSUT1, SYSUT2 and SYSUT3, along with the direct access volume(s) that contains the operating system. With these five data sets, only a listing is generated by the compiler. If an object module is to be punched, a SYSPUNCH DD statement must be supplied. If an object module is to be passed to the linkage editor, a SYSGO DD statement must be supplied.

For the DD statement SYSIN or SYSPRINT or SYSPUNCH an intermediate storage device may be specified instead of the card reader or printer or card punch. The intermediate storage device can be magnetic tape or a direct access device.

If an intermediate device is specified for SYSIN, the compiler assumes that the source module deck was placed on intermediate storage by a previous job or job step. If an intermediate device is specified for SYSPRINT, the listing, and error/warning messages are written on that device; a new job or job step can print the contents of the data set. When the SYSPRINT data set is written on an intermediate storage device, carriage control characters are placed in the records.

ddname	Function	Device Requirements
SYSIN	Reading the source program	● Card reader ● Intermediate storage
SYSPRINT	Writing the storage map, listing, and messages	● Printer ● Intermediate storage
SYSPUNCH	Output data set for the object module deck	● Card punch ● Intermediate storage
SYSUT1	Work data set needed by the compiler during compilation	● Direct access ● Magnetic tape
SYSUT2	Work data set needed by the compiler during compilation	● Direct access ● Magnetic tape
SYSUT3	Work data set needed by the compiler during compilation	● Direct access ● Magnetic tape
SYSGO	Output data set for the object module used as input to the linkage editor	● Direct access ● Magnetic tape

Figure 189. Compiler DDnames

Note: When using split cylinders, SYSUT1 and SYSUT2 must be assigned an equal number of tracks per cylinder.

Compiler Device Classes

Names for input/output device classes used for compilation are also specified by the operating system when the system is generated. The class names, functions, and types of devices are shown in Figure 190.

The data sets used by the compiler must be assigned to the device classes listed in Figure 191.

Class Name	Class Functions	Device Type
SYSSQ	Writing, reading, backspacing (sequential)	● Magnetic tape ● Direct access
SYSDA	Writing, reading, backspacing, updating records in place (direct)	● Direct access
A	SYSOUT output	● Printer ● Magnetic tape ● Direct access
B	SYSOUT output	● Card punch ● Magnetic tape ● Direct access

Figure 190. Device Class Names

ddname	Possible Device Classes
SYSIN	SYSSQ, or the input stream device (specified by DD * or DD DATA)
SYSPRINT	A, SYSSQ, SYSDA
SYSPUNCH	B, SYSSQ, SYSDA
SYSUT1	SYSSQ, SYSDA
SYSUT2	SYSSQ, SYSDA
SYSUT3	SYSSQ, SYSDA
SYSGO	SYSSQ, SYSDA

Figure 191. Correspondence Between Compiler ddnames and Device Classes

Compiler Options

Options are passed to the compiler through the PARM parameter in the EXEC statement.

$$\text{PARM} = \begin{Bmatrix} \text{DECK} \\ \underline{\text{NODECK}} \end{Bmatrix} \begin{Bmatrix} \underline{\text{,LOAD}} \\ \text{,NOLOAD} \end{Bmatrix} \begin{Bmatrix} \underline{\text{,LIST}} \\ \text{,NOLIST} \end{Bmatrix} \begin{Bmatrix} \text{,SEQN} \\ \underline{\text{,NOSEQN}} \end{Bmatrix}$$

The programmer specifies the options which are defined as:

DECK The object module is placed on the device specified in the SYSPUNCH DD statement (usually the card punch).

LOAD The object module is placed on the device specified in the SYSGO DD statement (usually intermediate storage).

LIST An output listing is written on the device specified in the SYSPRINT DD statement (see *Compiler Output,* in this section).

SEQN The source deck is checked to ensure that the combined page/line field value in each specification is greater than the combined page/line field value of the previous specification.

The prefix NO is used with any of these options to specify that the option is not wanted. If any entry for any of the four options is omitted from the PARM parameter, the corresponding underlined entry (NODECK, LOAD, LIST, or NOSEQN) is assumed for that option. If contradictory options are entered (for example, LIST, NOLIST) the entry without the underline (DECK, NOLOAD, NOLIST, or SEQN) is used for that option.

The DECK option specifies that the compiler output (that is, the object module) is written on the data set specified by the SYSPUNCH DD statement. NODECK specifies that no object module is written. A description of the deck is given in *Compiler Output* in this section.

The LOAD option indicates that the object module is written on the data set specified by the SYSGO DD statement. This option must be used if a cataloged procedure to compile, link edit, and execute is used.

The NOLOAD option indicates that the object module is not written on a sequential data set. This option must not be used if a cataloged procedure to compile, link edit, and execute is used. If NOLOAD and DECK are specified, the resulting object deck may be used as input to the linkage editor.

If the LOAD and DECK options are specified, the object module is written on the two data sets, indicated by the SYSGO and SYSPUNCH DD statements.

The cataloged, procedures assume the options of NODECK, LOAD, LIST, NOSEQN. However, the programmer can override any or all of the assumed options as described in *Overriding Statements in Cataloged Procedures* in this section.

Compiler Output

The RPG compiler can generate a listing of the source specifications, diagnostic messages, and a memory map showing the names and addresses of object program routines. The compiler can also produce an object module card deck (Figure 192).

The output listing (see *Sample Programs, Sample Program One)* provides:

1. Listing of each specification with related statement number and error notes.
2. Resulting indicator table showing:
   a. Names of all RPG processor and user defined resulting and record identifying indicators

   b. Address (6 places) of each defined resulting and record identifying indicator
3. Field name table containing:
   a. Names of all fields.

   b. Address (6 places) of each field. ENTRY or EXTERN type field names are denoted by ENTRY or EXTRN.
4. Literal table containing:
   a. All literals and edit words.

   b. Address (6 places) of each literal or edit word.
5. Diagnostic listing of erroneous entries showing:
   a. Statement number.

   b. Name of the erroneous field or resulting indicator.

   c. Appropriate error note.
6. Further specification diagnostics.
   a. Statement number.

   b. Appropriate error note.
7. Diagnostic messages. Refer to *Appendix C.*
8. Memory map listing names and addresses of RPG object program routines.
9. The length (in hexadecimal) of the compiled program not including the IOCS modules or any user subroutines.
10. END OF COMPILATION message.

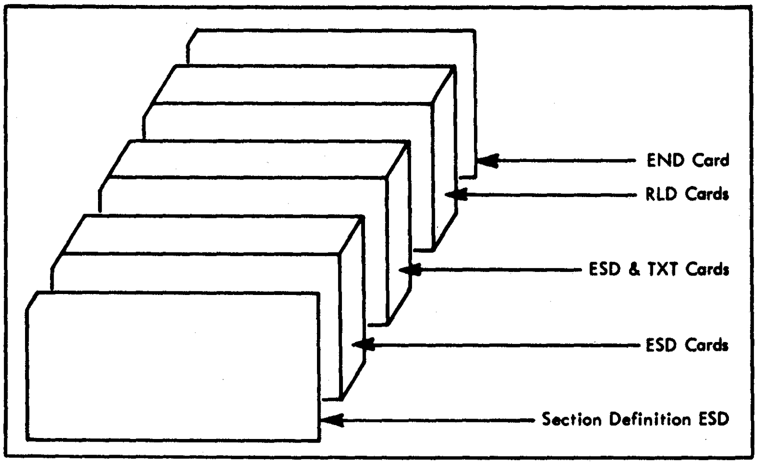


Figure 192. RPG Output Object Module Card Deck

Compiler Return Codes

Figure 193 shows the return codes issued by the RPG compiler for use with the COND=parameter of JOB or EXEC statements.

Linkage Editor Processing

The linkage editor processes RPG object modules, resolves any references to subprograms, and constructs a load module. To communicate with the linkage editor, the programmer supplies an EXEC statement and DD statements that define all required data sets; he may also supply linkage editor control statements.

Linkage Editor Name

The program name for the linkage editor is IEWL. If the linkage editor is executed without using cataloged procedures in a job step, the EXEC statement parameter, PGM=IEWL, must be used.

Linkage Editor Input and Output

The modules that are processed by the linkage editor are contained in data sets as:

1. Primary input data set (principal input).
2. Call library (for automatic library call).
3. Additional primary input or call library data sets.

The primary input data set is required for all linkage editor job steps. The call library is defined only if the automatic library call function is used. (Refer to *IBM System/360 Operating System, Linkage Editor,* Form C28-6538) Additional sequential data sets or partitioned data sets are defined only as required.

The primary input is a sequential data set that contains object modules and linkage editor control statements. In an RPG compile, link, and execute job, the primary input data set contains the object modules produced by the job steps. The primary input can be a chain of sequential data sets. A library member can be specified on a DD statement to be processed as a sequential data set. For details refer to *IBM System/360 Operating System, Job Control Language,* Form C28-6539. The primary input data set must be specified by the ddname SYSLIN.

The linkage editor can accept input from other than the primary input source. Additional input sourcesssuch as user subroutines can be specified by the INCLUDE statement or automatic library call. For details of linkage editor control statements refer to *IBM System/360 Operating System, Linkage Editor,* Form C28-6538. Variations to incorporate user subroutines are discussed in *Executing RPG – Input Stream Variations,* in the next subsection.

The output of the linkage editor is always placed in a PDS. Error messages and optional diagnostic messages are written on an intermediate storage device or a printer. In addition, a work data set is required by the linkage editor to do its processing.

Return Code	Explanation
0	No errors detected
4	Minor errors detected; successful program execution is probable
8	Errors detected; unsuccessful program execution is possible
12	Serious errors detected; unsuccessful program executioı is probable
16	Critical errors detected; normal execution is impossible
20	Unrecoverable I/O error occurred during compilation; compilation terminated
24	Specified program interruption exit. For details refer to *IBM System/360, Operating System, Control Program Services.*

Note: the COND=parameter is explained in *IBM System/360 Operating System, Job Control Language (Form C28-6539)*

Figure 193. Compiler Return Codes

Linkage Editor ddnames and Device Classes

The programmer communicates data set information to the linkage editor through DD statements identified by specific ddnames (similar to the ddnames used by the compiler). The ddnames, functions, and requirements for data sets are shown in Figure 194.

Any data sets specified by SYSLIB or SYSLMOD must be partitioned data sets. (Additional inputs are partitioned data sets or sequential data sets.) The ddname for the DD statement that retrieves any additional libraries is written in INCLUDE and LIBRARY statements and is not fixed by the linkage editor.

The device classes used by the compiler (see Figure 190) must also be used with the linkage editor. The data sets used by linkage editor may be assigned to the device classes listed in Figure 195.

Load Module Execution

Execution Options

Options are passed to the object module through the PARM parameter of the EXEC card.

$$\text{PARM} = \left\{ \begin{array}{l} \text{'DATE=mmddyy'} \\ \text{'DATE=ddmmyy'} \end{array} \right\}$$

where mm=month, dd=day, and yy=year.

The user specifies the option by replacing mmddyy or ddmmyy with a six digit numeric date which will initialize the UDATE, UMONTH, UDAY, and UYEAR fields. If no date is furnished through the PARM field, the system date will be used for UDATE, UMONTH, UDAY, and UYEAR initialization. The system date will be in the domestic format (mmddyy) if a blank is specified in position 21 of the RPG control card, or in European format (ddmmyy) if either an I or a D is specified in position 21.

Execution ddnames

In the RPG source program the file description specifications entries are used to identify the data sets (files). Data sets processed by the RPG load module must be defined by DD statements. The relationship is established between the data set defined in the source program and the DD statement by the ddname. The ddname must be the same as the entry in Filename (positions 7-14) on the File Description Specifications form.

ddname	Function	Device Requirements
SYSLIN	Primary input data, normally the output of the compiler	• Direct access • Magnetic tape • Card reader
SYSLIB	Automatic call library	• Direct access
SYSUT1	Work data set	• Direct access
SYSPRINT	Diagnostic messages	• Printer • Intermediate storage device
SYSLMOD	Output data set for the load module	• Direct access
User-specified	Additional libraries and object modules	• Direct access • Magnetic tape

Figure 194. Linkage Editor DDnames

ddname	Possible Device Classes
SYSLIN	SYSSQ, SYSDA, or the input stream device (specified by DD* or DD DATA)
SYSLIB	SYSDA
SYSUT1	SYSDA
SYSLMOD	SYSDA
SYSPRINT	A, SYSSQ, SYSDA
User-specified	SYSDA, SYSSQ

Figure 195. Correspondence Between Linkage Editor ddnames and Device Classes

Execution Cancellation

During execution of the load module (object program), job processing is cancelled when a halt indicator (H0-H9) is detected on (see *Debugging an OS Core Dump of an RPG Program*).

Execution Return Codes

Figure 196 shows the return codes issued by the RPG load module (object program) for use with the COND= parameter of the JOB or EXEC statements.

Code	Explanation
0	Normal execution, no errors
28	H0 - initialized on or on due to programmer request
32	H0 - invalid chaining request
36	H0 - collating sequence error (matching records)
40	H0 - record sequence error (predetermined sequence)
44	H0 - undefined record type
48	H0 - BSAM I/O error (combined file)
52	H0 - DAM I/O error
56	H0 - ISAM I/O error (random processing)
60	H0 - ISAM I/O error (sequential processing)
64	H0 - QSAM I/O error
68	H1 - on
72	H2 - on
76	H3 - on
80	H4 - on
84	H5 - on
88	H6 - on
92	H7 - on
96	H8 - on
100	H9 - on
132	Program interrupt (data): may occur when using non-numeric decimal data in a field specified as numeric data or when using packed decimal data in a field specified as unpacked data
144	Program interrupt (decimal divide): will occur if the value of Factor 2 of a DIV operation is zero

Note: The COND=parameter is explained in the *Job Control Language* publication.

Figure 196. Completion Codes

USING CATALOGED PROCEDURES

Because writing job control statements can become time-consuming work for the programmer, IBM supplies three cataloged procedures to aid in the compiling, link editing, and executing of RPG programs.

Compile

The cataloged procedure for compilation is RPGEC. It is invoked by specifying the name RPGEC as the first parameter in an EXEC statement (refer to *Cataloged Procedures* in this section).

With the procedure RPGEC, a DD statement RPG.SYSIN indicating the location of the source program must be supplied.

The control statements that can be used to invoke the procedure are:

```
//jobname JOB
//stepname EXEC RPGEC
//RPG.SYSIN DD *

                    RPG Source Program

/*
```

Link Edit and Execute

The cataloged procedure to link edit RPG object modules and execute the resulting load module is RPGELG. It is invoked by specifying the name RPGELG as the first parameter in an EXEC statement.

The cataloged procedure to link edit and execute consists of the control statements shown in this section in *Cataloged Procedures.*

With the procedure RPGELG, a DD statement LKED. SYSIN, which indicates the location of the object module, must be supplied.

The control statements that can be used to invoke the RPGELG cataloged procedure are:

```
//jobname JOB
//stepname EXEC RPGELG
//LKED.SYSIN DD *

                ┌───────────────────┐
                │ RPG Object Module │
                └───────────────────┘

/*
```

When the RPG compiler is maintained in a private library, a JOBLIB DD statement is required in the input stream to process an RPG program. The JOBLIB DD statement is used to temporarily chain the private library with the system library. The format of the DD statement is:

```
//JOBLIB DD DSNAME= libname,          X
//              DISP= (OLD,PASS), etc.
```

with libname being the fully qualified name of the RPG library. The JOBLIB DD statement must appear immediately before the first EXEC statement for the job. The concatenation is in effect only for the duration of the job.

Compile, Link Edit, and Execute

The cataloged procedure RPGECLG passes a source module through three procedure steps: compile, link edit, and execute. The cataloged procedure is invoked by specifying the name RPGECLG as the first parameter in an EXEC statement.

The statements that invoke the cataloged procedure RPGECLG are:

```
//jobname JOB
//stepname EXEC RPGECLG
//RPG.SYSIN DD *

                ┌────────────────────┐
                │ RPG Source Program │
                └────────────────────┘

/*
```

The SYSIN data set (source module) must be defined to the compiler as a separate DD statement not contained in the cataloged procedure.

Cataloged Procedures

This section contains figures showing the job control statements used in the RPG cataloged procedures and a brief description of each procedure.

If the user selects MVT (option 4) of the control program, he must be aware of several items; the direct access space requirements of SYSOUT data sets, that SPACE= and UNIT= keyword parameters are valid for any SYSOUT data set, the size of the region that is selected by default when the REGION= parameter is not specified for any job step. For additional information refer to *IBM System/360 Operating System, System Generation,* Form C28-6554.

The RPG cataloged procedures are set up to take advantage of the dedicated workfiles option of the Initiator cataloged procedure for the MVT option of the control program. Refer to *IBM System/360 Operating System: System Programmer's Guide,* Form C28-6550, for a complete description of the dedicated workfiles option.

The cataloged procedures make use of the OS/360 job control language symbolic parameter facility. Refer to *Modifying Cataloged Procedures Symbolic Parameter Values* in this section.

Compile

The cataloged procedure for compilation (RPGEC) is shown in Figure 197. The numbers to the right correspond to the numbered explanations that follow.

1. This statement assignes default values to the symbolic parameters defined in this procedure.

2. The system name IESRPG identifies the RPG compiler. The REGION= keyword parameter specifies the region size that is to be allocated to the job step, provided MVT in the operating system is specified. If MVT is not specified, the REGION= parameter is ignored. The size of the region required for any job step depends on the size of the program, the access methods used, the size and number of buffers requested, and the work areas requied by the various system functions (for example, OPEN, EOV, CLOSE, etc.). Refer to *IBM System/360 Operating System, Storage Estimates,* Form C28-6551).

   The COND= parameter can be added to this statement by the EXEC statement that calls the procedure.

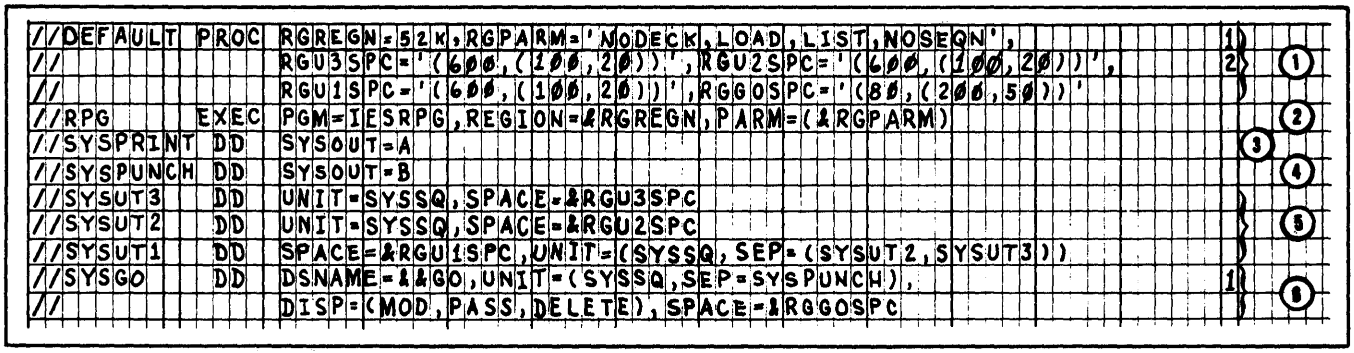
3. The destination for the compiler output listing is defined by this statement as the standard system output class, SYSOUT=A.

4. The destination for the object module (provided the option DECK has been specified) is defined as the standard system output class, SYSOUT=B. Usually the device assigned to this output class is a card punch.

5. The three compiler utility data sets are described by these statements. The SEP= subparameter in the last statement and the SPACE= parameter in each statement are effective only if the device assigned is a direct access device. The number of specifications in the source program determines the space requirement. The procedure provides an initial allocation of 60,000 bytes and additional allocations (if required) of 12,000 bytes.

6. This statement describes the compiler output (object module) produced for input to the linkage editor. The compiler option LOAD causes the object module to be written on this data set. The third term in the DISP parameter causes the data set to be deleted if the job step terminates abnormally.

Note: DCB parameters must not be overridden for any data set used in the compile step. To override the device assignments of SYSUT1 and/or SYSUT2 and/or SYSUT3 data sets, the user may assign the data sets to any direct access storage device supported by the operating system access method, BSAM (for example, 2311, 2314, 2301, 2302, 2303, or 2321). SYSUT1 and SYSUT2 must be assigned to the same type of device.



```
//DEFAULT PROC  RGREGN=52K,RGPARM='NODECK,LOAD,LIST,NOSEQN',              1
//              RGU3SPC='(6ØØ,(1ØØ,2Ø))',RGU2SPC='(6ØØ,(1ØØ,2Ø))',        2
//              RGU1SPC='(6ØØ,(1ØØ,2Ø))',RGGOSPC='(8Ø,(2ØØ,5Ø))'
//RPG     EXEC  PGM=IESRPG,REGION=&RGREGN,PARM=(&RGPARM)
//SYSPRINT DD   SYSOUT=A
//SYSPUNCH DD   SYSOUT=B
//SYSUT3   DD   UNIT=SYSSQ,SPACE=&RGU3SPC
//SYSUT2   DD   UNIT=SYSSQ,SPACE=&RGU2SPC
//SYSUT1   DD   SPACE=&RGU1SPC,UNIT=(SYSSQ,SEP=(SYSUT2,SYSUT3))
//SYSGO    DD   DSNAME=&&GO,UNIT=(SYSSQ,SEP=SYSPUNCH),                   1
//              DISP=(MOD,PASS,DELETE),SPACE=&RGGOSPC
```

Figure 197. Compile Catalogued Procedure (RPGEC)

Link Edit and Execute

The cataloged procedure to link edit RPG object modules and execute the resulting load modules (RPGELG) is shown in Figure 198. The numbers to the right correspond to the numbered explanations that follow.

1. This statement assigns default values to the symbolic parameters defined in this procedure.

2. This statement initiates linkage editor execution. The options in the PARM= field cause the linkage editor to put out a cross-reference table, and a list of all control statements processed. The LET option causes the linkage editor to mark the load module as executable even though errors were encountered during processing. The REGION= parameter specifies the region size required for the execution of this job step (provided the user has selected MVT in the operating system control program). Refer to *IBM System/360 Operating System, Storage Estimates,* Form C28-6551.

3. This statement indicates that the input to the linkage editor is from the input stream.

4. The output from the linkage editor is specified as a member of a temporary data set, residing on a direct access storage device, and is passed to a succeeding job step. The third term in the DISP parameter causes the data set to be deleted if the job step terminates abnormally.

5. The utility data set for the linkage editor is described by this statement.

6. The destination for the linkage editor output listing is defined by this statement as the standard output class, SYSOUT=A.

7. This statement initiates execution of the link edited load module. The notation *.LKED.SYSLMOD identifies the load module as being in the data set described in the job step LKED by the DD statement named SYSLMOD. The REGION= keyword parameter is not included in this job step. The effect of this omission is that the default region size is selected (providing the user has selected MVT in the operating system control program). If the default region size is not sufficient, refer to the preceeding subsection *Overriding Parameters in the EXEC Statement.*

Compile, Link Edit, and Execute

The cataloged procedure (RPGECLG) to compile, link edit, and execute RPG source programs is shown in Figure 199.

The cataloged procedure RPGECLG consists of the statements in the RPGEC and RPGELG procedures, with the exception: the DD statement SYSLIN identifies the linkage editor input data set as the same data set produced as output by the compiler. Further, the third term in the DISP parameter of the DD statement SYSLMOD will cause the data set to be deleted if the LKED step is terminated abnormally. The programmer must define the data set SYSIN (as a separate DD statement) for the compiler so that the source program can be read. He can also concatenate any input to the linkage editor from the input stream with the input from the compiler by using a DD statement LKED .SYSIN.

User Cataloged Procedures

The programmer can write his own cataloged procedures and tailor them to the facilities in his installation. He can also permanently modify the IBM-supplied cataloged procedures. For information about permanently modifying cataloged procedures, see the publication *IBM System/360 Operating System, Utilities,* Form C28-6586.

The RPG cataloged procedures may be temporarily modified by:

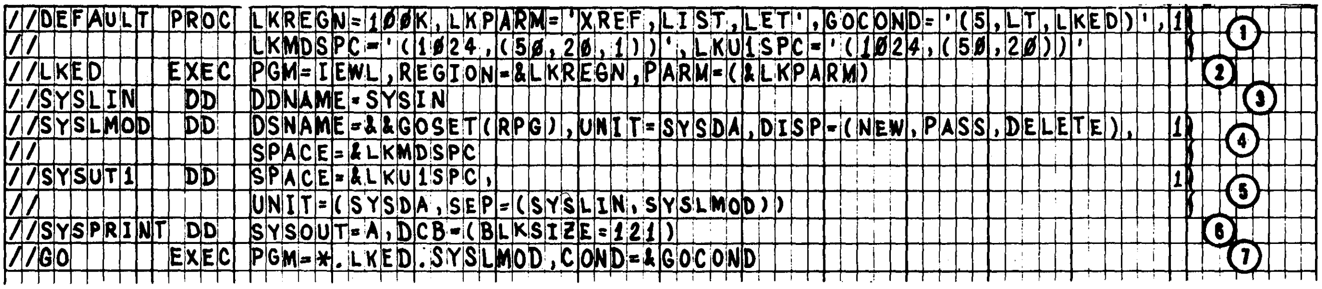
1. Specifying new values for symbolic parameters.

2. Overriding complete DD or EXEC statements.

Modifying Cataloged Procedures Symbolic Parameter Values

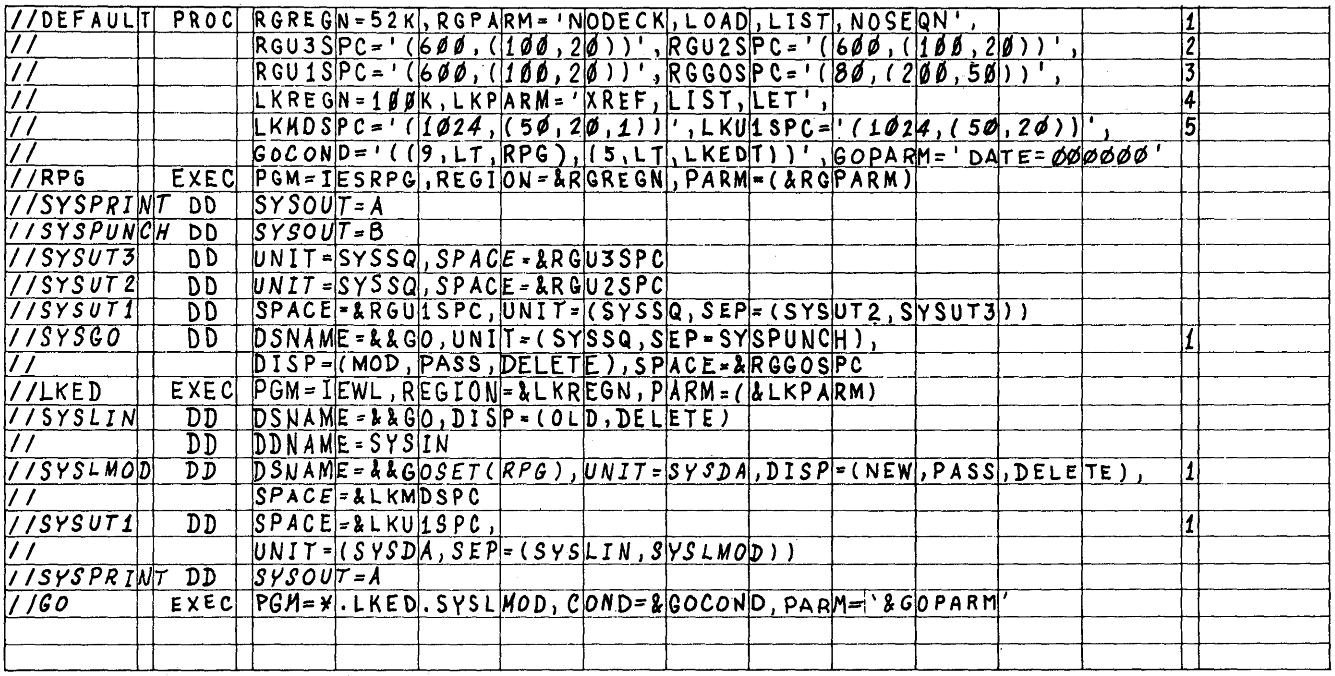
The RPG cataloged procedures make use of the job control language symbolic parameter facility. Refer to *IBM System/360 Operating System, Job Control Langauge,* Form C28-6539. A symbolic parameter is a symbol that appears in the operand of a cataloged procedure statement. Symbolic parameters are assigned values either on the EXEC statement that invokes the procedure or on a PROC statement in the procedure.

In the RPG procedures, symbolic parameters are used for REGION, COND, and PARM specifications on the EXEC statement, and for SPACE specifications on the DD statements. These values may be temporarily changed by coding the symbolic parameter and the desired value on the EXEC statement used to invoke the procedure.



```
//DEFAULT  PROC LKREGN=1ØØK,LKPARM='XREF,LIST,LET',GOCOND='(5,LT,LKED)',1
//              LKMDSPC='(1Ø24,(5Ø,2Ø,1))',LKU1SPC='(1Ø24,(5Ø,2Ø))'
//LKED     EXEC PGM=IEWL,REGION=&LKREGN,PARM=(&LKPARM)
//SYSLIN    DD  DDNAME=SYSIN
//SYSLMOD   DD  DSNAME=&&GOSET(RPG),UNIT=SYSDA,DISP=(NEW,PASS,DELETE),    1
//              SPACE=&LKMDSPC
//SYSUT1    DD  SPACE=&LKU1SPC,                                           1
//              UNIT=(SYSDA,SEP=(SYSLIN,SYSLMOD))
//SYSPRINT  DD  SYSOUT=A,DCB=(BLKSIZE=121)
//GO       EXEC PGM=*.LKED.SYSLMOD,COND=&GOCOND
```

Figure 198. Link Edit and Execute Catalogued Procedure (RPGELG)



```
//DEFAULT  PROC RGREGN=52K,RGPARM='NODECK,LOAD,LIST,NOSEQN',                1
//              RGU3SPC='(6ØØ,(1ØØ,2Ø))',RGU2SPC='(6ØØ,(1ØØ,2Ø))',          2
//              RGU1SPC='(6ØØ,(1ØØ,2Ø))',RGGOSPC='(8Ø,(2ØØ,5Ø))',           3
//              LKREGN=1ØØK,LKPARM='XREF,LIST,LET',                         4
//              LKMDSPC='(1Ø24,(5Ø,2Ø,1))',LKU1SPC='(1Ø24,(5Ø,2Ø))',        5
//              GOCOND='((9,LT,RPG),(5,LT,LKED))',GOPARM='DATE=ØØØØØØ'
//RPG      EXEC PGM=IESRPG,REGION=&RGREGN,PARM=(&RGPARM)
//SYSPRINT  DD  SYSOUT=A
//SYSPUNCH  DD  SYSOUT=B
//SYSUT3    DD  UNIT=SYSSQ,SPACE=&RGU3SPC
//SYSUT2    DD  UNIT=SYSSQ,SPACE=&RGU2SPC
//SYSUT1    DD  SPACE=&RGU1SPC,UNIT=(SYSSQ,SEP=(SYSUT2,SYSUT3))
//SYSGO     DD  DSNAME=&&GO,UNIT=(SYSSQ,SEP=SYSPUNCH),                      1
//              DISP=(MOD,PASS,DELETE),SPACE=&RGGOSPC
//LKED     EXEC PGM=IEWL,REGION=&LKREGN,PARM=(&LKPARM)
//SYSLIN    DD  DSNAME=&&GO,DISP=(OLD,DELETE)
//          DD  DDNAME=SYSIN
//SYSLMOD   DD  DSNAME=&&GOSET(RPG),UNIT=SYSDA,DISP=(NEW,PASS,DELETE),      1
//              SPACE=&LKMDSPC
//SYSUT1    DD  SPACE=&LKU1SPC,                                             1
//              UNIT=(SYSDA,SEP=(SYSLIN,SYSLMOD))
//SYSPRINT  DD  SYSOUT=A
//GO       EXEC PGM=*.LKED.SYSLMOD,COND=&GOCOND,PARM='&GOPARM'
```

Figure 199. Compile, Link Edit and Execute Catalogued Procedure (RPGECLG)

Examples

In procedure RPGECLG (Figure 199), suppressing producation of a listing might be desired. In this case, the EXEC statement in the input stream would be:

```
//stepname EXEC RPGECLG,RGPARM='NOLIST'
```

In the procedure RPGEC (Figure 197), a punched object deck can be obtained, the source deck sequence checked, and SPACE= parameters of the data set SYSUT1 respecified by:

```
//stepname EXEC RPGEC,                        X
//              RGPARM='DECK,SEQN'            X
//              RGU1SPC='(200,(80,40))'
```

In the procedure RPGELG (Figure 198), the link edit region (LKREGN) and the execution condition (GOCOND) specifications may be changed with:

```
//stepname EXEC RPGELG,                       X
//              GOCOND='(3,LT,LKED)',         X
//              LKREGN=90K
```

In the procedure (Figure 199) if a user date is to be specified at object time for the six digit numeric field UDATE, the EXEC statement would appear as:

```
//              EXEC RPGECLG
//                   GOPARM= 'DATE=080568'
```

Otherwise, the default value of DATE= 000000 would cause the UDATE field to be initialized with the system date.

If it is desired to modify an EXEC or DD statement parameter not specified symbolically, it is necessary to override the desired statement with an appropriate statement in the input stream.

The symbolic parameter names are listed in Figure 200.

Symbolic Name	Job Step	Description	Assigned Value
&RGREGN	Compiler	Region size	52K
&RGPARM		PARM = field	'NODECK, LIST, LOAD, NOSEQN'
&RGU3SPC		SYSUT3 space	'(600, (100,20))'
&RGU2SPC		SYSUT2 space	'(600, (100,20))'
&RGU1SPC		SYSUT1 space	'(600, (100,20))'
&RGGOSPC		SYSGO space	'(80, (200,50))'
&LKREGN	Link Edit	Region size	100K
&LKMDSPC		SYSLMOD space	'(1024, (50,20, 1))'
&LKU1SPC		SYSUT1 space	'(1024, (50,20))'
&LKPARM		PARM = field	'XREF, LIST, LET'
&GOCOND	Go	COND = field for RPGECLG	'((9,LT,RPG), (5,LT,LKED))'
&GOCOND		COND = field for RPGELG	'(5,LT,LKED)'
&GOPARM		PARM=field	'DATE=000000'

Figure 200. Symbolic Parameter Name Assignments

Overriding Statements in Cataloged Procedures

EXEC and DD statements appearing in cataloged procedures can be overridden. Such overriding of statements of fields is effective only for the duration of the job step in which the statements appear. The statements, as stored in the procedure library of the system, remain unchanged.

Overriding for the purposes of respecification, addition, or nullification is accomplished by including in the input stream statements containing the desired changes and identifying the statements to be overridden.

Overriding Parameters in the EXEC Statement: It may be desirable to add the PARM=, COND=, or REGION= parameters to an RPG procedure EXEC statement which does not have the parameter. This can be done by including in the EXEC statement calling the procedure the notation PARM.stepname=,COND.stepname=, or REGION.stepname=, followed by the desired change. Stepname identifies the EXEC statement within the procedure to which the modification applies. Overriding the PGM= parameter is not possible.

Overriding and Adding DD Statements: A DD statement with name procstep.ddname is used to override parameters not containing symbolic parameters in DD statements in cataloged procedures, or to add DD statements to cataloged procedures. The procstep identifies the step in the cataloged procedure. If ddname is the name of a DD statement:

1. Present in the step, the parameters in the new DD statement override parameters in the DD statement in the procedure step.

2. Not present in the step, the new DD statement is added to the step.

In any case, the modification is only effective for the current execution of the cataloged procedure.

When overriding, the original DD statement in the cataloged procedure is copied, and the parameters specified in it are replaced by the corresponding parameters in the new DD statement. Therefore, only parameters that must be changed are specified in the overriding DD statement.

If more than one DD statement is modified, the overriding DD statements must be in the same order as the DD statements appear in the cataloged procedure. Any DD statements that are added to the procedure must follow overriding DD statements.

When the procedures RPGEC and RPGECLG are used, a DD statement must be added to define the SYSIN data set to the compile step in the procedures. When the procedure RPGELG is used, a DD statement must be added to define the SYSLIN data set (see *Using Cataloged Procedures* in this section).

Examples: In the procedure RPGEC (Figure 197), the UNIT=parameter of the data set SYSUT1 can be respecified by including in the input stream:

```
//stepname  EXEC RPGEC
//RPG.SYSUT1 DD  UNIT=(2311,SEP=(SYSUT2,SYSUT3))
```

In procedure RPGECLG (Figure 199) chaining the COND=parameter to the EXEC statement which specifies execution of the linkage editor and specifying a region size of 60K in the GO step might be desired. In this case, the EXEC statement in the input stream would appear as:

```
//stepname EXEC RPGECLG,                        X
//              COND.LKED=(9,LT,stepname.RPG),X
//              REGION.GO=60K
```

Execution of the linkage editor job step //LKED is suppressed if the return code issued by the compiler (step RPG) was greater than 8 and the region size requested for the GO step was 60K.

IBM System/360 Operating System Utilities, Form C28-6586 and *IBM System/360 Operating System, Job Control Language* (Form C28-6539), provide additional description of overriding techniques.

RPG SOURCE PROGRAM DECK ARRANGEMENT

The deck prepared by the programmer is arranged as shown in Figure 201. The contents of the operating system job control statements are listed in *IBM System/360 Operating System, Job Control Language* (Form C28-6539).

The order in which the programmer places his control cards and source deck is:

1. Operating system job control statements.
2. Control card specifications. This card is required.
3. File description specifications.
4. Extension specifications.
5. Line counter specifications.
6. Input specifications.
7. Calculation specifications.
8. Output-format specifications.

EXECUTING RPG – INPUT STREAM VARIATIONS

The input stream varies considerably depending on the user's application. This section illustrates the input stream for various combinations of control statements, source decks, user subroutines, and data decks.

Note: When using a sequential scheduler, only one data set can be included in the input stream for each job step or procedure step. For additional information, refer to *IBM System/360 Operating System, Job Control Language,* Form

IBM Supplies three cataloged procedures that are explained in this section in *Cataloged Procedures* and illustrated in *Using Cataloged Procedures.* Additional examples are included in this section to illustrate inclusion of user subroutines for execution with the RPG load module.

The EXIT operand can be used to incorporate subroutines in assembler language. For convenience, frequently used subroutines are maintained in a library. The following illustrates three methods for including user subroutines with the RPG object module as input to the linkage editor. In these examples the user must supply DD statements

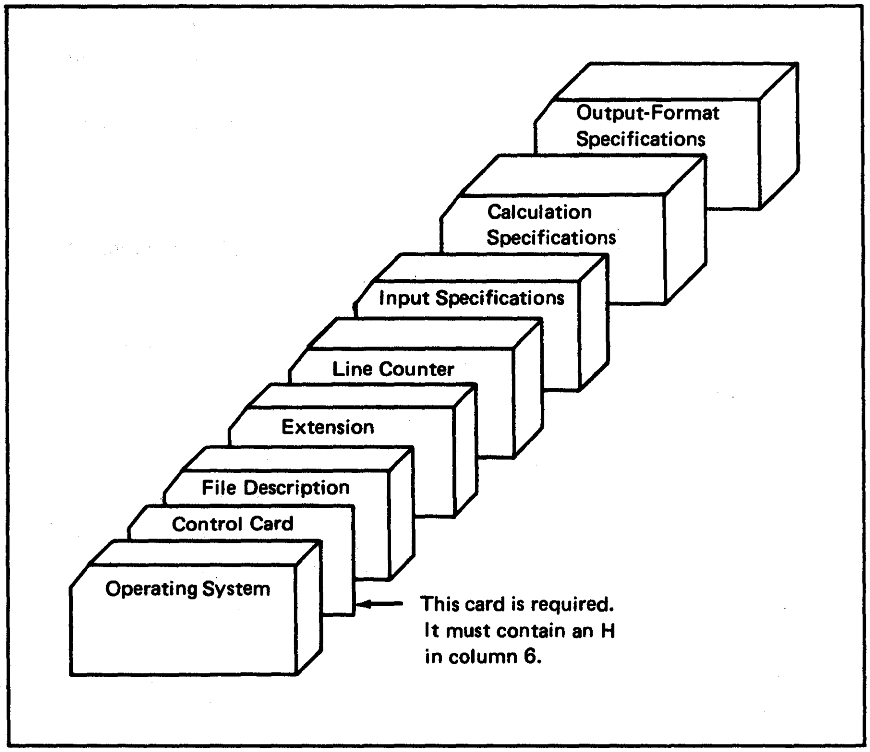

Output-Format Specifications
Calculation Specifications
Input Specifications
Line Counter
Extension
File Description
Control Card
Operating System
This card is required. It must contain an H in column 6.

Figure 201. RPG Deck Arrangement in the Operating System Input Stream

corresponding to the file description specifications. The LIBRARY statement is used to call the library in order to resolve the designated external references (SUBR1, SUBR2). The LKED.SUBLIB DD statement defines the private library containing the subroutines. These subroutines can be either load modules or object modules with control statements.

```
//jobname  JOB
//stepname  EXEC RPGECLG
//RPG.SYSIN  DD *

                    RPG source program

/*
//LKED.SUBLIB  DD  DSNAME=libname,DISP=OLD
//LKED.SYSIN  DD  *
  LIBRARY SUBLIB(SUBR1,SUBR2)
/*
```

The INCLUDE statement is used to cause the linkage editor to process the subroutine modules (SUBR1, SUBR2). The LKED.SUBLIB DD statement identifies the private library that contains the subroutines. These subroutines can be either load modules or object modules with control statements.

```
//jobname  JOB
//stepname  EXEC RPGECLG
//RPG.SYSIN  DD  *

                    RPG source program

/*
//LKED.SUBLIB  DD  DSNAME=libname,DISP=OLD
//LKED.SYSIN  DD  *
  INCLUDE SUBLIB(SUBR1,SUBR2)
/*
```

The placement of the user subroutines that are included as object modules in the input stream of a compile and execute job is:

```
//jobname  JOB
//stepname  EXEC RPGECLG
//RPG.SYSIN  DD  *

                    RPG source program

/*
//LKED.SYSIN  DD  *

                    SUBR1 object module

                    SUBR2 object module

                    SUBR3 object module

/*
```

Subroutines can be included both from a library and in the input stream. The job shown below includes subroutines SUBR1 and SUBR2 from the library and SUBR3 from the input stream.

```
//jobname  JOB
//stepname  EXEC  RPGECLG
//RPG.SYSIN  DD  *

                    RPG source program

/*
//LKED.SUBLIB  DD  DSNAME=libname,DISP=OLD

//LKED.SYSIN  DD  *
  INCLUDE SUBLIB(SUBR1, SUBR2)

                    SUBR3 object module

/*
```

The operating system provides the capability to assemble user subroutines within the same job as the RPG compilation. The assembly can either precede or follow or both precede and follow the RPG compilation. The job setup for an assembly following the RPG compilation is:

```
//jobname  JOB
//stepname  EXEC RPGEC

//RPG.SYSIN  DD  *

          RPG source program

/*
//stepname  EXEC  ASMECLG
//ASM.SYSIN  DD  *

          Source program for assembly

/*
```

The first EXEC statement invokes the cataloged procedure for compilation (RPGEC).

The subparameter MOD (in RPGEC) specifies that the data set is added to and causes logical positioning after the last record in the data set.

The second EXEC statement invokes the cataloged procedure for assembly, link edit, and execute (ASMECLG). Refer to *IBM System/360 Operating System, Assembler (E) Programmer's Guide* (Form C28-6595) for a detailed description of cataloged procedures for assembly.

The object module that is derived from the assembly is added to the data set that contains the object module from the RPG compilation and this data set is subsequently input to the linkage editor.

Execution must begin in the RPG produced object module. When an assembly precedes the RPG compilation, the RPG compiler is not the first program executed. Therefore, an ENTRY statement is required to specify the name of the RPG program. In order to link edit the assembly object module with the RPG object module, the SYSPUNCH DD statement in the ASMEC catalog procedure must be overriden. The assembly object module must be on SYSSQ. The job setup for assembly of the user subroutine preceding the RPG compilation is:

```
//jobname  JOB
//stepname  EXEC ASMEC
//ASM.SYSPUNCH  DD DSNAME=&GO,UNIT=SYSDA,  X
//                SPACE=(80,(200,50)),DISP=       X
//                (MOD,PASS)
//ASM.SYSIN DD *

          Source program for assembly

/*
//stepname  EXEC  RPGECLG
//RPG.SYSIN  DD  *

          RPG source program

/*
//LKED.SYSIN  DD  *
  ENTRY        source program identification
/*
```

The deck arrangement for jobs processing data files is:

```
//jobname  JOB
//stepname  EXEC  RPGECLG
//RPG.SYSIN  DD  **

          RPG source program

/*
//GO.ddname (parameters)
//GO.ddname (parameters)
      .
      .
      .
      .
      .
//GO.ddname (parameters)
/*
```

Data sets (files) processed by an RPG program must be defined by DD statements (see *Load Module Execution* in this section).

The deck arrangement for jobs processing table files and data files from the same card input device (SYSIN) is shown below. Under RPG support, the table data must correlate exactly with the amount specified on the Extension Specifications form.

In the following example, the TAB1 and TAB2 files must be loaded before the TRANS data file is loaded. The example is applicable to either MFT or MVT processing.

```
//jobname  JOB
//stepname  EXEC RPGECLG
//RPG.SYSIN  DD  *


                    RPG source program

/*
//GO.TAB1 DD *

                    table cards (TAB1)

//GO.TAB2 DD *

                    table cards (TAB2)

//GO.TRANS


                    Data

/*
```

Note: Table files and input data can not come from the same device during non-MFT or non-MVT processing.

Using the OS/360 Loader

The OS/360 Loader can be used to load and execute the object program generated by the RPG compiler. The RPG compile procedure, RPGEC, is used with the necessary JCL to invoke the loader. The loader combines the basic editing and loading functions of the linkage editor and program fetch in one job step. It is designed for high performance loading of modules that do not requires the special processing facilities of the linkage editor and fetch, such as overlay. The loader does not produce load modules for program libraries. Additional information on the OS/360 Loader can be found in *IBM System/360 Operating System Linkage Editor and Loader,* Form C28-6538.

This example shows the necessary JCL to compile and execute an RPG program.

```
//jobname  JOB
//stepname  EXEC  RPGEC
//RPG.SYSIN  DD  *

                      RPG source deck

/*
//stepname EXEC    PGM=LOADER,PARM=              1
//                 'loaderparms/                 1
//                 RPGobjectparms'
//SYSLIN  DD       DSNAME=*.stepname.RPG.        1
//                 SYSGO,DISP=(parameters)
//SYSLOUT  DD      (parameters)
//ddname    DD     (parameters)
//ddname    DD     *

                  Data
/*
```

The SYSLIN DSNAME refers back to the SYSGO DD statement in the RPG procedure RPGEC. The stepname must correspond to the stepname on the EXEC RPGEC statement. The files described in the RPG source program are described by the DD statements following the SYSLOUT statement.

The following examples illustrate three methods for including user subroutines with the RPG object module as input to the Loader. In these examples the user must supply DD statements corresponding to the file description specifications. In the first example, SYSLIN describes the RPG compiler output data set as input to the loader. Stepname in the DSNAME parameter must be same as stepname of the EXEC RPGEC statement. The loader entry point parameter EP=entry point name must specify the name given the RPG object program as specified on the RPG control card, positions 75-80, or RPGOBJ if no name is specified. This will assure that control is given to the RPG object module and not the subroutine. The SYSLIB DD statement defines the library which the loader will search in order to resolve external references in the RPG object program.

```
//jobname  JOB
//stepname  EXEC  RPGEC
//RPG.SYSIN  DD  *
```

RPG source deck

```
/*
//stepname    EXEC   PGM=LOADER,PARM='MAP,     1
//                   EP=RPGOBJ'
//SYSLIN      DD     DSNAME=*.stepname.RPG.    1
//                   SYSGO,DISP=(OLD,DELETE)
//SYSLIB      DD     DSNAME=libname,DISP=SHR
//SYSLOUT     DD     SYSOUT=A
//ddname      DD     *
```

Data

```
/*
```

The next example shows the setup necessary to assemble an external subroutine, compile an RPG source program, and load and execute the combined program. The output of the assembler will be placed in a data set followed by the compiler output. This data set is then described as input to the loader by the SYSLIN DD statement. Again, the load entry point parameter EP= must contain the RPG program name.

```
//jobname  JOB
//stepname  EXEC  ASMEC
//ASM.SYSIN,DD  *
```

Source program for assembly

```
/*
//stepname  EXEC  RPGEC
//RPG.SYSIN  DD  *
```

RPG source program

```
/*
//stepname    EXEC   PGM=LOADER,PARM='MAP      1
//                   LET,EP=programname'
//SYSLIN      DD     DSNAME=*.stepname.RPG.    1
//                   SYSGO,DISP=(OLD,DELETE)
//SYSLOUT     DD     (parameters)
//ddname      DD     (parameters)
//ddname      DD     *
```

Data

```
/*
```

This example shows the necessary JCL to combine a subroutine object deck with an RPG generated object program and load and execute it. The first step loads the subroutine object deck onto the data set specified by SYSUT2. This data set is described to be identical to the data set described by SYSGO DD statement in the RPG procedure RPGEC. Second step is an RPG compilation. The third step is a load and execute of the combined program. Once again the loader parameter EP= must specify the RPG program name as specified in positions 75-80 or RPGOBJ.

```
//jobname  JOB
//stepname EXEC  PGM=IEBGENER
//SYSPRINT DD    (parameters)
//SYSIN     DD   DUMMY
//SYSUT2    DD   DSNAME=&GO,UNIT=SYSDA,    1
//               SPACE=(80,(200,50)),      1
//               DISP=(MOD,PASS),DCB=      1
//               (LRECL=80,RECFM=F,        1
//               BLKSIZE=80)
//SYSUT1   DD    *

              Object deck

/*
//stepname EXEC  RPGEC
//RPG.SYSIN DD  *

              RPG source deck

/*
//stepname    EXEC   PGM=LOADER,PARM='MPA   1
//                   EP=RPGOBJ'
//SYSLIN      DD     DSNAME=*.stepname.RPG.  1
//                   SYSGO,DISP=(OLD,DELETE)
//SYSLOUT     DD     (parameters)
//ddname      DD     (parameters)
//ddname      DD     *

              Data

/*
```

When an RPG program is terminated, a dump of the program is provided by the operating system (OS). This core dump can be analyzed to determine the reason that the program was terminated.

Normally a core dump will be obtained by specifying the SYSUDUMP option in an OS job control statement.

The address at which the RPG object program is loaded will be used throughout this discussion to find information in the core dump. This address may be found as the contents of general register 3 on the core dump (Figure 204). This entry on the core dump listing is preceded by the heading Regs at Entry to ABEND. The discussion of the core dump will be related to the listing shown in Figure 204. The value of general register 3 may vary between programs, but the displacement values shown in the RPG symbol tables and the memory map remain fixed.

USING THE ERROR ANALYSIS TABLE

Figure 204, part two, shows the user completion code entry on a typical OS RPG core dump. In order to interpret this completion code, look down the left side of the error analysis table (Figure 202) to find a matching entry. The error type section of the table will indicate what kind of error caused the program to terminate.

For some types of errors an address is located at a displacement of X'11C' from the address in register three. The right side of the error analysis table explains what this address is. This address may be an IORB address, a DCB address, or a DECB address.

In this example a completion code of 0044 has an error type entry of "undefined record type" (this means that a record read by the RPG program was not defined on the input specifications). The four byte address at a displacement of '11C' from the address in register 3 (1FA38+11C=1FB54) is an IORB address.

For the user completion code values of 52, 56, and 60, there is a note on the error analysis table to see the I/O error analysis table (Figure 203). The contents of the two bytes found at a displacement of X'121' and X'122' from register 3 may be used to find more precisely what type of error caused the dump. The four-digit hexadecimal value found at this location should have a matching entry on the I/O error analysis table. The message associated with this number should pinpoint the type of error.

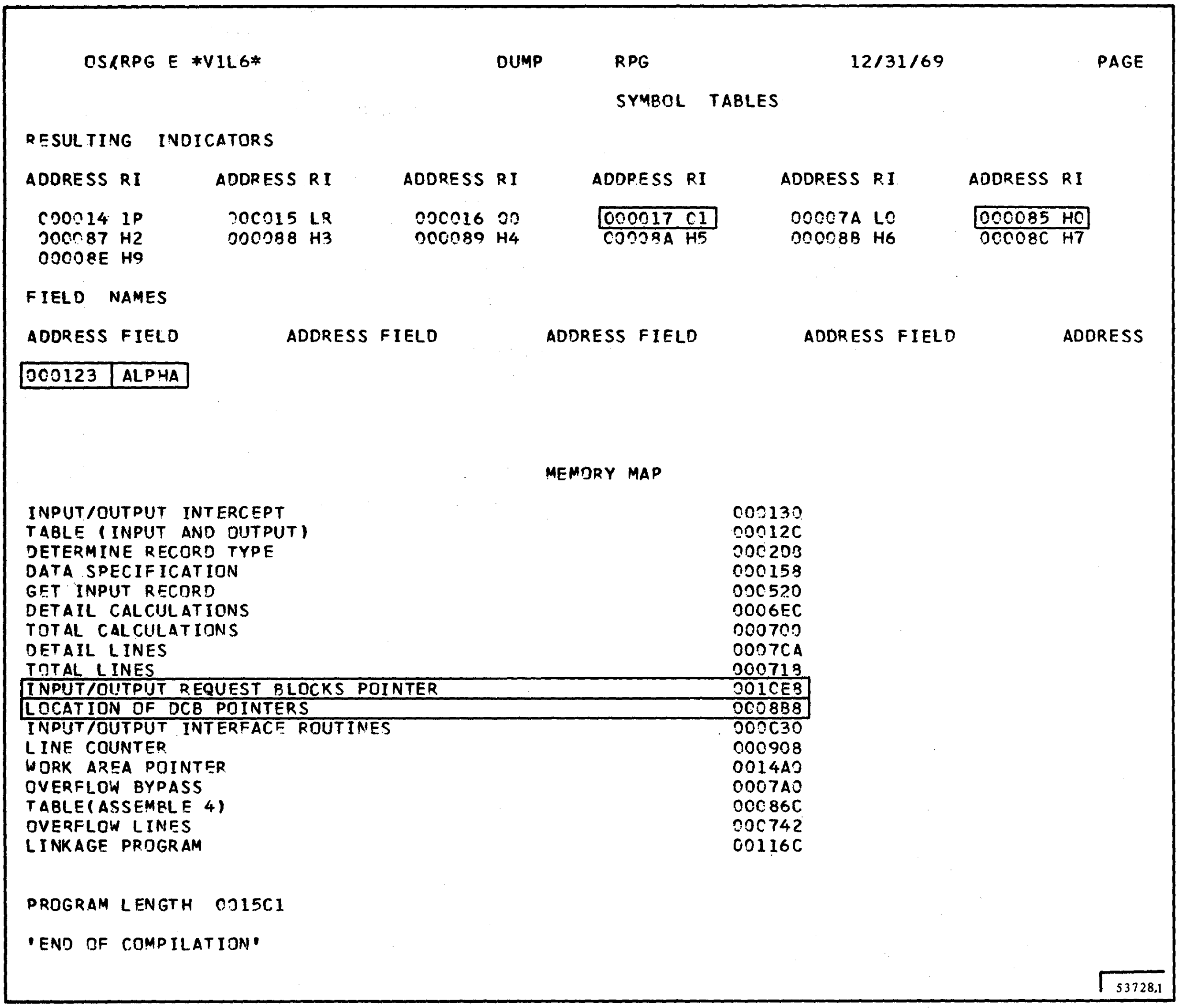
Completion Code	Error Type	Displacement From Register 3 X'11C' X'11'F
0	Normal execution, no errors	N/A
28	H0 – initialized on or on due to programmer request	N/A
32	H0 – invalid chaining request	Chaining identifier address
36	H0 – collating sequence error (matching records)	N/A
40	H0 – record sequence error (predetermined sequence	N/A
44	H0 – undefined record type	IORB address
48	H0 – BSAM I/O error (combined file)	DECB address
52*	H0 – DAM I/O error	DECB address
56*	H0 – ISAM I/O error (random processing)	DECB address
60*	H0 – ISAM I/O error (sequential processing)	DCB address
64	H0 – QSAM I/O error	DCB address
68	H1 – on	N/A
72	H2 – on	N/A
76	H3 – on	N/A
80	H4 – on	N/A
84	H5 – on	N/A
88	H6 – on	N/A
92	H7 – on	N/A
96	H8 – on	N/A
100	H9 – on	N/A
132	Program interrupt (data): may occur when using non-numeric decimal data in a field specified as numeric data or when using packed decimal data in a field specified as unpacked data	N/A
144	Program interrupt (decimal divide): will occur if the value of Factor 2 of a DIV operation is zero	N/A

* See the I/O Error Analysis Table.

Figure 202. Error Analysis Table

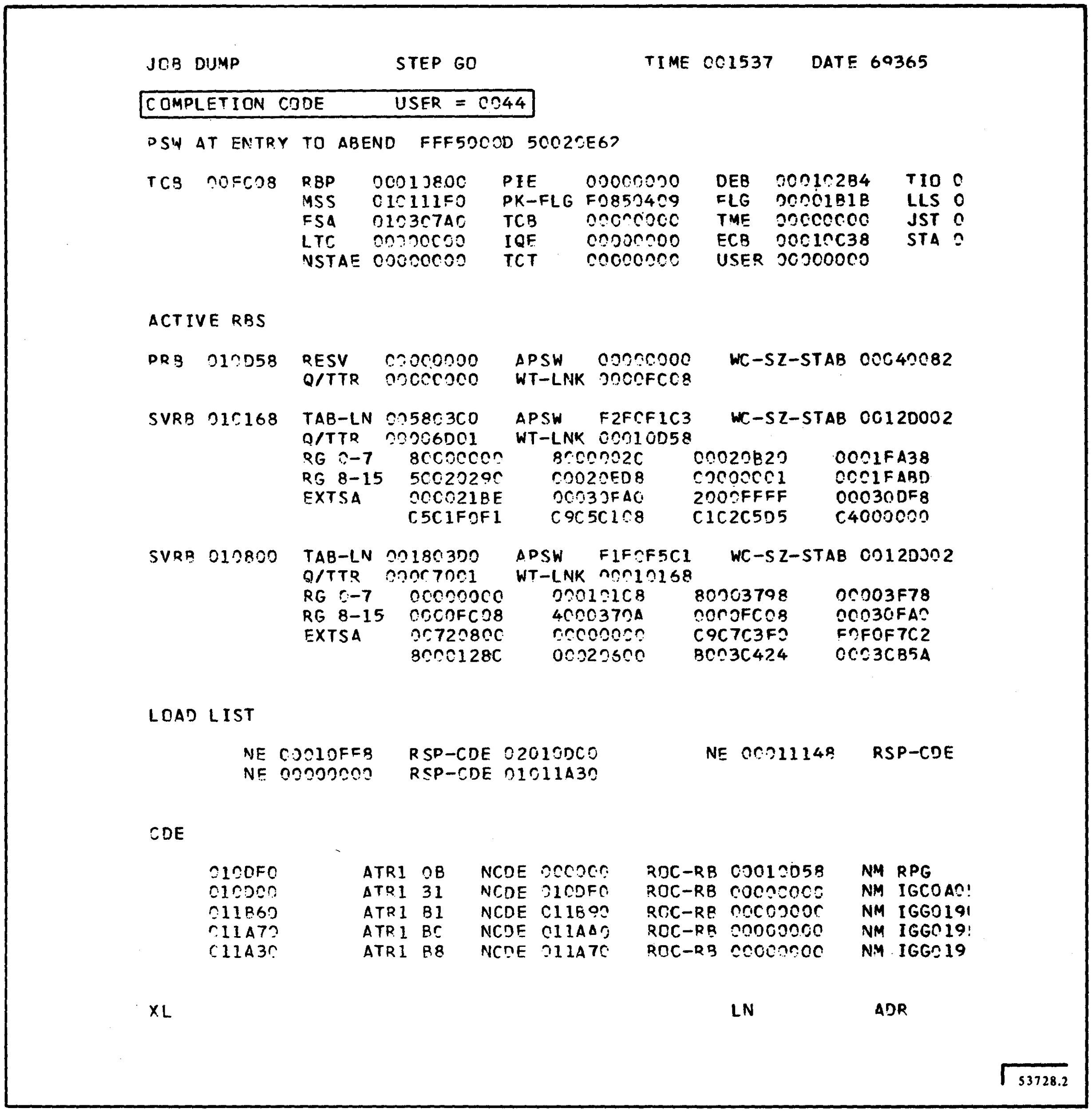
Completion Code	Error Type (Specific)	Displacement from Register 3	
		X'121 –	122'
52	Record not found.	80	00
	Record length check.	40	00
	Space not found	20	00
	Invalid request.	18	00
	Uncorrectable I/O error.	08	00
	End of data.	04	00
	Uncorrectable error other than an I/O error.	02	00
	A READ with exclusive control was not preceded by a WRITE with exclusive control.	01	00
	A WRITE macro instruction was addressed to an input data set.	00	40
	An extended search was specified with the DCBLIMCT field set to zero.	00	20
	The block requested is not within the data set.	00	10
	A write-by-identification (DI) addressed record zero.	00	08
	A search-on-key (DK) was specified with the DCBKEYLE field set to zero or without an address for the key.	00	04
	A macro instruction used an option not set in the DCB.	00	02
	The key for the fixed-length record to be added begins with hex. FF.	00	01
56	Record not found.	N/A	80
	Record length check.	N/A	40
	Space not found in which to add a record.	N/A	20
	Invalid request.	N/A	10
	Uncorrectable I/O error.	N/A	08
	Unreachable block.	N/A	04
	Overflow record.	N/A	02
	Duplicate record presented for inclusion in the data set.	N/A	01
60	Lower key limit not found.	80	00
	Invalid device address for lower limit.	40	00
	Space not found.	20	00
	Invalid request.	10	00
	Uncorrectable input error.	08	00
	Uncorrectable output error.	04	00
	Unreachable block.	02	00
	Overflow record.	01	00
	Sequence check.	00	80
	Duplicate record.	00	40
	DCB closed when error was detected.	00	20
	Overflow record.	00	10

Figure 203. I/O Error Analysis Table



```
     OS/RPG E *V1L6*                        DUMP       RPG               12/31/69            PAGE
                                                       SYMBOL  TABLES

  RESULTING  INDICATORS

  ADDRESS RI        ADDRESS RI        ADDRESS RI        ADDRESS RI        ADDRESS RI        ADDRESS RI

   000014 1P         000015 LR         000016 00        000017 C1          00007A L0         000085 H0
   000087 H2         000088 H3         000089 H4         00008A H5         00008B H6         00008C H7
   00008E H9

  FIELD  NAMES

  ADDRESS FIELD          ADDRESS FIELD          ADDRESS FIELD          ADDRESS FIELD          ADDRESS

  000123  ALPHA



                                                   MEMORY MAP

  INPUT/OUTPUT INTERCEPT                                          000130
  TABLE (INPUT AND OUTPUT)                                        00012C
  DETERMINE RECORD TYPE                                           0002D0
  DATA SPECIFICATION                                              000158
  GET INPUT RECORD                                                000520
  DETAIL CALCULATIONS                                             0006EC
  TOTAL CALCULATIONS                                              000700
  DETAIL LINES                                                    0007CA
  TOTAL LINES                                                     000718
  INPUT/OUTPUT REQUEST BLOCKS POINTER                             001CE8
  LOCATION OF DCB POINTERS                                        0008B8
  INPUT/OUTPUT INTERFACE ROUTINES                                 000C30
  LINE COUNTER                                                    000908
  WORK AREA POINTER                                               0014A0
  OVERFLOW BYPASS                                                 0007A0
  TABLE(ASSEMBLE 4)                                               00086C
  OVERFLOW LINES                                                  000742
  LINKAGE PROGRAM                                                 00116C


  PROGRAM LENGTH  0015C1

  'END OF COMPILATION'
```

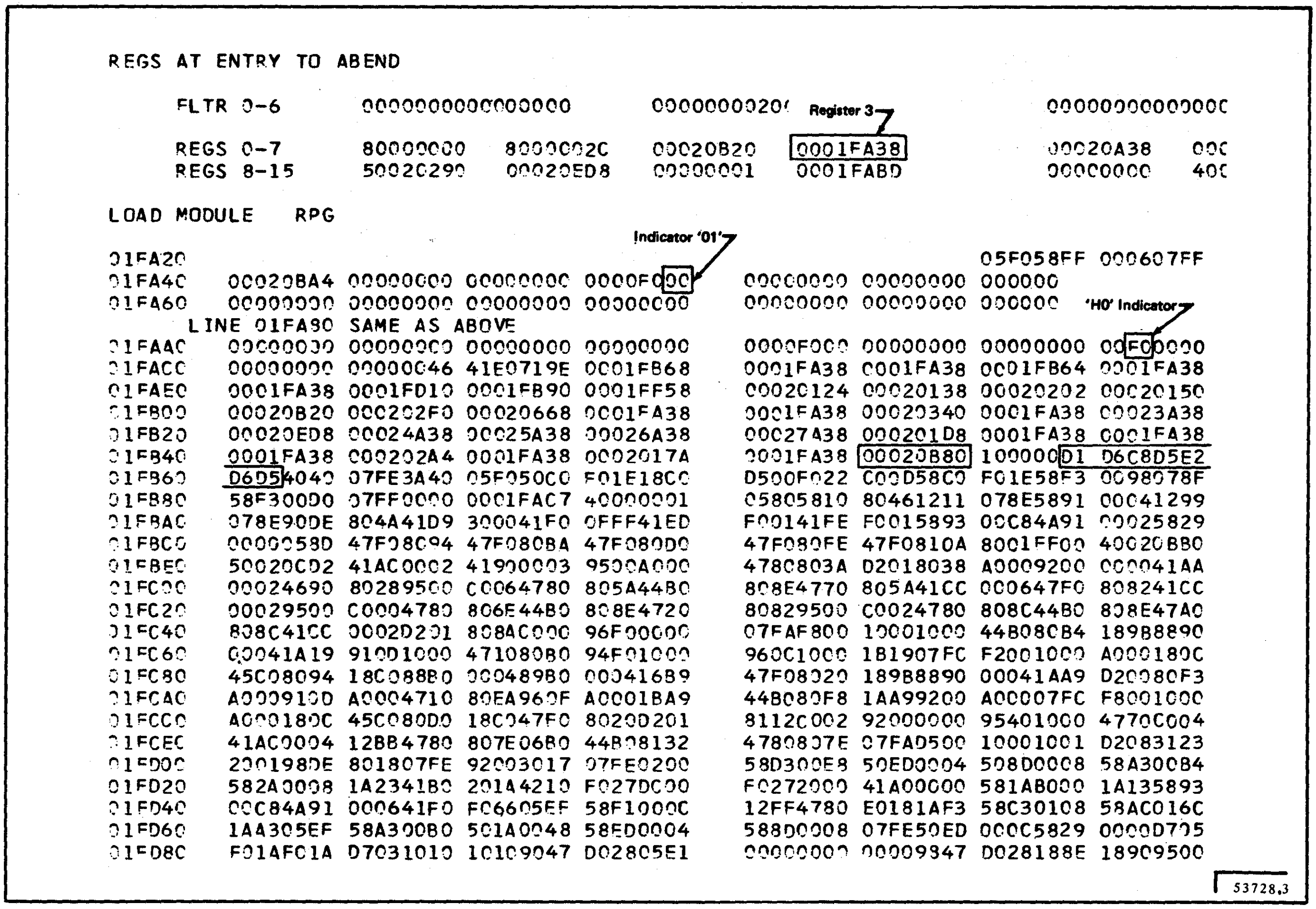
Figure 204. An OS Core Dump of an RPG Program (Part 1 of 5)



```
JOB DUMP              STEP GO                 TIME 001537   DATE 69365

COMPLETION CODE       USER = 0044

PSW AT ENTRY TO ABEND  FFF5000D 50020E62

TCB   00FC08  RBP    00010800   PIE    00000000   DEB  00010284   TIO 0
              MSS    010111F0   PK-FLG F0850409   FLG  00001B1B   LLS 0
              FSA    01030 7A0  TCB    00000000   TME  00000000   JST 0
              LTC    00000000   IQE    00000000   ECB  00010C38   STA 0
              NSTAE 00000000    TCT    00000000   USER 00000000


ACTIVE RBS

PRB   010D58  RESV    00000000   APSW    00000000   WC-SZ-STAB 00040082
              Q/TTR   00000000   WT-LNK  0000FC08

SVRB 010168   TAB-LN 005803C0    APSW    F2FCF1C3   WC-SZ-STAB 0012D002
              Q/TTR   00006D01   WT-LNK  00010D58
              RG 0-7     80000000      8000002C      00020B20      0001FA38
              RG 8-15    50020290      00020ED8      00000001      0001FABD
              EXTSA      000021BE      00030FA0      2000FFFF      00030DF8
                         C5C1F0F1      C9C5C108      C1C2C5D5      C4000000

SVRB 010800   TAB-LN 001803D0    APSW    F1F0F5C1   WC-SZ-STAB 0012D002
              Q/TTR   00007001   WT-LNK  00010168
              RG 0-7     00000000      000101C8      80003798      00003F78
              RG 8-15    0000FC08      4000370A      0000FC08      00030FA0
              EXTSA      00720800      00000000      C9C7C3F0      F0F0F7C2
                         8000128C      00020600      B003C424      0003CB5A


LOAD LIST

          NE 00010FF8   RSP-CDE 02010DC0           NE 00011148   RSP-CDE
          NE 00000000   RSP-CDE 01011A30


CDE

      010DF0        ATR1  0B   NCDE 000000   ROC-RB 00010D58   NM RPG
      010D00        ATR1  31   NCDE 010DF0   ROC-RB 00000000   NM IGC0A0
      011B60        ATR1  B1   NCDE 011B90   ROC-RB 00000000   NM IGG019
      011A70        ATR1  BC   NCDE 011A40   ROC-RB 00000000   NM IGG019
      011A30        ATR1  B8   NCDE 011A70   ROC-RB 00000000   NM IGG019

XL                                                LN           ADR
```

53728.2

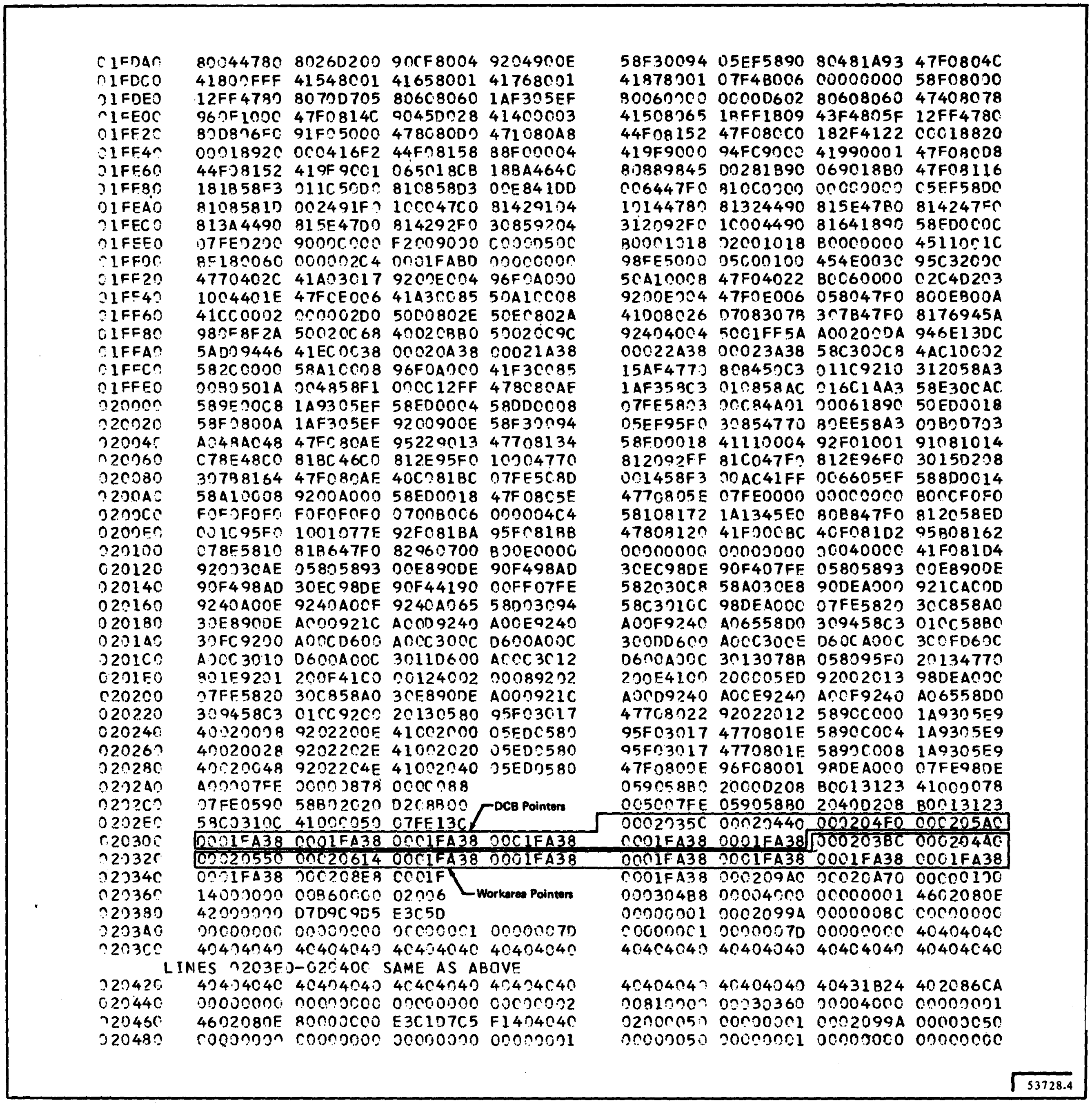
Figure 204. An OS Core Dump of an RPG Program (Part 2 of 5)



```
REGS AT ENTRY TO ABEND

      FLTR 0-6        0000000000000000          00000000020(                     000000000000000C

      REGS 0-7        80000000   8000002C     00020B20   0001FA38               00020A38    000
      REGS 8-15       50020290   00020ED8     00000001   0001FABD               00000000    400

LOAD MODULE    RPG

01FA20                                                                      05F058FF 000607FF
01FA40     00020BA4 00000000 0000000C 0000F000     00000000 00000000 000000
01FA60     00000000 00000000 00000000 00000C00     00000000 00000000 000000
      LINE 01FA80  SAME AS ABOVE
01FAA0     00000000 00000000 00000000 00000000     0000F000 00000000 00000000 00F00000
01FAC0     00000000 00000046 41E0719E 0001FB68     0001FA38 0001FA38 0001FB64 0001FA38
01FAE0     0001FA38 0001FD10 0001FB90 0001FF58     00020124 00020138 00020202 00020150
01FB00     00020B20 000202F0 00020668 0001FA38     0001FA38 00020340 0001FA38 00023A38
01FB20     00020ED8 00024A38 00025A38 00026A38     00027A38 000201D8 0001FA38 0001FA38
01FB40     0001FA38 000202A4 0001FA38 0002017A     0001FA38 00020B80 100000D1 D6C8D5E2
01FB60     D6D54040 07FE3A40 05F050C0 F01E18C0     D500F022 C00D58C0 F01E58F3 0098078F
01FB80     58F30000 07FF0000 0001FAC7 40000001     05805810 80461211 078E5891 00041299
01FBA0     078E900E 804A41D9 300041F0 0FFF41ED     F00141FE F0015893 00C84A91 00025829
01FBC0     0000058D 47F08094 47F080BA 47F080D0     47F080FE 47F0810A 8001FF00 40020BB0
01FBE0     50020CD2 41AC0002 41900003 9500A000     4780803A D2018038 A0009200 000041AA
01FC00     00024690 80289500 C0064780 805A44B0     808E4770 805A41CC 000647F0 808241CC
01FC20     00029500 C0004780 806E44B0 808E4720     80829500 C0024780 808C44B0 808E47A0
01FC40     808C41CC 0002D201 808AC000 96F00000     07FAF800 10001000 44B080B4 189B8890
01FC60     00041A19 910D1000 47108080 94F01000     960C1000 1B1907FC F2001000 A000180C
01FC80     45C08094 18C088B0 000489B0 000416B9     47F08020 189B8890 00041AA9 D20080F3
01FCA0     A000910D A0004710 80EA960F A0001BA9     44B080F8 1AA99200 A00007FC F8001000
01FCC0     A000180C 45C080D0 18C047F0 8020D201     8112C002 92000000 95401000 4770C004
01FCE0     41AC0004 12BB4780 807E06B0 44B08132     4780807E 07FAD500 10001001 D2083123
01FD00     200198DE 801807FE 92003017 07FE0200     58D300E8 50ED0004 508D0008 58A300B4
01FD20     582A0008 1A2341B0 201A4210 F027DC00     F0272000 41A00000 581AB000 1A135893
01FD40     00C84A91 000641F0 F06605EF 58F1000C     12FF4780 E0181AF3 58C30108 58AC016C
01FD60     1AA305EF 58A300B0 501A0048 58ED0004     588D0008 07FE50ED 000C5829 0000D705
01FD80     F01AF01A D7031010 10109047 D02805E1     00000000 00009347 D028188E 18909500
```

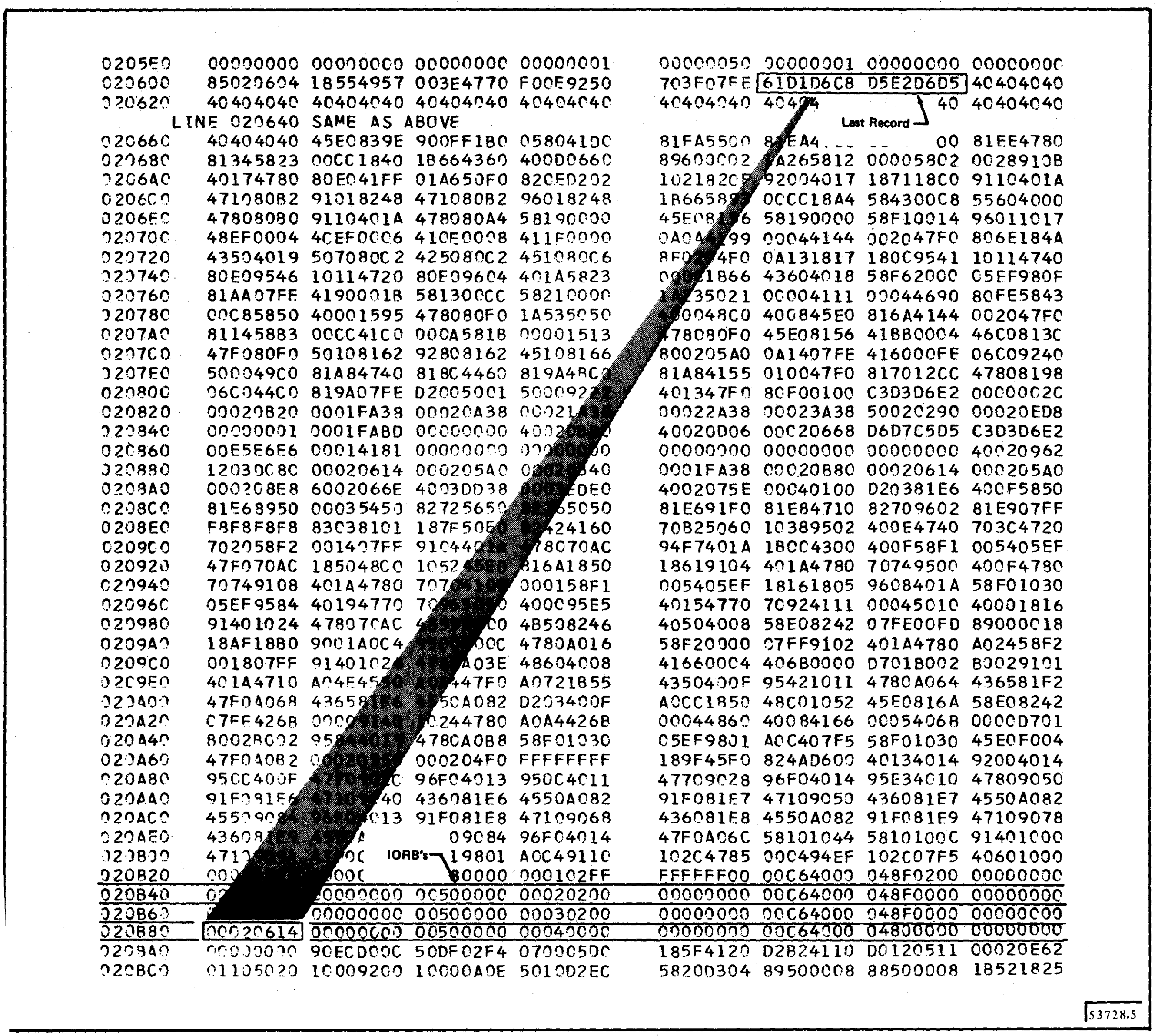
53728.3

Figure 204. An OS Core Dump of an RPG Program (Part 3 of 5)


DCB Pointers
Workarea Pointers
53728.4

```
01FDA0    80044780 8026D200 90CF8004 9204900E    58F30094 05EF5890 80481A93 47F0804C
01FDC0    41800FFF 41548001 41658001 41768001    41878001 07F4B006 00000000 58F08000
01FDE0    12FF4780 8070D705 80608060 1AF305EF    80060000 0000D602 80608060 47408078
01FE00    960F1000 47F0814C 9045D028 41400003    41508065 18FF1809 43F4805F 12FF4780
01FE20    80D806F0 91F05000 478080D0 471080A8    44F08152 47F080C0 182F4122 00018820
01FE40    00018920 000416F2 44F08158 88F00004    419F9000 94FC9000 41990001 47F080D8
01FE60    44F08152 419F9001 065018CB 18BA4640    80889845 D0281B90 069018B0 47F08116
01FE80    181B58F3 011C50D0 810858D3 00E841DD    006447F0 810C0000 00000000 C5EF58D0
01FEA0    8108581D 002491F0 100047C0 81429104    10144780 81324490 815E47B0 814247F0
01FEC0    813A4490 815E47D0 814292F0 30859204    312092F0 10004490 81641890 58FD000C
01FEE0    07FED200 9000C000 F2009000 C000D500    B0001018 02001018 B0000000 4511001C
01FF00    8F180060 000002C4 0001FABD 00000000    98FE5000 05C00100 454E0030 95C32000
01FF20    4770402C 41A03017 9200E004 96F0A000    50A10008 47F04022 B0C60000 02C4D203
01FF40    1004401E 47FCE006 41A30085 50A10008    9200E004 47F0E006 058047F0 800EB00A
01FF60    41CC0002 000002D0 50D0802E 50E0802A    41D08026 D708307B 307B47F0 8176945A
01FF80    980F8F2A 50020C68 40020BB0 50020C9C    92404004 5001FF5A A00200DA 946E13DC
01FFA0    5AD09446 41EC0C38 00020A38 00021A38    00022A38 00023A38 58C300C8 4AC10002
01FFC0    582C0000 58A10008 96F0A000 41F30085    15AF4770 808450C3 011C9210 312058A3
01FFE0    0080501A 004858F1 000C12FF 478080AE    1AF358C3 010858AC 016C1AA3 58E30CAC
020000    589E00C8 1A9305EF 58ED0004 58DD0008    07FE5803 00C84A01 00061890 50ED0018
020020    58F0800A 1AF305EF 9200900E 58F30094    05EF95F0 30854770 80EE58A3 00B0D703
020040    A048A048 47FC80AE 95229013 47708134    58FD0018 41110004 92F01001 91081014
020060    C78E48C0 81BC46C0 812E95F0 10004770    812092FF 81C047F0 812E96F0 3015D208
020080    307B8164 47F080AE 40C081BC 07FE5C8D    001458F3 00AC41FF 006605EF 588D0014
0200A0    58A10008 9200A000 58ED0018 47F08C5E    4770805E 07FE0000 00000000 B00CF0F0
0200C0    F0F0F0F0 F0F0F0F0 0700B0C6 000004C4    58108172 1A1345E0 80B847F0 812058ED
0200E0    001C95F0 1001077E 92F081BA 95F081BB    47808120 41F000BC 40F081D2 95B08162
020100    078E5810 81B647F0 82960700 B00E0000    00000000 00000000 00040000 41F081D4
020120    920030AE 05805893 00E890DE 90F498AD    30EC98DE 90F407FE 05805893 00E890DE
020140    90F498AD 30EC98DE 90F44190 00FF07FE    582030C8 58A030E8 90DEA000 921CAC0D
020160    9240A00E 9240A00F 9240A065 58D03094    58C3010C 98DEA000 07FE5820 30C858A0
020180    30E890DE A000921C A00D9240 A00E9240    A00F9240 A06558D0 309458C3 010C58B0
0201A0    30FC9200 A00CD600 A00C300C D600A00C    300DD600 A00C300E D60CA00C 300FD60C
0201C0    A00C3010 D600A00C 3011D600 A00C3012    D600A00C 3013078B 058095F0 20134770
0201E0    801E9201 200F41C0 00124002 00089202    200E4100 200005ED 92002013 98DEA000
020200    07FE5820 30C858A0 30E890DE A000921C    A00D9240 A00E9240 A00F9240 A06558D0
020220    309458C3 010C9200 20130580 95F03017    47708022 92022012 589C0000 1A9305E9
020240    40020008 9202200E 41002000 05ED0580    95F03017 4770801E 5890C004 1A9305E9
020260    40020028 9202202E 41002020 05ED0580    95F03017 4770801E 5890C008 1A9305E9
020280    40020048 9202204E 41002040 05ED0580    47F0800E 96F08001 98DEA000 07FE98DE
0202A0    A00007FE 00000878 0000088             059058B0 2000D208 B0013123 41000078
0202C0    07FE0590 58B02020 D208B00             005007FE 059058B0 2040D208 B0013123
0202E0    58C0310C 41000050 07FE13C             0002035C 00020440 000204F0 000205A0
020300    0001FA38 0001FA38 0001FA38 0001FA38    0001FA38 0001FA38 000203BC 000204A0
020320    00020550 00020614 0001FA38 0001FA38    0001FA38 0001FA38 0001FA38 0001FA38
020340    0001FA38 000208E8 0001F                0001FA38 000209A0 00020A70 00000100
020360    14000000 00B60000 02006                000304B8 00004000 00000001 4602080E
020380    42000000 D7D9C9D5 E3C5D                00000001 0002099A 0000008C 00000000
0203A0    00000000 00000000 00000001 0000007D    000000C1 0000007D 00000000 40404040
0203C0    40404040 40404040 40404040 40404040    40404040 40404040 40404040 40404040
      LINES 0203E0-020400 SAME AS ABOVE
020420    40404040 40404040 40404040 40404040    40404040 40404040 40431B24 402086CA
020440    00000000 00000000 00000000 00000002    00810000 00030360 00004000 00000001
020460    4602080E 80000C00 E3C1D7C5 F1404040    02000050 00000001 0002099A 00000050
020480    00000000 00000000 00000000 00000001    00000050 00000001 00000000 00000000
```

Figure 204. An OS Core Dump of an RPG Program (Part 4 of 5)



```
020SE0   00000000 00000000 00000000 00000001   00000050 00000001 00000000 00000000
020600   85020604 18554957 003E4770 F00E9250   703F07FE|61D1D6C8 D5E2D6D5|40404040
020620   40404040 40404040 40404040 40404040   40404040 4040              40 40404040
                                                                Last Record
     LINE 020640 SAME AS ABOVE
020660   40404040 45E0839E 900FF1B0 0580410C   81FA5500 8 EA4.-- --       00 81EE4780
020680   81345823 00CC1840 18664360 400D0660   89600002  A265812 00005802 0028910B
0206A0   40174780 80E041FF 01A650F0 820ED202   1021820  92004017 187118C0 9110401A
0206C0   471080B2 91018248 471080B2 96018248   1B6658   0CCC18A4 584300C8 55604000
0206E0   478080B0 9110401A 478080A4 58190000   45E08  6 58190000 58F10014 96011017
020700   48EF0004 4CEF0006 410E0008 411F0000   0A0A  99 00044144 002047F0 806E184A
020720   43504019 507080C2 425080C2 451080C6   8F0  4F0 0A131817 180C9541 10114740
020740   80E09546 10114720 80E09604 401A5823   0   1B66 43604018 58F62000 05EF980F
020760   81AA07FE 4190001B 581300CC 58210000   1  35021 00004111 00044690 80FE5843
020780   00C85850 40001595 478080F0 1A535050    00048C0 400845E0 816A4144 002047F0
0207A0   811458B3 00CC41C0 00CA581B 00001513    78080F0 45E08156 41BB0004 46C0813C
0207C0   47F080F0 50108162 92808162 45108166   800205A0 0A1407FE 416000FE 06C09240
0207E0   500049C0 81A84740 818C4460 819A4BC    81A84155 010047F0 817012CC 47808198
020800   06C044C0 819A07FE D2005001 500092     401347F0 80F00100 C3D3D6E2 00C0002C
020820   00020B20 0001FA38 00020A38 00021A3    00022A38 00023A38 50020290 00020ED8
020840   00000001 0001FABD 00000000 40020      40020D06 00C20668 D6D7C5D5 C3D3D6E2
020860   00E5E6E6 00014181 00000000 00    00   00000000 00000000 00000000 40020962
020880   12030C8C 00020614 000205A0 0     40   0001FA38 00020B80 00020614 000205A0
0208A0   000208E8 6002066E 4003DD38     EDE0   4002075E 00040100 D20381E6 400F5850
0208C0   81E68950 00035450 82725650   65050    81E691F0 81E84710 82709602 81E907FF
0208E0   F8F8F8F8 83038101 187F50E    424160   70B25060 10389502 400E4740 703C4720
020900   702058F2 001407FF 91044     78070AC   94F7401A 1B0C4300 400F58F1 005405EF
020920   47F070AC 185048C0 1052      16A1850   18619104 401A4780 70749500 400F4780
020940   70749108 401A4780 707       000158F1  005405EF 18161805 9608401A 58F01030
020960   05EF9584 40194770 70       0 400095E5  40154770 70924111 00045010 40001816
020980   91401024 478070AC 4       00 4B508246  40504008 58E08242 07FE00FD 89000018
0209A0   18AF18B0 9001A0C4        00C 4780A016  58F20000 07FF9102 401A4780 A02458F2
0209C0   001807FF 9140102     A03E 48604008    416600C4 406B0000 D701B002 B0029101
0209E0   401A4710 A04E45     447F0 A0721B55    4350400F 95421011 4780A064 436581F2
020A00   47F0A068 43658     50A082 D203400F    A0CC1850 48C01052 45E0816A 58E08242
020A20   C7FE426B 0000     0244780 A0A4426B    00044860 40084166 0005406B 0000D701
020A40   8002B002 95      4780A0B8 58F01030    05EF9801 A0C407F5 58F01030 45E0F004
020A60   47F0A082 0      000204F0 FFFFFFFF     189F45F0 824AD600 40134014 92004014
020A80   95C0400F       C 96F04013 950C4011    47709028 96F04014 95E34010 47809050
020AA0   91F081E6     40 436081E6 4550A082     91F081E7 47109050 436081E7 4550A082
020AC0   455090     013 91F081E8 47109068      436081E8 4550A082 91F081E9 47109078
020AE0   43608     A        09084 96F04014     47F0A06C 58101044 5810100C 91401000
020B00   471      0C  IORB's 19801 A0C49110    102C4785 000494EF 102C07F5 40601000
020B20   00      00C          0000 000102FF    FFFFFF00 00C64000 048F0200 00000000
020B40   0       0000000 0C500000 00020200     00000000 00C64000 048F0000 00000000
020B60           00000000 00500000 00030200    00000000 00C64000 048F0000 00000000
020B80  |00020614|00000000 00500000 00040000   00000000 00C64000 04800000 00000000
020BA0   00000000 90ECD00C 50DF02F4 070005D0   185F4120 D2B24110 D0120511 00020E62
020BC0   01105020 10009200 10000A0E 5010D2EC   5820D304 89500008 88500008 1B521825
```

53728.5

Figure 204. An OS Core Dump of an RPG Program (Part 5 of 5)

DETERMINING THE LAST FILE PROCESSED

If the DCB address or the IORB address is indicated in the error analysis table, the last file processed may be determined.

Note: The DCB address may be found if the DECB address is known. This is discussed under *Location of DCB Pointers* in this section.

Input/Output Request Blocks Pointers

The displacement given in the RPG memory map for Input/Output Request Blocks Pointers (in this case 1FA38+10E8= 20B20) points to the beginning location of a continuous series of Input/output request blocks (IORBs). Each IORB is 20 bytes in length.

An IORB is where information about the current use of a data set is stored; this block is used by the RPG program. One IORB is created in sequence for each file specified on the file description specifications. The fourth IORB would be associated with the fourth file specified.

If an IORB address is indicated by the error analysis table, you can determine the last file and the last record to be processed. Increment the IORB pointer (20B20 from the memory map) by X'20' until it matches the IORB address (20B80 indicated by error analysis table).

IORB Pointer	X'20B20'	first file
	X'20B40'	second file
	X'20B60'	third file
	X'20B80'	fourth file

In this case the fourth file was the last processed. Figure 205 shows the format of an IORB. The first four bytes are the address of the last record processed (020614). In this case the last record is /JOHNSON (X'61D1D6C8D5E2D6D5').

Displacement	*Comments*
0	Record address
4	Address of list of parameters for direct access method or index sequential.
8	Record length
10	Line counter value
12	Line counter TAC. X'11' Line counter present X'EE' Error in line counter routine
13	File number
14	Action Byte X'00' READ X'02' WRITE X'04' STACKER SELECT
15	Skip before or stacker number
16	Space before
17	Skip after
18	Space after
19	Overflow or end-of-file switch X'00' SWITCH OFF X'11' PRINTER OVERFLOW X'22' EOF or END of Extents X'33' END OF ISFMS LIMITS
20	Overflow switch two
21	Sense overflow
22	Index sequential with limits
23	First pass switch used with input specifications
24	File type
25	Stacker select codes for the device assigned to the file
27	Record read from index sequential file X'11' Record read X'00' Record not read
28	SETL switch X'FF' 'SETL' or 'SETFL' have been executed X'00' 'ESETL' or 'ENDFL' have been executed
29	Index-sequential key length
30	Index-sequential key location within data

Figure 205. Format of an IORB Table

Location of DCB Pointers

The displacement address associated with this entry in the RPG memory map points to the beginning location of a list of data control block (DCB) addresses for all files in the program. Each address is four bytes in length and there are ten addresses. The order in which they appear is the same as the order in which the files were specified on the File Description Specifications form. If there are less than ten DCBs used by the program, the unused DCB pointer address areas of this list contain the address at which the RPG object program is loaded (the same as register 3). Following the DCB pointers are ten 4-byte addresses: the work area addresses. There is one work area for each DCB. Work area addresses are in the same order as the DCB addresses. Work areas contain the records (input or output depending on the type of file) currently used within the RPG program.

A DCB is where information about the current use of a data set is stored. When the error analysis table (Figure 202) gives the address of a DCB, the address must be compared to the addresses found in the list of DCB addresses. When the equal address is found, the number of the DCB table pointer entry will denote which file specified on File Description Specification form caused the error. Further information on DCBs may be found in *IBM System/360 Operating System: System Control Blocks*, Form C28-6628.

Note: The DECB address plus nine points to a three-byte address. This address is the DCB address.

DETERMINING FIELD STATUS

You can find the contents of a field by using the following technique.

Below the Symbol Tables heading are the Field Names. The first entry in the example is for the field ALPHA. X'000123' is a displacement for the field ALPHA; by adding register 3 to this displacement an address is produced (000123+1FA38=1FB5B). This address points to the contents of the field ALPHA just before the program terminated. In this example ALPHA contains JOHNSON (X'D1D6C8D5E2D6D5'). The input record '/JOHNSON' caused the program termination because '/' is an invalid character in this program.

DETERMINING INDICATOR STATUS

Figure 204, part 1, has a heading Resulting Indicators. Listed below this heading are indicators and hexadecimal displacements. The following examples show how to use this information.

Indicator '01' has a displacement of X'17' associated with it. X'1FA38' (register 3) added to X'17' points to a byte containing X'00'. This value means that indicator 01 is off. The H0 indicator byte, located at a displacement of X'85' from register 3 (1FA38+85=1FABD) contains X'F0'. This means that H0 is on.

This summary contains a brief position-by-position description of each of the five RPG specification forms. The purpose of the summary is to provide the user with a concise reference guide.

The first five items are common to all five specification forms.

Page (1-2)

Enter number of specification page. Assign ascending numbers to the pages of each program specification set to collate in the following sequence:

1. Control card specifications
2. File description specifications
3. Extension specifications
4. Line counter specifications
5. Input specifications
6. Calculation specifications
7. Output-format specifications

Line (3-5)

First two digits of line number are preprinted. Use third position (5) to identify additional lines to be inserted between two preprinted lines.

Form Type (6)

Contains a preprinted code (H, F, E, L, I, C, or O) which must be punched into all RPG specification cards.

Comments (7)

Enter an asterisk (*) in each line to be used exclusively as a comments line.

Program Identification (75-80)

Insert any information to identify certain cards or portions of an RPG source program.

CONTROL CARD SPECIFICATIONS

Positions 7-16

This positions are not used.

Sterling 17-20

Blank entries in these positions indicate that sterling currency is not being used.

Input Shillings (17)

1=Input shilling field is in IBM format.

2=Input shilling field is in BSI format.

Input Pence (18)

1=Input shilling field is in IBM format.

2=Input shilling field is in BSI format.

Output Shillings (19)

0 or blank=Output shilling field is to be printed only.

1=Output shilling field is to be in IBM format.

2=Output shilling field is to be in BSI format.

Output Pence (20)

0 or blank=Output pence field is to be printed only.

1=Output pence field is to be in the IBM format.

2=Output pence field is to be in the BSI format.

Inverted Print (21)

Leave blank for domestic format of month/day/year of UDATE and decimal point notation for numeric fields.

Enter D for United Kingdom format of day/month/year and decimal point notation.

Enter I for European format of day.month.year and decimal comma notation.

Positions 22-25

These positions are not used.

Alternate Collating Sequence (26)

Enter A if an external subroutine is used to translate the sequence of a matching field to the collating sequence of System/360. Otherwise, leave this position blank.

Positions 27-74

These positions are not used.

FILE DESCRIPTION SPECIFICATIONS

Filename (7-14)

Enter a name for each file used in the program. Names must be left justified.

File Type (15)

Enter one of the letters I, O, U, or C to identify input, output, update, or combined file. D is not used.

File Designation (16)

P for primary, and S is for secondary input or combined files. Enter P if only one input file or combined file is used. C for chained file, R for RA file, or T for table file. Leave blank for output files. D is unused.

End of File (17)

Enter E for each input file or combined file that is to be checked to determine when the last record has been read and processed (LR indicator on). Leave blank if LR is to be set on when the last record of all input and combined files has been read.

Sequence (18)

Required when Matching Fields is used. A if the input file or combined file is in ascending sequence, D if it is in descending sequence. Leave blank for output files, or for input or combined files without Matching Fields.

File Format (19)

F = Fixed-length records.

V = Variable-length records.

S = Spanned, unblocked, variable-length records.

M = Spanned, blocked, variable-length records.

Block Length (20-23)

Unblocked

If the records are unblocked, enter the length of a record. If variable-length records are used, enter the length of the longest record.

Blocked

If the records are blocked, enter the length of the largest block.

Record Length (24-27)

Used to enter the length of the logical records contained in the file. If the file contains records that are variable in length, enter the length of the largest record. Maximum record size is 4000 characters.

Mode of Processing (28)

Used to indicate the method or mode by which the file is processed. Enter an L in this position if a segment of the file is to be processed. Enter an R in this position if the records are to be processed randomly. If no entry is made in this position for the file, the entire file will be processed sequentially.

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

If an input, output, or update indexed-sequential file identifies records by key, enter the number of positions in the key. If the file is a record address file, enter the number of positions that each entry in theRA file occupies.

Record Address Type (31)

If the records from the file are retrieved by using record keys, enter a K in this position. If record ID is used to retrieve records, enter 1 in this position. A is unused.

Type of File Organization (32)

Leave blank if the file is organized sequentially. Enter an I if the file has indexed-sequential organization. Enter a D if the file has direct organization. T and 1-9 are unused.

Overflow Indicator (33-34)

If overflow indicators are used, enter the overflow indicator associated with the file. A maximum of eight overflow indicators is allowed: OA, OB, OC, OD, OE, OF, OG, and OV.

Key Field Starting Location (35-38)

Indicates the location of the key field within the data record. Enter the starting position of the key field. This entry is required for indexed-sequential files unless a key area, separate from the data area, is to be defined for unblocked indexed-sequential files on either the input or output-format specifications. This entry must then be omitted.

This entry must not be 1 for unblocked indexed-sequential files.

Extension Code (39)

Indicates that additional information about the file is coded on the Extension Specifications form, or Line Counter Specifications form.

Enter an E if the file defined on the line is a chaining file, table file, or a record address file (RA file).

Device (40-46)

Relates a file to a specific type of input or output unit.

If the output file is a printer, enter PRINTER.

If the file is an I/O file and it is associated with a card reader or card punch unit, enter:

1.	READ01	IBM 2501 Card Reader
2.	READ20	IBM 2520 Card Read-Punch
3.	READ40	IBM 2540 Card Read-Punch
4.	READ42	IBM 1442 Card Read-Punch

If the file is an I/O file and it is associated with a tape unit enter TAPE.

If the file is associated with a direct access storage device, enter DADEVT. RPG will also recognize the following as specific devices related to a file:

1.	DISK11	IBM 2311 Disk Storage Drive
2.	DISK14	IBM 2314 Direct Access Storage Facility
3.	*CEL01	IBM 2321 Data Cell Drive
4.	*CEL0111	IBM 2321 Data Cell Drive
5.	*CEL0114	IBM 2321 Data Cell Drive

*Accepted to provide compatability with DOS RPG.

Symbolic Device (47-52)

These positions are not used.

Labels (53)

S = Standard labels

E = Standard labels followed by user-standard labels

N = Unused

b = No labels

Name of Label Exit (54-59)

Enter the name of the routine to process user-standard labels. If the entry is shorter than six characters, it must be left justified.

Extent Exit for DAM/Core Index (60-65)

These positions are not used.

File Addition/Unordered (66)

Entries in this position are used in combination with those in position 15 to determine indexed-sequential functions. To add records to a file, enter an A. Also refer to the entry for positions 16-18 of the Output-Format Specifications form.

Number of Tracks for Cylinder Overflow (67)

For an indexed-sequential load function, enter the number of tracks (between 0 and 8 for 2311; 0 and 9 for 2314/2321) per data cylinder that are to be used for overflow records. This information can be supplied at object program execution through the DCB parameter in the file's DD card. If no entry is made in position 67, the type and size of overflow area can be specified in the file's DD card(s) at object execution time. See the section *RPG User's Guide for ISAM Processing* for a complete discussion of overflow.

Number of Extents (68-69)

These positions are not used.

Tape Rewind (70)

This position is not used by OS RPG, but tape positioning may be controlled by appropriate parameters in the DD card.

EXTENSION SPECIFICATIONS

Record Sequence of the Chaining File (7-8)

Used only for chaining files. Enter the same entry that is made for the chaining file in Sequence on the Input Specifications form.

Number of Chaining Field (9-10)

Used only for chaining files. Enter the identifying number of the chaining field. This number is entered in Chaining Field of the Input Specifications form.

From Filename (11-18)

Used in conjunction with To Filename to identify, for the RPG program, the relationship between two files. For example, they provide the name of a chaining file and the name of the file that is chained to it.

To Filename (19-26)

See description of From Filename.

Table or Array Name (27-32)

Enter name of table: for alternating input format, enter name of table to which the first entry in each input record belongs; for non-alternating input format, enter name of single table.

For an RA file or chaining file, specify the label of the address conversion routine. Table name must consist of 6 characters, the first 3 must be TAB, the remaining 3 may be any alphabetic or numeric characters.

Number of Entries Per Record (33-35)

Enter the maximum number of table entries (arguments or functions) that are contained in each input record. The entry must be right justified.

Number of Entries Per Table or Array (36-39)

Enter the exact number of table or array entries (arguments or functions) contained in the table. The entry must be right justified.

Length of Entry (40-42)

Enter the length of each table entry. The maximum size of a numeric entry is 15 characters, of an alphameric entry 256 characters. The entry must be right justified.

P = Packed/B = Binary (43)

If the data for the table is in the packed decimal format, enter P in this position. Otherwise leave this position blank. Binary is unused.

Decimal Position (44)

If the data contained in the table is numeric, enter the number of decimal positions (1-9). Enter a zero if there are no decimal positions. If the field is alphameric, leave this position blank.

Sequence A/D (45)

If the data contained in the table is in ascending sequence, enter an A; in descending sequence, enter a D. Leave blank if the data is not in ascending or descending sequence or if this specification is not required.

Table or Array Name -- Alternating Format (46-51)

If alternating arguments and function tables are used, enter the second table name. It must be of the form TABnnn. The entry must be left justified.

Length of Entry (52-54)

Enter the length of each table entry. The maximum size of a numeric entry is 15 characters; of an alphameric entry 256 characters. The entry must be right justified.

P = Packed/B = Binary (55)

Enter a P if the data for the table is in the packed decimal format. Otherwise leave this column blank. B is unused.

Decimal Positions (56)

If the data contained in the table is numeric, enter the number of decimal positions (1-9). Enter a zero if there are no decimal positions. If the field is alphameric, leave blank.

Sequence (57)

If the data contained in the table is in ascending sequence, enter an A; in descending sequence, enter a D. Leave blank if the data is not in ascending or descending sequence or if this specification is not required.

Comments (58-74)

Leave positions 58-74 blank, unless comments are entered in these positions.

LINE COUNTER SPECIFICATIONS

Filename (7-14)

Enter the name of the output file.

Line Number (15-17)

Enter the number of the first line controlled by the carriage tape in positions 15-17.

The remaining specifications (items 2-12) perform the same function.

FL or Channel Number (18-19)

Enter the channel number corresponding to the line number in positions 15-17. The remaining specifications (items 2-12) perform the same function.

INPUT SPECIFICATIONS

Filename (7-14)

Enter a filename for each input or combined file, one entry per file. Must be left justified; it may have up to 8 characters; first character must be alphabetic; remaining characters may be alphabetic or numeric; special characters or embedded blanks may not be used.

AND/OR Relationship (14-16)

Enter AND to indicate that the AND relationship of Record Identification Codes in the preceding line is to be continued. Enter OR (positions 14 and 15) to indicate that the entries in Record Identification Codes of this line are to be in an OR relationship to the entries in 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 card types. Enter leading zeros. Enter any two alphabetic characters to indicate that sequence checking is not required. Lines with alphabetic entries in Sequence must precede lines with numeric entries. Any numeric entry in Sequence requires an entry in Number.

Number (17)

1 = One and only one record of a specific record type should be present in each group.

N = One or more records of a specific record type may be present in each group (used only with numeric entry in positions 15-16).

Option (18)

0 = Indicates that a record of a specific control group used need not be present. Leave blank if a record must be present, or if records are nonsequential (used only with numeric entry in positions 15-16).

Record Identifying Indicator or ** (19-20)

Enter any indicator from 01 through 99 to establish a two-digit code for the input record type defined in Record Identification Codes. This sets special condition(s) in the object program each time the input record is read. ** is unused.

Record Identification Codes (21-41)

This field is divided into three identical subfields: positions 21-27, 28-34, and 35-41. An AND relationship exists between these three subfields.

Position: Enter the number of the input record position containing the identifying code. Must be right justified.

NOT (N): Enter N if the code described must not be present in the position specified. Otherwise leave blank.

C/Z/D: Enter D if only the digit portion of the specified position is to be checked. Enter Z if only the zone portion is to be checked. Enter C if both portions are to be checked.

Character: If C/Z/D contains C or D, enter any one of the 256 EBCDIC characters.

If C/Z/D contains Z, enter: &, A through I, or 0 to check for a 12-zone; –, J through R, or 0 to check for an 11-zone; S-Z for 0-zone; 0 through 9, or blank to check for the absence of zones.

Note: Record Identification Codes may be continued in the subsequent specification line by means of an AND entry for AND relationships or an OR entry for OR relationships.

Stacker Select (42)

Enter number of stacker to which input cards are to be selected. Leave blank for single stacker devices and for combined files.

P = Packed/B = Binary (43)

P if input data in packed decimal format. Blank if input data in standard format. B is unused.

Field Location (44-51)

This specification describes location of fields in input records.

From (44-47): Enter the number of the input record position containing the first position of the field specified in Field Name. To define a field within the key area for each data record in an unblocked indexed-sequential file (with key not part of the data), enter a K to the immediate left of the high-order digit.

To (48-51): Enter the number of the input record position containing the last position of the field defined in Field Name. Entries must be right justified; leading zeros may be omitted. To define a field within the key area for each data record in an unblocked indexed-sequential file (with key not part of the data), enter a K to the immediate left of the high-order digit. If a K was specified in the From entry, then it must also be specified in the To entry, and vice versa. See the section *RPG User's Guide for ISAM Processing* for more information on key formats.

Decimal Positions (52)

Used only for numeric fields. Enter a digit 0 through 9 to indicate number of decimal positions in input field. Leave blank for alphameric fields.

Note: Each input field that is to be used in arithmetic operations must have an entry in Decimal Positions. Also use this specification for fields that are to be edited or zero suppressed.

Field Name (53-58)

Enter the name of each field defined in Field Location. Field names may be up to 6 characters in length; left justified. First character must be alphabetic; remaining characters may be alphabetic or numeric. Special characters or embedded blanks may not be used.

Control Level (59-60)

Enter any one of the control level indicators L1 through L9 to identify control fields (L1 for lowest level, L9 for highest level of control).

Matching Fields or Chaining Fields (61-62)

Enter any one of the codes M1, M2, or M3 to specify record matching for two input files or to specify sequence checking for the fields of a single input file.

Enter the codes C1 through C9 to specify a chaining field.

Field Record Relation (63-64)

Enter any one of the indicators defined in positions 19-20 of the input specifications to provide field record relation for identical fields contained in different locations (OR relationships) or for selective processing of chaining fields.

External indicators U1-U8 may also be used here but not with positions 59-60 (Control Level) and 61-62 (Matching Fields or Chaining Fields).

Field Indicators (65-70)

If the field is alphameric, that is, if position 52 is blank, only the Zero or Blank specification may be used. Enter any one of the indicators 01 through 99 or H0 through H9, as required, in each of the fields. Indicator in Plus (positions 65 and 66) is set on if the field specified in positions 53-58 contains a positive value, except +0. Indicator in Minus (positions 67 and 68) is set on if the field contains a negative value, except –0. Indicator in Zero or Blank (positions 69 and 70) is set on if the field contains no other character than zero or blank. It is also set on if the field is numeric and contains no other character than +0 or –0, or when a Blank After output specification is executed.

Sterling Sign Position (71-74)

Used only for programs processing sterling currency amounts. If sign of Sterling field is in the usual place, enter S in position 74. If sign is not in the usual place, enter the place in the record that contains the sign. Leave blank in all other RPG programs.

CALCULATION SPECIFICATIONS

Control Level (7-8)

Enter one of control level indicators L1 through L9, L0, or LR to specify that the calculation contained in this line is to be performed at total time. Leave blank if calculation is to be performed at detail time.

Indicators (9-17)

Enter one to three indicators to establish conditions controlling calculations specified in the line. Any of the indicators 01 through 99, L1 through L9, U1 through U8, L0, LR, MR, or any halt or overflow indicator may be used. Positions 9, 12, and 15 may contain blank or N.

Note: If there is more than one indicator in a line, RPG assumes an AND relationship between the individual indicators.

Factor 1 (18-27)

Field name Left justified, maximum length of 6 characters. Do not use special characters or embedded blanks. First character must be alphabetic. Must be defined in input specifications or as a Result Field in another calculation specification.

Literal Numeric: left justified, maximum length of ten characters, characters must be numeric (0 through 9), one decimal symbol and/or one sign (plus or minus) allowed. If a sign is present, it must be in the first position of the field.

Alphameric: left justified, must be enclosed in apostrophe symbol ('). maximum length 8 characters, any one of the 256 EBCDIC characters may be used.

Apostrophe symbols required within constants must be represented as two consecutive apostrophe symbols.

Operation (28-32)

Enter one of the RPG operation codes. An entry in this specification is required in each line, except in a comments line defined by an asterisk in position 7. Entries must be left justified.

Factor 2 (33-42)

Enter field name or literal to be used in the specified operation. (For a definition of field name and literal, see Factor 1.) Entries must be left justified.

Result Field (43-48)

Enter field name to designate location in storage into which result of pertinent operation is to be placed. First character of field name must be alphabetic; remaining characters may be alphabetic or numeric, cannot contain special characters or embedded blanks. Must be left justified.

Field Length (49-51)

Enter number of storage positions to be reserved for Result Field on this line. Maximum length of numeric Result Fields is 15 digits. Maximum length of alphameric Result Fields is 256 characters. Must be right justified. May be left blank, if length of Result Field specified in this line is defined in a previous line of calculation specifications or in input specifications.

Decimal Positions (52)

Enter number of decimal positions (0 through 9) to be reserved in result field. Required for all numeric result fields used with arithmetic operations. Must be blank if the result field is alphameric.

Note: The number of decimal positions is included in Field Length specification in positions 49-51. For example, if a number of decimal places specified is 7, the maximum number of places to the left of the decimal symbol is 15 – 7 = 8. (The maximum length of a numeric field is 15.)

Half Adjust (53)

Enter H to specify half adjustment of Result Field. Must be blank if Result Field is alphameric.

Resulting Indicators (54-59)

Any one of the indicators 01 through 99, H0 through H9, and L1 through L9 may be used for this specification.

Arithmetic Operations: Enter up to 3 indicators to be set on whenever the result of an arithmetic operation is positive (Plus, positions 54 and 55), or negative (Minus, positions 56 and 57), or zero (Zero, positions 58 and 59).

Compare Operations: Enter up to 3 indicators to be set on whenever result of COMP operation is Factor 1 > Factor 2 (High, positions 54 and 55), or Factor 1 < Factor 2 (Low, positions 56 and 57), or Factor 1 = Factor 2 (Equal, positions 58 and 59).

LOKUP Operations: Enter one or two indicators in High, or Low, or Equal, or High and Equal, or Low and Equal. (This defines type of entry to be located by means of LOKUP operation.)

TESTZ Operations: Enter up to 3 indicators to be set on whenever a 12-zone (High, positions 54 and 55), or an 11-zone (Low, positions 56 and 57), or any other zone or no zone (Equal, positions 58 and 59) is detected in the field specified in Result Field of this line.

SETOF, SETON Operations: Enter up to 3 indicators to be set on (SETON) or off (SETOF). If more than 3 indicators are to be set on (or off), specify another SETON (SETOF) statement in subsequent line. Any RPG indicator can be used, except L0, 00, and U1 through U8.

Note: Headings High, Low, Equal do not apply to indicators specified in conjunction with SETOF or SETON operations.

Comments (60-74)

Enter any desired comments. An asterisk in position 7 must not be used if this type of comment is in a line containing specifications.

OUTPUT-FORMAT SPECIFICATIONS

Filename (7-14)

Enter name of each output file. Names must be left justified.

AND/OR Relationships (14-16)

Enter AND for records in an AND relationship. Enter OR for records in an OR relationship (positions 14 and 15).

Type (15)

H = Heading line

D = Detail time output

T = Total time output

E = Unused

Stacker Select/Fetch Overflow (16)

Enter number of stacker to which cards are to be selected. Leave blank for single stacker devices. F is unused.

Space (17-18)

At least one entry is required in positions 17-22 if the line is to be printed.

Before: Enter 0, 1, 2, or 3 to specify 0, 1, 2, or 3 lines spacing before printing. Leave blank if no space before printing is required.

After: Enter 0, 1, 2, or 3 to specify 0, 1, 2, or 3 lines spacing after printing. Enter zero to specify no space after printing.

Indexed-Sequential ADD (16-18)

The entry ADD should be made in these positions on the specification line that identifies the type of output record (H, D, or T in position 15) if the output record is being added to an indexed-sequential file.

Skip (19-22)

Before: Enter any number from 01 through 12 to specify skipping before printing to channel 01 through 12 of the carriage control tape. Leave blank for no skip before printing.

After: Enter any number from 01 through 12 to specify skipping after printing 01 through 12 of the carriage control tape.

Output Indicators (23-31)

Enter up to three RPG indicators to identify files or to describe fields. Positions 23, 26, 29 must contain blank or the letter N.

Field Name (32-37)

Enter any field name defined in either input specifications or calculation specifications. Leave blank for constants specified in Constant or Edit Word. Use field name PAGE to cause automatic page numbering.

Edit Codes (38)

Enter Z if leading zeros of a numeric field are to be suppressed and the sign is to be stripped from the rightmost position. Leave blank if leading zeros are not suppressed, or if field specified in Field Name is alphameric or if line contains a constant or an edit word. Other entries are invalid.

Blank After (39)

Enter B to 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 position in output record to contain rightmost character of output field. To define a field within the key area for each data record in an unblocked indexed-sequential file (with key not part of the data), enter a K to the immediate left of the high-order digit. See the section *RPG User's Guide for ISAM Processing* for an explanation of key formats.

P=Packed/B=Binary (44)

Enter P if output data in packed decimal format. Leave blank if data in standard format. B is unused.

Constant or Edit Word (45-70)

Constant: Enter any required constant. Must be enclosed in apostrophe symbols. May consist of any of the 256 EBCDIC characters. Maximum length is 24 characters.

Apostrophe symbols required within constants must be represented as two consecutive apostrophe symbols.

Edit Word: Enter any edit word to specify editing with respect to punctuation, printing of dollar symbols, sign status, zero suppression, etc. Edit words must be enclosed in apostrophe symbols.

Sterling Sign Position (71-74)

Provided for programs processing sterling currency amounts. Leave blank for all other RPG programs.

Appendix A. Conversion Routine Operation Codes

The following list shows the relationship between conversion routine operation codes and how they are specified.

Code	Factor 1	Factor 2	Result Field	Specifications That May Follow This Entry
RPGCV	Reference Name	No entry	Label of the field which will contain the track address of the record.	KEYCV
EXTCV	Reference Name	Name or label of the user's routine	Label of the field which will contain the track address of the record.	KEYCV
KEYCV	None	None	Label of the field which will contain the key of the record to be located.	If RPGCV has been used, the conversion steps follow KEYCV. If EXTCV has been used, any other RPG calculations follow.
ERPGC	None	None	None	Any other calculations in the RPG program.

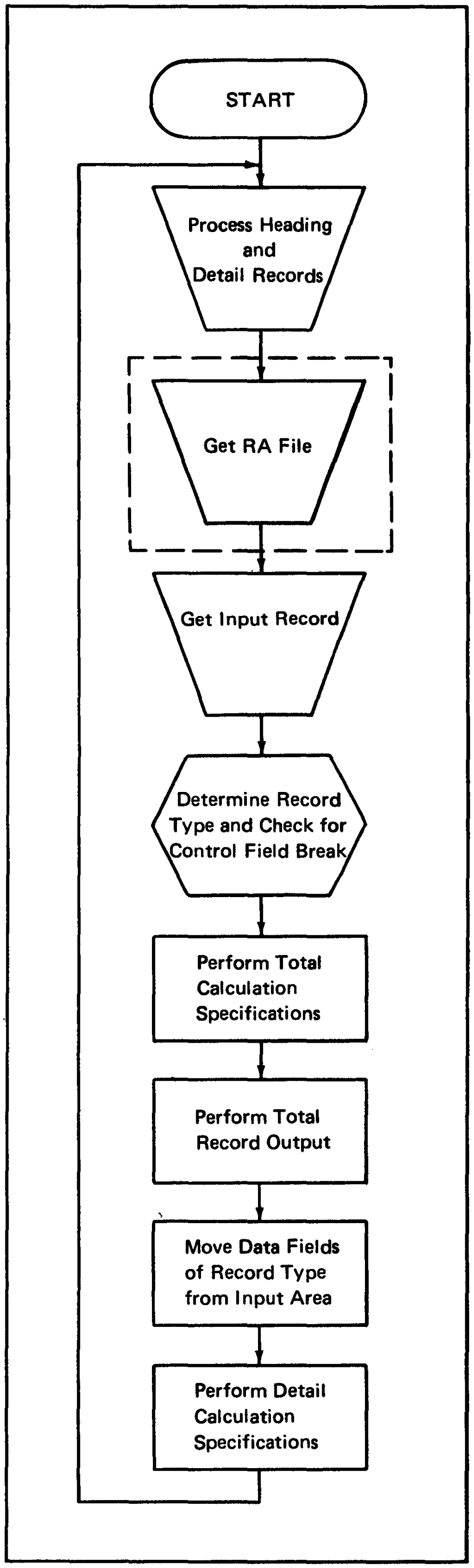

START
Process Heading and Detail Records
Get RA File
Get Input Record
Determine Record Type and Check for Control Field Break
Perform Total Calculation Specifications
Perform Total Record Output
Move Data Fields of Record Type from Input Area
Perform Detail Calculation Specifications

Chart 1. General Logic for RPG Object Module with RA File

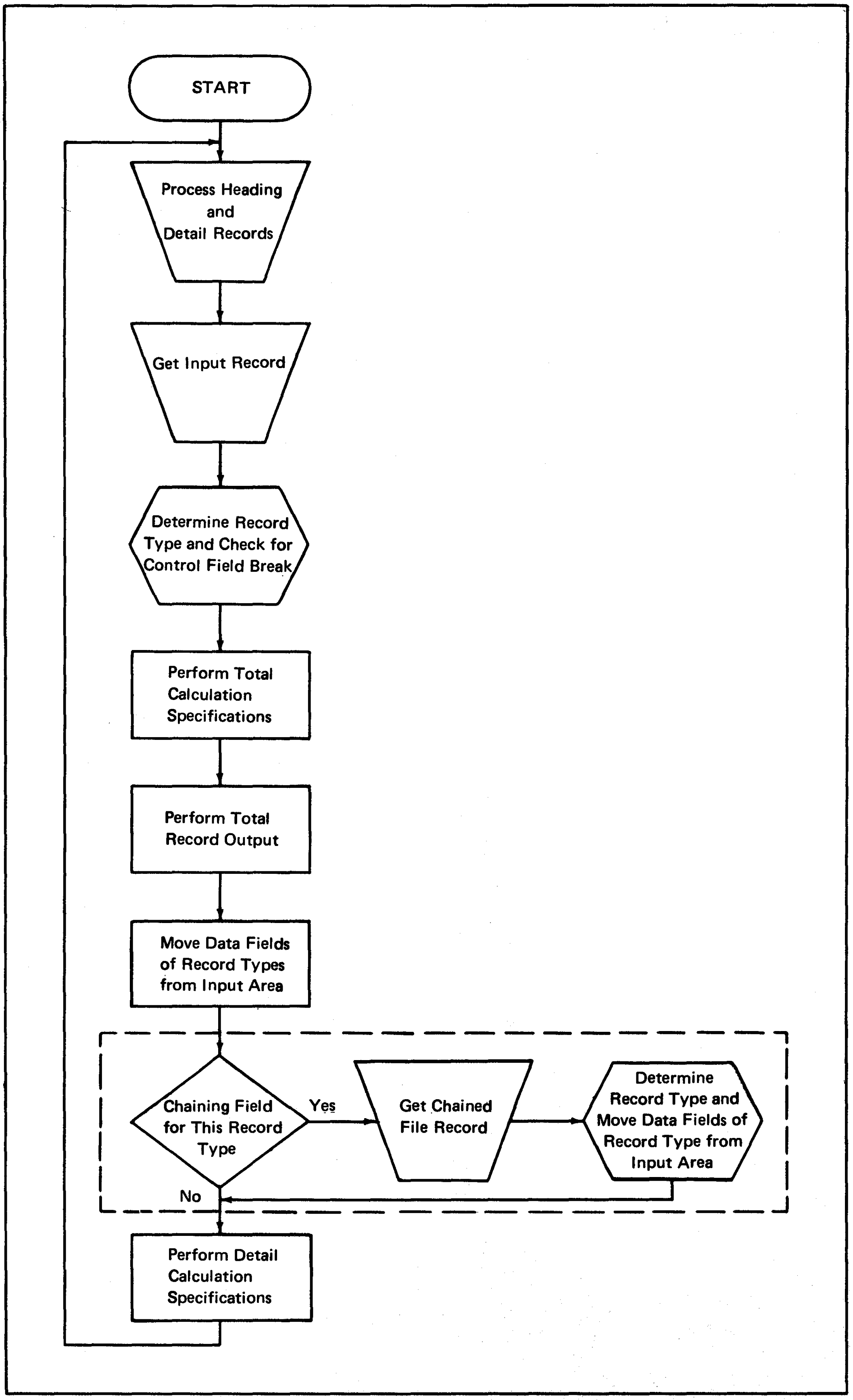

START
Process Heading and Detail Records
Get Input Record
Determine Record Type and Check for Control Field Break
Perform Total Calculation Specifications
Perform Total Record Output
Move Data Fields of Record Types from Input Area
Chaining Field for This Record Type
Yes
Get Chained File Record
Determine Record Type and Move Data Fields of Record Type from Input Area
No
Perform Detail Calculation Specifications

Chart 2. General Logic for RPG Object Module with a Chaining File

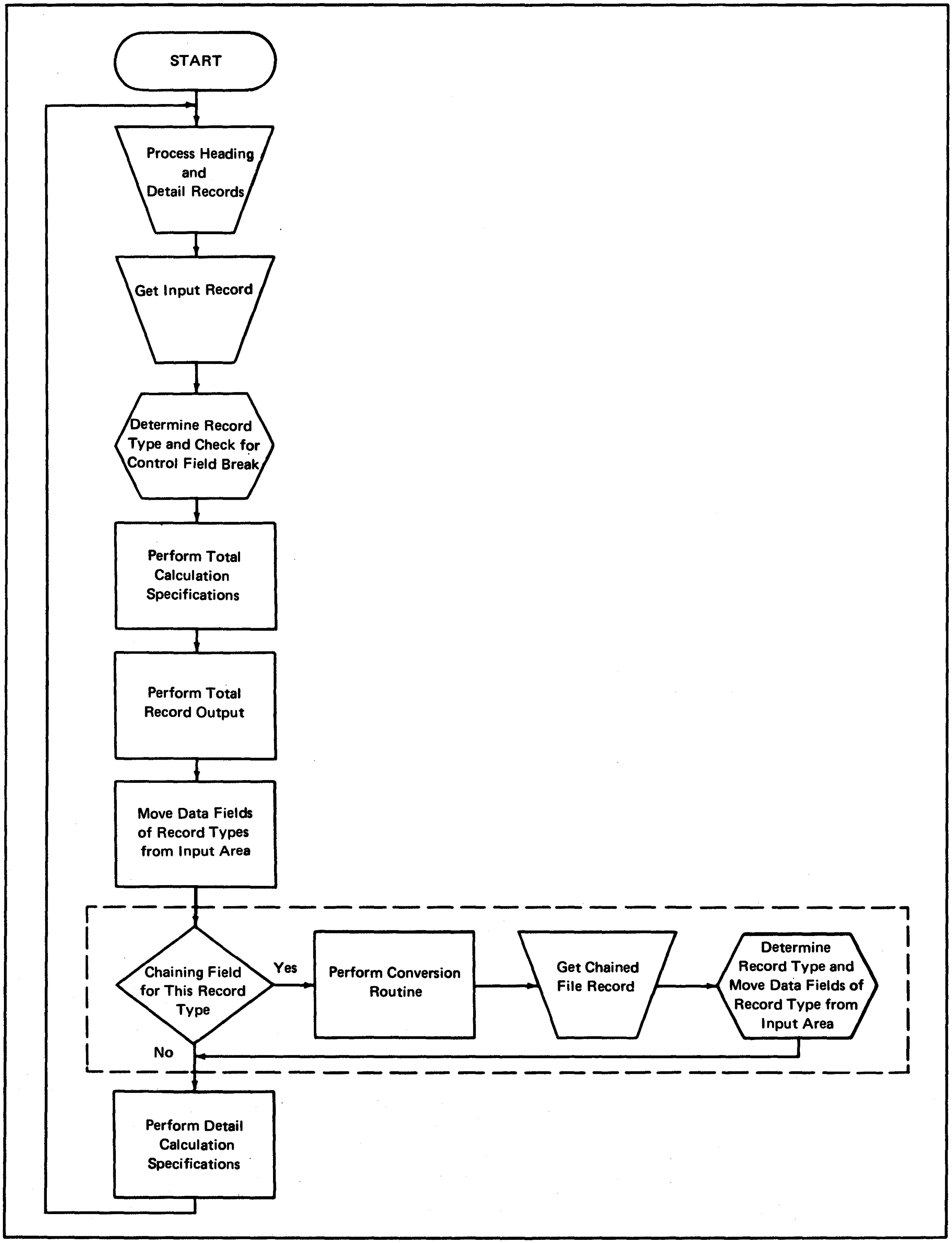

START
Process Heading and Detail Records
Get Input Record
Determine Record Type and Check for Control Field Break
Perform Total Calculation Specifications
Perform Total Record Output
Move Data Fields of Record Types from Input Area
Chaining Field for This Record Type
Yes
No
Perform Conversion Routine
Get Chained File Record
Determine Record Type and Move Data Fields of Record Type from Input Area
Perform Detail Calculation Specifications

Chart 3. General Logic for RPG Object Module with Chaining File Which Requires Conversion

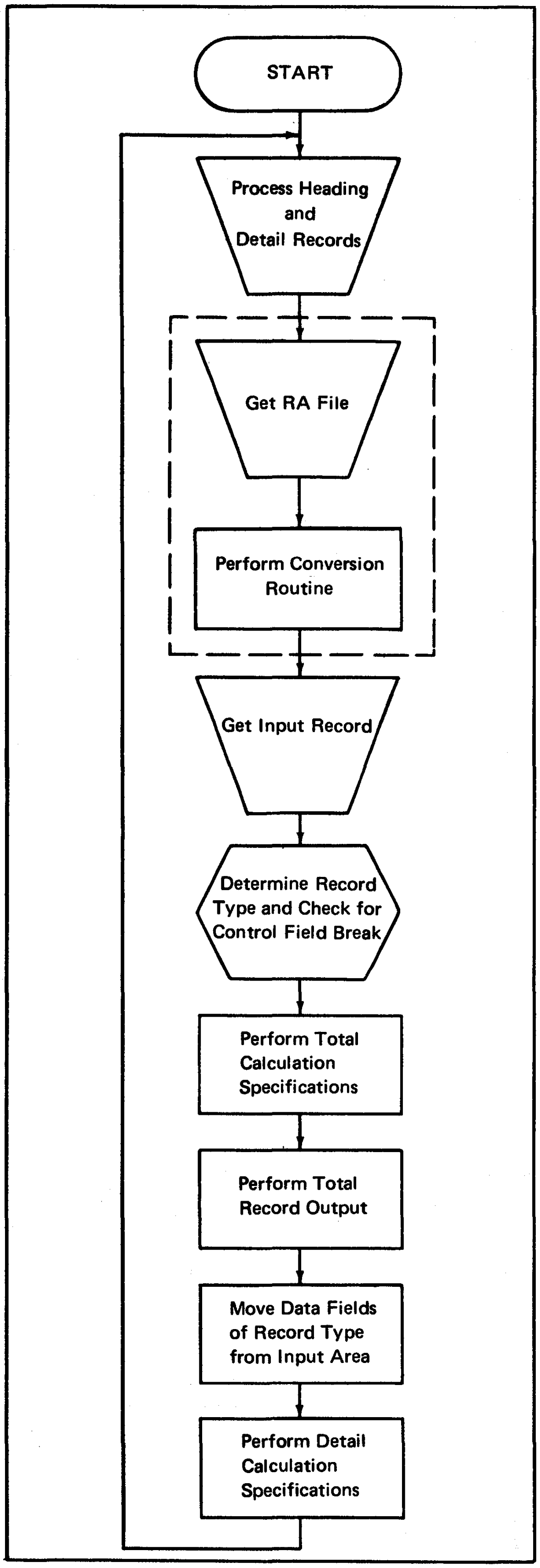

START
Process Heading and Detail Records
Get RA File
Perform Conversion Routine
Get Input Record
Determine Record Type and Check for Control Field Break
Perform Total Calculation Specifications
Perform Total Record Output
Move Data Fields of Record Type from Input Area
Perform Detail Calculation Specifications

Chart 4. General Logic for RPG Object Module with RA File which Requires Conversion

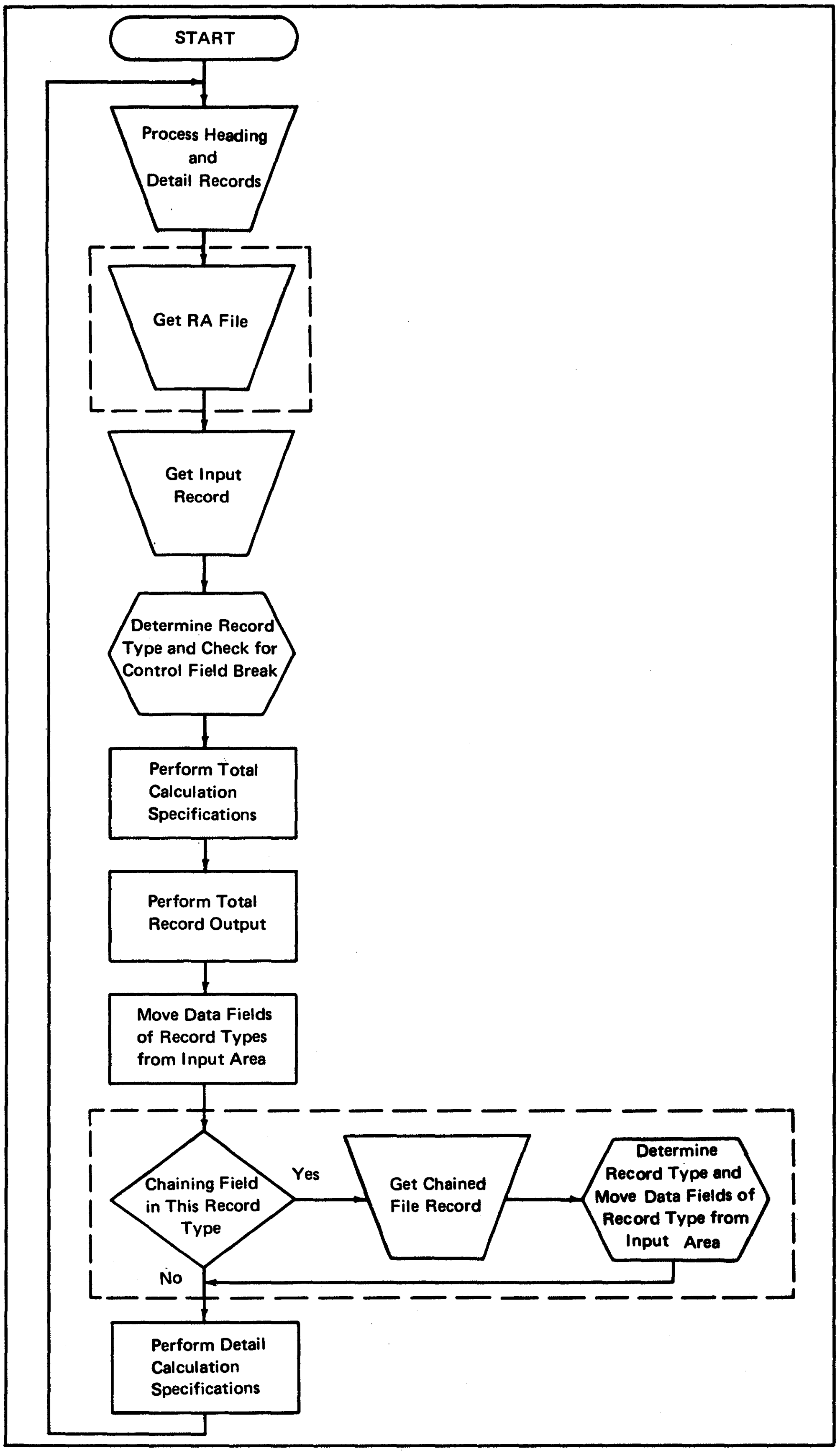

START
Process Heading and Detail Records
Get RA File
Get Input Record
Determine Record Type and Check for Control Field Break
Perform Total Calculation Specifications
Perform Total Record Output
Move Data Fields of Record Types from Input Area
Chaining Field in This Record Type
Yes
No
Get Chained File Record
Determine Record Type and Move Data Fields of Record Type from Input Area
Perform Detail Calculation Specifications

Chart 5. General Logic for RPG Object Module with RA File and Chaining Files

IES000I NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

IES001I FILE TYPE (COLUMN 15) IS INVALID. SPECIFICATION IS NOT PROCESSED.

IES002I INVALID ENTRY IN COLUMNS 28, 31, OR 32. SPECIFICATION IS NOT PROCESSED.

IES003I RECORD ADDRESS FIELD (COLUMNS 29-30) HAS INVALID LENGTH, IS MISSING OR IS NOT RIGHT-JUSTIFIED. ENTRY OF 03 IS ASSUMED.

IES004I MORE THAN ONE RECORD ADDRESS FILE IS PRESENT. SUCCEEDING ONES ARE NOT PROCESSED.

IES005I EXTENSION CODE (COLUMN 39) IS INVALID. ENTRY OF L IS ASSUMED FOR SEQUENTIAL, ENTRY OF BLANK IS ASSUMED FOR INDEXED-SEQUENTIAL.

IES006I INPUT FILE DESIGNATION (COLUMN 16) IS INVALID OR MISSING. ENTRY OF S IS ASSUMED.

IES007I OVERFLOW INDICATOR (COLUMN 33) IS NOT O. ENTRY OF O IS ASSUMED.

IES008I OVERFLOW INDICATOR (COLUMNS 33-34) IS INVALID. ENTRY OF OA IS ASSUMED.

IES009I MORE THAN ONE PRIMARY FILE IS SPECIFIED. FILE IS ASSUMED TO BE A SECONDARY FILE.

IES010I MODE OF PROCESSING (COLUMN 28) IS INVALID. ENTRY OF R IS ASSUMED.

IES011I FIXED FORMAT IS SPECIFIED, YET BLOCK LENGTH IS NOT A MULTIPLE OF RECORD LENGTH. BLOCK LENGTH IS INCREASED TO NEXT HIGHEST MULTIPLE.

IES012I TYPE OF FILE ORGANIZATION (COLUMN 32) IS NOT BLANK. ENTRY OF BLANK IS ASSUMED.

IES013I END-OF-FILE CODE (COLUMN 17) IS INVALID. ENTRY OF BLANK IS ASSUMED.

IES014I SEQUENCE (COLUMN 18) IS INVALID. ENTRY OF BLANK IS ASSUMED.

IES015I MODE OF PROCESSING (COLUMN 28) IS NOT BLANK. ENTRY OF BLANK IS ASSUMED.

IES016I RECORD ADDRESS TYPE (COLUMN 31) IS NOT BLANK. ENTRY OF BLANK IS ASSUMED.

IES017I EXTENSION CODE (COLUMN 39) IS INVALID. ENTRY OF E IS ASSUMED.

IES018I FILE FORMAT (COLUMN 19) IS INVALID. ENTRY OF F IS ASSUMED FOR INDEXED-SEQUENTIAL. OTHERWISE ENTRY OF V IS ASSUMED.

IES019I BLOCK LENGTH (COLUMNS 20-23) IS MISSING, INVALID, LESS THAN THE RECORD LENGTH, OR NOT RIGHT-JUSTIFIED. BLOCK LENGTH IS ASSUMED EQUAL TO RECORD LENGTH.

IES020I RECORD LENGTH (COLUMNS 24-27) IS MISSING, INVALID, OR NOT RIGHT-JUSTIFIED. RECORD LENGTH OF 0080 IS ASSUMED.

IES021I FILENAME (COLUMNS 7-14) IS MISSING, INVALID, OR NOT LEFT-JUSTIFIED. SPECIFICATION IS NOT PROCESSED.

IES022I OUTPUT FILE DESIGNATION (COLUMN 16) IS NOT BLANK. ENTRY OF BLANK IS ASSUMED.

IES023I PROGRAM EXCEEDS LIMIT OF TEN VALID FILE NAMES. ADDITIONAL FILE DESCRIPTION SPECIFICATIONS WILL BE TREATED AS COMMENTS.

IES024I KEY FIELD STARTING LOCATION (COLUMNS 35-38) IS INVALID, NOT RIGHT-JUSTIFIED, OR NOT LESS THAN RECORD LENGTH. ENTRY OF 0001 IS ASSUMED.

IES025I DEVICE (COLUMNS 40-46) IS INVALID. SPECIFICATION IS NOT PROCESSED.

53194.1

```
IES026I   NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

IES027I   'LABELS' (COLUMN 53) IS INVALID.  ENTRY OF S IS ASSUMED.

IES028I   NAME OF LABEL EXIT (COLUMNS 54-59) IS MISSING, INVALID, OR NOT LEFT-JUSTIFIED.
           SPECIFICATION IS NOT PROCESSED.

IES029I   NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

IES030I   MORE THAN ONE RECORD ADDRESS FILE IS SPECIFIED ON FILE EXTENSION SPECIFICATION.
           SPECIFICATION IS NOT PROCESSED.

IES031I   'FROM FILENAME' (COLUMNS 11-18) IS NOT SPECIFIED AS ON FILE DESCRIPTION
           SPECIFICATION.  SPECIFICATION IS NOT PROCESSED.

IES032I   EXTENSION CODE (COLUMN 39) IS NOT E.  SPECIFICATION IS NOT PROCESSED.

IES033I   LENGTH OF TABLE ENTRY (COLUMNS 40-42 OR 52-54) EXCEEDS 256 CHARACTERS FOR AN
           ALPHAMERIC FIELD.  ENTRY OF 256 IS ASSUMED.

IES034I   CHAINING FIELD (COLUMNS 9-10) IS MISSING, INVALID, OR NOT RIGHT-JUSTIFIED.
           SPECIFICATION IS NOT PROCESSED.

IES035I   NUMBER OF CONVERSION ROUTINES AND TABLE NAMES EXCEEDS ALLOCATED CORE STORAGE.
           ADDITIONAL SPECIFICATIONS CONTAINING CONVERSION ROUTINES OR TABLE NAMES WILL
           NOT BE PROCESSED.

IES036I   'TO FILENAME' (COLUMNS 19-26) IS NOT SPECIFIED AS ON CORRESPONDING FILE
           DESCRIPTION SPECIFICATION, OR FILE DESCRIPTION SPECIFICATION WAS NOT PROCESSED.
            SPECIFICATION IS NOT PROCESSED.

IES037I   'TO FILENAME' (COLUMNS 19-26) IS NOT SPECIFIED AS A CHAINED FILE ON FILE
           DESCRIPTION SPECIFICATION.  SPECIFICATION IS NOT PROCESSED.

IES038I   LENGTH OF TABLE ENTRY (COLUMNS 40-42 OR 52-54) EXCEEDS 15 DIGITS FOR A NUMERIC
           FIELD.  ENTRY OF 15 IS ASSUMED.

IES039I   CONVERSION ROUTINE (COLUMNS 27-32) IS MISSING, INVALID, OR NOT LEFT-JUSTIFIED.
           SPECIFICATION IS NOT PROCESSED.

IES040I   'TO FILENAME' (COLUMNS 19-26) IS NOT SPECIFIED AS A PRIMARY OR SECONDARY FILE OR
           'MODE OF PROCESSING' IS NOT BETWEEN LIMITS OR RANDOM.  SPECIFICATION IS NOT
           PROCESSED.

IES041I   TABLE SEQUENCE (COLUMNS 45 OR 57) IS INVALID.  ENTRY OF BLANK IS ASSUMED.

IES042I   TABLE NAME (COLUMNS 27-32 OR 46-51) IS MULTI-DEFINED.  SPECIFICATION IS NOT
           PROCESSED.

IES043I   'TO FILENAME' (COLUMNS 19-26) IS NOT SPECIFIED AS ON FILE DESCRIPTION
           SPECIFICATION.  ENTRY OF BLANKS IS ASSUMED.

IES044I   'TO FILENAME' (COLUMNS 19-26) IS NOT SPECIFIED AS AN OUTPUT FILE ON FILE
           DESCRIPTION SPECIFICATION.  ENTRY OF BLANKS IS ASSUMED.

IES045I   TABLE NAME (COLUMNS 27-32 OR 46-51) IS MISSING OR NOT LEFT-JUSTIFIED.
           SPECIFICATION IS NOT PROCESSED.

IES046I   LETTERS 'TAB' ARE MISSING IN TABLE NAME (COLUMNS 27-29 OR 46-48).  ENTRY OF
           'TAB' IS ASSUMED.

IES047I   NUMBER OF TABLE ENTRIES PER RECORD (COLUMNS 33-35) IS MISSING, INVALID OR NOT
           RIGHT-JUSTIFIED.  ENTRY OF 008 IS ASSUMED.

IES048I   NUMBER OF TABLE ENTRIES PER TABLE (COLUMNS 36-39) IS MISSING, INVALID OR NOT
           RIGHT-JUSTIFIED.  ENTRY OF 0150 IS ASSUMED.

IES049I   'LENGTH OF TABLE' ENTRY (COLUMNS 40-42 OR 52-54) IS MISSING, INVALID, OR NOT
           RIGHT-JUSTIFIED.  ENTRY OF 010 IS ASSUMED.
```

53194.2

IES050I 'PACKED' (COLUMN 43 OR 55) IS INVALID. ENTRY OF BLANK IS ASSUMED.

IES051I 'DECIMAL POSITIONS' (COLUMN 44 OR 56) IS INVALID. ENTRY OF ZERO IS ASSUMED.

IES052I RECORD SEQUENCE OF THE CHAINING FILE (COLUMNS 7-8) IS INVALID. BOTH POSITIONS MUST BE EITHER NUMERIC OR ALPHABETIC. SPECIFICATION IS NOT PROCESSED.

IES053I NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

IES054I MAXIMUM NUMBER OF FILE EXTENSION SPECIFICATIONS CONTAINING TABLES HAS BEEN EXCEEDED. ADDITIONAL TABLE FILE EXTENSION SPECIFICATIONS WILL BE TREATED AS COMMENTS.

IES055I NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

IES056I INVALID ENTRY FOR INDEXED-SEQUENTIAL IN COLUMN 66. SPECIFICATION IS NOT PROCESSED FOR OUTPUT FILE. ENTRY OF BLANK ASSUMED FOR INPUT/UPDATE FILE.

IES057I KEY LENGTH INVALID OR MISSING. SPECIFICATION NOT PROCESSED FOR LOAD OR ADD FILE, OTHERWISE LENGTH OF 8 IS ASSUMED.

IES058I INVALID ENTRY IN ONE OR MORE OF COLUMNS 66-70. ENTRY IS IGNORED.

IES059I KEY LENGTH GREATER THAN RECORD LENGTH. SPECIFICATION IS NOT PROCESSED.

IES060I INVALID ENTRY FOR INDEXED-SEQUENTIAL IN COLUMN 67. ZERO IS ASSUMED FOR LOAD OR ADD.

IES061I WARNING - MULTI-FILE PROGRAM (WITH PRIMARY AND SECONDARY FILES) IS SPECIFIED WITHOUT MATCHING FIELDS FOR THE PRIMARY FILE.

IES062I WARNING - MULTI-FILE PROGRAM (WITH PRIMARY AND SECONDARY FILES) IS SPECIFIED WITHOUT MATCHING FIELDS FOR THE SECONDARY FILE(S).

IES063I THE SUM OF THE LENGTHS OF THE MATCHING FIELDS FOR THE PRIMARY FILE DOES NOT EQUAL THAT OF EACH SECONDARY FILE. EXECUTION IS DELETED.

IES064I THE SUM OF THE LENGTHS OF THE MATCHING FIELDS IS NOT CONSTANT IN EACH RECORD WHICH SPECIFIED MATCHING FIELDS FOR A FILE. EXECUTION IS DELETED.

IES065I WARNING- PAGE/LINE ENTRY IS OUT OF SEQUENCE.

IES066I IMPROPER USE OF A USER DATE RESERVED WORD.SPECIFICATION IS NOT PROCESSED.

IES067I USER DATE FORMAT ENTRY IS INVALID. ENTRYOF BLANK IS ASSUMED.

IES068I NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

IES069I NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

IES070I NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

IES071I DETAIL CALCULATION SPECIFICATION FOLLOWS A TOTAL CALCULATION SPECIFICATION. DETAIL SPECIFICATION IS NOT PROCESSED.

IES072I UNDEFINED TABLE SPECIFIED IN LOKUP OPERATION. SPECIFICATION IS NOT PROCESSED.

IES073I KEYCV IS VALID ONLY WHEN PRECEDED BY RPGCV OR EXTCV. SPECIFICATION IS NOT PROCESSED.

IES074I NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

IES075I NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

IES076I NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

IES077I THERE ARE NO VALID INPUT SPECIFICATIONS IN THIS PROGRAM. EXECUTION IS DELETED.

53194.3

IES078I DECIMAL POSITION IS INVALID. ENTRY OF ZERO IS ASSUMED FOR NUMERIC FIELD. ENTRY OF BLANK IS ASSUMED FOR ALPHAMERIC FIELD.

IES079I CONVERSION NAME CANNOT BE USED TO DEFINE A FIELD. SPECIFICATION IS NOT PROCESSED.

IES080I FIELD INDICATOR IS SPECIFIED BUT IS NOT VALID. INDICATOR IS NOT PROCESSED.

IES081I NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

IES082I NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

IES083I FIELD LENGTH IS IMPROPERLY SPECIFIED OR IS NOT SPECIFIED. ENTRY OF ZERO IS ASSUMED FOR INVALID CHARACTER. WHEN REQUIRED LENGTH IS NOT SPECIFIED, ENTRY OF 3 IS ASSUMED FOR EXTCV AND RPGCV. ENTRY OF 4 IS ASSUMED FOR ALL OTHER OPERATION CODES.

IES084I NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

IES085I RESULT FIELD LENGTH (COLUMNS 49-51) IS GREATER THAN ALLOWED. A LENGTH OF 256 IS ASSUMED FOR AN ALPHAMERIC FIELD. A LENGTH OF 15 IS ASSUMED FOR A NUMERIC FIELD.

IES086I OPERATION CODE (COLUMNS 28-32) IS INVALID OR MISSING. SPECIFICATION IS NOT PROCESSED.

IES087I REQUIRED ENTRY IN FACTOR 1 (COLUMNS 18-27) IS MISSING OR INVALID. SPECIFICATION IS NOT PROCESSED.

IES088I REQUIRED ENTRY IN FACTOR 2 (COLUMNS 33-42) IS MISSING OR INVALID. SPECIFICATION IS NOT PROCESSED.

IES089I REQUIRED ENTRY IN RESULT FIELD (COLUMNS 43-48) IS MISSING OR INVALID. SPECIFICATION IS NOT PROCESSED.

IES090I FORM TYPE (COLUMN 6) IS INVALID. SPECIFICATION IS NOT PROCESSED.

IES091I 'NOT' (COLUMNS 9, 12, OR 15) IS NOT N OR BLANK. ENTRY OF N IS ASSUMED.

IES092I CONTROL LEVEL IS IMPROPERLY SPECIFIED. ENTRY OF L0 IS ASSUMED.

IES093I RESULTING INDICATOR IS INVALID. INDICATOR IS NOT PROCESSED.

IES094I INDICATOR L0 OR U1-U8 IS SPECIFIED AS A FIELD INDICATOR, BUT IS NOT ALLOWED. INDICATOR IS IGNORED.

IES095I FIELD-RECORD RELATION (COLUMNS 63-64) IS INVALID. ENTRY OF 99 IS ASSUMED.

IES096I 'HALF ADJUST' ENTRY (COLUMN 53) IS INVALID. ENTRY OF H IS ASSUMED.

IES097I FIELD NAME IS IMPROPERLY USED. SPECIFICATION IS NOT PROCESSED.

IES098I INDICATOR (COLUMNS 10-11, 13-14, OR 16-17) IS INVALID. ENTRY OF 00 IS ASSUMED.

IES099I REQUIRED RESULTING INDICATOR (COLUMNS 54-55, 56-57, OR 58-59) IS NOT SPECIFIED. SPECIFICATION IS NOT PROCESSED.

IES100I 'MVR' DOES NOT FOLLOW 'DIV', OR FOLLOWS A 'DIV' WITH HALF ADJUST SPECIFIED. SPECIFICATION IS NOT PROCESSED.

IES101I 'FROM' (COLUMNS 44-47), 'TO' (COLUMNS 48-51) OR RECORD IDENTIFICATION POSITION (COLUMNS 21-24, 28-31, OR 35-38) IS ZERO. ENTRY OF 1 IS ASSUMED.

IES102I RESULT FIELD OF RPGCV OR EXTCV DOES NOT HAVE LENGTH OF 3. ENTRY OF 3 IS ASSUMED.

53194.4

IES103I IF IBM SHILLING IS SPECIFIED, STERLING INPUT FIELD MUST HAVE MORE THAN THREE NON-DECIMAL POSITIONS. IF BSI SHILLING IS SPECIFIED, STERLING INPUT FIELD MUST HAVE MORE THAN TWO NON-DECIMAL POSITIONS. FIELD IS INITIALIZED TO ZERO AND NO INPUT IS ACCEPTED.

IES104I WARNING - INDICATOR 00 SHOULD BE USED ONLY IN OUTPUT SPECIFICATIONS.

IES105I FIELDS USED IN AN ALPHAMERIC COMPARE MUST BE EQUAL IN LENGTH OR MUST BE LESS THAN OR EQUAL TO 200 BYTES.

IES106I FIELD LENGTHS ARE INVALID FOR THIS OPERATION. SPECIFICATION IS NOT PROCESSED.

IES107I PLUS AND/OR MINUS RESULTING INDICATORS (COLUMNS 54-55 OR 56-57) ARE NOT ALLOWED FOR TESTING ALPHAMERIC FIELDS. INDICATORS ARE IGNORED.

IES108I FIELD TYPE IS INVALID FOR THIS OPERATION. SPECIFICATION IS NOT PROCESSED.

IES109I NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

IES110I FORM TYPE (COLUMN 6) DOES NOT CONTAIN I, C, OR O, AND COLUMN 7 IS NOT AN ASTERISK. SPECIFICATION IS NOT PROCESSED.

IES111I UNDEFINED FILENAME. SPECIFICATION IS NOT PROCESSED.

IES112I FILENAME HAS BEEN PREVIOUSLY REFERENCED OR DEFINED AS A TABLE OR OUTPUT FILE TYPE. SPECIFICATION IS NOT PROCESSED.

IES113I 'AND' SPECIFICATION (COLUMNS 14-16) IS FIRST INPUT SPECIFICATION OR FOLLOWS FIELD DESCRIPTION SPECIFICATION, INVALID RECORD IDENTIFICATION, OR INVALID FILE NAME. SPECIFICATION IS NOT PROCESSED.

IES114I THERE ARE NO RECORD IDENTIFICATION CODES (COLUMNS 21-41) IN THE CARD BEFORE AN 'AND' CARD. SPECIFICATION IS NOT PROCESSED.

IES115I 'OR' SPECIFICATION (COLUMNS 14-15) IS FIRST INPUT SPECIFICATION OR FOLLOWS FIELD DESCRIPTION SPECIFICATION, INVALID RECORD IDENTIFICATION, OR INVALID FILE NAME. SPECIFICATION IS NOT PROCESSED.

IES116I RECORD IDENTIFICATION SPECIFICATION FOLLOWS INVALID FILE TYPE SPECIFICATION. SPECIFICATION IS NOT PROCESSED.

IES117I FIELD NAME CONTAINS EMBEDDED BLANK. SPECIFICATION IS NOT PROCESSED.

IES118I FILE AND FIELD NAMES ARE BOTH PRESENT ON THE SAME SPECIFICATION. FILENAME IS ASSUMED.

IES119I SEQUENCE (COLUMNS 15-16) IS BLANK. ENTRY OF AA IS ASSUMED.

IES120I ALPHAMERIC SEQUENCE FOUND AFTER NUMERIC SEQUENCE. NUMERIC SEQUENCE 1 GREATER THAN PREVIOUS NUMERIC SEQUENCE IS ASSUMED.

IES121I NUMERIC SEQUENCE (COLUMNS 15-16) IS INVALID. ENTRIES MUST BEGIN WITH 01 AND BE CONSECUTIVE IN ASCENDING ORDER. NUMERIC SEQUENCE 1 GREATER THAN PREVIOUS VALID NUMERIC SEQUENCE IS ASSUMED.

IES122I NUMBER (COLUMN 17) IS NOT N OR 1, AND NUMERIC SEQUENCE IS FOUND. ENTRY OF N IS ASSUMED.

IES123I OPTION (COLUMN 18) IS NOT O OR BLANK. ENTRY OF O IS ASSUMED.

IES124I RESULTING INDICATOR (COLUMNS 19-20) IS BLANK OR INVALID. INDICATOR OF 99 IS ASSUMED.

IES125I STACKER SELECT (COLUMN 42) IS NOT BLANK OR NUMERIC. ENTRY OF BLANK IS ASSUMED.

IES126I 'POSITION' (COLUMNS 21-24, 28-31, OR 35-38) CONTAINS EMBEDDED BLANK. ENTRY OF ZERO IS ASSUMED.

IES127I 'POSITION' (COLUMNS 21-24, 28-31, OR 35-38) CONTAINS NON-NUMERIC CHARACTER. ENTRY OF ZERO IS ASSUMED.

53194.5

IES128I 'NOT' (COLUMNS 25, 32, OR 39) IS NOT BLANK OR N. ENTRY OF N IS ASSUMED.

IES129I 'C/Z/D' (COLUMNS 26, 33, OR 40) IS NOT C, Z, OR D. ENTRY OF C IS ASSUMED.

IES130I FIELD DESCRIPTION SPECIFICATION IS FIRST INPUT SPECIFICATION OR FOLLOWS AN INVALID RECORD IDENTIFICATION OR FILE NAME. SPECIFICATION IS NOT PROCESSED.

IES131I FIELD NAME (COLUMNS 53-58) IS MISSING OR NOT LEFT-JUSTIFIED. SPECIFICATION IS NOT PROCESSED.

IES132I FIELD NAME (COLUMNS 53-58) BEGINS WITH A NUMERIC CHARACTER. SPECIFICATION IS NOT PROCESSED.

IES133I 'FROM' (COLUMNS 44-47) OR 'TO' (COLUMNS 48-51) IS BLANK. ENTRY OF 1 IS ASSUMED.

IES134I 'FROM' (COLUMNS 44-47) OR 'TO' (COLUMNS 48-51) CONTAINS EMBEDDED BLANK. ENTRY OF ZERO IS ASSUMED FOR EACH BLANK.

IES135I 'FROM' (COLUMNS 44-47) OR 'TO' (COLUMNS 48-51) CONTAINS NON-NUMERIC CHARACTER. ENTRY OF ZERO IS ASSUMED.

IES136I 'FROM' (COLUMNS 44-47) SHOULD BE LESS THAN OR EQUAL TO 'TO' (COLUMNS 48-51). FIELD LENGTH OF 1 IS ASSUMED. 'TO' IS ASSUMED EQUAL TO 'FROM'.

IES137I DECIMAL POSITION (COLUMN 52) IS NOT NUMERIC. ENTRY OF ZERO IS ASSUMED.

IES138I EITHER AN UNPACKED NUMERIC FIELD IS MORE THAN 15 BYTES LONG, OR A PACKED NUMERIC FIELD IS GREATER THAN 8 BYTES LONG. LENGTH OF 15 IS ASSUMED FOR UNPACKED NUMERIC FIELD. LENGTH OF 8 IS ASSUMED FOR PACKED NUMERIC FIELD.

IES139I STERLING FIELD IS INDICATED WITH MORE THAN THREE DECIMAL POSITIONS. THE DECIMAL PORTION OF THE FIELD IS TRUNCATED TO THREE POSITIONS. THE 'TO' POSITION OF THE FIELD IS ALTERED TO ALLOW FOR THIS TRUNCATION.

IES140I PACKED INDICATOR (COLUMN 43) IS NEITHER BLANK NOR P. ENTRY OF P IS ASSUMED.

IES141I CONTROL LEVEL DOES NOT START WITH L IN COLUMN 59 (L IS ASSUMED), OR COLUMN 60 IS NOT NUMERIC (1 IS ASSUMED).

IES142I MATCHING/CHAINING FIELD (COLUMN 61) IS NOT C OR M (M IS ASSUMED), OR COLUMN 62 IS NOT NUMERIC (1 IS ASSUMED).

IES143I MATCHING VALUE (COLUMN 62) IS INVALID. ENTRY OF 3 IS ASSUMED IF COLUMN IS NUMERIC. ENTRY OF 1 IS ASSUMED IF COLUMN IS NON-NUMERIC.

IES144I ALPHAMERIC FIELD LENGTH IS GREATER THAN 256. LENGTH OF 256 IS ASSUMED.

IES145I NUMERIC RESULTING INDICATOR (COLUMNS 65-66 OR 67-68) IS SPECIFIED FOR AN ALPHAMERIC FIELD. INDICATOR IS SET OFF.

IES146I STERLING SIGN POSITION (COLUMNS 71-74) IS INVALID. NORMAL POSITION FOR SIGN IS ASSUMED.

IES147I STERLING SPECIFIED IN COLUMNS 71-74, BUT NOT SPECIFIED ON PROCESSOR CONTROL CARD. ENTRY OF BLANKS IS ASSUMED.

IES148I THE CALCULATED LENGTH OF THE STERLING FIELD IS GREATER THAN 15. LENGTH OF 15 IS ASSUMED.

IES149I FIELD EXTENDS BEYOND RECORD LENGTH. FIELD IS ASSUMED TO END IN LAST CHARACTER POSITION OF RECORD.

IES150I COLUMNS 16-18 OF 'OR' TYPE INPUT SPECIFICATION MUST BE BLANK. ENTRY OF BLANK IS ASSUMED.

IES151I COLUMNS 17-20 AND 42 OF 'AND' TYPE INPUT SPECIFICATION MUST BE BLANK. ENTRY OF BLANK IS ASSUMED.

IES152I KEY 'TO' POSITION (COLUMNS 48-51) OUTSIDE KEY AREA. FIELD TRUNCATED.

IES153I KEY AREA FIELD CANNOT BE DEFINED FOR THIS INDEXED-SEQUENTIAL FILE (COLUMNS 44-51). 'K' ENTRY(S) IGNORED.

IES154I 'K' MUST PRECEDE NUMERIC VALUES IN BOTH 'FROM' AND 'TO' FIELD ENTRIES (COLUMNS 41-55). SPECIFICATION IS NOT PROCESSED.

IES155I ENTRY IN COLUMNS 16-18 VALID FOR INDEXED-SEQUENTIAL ADD ONLY. ENTRY IS IGNORED.

IES156I REQUIRED 'ADD' ENTRY (COLUMNS 16-18) MISSING. EXECUTION IS DELETED.

IES157I U INDICATOR IS IMPROPERLY SPECIFIED (COLUMNS 63-64) WITH OTHER INDICATORS (COLUMNS 59-62). THE INDICATOR IS IGNORED.

IES158I NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

IES159I INAPPROPRIATE OUTPUT FIELD. SPECIFICATION IS NOT PROCESSED.

IES160I FILENAME (COLUMNS 7-14) IS MISSING, OR RECORD TYPE (COLUMN 15) IS IN WRONG ORDER. SPECIFICATION IS NOT PROCESSED.

IES161I CORRESPONDING FILENAME (COLUMNS 7-14) CANNOT BE DETERMINED. SPECIFICATION IS NOT PROCESSED.

IES162I 'STACKER SELECT' (COLUMN 16) IS INVALID. ENTRY OF BLANK IS ASSUMED.

IES163I 'SPACE BEFORE' (COLUMN 17) IS INVALID. ENTRY OF 1 IS ASSUMED.

IES164I 'SPACE AFTER' (COLUMN 18) IS INVALID. ENTRY OF 1 IS ASSUMED.

IES165I 'SKIP BEFORE' (COLUMNS 19-20) IS INVALID. ENTRY OF 01 IS ASSUMED.

IES166I 'SKIP AFTER' (COLUMNS 21-22) IS INVALID. ENTRY OF 01 IS ASSUMED.

IES167I RECORD TYPE (COLUMN 15) IS NOT H, D, OR T. SPECIFICATION IS NOT PROCESSED.

IES168I COLUMNS 17-22 MUST BE BLANK FOR 'AND' TYPE OUTPUT SPECIFICATIONS. ENTRY OF BLANK IS ASSUMED.

IES169I COLUMNS 7-13 MUST BE BLANK FOR 'AND' OR 'OR' TYPE OUTPUT SPECIFICATIONS. ENTRY OF BLANK IS ASSUMED.

IES170I CORRESPONDING RECORD IDENTIFICATION SPECIFICATION IS MISSING OR INVALID. SPECIFICATION IS NOT PROCESSED.

IES171I 'ZERO SUPPRESS' (COLUMN 38) IS INVALID. ENTRY OF BLANK IS ASSUMED.

IES172I 'PACKED FIELD' (COLUMN 44) IS INVALID. ENTRY OF BLANK IS ASSUMED.

IES173I FIELD NAME (COLUMNS 32-37) IS NOT LEFT-JUSTIFIED. SPECIFICATION IS NOT PROCESSED.

IES174I 'END POSITION' (COLUMNS 40-43) IS INVALID OR MISSING. SPECIFICATION IS NOT PROCESSED.

IES175I LEADING OR CLOSING APOSTROPHE (') IN EDIT WORD IS NOT CORRECT. ENTRY OF BLANKS IN COLUMNS 45-70 IS ASSUMED.

IES176I 'BLANK AFTER' (COLUMN 39) IS INVALID. ENTRY OF BLANK IS ASSUMED.

IES177I PUNCH AND PRINT FUNCTIONS ARE SPECIFIED FOR THE SAME FILE. ENTRY OF BLANKS IS ASSUMED FOR COLUMNS 17-22.

IES178I ZERO SUPPRESSION (COLUMN 38) MAY NOT BE SPECIFIED FOR CONSTANTS OR EDIT WORDS. ENTRY OF BLANK IN COLUMN 38 IS ASSUMED.

IES179I FIELD NAME (COLUMNS 32-37) IS UNDEFINED. SPECIFICATION IS NOT PROCESSED.

IES180I WARNING - 'BLANK AFTER' (COLUMN 39) IS SPECIFIED FOR CONSTANT. ALL IDENTICAL CONSTANTS WILL BE BLANKED.

53194.7

IES181I CONSTANT (COLUMNS 45-70) IS NOT LEFT-JUSTIFIED. SPECIFICATION IS NOT PROCESSED.

IES182I EDIT WORD (COLUMNS 45-70) IS NOT LEFT-JUSTIFIED. ENTRY OF BLANKS IN COLUMNS 45-70 IS ASSUMED.

IES183I 'PACKED FIELD' (COLUMN 44) MAY NOT BE SPECIFIED WITH CONSTANT, EDIT WORD OR STERLING ENTRY. ENTRY OF BLANK IN COLUMN 44 IS ASSUMED.

IES184I FILENAME (COLUMNS 7-14) IS NOT LEFT-JUSTIFIED. SPECIFICATION IS NOT PROCESSED.

IES185I FILENAME (COLUMNS 7-14) CONTAINS NON-ALPHABETIC CHARACTER IN FIRST POSITION. SPECIFICATION IS NOT PROCESSED.

IES186I EDIT WORD (COLUMNS 45-70) CONTAINS NO DIGIT POSITIONS OR MORE THAN FIFTEEN (SIXTEEN FOR STERLING). ENTRY OF BLANKS IN COLUMNS 45-70 IS ASSUMED.

IES187I LEADING OR CLOSING APOSTROPHE (') IN CONSTANT IS NOT CORRECT. SPECIFICATION IS NOT PROCESSED.

IES188I 'AND' OR 'OR' FOLLOWING A FIELD NAME SPECIFICATION OR AS FIRST OUTPUT SPECIFICATION IS INVALID. SPECIFICATION IS NOT PROCESSED.

IES189I FIELD NAME (COLUMNS 32-38) CONTAINS NON-ALPHABETIC CHARACTER IN FIRST POSITION. SPECIFICATION IS NOT PROCESSED.

IES190I STERLING ENTRY (COLUMNS 71-74) MAY NOT BE SPECIFIED WITH CONSTANT OR PAGE(N). ENTRY OF BLANK IN COLUMNS 71-74 IS ASSUMED.

IES191I STERLING ENTRY (COLUMNS 71-74) IS INVALID. ENTRY OF BLANKS IS ASSUMED.

IES192I OUTPUT INDICATOR (COLUMNS 24-25, 27-28, OR 30-31) IS INVALID OR UNDEFINED. ENTRY OF LO IS ASSUMED.

IES193I OUTPUT INDICATORS SHOULD START IN COLUMNS 23-25, THEN 26-28, AND FINALLY 29-31. ENTRY IS SHIFTED LEFT.

IES194I 'NOT' (COLUMNS 23, 26, OR 29) IS NOT BLANK OR N. ENTRY OF N IS ASSUMED.

IES195I WARNING - OVERFLOW INDICATOR IS SPECIFIED IN 'AND' TYPE SPECIFICATION. RECORD WILL NOT BE PUT OUT AS OVERFLOW LINE.

IES196I DECIMAL POSITIONS MUST BE ZERO FOR PAGE(N) FIELD. ENTRY OF ZERO IS ASSUMED.

IES197I SPECIFICATION TYPE CANNOT BE DETERMINED. RECORD AND FIELD DEFINITIONS ARE SPECIFIED IN SAME LINE OR BOTH ARE BLANK. SPECIFICATION IS NOT PROCESSED.

IES198I FORM TYPE (COLUMN 6) IS INVALID (NOT O). SPECIFICATION IS NOT PROCESSED.

IES199I NO OUTPUT INDICATOR (COLUMNS 24-25, 27-28, OR 30-31) IS SPECIFIED FOR 'AND' OR 'OR' TYPE SPECIFICATION. SPECIFICATION IS NOT PROCESSED.

IES200I FORM TYPE (COLUMN 6) IS INVALID OR OUT OF SEQUENCE. SPECIFICATION IS NOT PROCESSED.

IES201I FILE DESCRIPTION SPECIFICATIONS ARE MISSING. COMPILATION IS ABORTED.

IES202I WARNING - LINE COUNTER SPECIFICATION IS MISSING.

IES203I PRIMARY FILE IS NOT SPECIFIED. EXECUTION IS DELETED.

IES204I WARNING - FILE EXTENSION SPECIFICATION IS MISSING.

IES205I FILENAME (COLUMNS 7-14) IS MULTI-DEFINED. SPECIFICATION IS NOT PROCESSED.

IES206I INVALID 'K' ENTRY (COLUMNS 40-43) FOR THIS INDEXED-SEQUENTIAL FILE (OR UPDATED RECORD) AND ALL NON-INDEXED-SEQUENTIAL FILES. EXECUTION IS DELETED.

IES207I NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

53194.8

IES208I INDEXED-SEQUENTIAL KEY NOT DEFINED FOR ONE OR MORE OUTPUT RECORD(S). EXECUTION IS DELETED.

IES209I END POSITION OF KEY FIELD (COLUMNS 40-43) IS GREATER THAN KEY LENGTH. ALL OR PART OF THE FIELD IS LOST, STARTING FROM RIGHTMOST POSITION.

IES210I WARNING - FILENAME (COLUMNS 7-14) IS DEFINED BUT NEVER USED.

IES211I FILENAME HAS BEEN REFERENCED PREVIOUSLY IN INPUT SPECIFICATIONS. SPECIFICATION IS NOT PROCESSED.

IES212I RESULTING INDICATOR IS INVALID OR UNDEFINED. ENTRY OF LO IS ASSUMED.

IES213I WARNING - RESULTING INDICATOR IS UNREFERENCED.

IES214I FIELD NAME IS UNDEFINED. FIELD IS PROCESSED WITH ASSUMED LENGTH OF 004.

IES215I WARNING - FIELD NAME IS MULTI-DEFINED.

IES216I WARNING - FIELD NAME IS UNREFERENCED.

IES217I THE COMBINED LENGTHS OF LITERALS AND FIELD NAMES EXCEEDS ALLOCATED CORE STORAGE. EXECUTION IS DELETED.

IES218I NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

IES219I NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

IES220I FILE SPECIFIED ON OUTPUT-FORMAT SPECIFICATIONS IS UNDEFINED OR NOT AN OUTPUT FILE (U, C, OR O). ENTIRE FILE IS DELETED FROM PROCESSING.

IES221I WARNING - FILENAME (COLUMNS 7-14) IS NOT REFERENCED ON OUTPUT SPECIFICATIONS.

IES222I NO VALID OUTPUT SPECIFICATIONS ARE PRESENT. COMPILATION IS ABORTED.

IES223I ALL OUTPUT LINES OF A PRINTER FILE MUST INDICATE SPACING AND/OR SKIPPING. SINGLE SPACING IS ASSUMED FOR ALL OUTPUT LINES OF NAMED FILE WHICH HAVE NO PRINT FUNCTION.

IES224I STACKER SELECT MAY NOT BE SPECIFIED WITH PRINT FILE. STACKER SELECT IS IGNORED AND SINGLE SPACING IS ASSUMED FOR ALL LINES OF NAMED FILE.

IES225I PRINT OR PUNCH FUNCTION MAY NOT BE SPECIFIED FOR AN OUTPUT RECORD OF TAPE OR DISK FILE. STACKER SELECT, SPACING, OR SKIPPING IS IGNORED ON ALL RECORDS OF NAMED FILE.

IES226I PRINT FUNCTION MAY NOT BE SPECIFIED FOR OUTPUT RECORD OF PUNCH FILE. SPACE AND SKIP ENTRIES ARE IGNORED FOR ALL RECORDS OF NAMED FILE.

IES227I IMPROPER USE OF PACKING OR ZERO SUPPRESSION ON ALPHAMERIC OR PACKED FIELD. ENTRY OF BLANK IS ASSUMED FOR INVALID CODE.

IES228I END POSITION SPECIFIED FOR THE FIELD IS GREATER THAN THE RECORD LENGTH. ALL OR PART OF THE FIELD IS LOST, STARTING FROM RIGHTMOST POSITION.

IES229I END POSITION IS LESS THAN THE FIELD LENGTH. FIELD IS NOT PROCESSED.

IES230I FIELD TO BE EDITED IS GREATER THAN THE EDIT WORD. RIGHTMOST DIGITS WILL BE LOST.

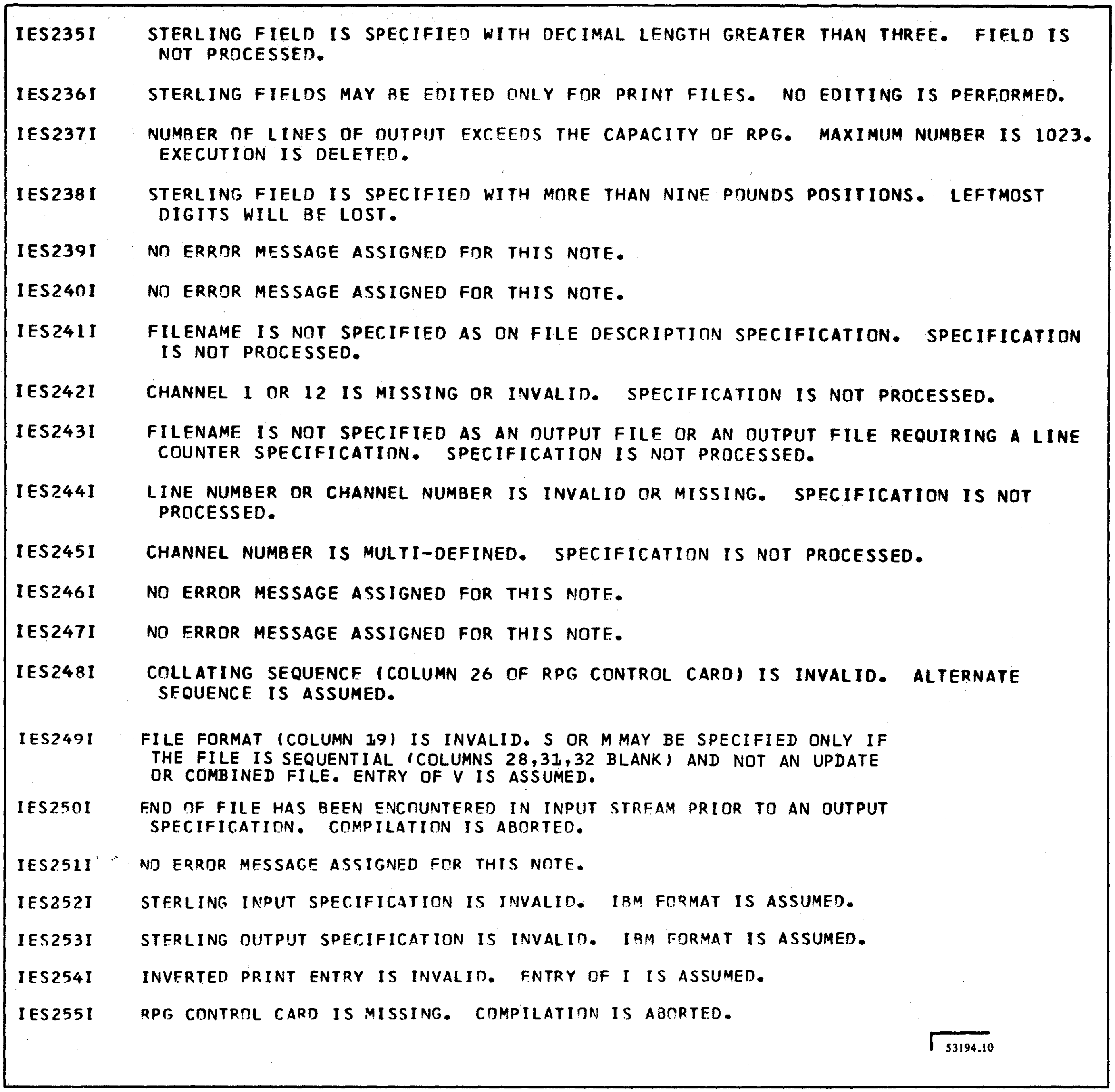
IES231I FIELD TO BE EDITED IS NOT NUMERIC. NO EDITING IS PERFORMED.

IES232I STERLING IS SPECIFIED FOR ALPHAMERIC FIELD. STERLING IS IGNORED.

IES233I STERLING SIGN POSITION IS SPECIFIED AS OTHER THAN NORMAL FOR PRINTED LINE. NORMAL POSITION IS ASSUMED.

IES234I LOCATION FOR STERLING SIGN EXCEEDS RECORD LENGTH. NORMAL POSITION FOR SIGN IS ASSUMED.

53194.9



```
IES235I   STERLING FIELD IS SPECIFIED WITH DECIMAL LENGTH GREATER THAN THREE.  FIELD IS
           NOT PROCESSED.

IES236I   STERLING FIELDS MAY BE EDITED ONLY FOR PRINT FILES.  NO EDITING IS PERFORMED.

IES237I   NUMBER OF LINES OF OUTPUT EXCEEDS THE CAPACITY OF RPG.  MAXIMUM NUMBER IS 1023.
           EXECUTION IS DELETED.

IES238I   STERLING FIELD IS SPECIFIED WITH MORE THAN NINE POUNDS POSITIONS.  LEFTMOST
           DIGITS WILL BE LOST.

IES239I   NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

IES240I   NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

IES241I   FILENAME IS NOT SPECIFIED AS ON FILE DESCRIPTION SPECIFICATION.  SPECIFICATION
           IS NOT PROCESSED.

IES242I   CHANNEL 1 OR 12 IS MISSING OR INVALID.  SPECIFICATION IS NOT PROCESSED.

IES243I   FILENAME IS NOT SPECIFIED AS AN OUTPUT FILE OR AN OUTPUT FILE REQUIRING A LINE
           COUNTER SPECIFICATION.  SPECIFICATION IS NOT PROCESSED.

IES244I   LINE NUMBER OR CHANNEL NUMBER IS INVALID OR MISSING.  SPECIFICATION IS NOT
           PROCESSED.

IES245I   CHANNEL NUMBER IS MULTI-DEFINED.  SPECIFICATION IS NOT PROCESSED.

IES246I   NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

IES247I   NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

IES248I   COLLATING SEQUENCE (COLUMN 26 OF RPG CONTROL CARD) IS INVALID.  ALTERNATE
           SEQUENCE IS ASSUMED.

IES249I   FILE FORMAT (COLUMN 19) IS INVALID. S OR M MAY BE SPECIFIED ONLY IF
           THE FILE IS SEQUENTIAL (COLUMNS 28,31,32 BLANK) AND NOT AN UPDATE
           OR COMBINED FILE. ENTRY OF V IS ASSUMED.

IES250I   END OF FILE HAS BEEN ENCOUNTERED IN INPUT STREAM PRIOR TO AN OUTPUT
           SPECIFICATION.  COMPILATION IS ABORTED.

IES251I   NO ERROR MESSAGE ASSIGNED FOR THIS NOTE.

IES252I   STERLING INPUT SPECIFICATION IS INVALID.  IBM FORMAT IS ASSUMED.

IES253I   STERLING OUTPUT SPECIFICATION IS INVALID.  IBM FORMAT IS ASSUMED.

IES254I   INVERTED PRINT ENTRY IS INVALID.  ENTRY OF I IS ASSUMED.

IES255I   RPG CONTROL CARD IS MISSING.  COMPILATION IS ABORTED.
```

53194.10

Appendix D. Conditions That Automatically Turn On Halt Indicator (H0)

The object program:

- Read an input record that was not defined on the Input Specifications form (positions 21-41).
- Found an input record out of the predetermined sequence of card types specified by the entry in *Sequence* (positions 15-16) on the Input Specifications form.
- Found an input record out of sequence when the entry in *Matching Fields* (positions 61-62) on the Input Specifications form was used for sequence checking a single input file.
- Encountered a chaining field in the chaining file that does not appear in the chained file during random processing of multiple input files.
- Did not find a record with the correct key at the designated track address during random processing by record key of a directly organized file.
- Did not find the record key that designates the lower limit (obtained from the RA file) during sequential processing between limits of an indexed-sequential file.
- Found a wrong length record during processing of an indexed-sequential file.
- Found an invalid length record (zero or too long) during random processing by record identification of a file on a DASD.
- Found a difference between the key length of a DASD record in an indexed-sequential file and the length as specified in *Length of Record Address Field* (positions 29-30) on the File Description form during processing with RA file support (random, or between limits).
- Found a difference between the key length in the chained indexed-sequential file and the length as specified (positions 44-51) on the Input Specifications form during chaining of multiple input files.
- Encountered a data check on the DASD during random processing of a directly organized file.
- Encountered a DASD error during sequential or random processing of an indexed-sequential file.
- Found a record identification outside the defined limits during random processing of a directly organized file.
- Did not find keys in a directly organized file when the entry in *Record Address Type* (position 31) on the File Description Specifications form specified processing by key.
- Did not find a record with the specified key during random processing of an indexed-sequential file.
- Encountered an irrecoverable input/output error during processing of a sequential file.
- Encountered an irrecoverable input/output error during processing of a combined file.
- Found the prime data area was filled while creating an indexed-sequential file.
- Found the cylinder and/or master index areas were filled while creating an indexed-sequential file.
- Found a duplicate record when creating or adding to an indexed-sequential file.
- Found a sequence error in record keys when creating an indexed-sequential file.
- Found the overflow area was filled when adding to an indexed-sequential file.

Note: Unless the H0 indicator is set off by a SETOF operation entry on the Calculation Specifications form (see *Setting Indicators On or Off*) the program terminates before the next input record is read.

Appendix E. Tips for Minimizing Storage Requirements

These are generalized suggestions. They do not necessarily hold true under all conditions, nor are they always practicable because of other factors involved.

1. If the majority of the input fields are identical, use OR lines and field record relation entries rather than completely separate file identifications.

2. Use the same field name for different input fields in different card types when field size and format (alphameric, or numeric and number of decimal places) are identical if the data need not be preserved from one card type to the other.

3. Use the same field name repeatedly in calculations as result field for various different items whenever the result need no longer be retained; that is, employ one work field in several calculations.

4. Use the minimum number of Record Identification Codes to identify a card type. Use C for Record Identification Code (positions 26, 33, and 40) in preference to D or Z.

5. Do not use Matching Fields specifications solely for purposes of sequence checking a single file on one or two fields. This requires more storage than sequence checking by comparing (COMP) in the calculation specifications.

   Sequence checking by COMP operation also permits detection of duplicates; this cannot be done using Matching Fields for sequence checking.

   Note: Sequence checking a single file by COMP operation, rather than by Matching Fields entries, also tends to improve throughput. A Matching Fields specification delays the reading of the next card.

6. Use field indicators in input specifications in preference to COMP operation in calculations.

   a. Where feasible, group together calculation specifications conditioned by the same indicators (positions 7-17). (In this context, same also implies identical status criterion for an indicator: either on or not on in all lines.)

   b. When successive specifications lines are conditioned by the same indicators, specify the indicators in the same order (in positions 9-11, 12-14, 15-17).

   When the same indicator is assigned as Resulting Indicator (positions 54-59) in a specifications line, and as conditioning indicator (positions 7-17) in that and the next line, there is no storage saving if the two lines contain identical conditioning indicators.

7. Try to define a uniform number of decimal places for all numeric fields or literals used in one arithmetic or compare operation. For ADD/SUB operations, also try to have fields of literals of equal length.

8. When a number of fields are to be edited for output, define as many of them as possible with the same length, and edit them uniformly. This requires storage of only a single edit word.

9. a. In ADD and SUB operations, you save significant storage space when the Result Field and Factor 1 and/or Factor 2 are the same, provided the other Factor is of equal or shorter length and all three have the same number of decimal places.

   b. In Z-ADD and Z-SUB, you have significant storage space if the two fields have the same number of decimal places.

   c. In MULT, DIV, and MVR, you save storage space if no decimal alignment is required.

10. To clear a numeric field to zero, use the SUB operation. Enter the name of the field to be cleared in Factor 1, Factor 2, and Result Field of the Calculation Specifications form.

11. To set indicators on or off (as Resulting Indicators) in calculation specifications, based on whether numeric field contents are plus, minus, or zero, add a literal zero rather than comparing with a zero.

    For the storage savings to be realized:

    a. The Result Field must be the same as one of the two Factors, and

    b. The literal zero must be specified with the same number of decimal places as in the other Factor.

12. Use GOTO to:

    a. Bypass a group of inapplicable calculation specifications rather than conditioning each specification by indicators, particularly when several conditioning indicators are required.

    b. Use the same series of calculation instructions at different points in the calculation specifications, rather than repeating identical, or near identical, specifications.

13. Punch constant data (such as date) into the input card in edited format rather than using an edit word in output specifications.

14. When testing for specific numeric codes in a field, define the field as alphameric and compare (COMP) to alphameric literals.

15. Whenever possible, use a series of compares rather than a LOKUP. The first table read and first LOKUP require over 1000 bytes.

16. Whenever possible on input specifications, test a record identifier of a character rather than a zone or digit. The first zone or digit test causes a general routine of 310 bytes to be included.

17. Whenever possible, use an odd length numeric field rather than even length.

18. When using an operation code of ADD, specify the factors and result as

    A ADD B A

    rather than as

    B ADD A A

    since the first example permits the opreation without moving the factors to work areas.

Appendix F. Running The OS RPG Compiler Under OS/VS1

In order to run the OS RPG compiler on OS/VSI, you must apply changes to the compiler. If you do not make these changes, a system or compiler ABEND could result.

First, prepare the necessary JCL to execute the program IMASPZAP and to address the system library which contains the RPG compiler. Apply the following changes:

(These changes apply *only* to the final release (V1M10) of the OS RPG compiler.)

Change 1

Name	RPG10000 IES00010
Verify	0510 00FFFFF8
Rep	0510 00002710

Change 2

Name	RPG10240 IESCEX
Verify	000A 50D061D0
Rep	000A 50D0602C
Verify	0022 58D061D0
Rep	0022 58D0602C

Change 3

A potential error also exists in each of the other CIOEX phases. If failures continue after you have applied the change to RPG10240, apply the following changes to each of the remaining CIOEX phases. The application of these changes can result in extraneous information printing in the heading lines on the compiler source listing.

Verify	000A 50D061D0
Rep	000A 50D060CC
Verify	0022 58D061D0
Rep	0022 58D060CC

The remaining CIOEX phases are:

Name	RPG10000	IESCEX
Name	RPG10030	IESCEX
Name	RPG10040	IESCEX
Name	RPG10050	IESCEX
Name	RPG10060	IESCEX
Name	RPG10070	IESCEX
Name	RPG10080	IESCEX
Name	RPG10090	IESCEX
Name	RPG10100	IESCEX
Name	RPG10120	IESCEX
Name	RPG10130	IESCEX
Name	RPG10140	IESCEX
Name	RPG10150	IESCEX
Name	RPG10160	IESCEX
Name	RPG10170	IESCEX
Name	RPG10180	IESCEX
Name	RPG10190	IESCEX
Name	RPG10200	IESCEX
Name	RPG10210	IESCEX
Name	RPG10220	IESCEX
Name	RPG10230	IESCEX

Phase RPG10110 should be changed as follows:

Name	RPG10110 IESCEX
Verify	0044 50D061D0
Rep	0044 50D060CC
Verify	005C 58D061D0
Rep	005C 58D060CC

Index

GC24-3337-6

IBM System/360 Printed in U.S.A. GC24-3337-6