

CONTENTS

IBM BASIC OPERATING SYSTEM/360 REPORT PROGRAM GENERATOR (8K DISK)	5	USING TABLES AND EXIT ROUTINES IN THE OBJECT PROGRAM.	95
Compatibility of Basic Operating System/360 RPG with Model 20 RPG or with System/360 Basic Programming Support RPG	5	Using Tables in the Object Program.	95
Function of RPG.	5	Exit to a User's Routine.	103
Using RPG.	5	DISK STORAGE CONCEPTS.	107
Machine Features Required.	6	Introduction and Terminology.	107
FUNDAMENTALS OF RPG PROGRAMMING.	8	File Organization.	108
Functions Described.	8	File Processing.	109
Correlation of the RPG Specification Sheets.	29	File Processing in RPG.	110
PROGRAM LOGIC.	32	PROCESSING SINGLE INPUT FILES.	111
PROBLEM DEFINITION.	34	Sequential Organization.	111
RPG SPECIFICATIONS SHEETS.	36	Indexed-Sequential Organization.	111
Common Fields.	38	Direct Organization.	114
INPUT SPECIFICATIONS SHEET.	40	Using the Addrout (Address Output) Option.	116
Record Identification Entries.	40	PROCESSING MULTIPLE INPUT FILES.	119
Field Description Entries.	47	Creating Record Address Files (RAF).	127
CALCULATION SPECIFICATION SHEET.	56	Multiple File Processing.	127
Arithmetic Operations.	60	RPG SOURCE DECK ARRANGEMENT.	130
Move Operations.	61	VARIABLE-LENGTH RECORDS.	132
Testing or Compare Operations.	63	DIAGNOSTICS.	135
Branching and Exit Operations.	64	Sample Program One.	145
Turning Indicators On or Off.	65	Sample Program Two.	151
Table Operations.	66	Sample Program Three.	156
Conversion Routine Operations.	66	APPENDIX A: INDICATOR CHART.	163
Testing the Results of a Calculation.	67	APPENDIX B: RPG LOGIC FLOW CHARTS.	164
OUTPUT-FORMAT SPECIFICATION SHEET.	71	APPENDIX C: STERLING ROUTINES FOR THE REPORT PROGRAM GENERATOR.	167
File Identification and Control.	71	Pound Sterling Formats.	168
FILE DESCRIPTION SPECIFICATION SHEET.	84	Editing of Sterling Fields.	168
FILE EXTENSION SPECIFICATION SHEET.	91	APPENDIX D: CONVERSION ROUTINE OPERATION CODES.	170
		INDEX.	171

RPG operates under control of the Basic Operating System/360. The Basic 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 Basic Operating System. The Decimal Arithmetic feature is required for RPG.

COMPATIBILITY OF BASIC OPERATING SYSTEM/360
RPG WITH MODEL 20 RPG OR WITH SYSTEM/360
BASIC PROGRAMMING SUPPORT RPG

The RPG source language is upward-compatible. The Basic Operating System/360 Report Program Generator can compile a Model 20 RPG source program, or a Basic Programming Support RPG source program that adheres to its language specifications. The object program will produce equivalent results, if the same input/output facilities exist.

- The File Description Specification sheet of the Basic Operating System/360 RPG may require additional information (record length and overflow indicators).
- The control card of the Model 20 RPG or Basic Programming Support RPG programs must be modified before the object program can be generated by the Basic Operating System/360 RPG.

FUNCTION OF RPG

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

1. Program-generating, and
2. Data processing.

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

In data processing, 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 RPG consists of the general operations illustrated 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 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 Specification sheet.

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

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

He must describe all files used by the object program (input files, output files, table files, etc.) by making entries on the File Description Specification sheet.

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

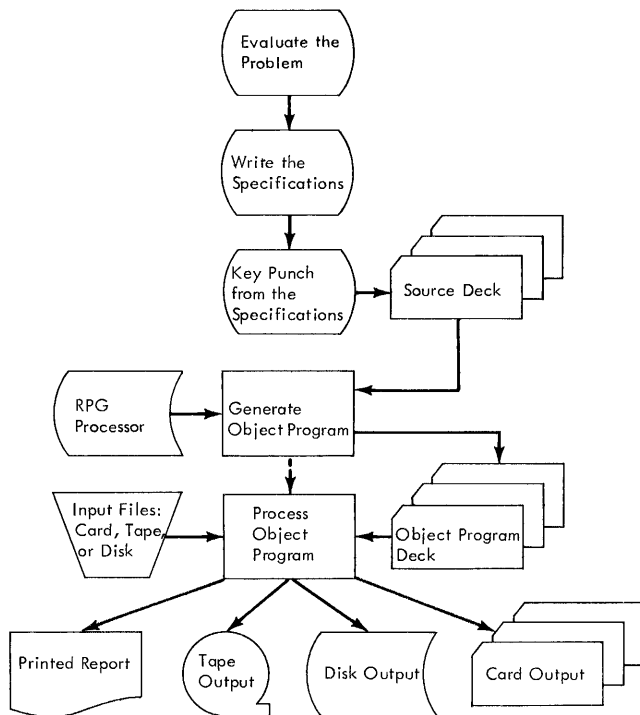


Figure 1. Producing Reports Using the RPG Program

4. These punched cards (called a source deck) are combined with the RPG Processor control card and the Basic Operating System/360 Job Control statements. The source deck and the control cards are supplied to an input device and are processed under control of the Basic Operating System. At the end of this processing run (known as a generating or compiling run), a program capable of preparing the report specified by the programmer has been produced. This program (known as an object program) 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 object run.

At the end of the object run, the report has been prepared and any other functions, such as file updating, are completed. Through facilities provided by the Basic Operating System, the object program can be retained for later runs without recompilation.

MACHINE FEATURES REQUIRED

PROGRAM GENERATION

1. 8K bytes of main storage
2. One of the following:
 - IBM 1442 Card Read-Punch
 - IBM 2540 Card Read-Punch
3. One of the following:
 - IBM 1403 Printer
 - IBM 1404 Printer (continuous forms operation only)
 - IBM 1443 Printer
4. IBM 2311 Disk Storage Drive
5. Standard Instruction Set
6. Decimal Arithmetic Feature

OBJECT PROGRAM EXECUTION

1. 8K bytes of main storage
2. I/O units as requested by the specifications
3. Standard Instruction Set
4. Decimal Arithmetic Feature
5. IBM 2311 Disk Storage Drive

ADDITIONAL MACHINE FEATURES SUPPORTED

PROGRAM GENERATION

1. 16K, 32K, or 64K bytes of main storage.
2. One IBM 2400-series Magnetic Tape Unit (7- or 9-track) (for source program input only).

OBJECT PROGRAM EXECUTION

1. 16K, 32K, or 64K bytes of main storage.
2. IBM 2311 Disk Storage Drive

3. IBM 2400-series Magnetic Tape Unit (7- or 9-track). If a 7-track tape unit is used, the Data Conversion feature is required.

For usage of tape units at either compile time or object time, refer to Appendix H of the programmer's guide publication listed on the front cover of this publication.

FUNDAMENTALS OF RPG PROGRAMMING

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 this introduction to RPG. Programmers familiar with the basic concepts of RPG may omit this section and concentrate on the detailed specifications descriptions contained in the section RPG Specification Sheets of this publication.

Because of the numerous fields on the five specification sheets, 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 sheets.

At the end of the discussion of basic RPG functions, a simple file-to-file listing application is used to illustrate how the specification sheets are related.

FUNCTIONS DESCRIBED

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

1. Describing a record and its fields to the system.
2. Adding and subtracting (crossfooting).
3. Detail printing.
4. Establishing control fields.

5. Total calculations.
6. Detail and total printing.
7. Group printing.
8. Group indication.
9. Overflow printing.
10. Testing for zero, plus and minus balance.
11. Using field indicators.
12. Comparing.
13. Multiplying and dividing.
14. Sequence-checking.

DESCRIBING A RECORD AND ITS FIELDS TO THE SYSTEM

Entries on the Input Specification sheet 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 columns 46-50; Field B is contained in columns 56-60; and Field C is contained in columns 66-70 of each record. The file that contains the records and the fields within each record are described on the Input Specification sheet.

DIV	DEPT	EMP NO.	EMPLOYEE NAME	C O D E	FIELD A	FIELD B	FIELD C	C O D E
0000	0000	0000000	0000000000000000	00000000	000000000000	000000000000	000000000000	00000000
1111	1111	1111111	1111111111111111	11111111	111111111111	111111111111	111111111111	11111111
2222	2222	2222222	2222222222222222	22222222	222222222222	222222222222	222222222222	22222222
3333	3333	3333333	3333333333333333	33333333	333333333333	333333333333	333333333333	33333333
4444	4444	4444444	4444444444444444	44444444	444444444444	444444444444	444444444444	44444444
5555	5555	5555555	5555555555555555	55555555	555555555555	555555555555	555555555555	55555555
6666	6666	6666666	6666666666666666	66666666	666666666666	666666666666	666666666666	66666666
7777	7777	7777777	7777777777777777	77777777	777777777777	777777777777	777777777777	77777777
8888	8888	8888888	8888888888888888	88888888	888888888888	888888888888	888888888888	88888888
9999	9999	9999999	9999999999999999	99999999	999999999999	999999999999	999999999999	99999999

• Figure 2. Input Cards for Detail Labor File

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. When these two conditions exist in the card record, the record is valid (that is, it contains fields A, B, and C).

reference the fields in text. The remaining fields on the form are described later.

1. Each file of cards must be given a name. In this example the input deck of cards represents a detail labor file (Figure 2). A file name can be used to describe only one 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

SPECIFICATIONS

The entries on the Input Specification sheet (Figure 3) that apply to this problem are circled and identified by numbers to

INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS
 IBM System/360

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)	Resulting Indicator	Record Identification Codes			Field Location		Field Name	Control Level (1-16)	Matching Fields or Chaining Fields	Field-Record Relation	Field Indicators		Sterling Sign Position
					Position	Net (N) C/Z/D Character	Position	Net (N) C/Z/D Character	Position					Net (N) C/Z/D Character	From	
1		DETLABORAA			14	35	D5	80	NZ							
2										46	50	FLDA				
3										56	60	FLDB				
4										66	70	FLDC				

• Figure 3. Input Specification Sheet

the fields called Record Identification Codes (columns 21-41 of the form). These fields provide for three identifying codes; however, more codes can be indicated (see Record Identification Codes).

In this example the card codes are specified by writing a 5 in Character and a 35 in Position for the first code; and by writing an 80 in Position and a minus in Character, and an N in Not 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 resulting indicator (columns 19-20 of the form), is used to refer to this specific card type on other specification sheets. Using this code reduces the number of entries required on other specification forms, as will be seen later.

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 (Column 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 main 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 will cause the contents of fields A, B, and C from the card to be read into the system during the processing of the object program. Calculations to be made from the input data are written on the Calculation Specification sheet.

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. This operation is sometimes referred to as crossfooting. The form required for this operation is the Calculation Specification sheet.

Specifications

Figure 4 shows the required entries using the fields from the previous example. The first line of the Calculation Specification sheet 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 on the Input Specification sheet in Resulting Indicator (Figure 3).
2. The card columns for FLDA and FLDB were defined by entries in columns 46-47 and 50-51 on the input forms and need not be specified again. The names assigned in columns 53-56 will be used to refer to the fields.
3. The result field FLDD is defined for the program by merely writing the name FLDD in Result Field (columns 43-48) and indicating in Field Length (columns 49-51) the number of positions that must be set aside for this field in the object program. Just as in the case of FLDA and FLDB, which were defined on the input form, the name FLDD can now be used in other calculation operations or used in defining output specifications.

Note: The result of $A + B$ is not placed back into FLDA. To do 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 program to subtract the contents of FLDC from the result just previously attained in FLDD, and to store the new result back into FLDD.

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)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators						Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position		
									And	And	And	And	And	And									
									Not	Not	Not	Not	Not	Not									
010	o	DAILYRPTD	D			2																	
020	o																						
030	o																						
040	o																						
050	o																						
06	o																						
07	o																						
08	o																						

● Figure 5. Output-Format Specification Sheet

Problem

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

stored in FLDA (00126) is printed as 126.)

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 file name. In this example it has been called DAILYRPTD.

The D in Type H/D/T indicates that the line being printed is a detail line. That is, it contains information from the record just read. (The other possible entries, H and T, indicating heading lines and total lines, 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 to be printed.
3. Each field to be printed must be specified under the heading Field Name. The Z in Zero Suppress means that zeros to the left of significant digits are not printed. (For example, the value

The printing positions for data to be printed on the output report are specified in End Position in Output Record (Columns 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 specifications, 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 Problem Definition.

Figures 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 pre-

pare a report. A significant item not yet described is how to specify a control field.

(Because as many as nine levels of totals are possible in RPG, the common terms of minor, intermediate, and major totals are inadequate.)

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.

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.

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, as illustrated in the following example.

The reading of the employee-name field (fifth line of the sheet) 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 column of the Input Specification sheet. When a numeric field is specified in Field Location, any digit (0 to 9) placed in Decimal Position causes zone punches in all positions of the field (except the units position) to be removed.

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

IBM
INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS
Form X24-3350
Printed in U. S. A.
IBM System/360

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)	Resulting Indicator	Record Identification Codes						Field Location		Field Name	Control Level (1-19)	Matching Fields or Chaining Fields	Field-Record Relation	Field Indicators			Sterling Sign Position			
						Position 1	Position 2	Position 3	Position 4	Position 5	Position 6	From	To					Plus	Minus	Zero or Blank				
1		DETLABORAAA	14			35	D5	80	NZ	-			2											
2													1	40	DIVSON	L3								
3													5	80	DEPT	L2								
4													9	14	EMPNO	L1								
5													15	28	NAME									
6													46	50	FLDA									
7													56	60	FLDB									
8													66	70	FLDC									
9																								

major intermediate minor

● Figure 6. Specifying Control Levels on the Input Sheet



INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS
 IBM System/360

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic								
	Punch								

Page 1 2

Line	Form Type	Control Level (L1-L3)	Indicators				Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Resulting Indicators		
			And		And								Plus	Minus	Zero or Blank
			Not	Not	Not	Not							High	Low	Equal
1	C		14			FLDA	ADD	FLDB	FLDD	7					
2	C		14			FLDD	SUB	FLDC	FLDD	7					
3	C		14			TOTE	ADD	FLDD	TOTE	8					
4	L1					TOTF	ADD	TOTE	TOTF	8					
5	L2					TOTG	ADD	TOTF	TOTG	8					
6	L3					FINTOT	ADD	TOTG	FINTOT	8					
7	C														

● Figure 7. Specifying Control Levels on the Calculation Sheet

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 7 contains the data shown in Figure 4. The additional information for this example is circled. During processing of the object program, 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 program each time a detail card is read (see Figure 7, 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 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 data processing 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-program report.

DIVSON	DEPT	EMPNO	NAME	FLDD	
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 9. Detailed Printed Report

- This information is similar to the specifications in the previous example in Figure 5.
- The data fields DIVSON, DEPT, EMPNO, NAME, and FLDD, are specified for printing.
- The specifications to print the total for the first control level shown on lines 7 and 8 are:
 - The T in H/D/T (column 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.

- A 3 in Space After (column 18) provides two blank lines after each printed line to make the report easier to read.
- The L1 in Output Indicators (columns 23-24) indicates that the line is to be printed only on a level-1 control break.
- The field to be printed (TOTE) is indicated under Field Name (columns 32-37).
- The Z in Zero Suppress (column 38) indicates that zeros to the left of significant digits are not to be printed.
- The B in Blank After (column 39) causes the main-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.

- The specifications for printing the final total contain the code LR (Last Record) in Output Indicators (columns 23-24). In this example, the indicator LR is turned 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 10.

DIV
DEPT
EMP #

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

Figure 13 illustrates how the detail-printed report specifications in Figure 11 could be altered to specify a group-indication report.

Figure 12. Example of a Group-Indication Report

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.

Figure 15 and 16 illustrate a test for a zero balance, and Figure 17 shows the specifications for a test for a minus balance.

Input Specifications

Three types of tests can be made on the Input Specification sheet:

1. Testing an input field to determine if it contains a plus value.
2. Testing an input field to determine if it contains a minus value.
3. Testing an input field to determine if it is blank or is punched with zero.

Figure 15 illustrates a typical test of the content of an input field. If FLDA (line 6) contains zeros, it is unnecessary to perform some of the operations entered on the Calculation Specification sheet. To make this test, the programmer places a number from 01 to 99 in Field Indicators: Zero or Blank (columns 69-70). In this example the number is 18. If a card read into the system contains zeros in columns 46-50 (FLDA), indicator 18 is turned on.

Turning 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 turned on by a specific record identification code from an input card.

Calculation Specifications

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

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

For example, the entries circled in Figure 16 are the additional specifications that bypass the calculation upon detail cards if the value in FLDA is zero or blank.

INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS
IBM System/360

Form X24-3350
Printed in U. S. A.

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 (OO)	Resulting Indicator	Record Identification Codes									Field Location		Field Name	Control Level (L1-L9) Marking Fields or Changing Fields	Field-Record Relation	Field Indicators		Sterling Sign Position	
					1			2			3			From	To				Plus	Minus		Zero or Blank
					Position	Not (N) C/Z/D Character	Character	Position	Not (N) C/Z/D Character	Character	Position	Not (N) C/Z/D Character	Character	Stacker Select Packed (P)	Decimal Positions				Field Name	Control Level (L1-L9)		Plus
01		DETLABORAA	14		35	D5		8	0	NZ-			2									
02														1	4	0	DIVISION	L3				
03														5	8	0	DEPT	L2				
04														9	14	0	EMPNO	L1				
05														15	28		NAME					
06														46	50	0	FLDA				18	
07														56	60	0	FLDB					
08														66	70	0	FLDC					

Figure 15. Specifying a Test for a Zero-Balance

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic						
	Punch						

Page

1	2
---	---

Program Identification

75	76	77	78	79	80
----	----	----	----	----	----

Line	Farm Type	Control Level (00, 19, LR)	Indicators														Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions Half Adjust (H)	Resulting Indicators			Comments
			And		And		And		And		Plus	Minus	Zero or Blank													
			Not	Not	Not	Not	Not	Not	Not	Not	1 > 2	1 < 2	1 = 2													
01	c																FLDA	ADD	FLDB	FLDD	7	0				
02	c																FLDD	SUB	FLDC	FLDD	7	0				
03	c																FLDD	ADD	TOTE	TOTE	8	0				
04	c																Z-ADD	0	FLDD	FLDD	7	0				
05	c	L1															TOTE	ADD	TOTF	TOTF	8	0				
06	c	L2															TOTF	ADD	TOTG	TOTG	8	0				
07	c	L3															TOTG	ADD	FINTOT	FINTOT	8	0				
08	c																									

● Figure 17. Testing for a Minus Condition

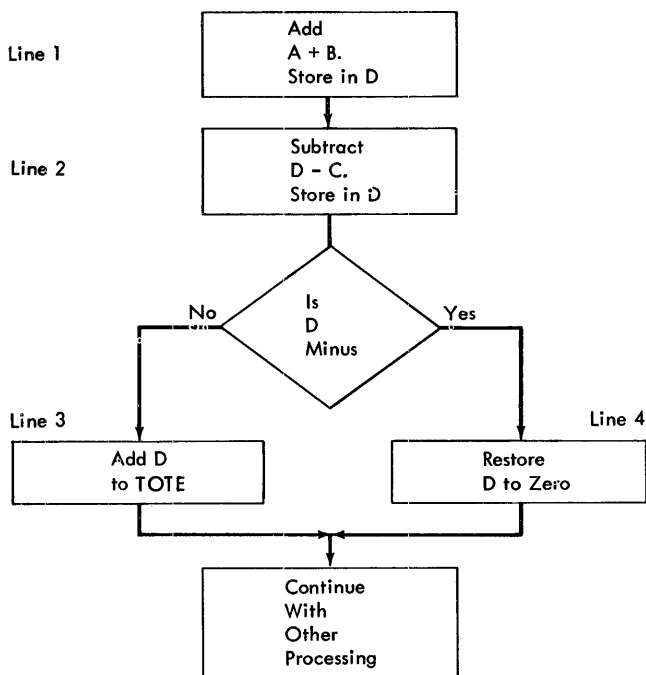


Figure 18. Testing Indicators to Govern Processing

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 sheet (Figure 17) by placing an indicator code (19) in Resulting Indicators: Minus on specification-line 2. Indicator 19 will be turned on for a minus condition.

The function of adding TOTE to FLDD is specified on line 3. It is accomplished only if there is a 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. This is accomplished by the zero and add (Z-ADD) operation. 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 specifications on the form in Figure 17 are the same as those from previous examples.

By using indicator 19, it is possible to suppress detail-card printing when 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 specifi-

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS
 IBM System/360

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)	Resulting Indicator	Record Identification Codes						Field Location		Field Name	Control Level (L1-L9) Matching Fields or Chaining Fields	Field Indicators			Sterling Sign Position			
						Position	Char (N) C/Z/D	Character	Position	Char (N) C/Z/D	Character	Position	Char (N) C/Z/D			Character	From	To		Plus	Minus	Zero or Blank
01	1	INVENTORY	A		29	80	K															
02	1												4	8	PARTN							
03	1												28	323	PRICE							
04	1												33	360	QUANTITY							
05	1												41	482	YDCOST							
06	1												49	540	YDUSE							

● Figure 22. Input Specifications

		No. of Decimal Positions
Part Number	xxxxx	None (alphabetic)
Price	xx.xxx	3
Quantity	xxxx.	0
Y/D Cost	xxxxxx.xx	2
Y/D Usage	xxxxxx.	0

Decimal Positions on the Input Specification sheet.

The calculation specifications for this example are shown in Figure 23. 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 (column 53). The following is an example of the arithmetic of this operation, using actual values:

As shown in Figure 22, the decimal positions of each field are entered under

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS
 IBM System/360

Date _____
 Program _____
 Programmer _____

Punching Instruction	Graphic				
	Punch				

Page 1 2

Line	Form Type	Control Level (L0-L9, LR)	Indicators			Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions Half Adjust (H)	Resulting Indicators		
			And	And	And							Plus	Minus	Zero or Blank
01	C					PRICE	MULT	QUANTITY	COST	9	2H			
02	C					YDCOST	DIV	YDUSE	PRICE	5	3H			

● Figure 23. Calculation Specifications

Price	1.213
Quantity	x 216
	<u>7 278</u>
	12 13
	<u>242 6</u>
	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. 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
<u>262.013</u>

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 Specification sheet 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, with 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.

		<u>1.3419</u>	(price)
(usage)	1296)	1739.23xx	(year-to-date 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
<u>1.3424</u>

The result, 1.342, is stored as the contents of PRICE. The position that was half-adjusted is 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 Using the Matching Fields Specification for Sequence-Checking.)

Sequence-Checking of Different Record Types within a Control Group

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

Card	Code
Master Customer	5 in col. 78
Shipped Via	3 in col. 78
Part Number	2 in col. 78
Comments	8 in col. 78

Figure 24 illustrates the specifications required to check the sequence of a group of four card-types. The numbers on the form refer to the following numbers:



INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS
 IBM System/360

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic						
	Punch						

Page

1	2
---	---

Line	Form Type	Filename	Sequence Number (1-N) Option (00)	Resulting Indicator	Record Identification Codes									Field Location		Field Name	Control Level (1-19) Mending Fields or Chaining Fields	Field-Record Relation
					1			2			3			From	To			
					Position	Net (N) C/Z/D	Character	Position	Net (N) C/Z/D	Character	Position	Net (N) C/Z/D	Character	Sticker Select	Packed (P)			
010		INVOICE	011	21	78	C	5											
020													1	16	ACCOUNT			
030													17	20	CUSNO			
040													40	46	ADDRS			
050			021	022	78	C	3											
060													1	6	SHTPER			
070													28	31	LOCATN			
080													60	66	RTE			
090			03W	23	78	C	2											
100													1	6	PARTNO			
110													7	10	NUMORD			
120													40	46	DESC			
130			04N	024	78	D	8											
140													1	6	DISCNT			
150													28	31	NEWSAL			
160													60	66	DUEDAT			

● Figure 24. Input Specifications

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.

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 for any one file.

These specifications are all that are required to cause the object program to perform a sequence-check of the various record-types. If a sequence-check error is detected, the object program is halted.

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 two record identification lines, the number 1 in Number indicates that no more than one record of that type must be present in each group. In this example, only one master card and no more

than one shipped-via card for each invoice must be present. If there is more than one of either of these cards, the program recognizes an error in the input file.

The letter N in Number of the specifications for the last two record-types means that multiple card-types for each part number may be present. In this case, multiple cards for part number and comments 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 was not specified, it means that the particular record must be present. This requirement applies only if Sequence has been specified as numeric.

CORRELATION OF THE RPG SPECIFICATION SHEETS

Figure 25 shows a file-to-file sample program. It illustrates the relationships of the RPG specification sheets. (This type of report is often called an 80/80 listing).

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS
 IBM System/360

Date _____
 Program _____
 Programmer _____

Punching Instruction: Graphic
 Punch

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

Line	Form Type	Filename	File Type			Mode of Processing						Device	Symbolic Device	Name of Label Exit	Extent Exit for DAM	Comments
			I/O/U/C	P/S/C/T	A/D	E/V	Length of Record	Address Field	Record Address Type	Type of File Organization	Overflow Indicator					
010	F	INPUT	I	P	F	80	80						READ42	SYSRDR		
020	F	OUTPUT	O	V		132	132						PRINTERS	SYSLSL		

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS
 IBM System/360

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)	Resulting Indicator	Record Identification Codes			Field Location		Field Name	Control Level (1-19)	Matching Fields or Chaining Fields	Field-Record Relation	Field Indicators			Sterling Sign Position
					Position 1	Position 2	Position 3	From	To					Plus	Minus	Zero or Blank	
010	F	INPUT	AA	01				1	80	CARDIN							

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS
 IBM System/360

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)	Space	Skip	Output Indicators			Field Name	Zero Suppress (Z) Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
						Stacker Select Before	Stacker Select After	And						
010	F	OUTPUT	D	1									CARDIN	80

● Figure 25. File-to-File Program

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 contents of the cards are to be printed. Three specifications sheets are required

for this program: File Description, Input, and Output-Format.

FILE DESCRIPTION SHEET

The two files are described on this sheet. 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 columns 1-5. An F in column 6 indicates that each entry is a File Description entry. The file names are entered in columns 7-14 (Filename). Column 15 contains an I or an O to indicate whether the file has an input or an output function.

Column 16 of the input file on the File Description Specification sheet contains a P because the file described is the primary input file for the job; the E in column 17 indicates that the end-of-file condition for this file occurs when this file is depleted.

Column 19 contains an F and a V to indicate that the file formats are fixed-length and variable-length respectively. (See Variable-Length Records.)

Block length (columns 20-23) is 80 for the input file because each card is a block of data. Record length (columns 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 columns 75-80.

If the input file was read in by an IBM 1442 Card Read-Punch, the code in Device (columns 40-46) would be READ42. The output printer would have a device code of PRINTER.

The symbolic device codes for the two files would be SYSRDR and SYSLST.

INPUT SPECIFICATION SHEET

The Input Specification sheet also has the program identification entered in columns 75-80 and the page and line numbers in

columns 1-5. An I in column 6 of each card indicates an Input Specification Entry.

The filename is again entered into columns 7-14. Columns 15-16 contain the sequence code AA. Indicator 01, entered into columns 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 columns 1-80.

OUTPUT SPECIFICATION SHEET

On the Output Format Specification sheet, the filename of the output file is entered in columns 7-14. The D in column 15 indicates that each line printed in this file is a detail line. A single space after printing is specified by the entry in column 18. Indicator 01 from the Input Specification sheet is specified in columns 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 columns 32-37. Data from the field labeled CARDIN will be printed (end position of 80) in the output record as specified in columns 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 sheets 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.

More specific information about each specification sheet is contained under RPG Specification Sheets.

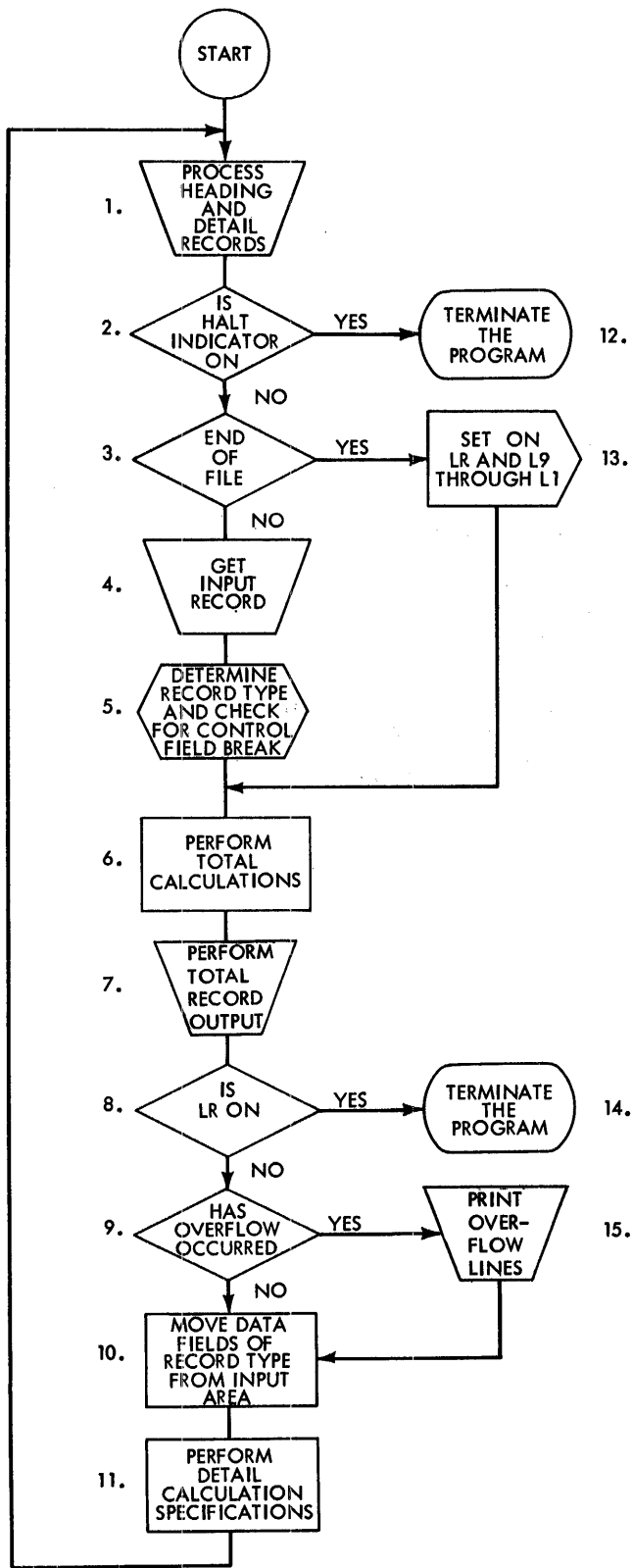
PROGRAM LOGIC

Each object 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 Specification sheets are performed. These instances are called detail and total time.

For the illustration of this concept, a generalized flowchart of an RPG object program is shown in Figure 26.

The following numbers correspond to the numbers on Figure 26. A program cycle begins with item 1 and continues through item 11. Steps 6 and 7 are referred to as total time. Steps 11 and 1 are referred to as detail time.

1. Before the first record is read, the object program prepares and prints any heading information to be printed on the first page. After the first record has been read, the object program prepares and prints any heading and detail information which is not conditioned on overflow.
2. The object program tests for any halt indicators. If any halt indicators are on, the program branches to item 12.
3. The object program tests for the end-of-file conditions. If the end-of-file condition has occurred, the program branches to item 13.
4. The object program then reads an input record.
5. Starting with line 1 of the Input Specification sheet, and with the record just read, the object program uses the record identification code to identify the record. When the identification code matches an entry on the input sheet, the object program turns on the resulting indicator that has been specified for the record. When a control-field break occurs, appropriate control-level indicators are turned on.
6. Next, all total calculations are performed. (This step is bypassed for the initial control break which is caused by reading the first input record.)
7. Next, all total output lines which are not conditioned on overflow are prepared and printed. (This step is also bypassed for the first control break.)
8. The object program tests for the last record indicator. If it is on, the program branches to item 14.
9. The object program tests for an overflow condition. If an overflow condition has occurred, the program branches to item 15.
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 Specification sheet.



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

12. Program execution stops.

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 stops.

15. If overflow has occurred, any lines conditioned by overflow are printed. If no overflow printing has been specified, an automatic skip to channel 1 occurs. The program then branches to item 10.

● Figure 26. General Logic Flow of an RPG Object Program

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 program writing.

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 sheets. 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 (Form X21-9011) or a direct-access storage-device layout chart (Form X24-1711) 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 27 shows an example of the printer spacing chart (Form X24-6436).

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

LINE DESCRIPTION	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
GLUE															
=====															
1															
2															
3															
4															
5															
6															
7															
8															
9															
10															
11															
12															
13															
14															
15															
16															
17															
18															
19															

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 X's 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.

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

Figure 27. Printer Spacing Chart

RPG SPECIFICATIONS SHEETS

This section provides detailed explanations of each specification field in the five RPG coding forms.

CROSS-REFERENCES

To make this reference manual a more effective learning tool, numerous crossreferences are placed for readers not familiar with disk-storage processing and related functions.

Disk storage, table lookup, matching field, and 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, are intended for 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.

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 RPG Specification Sheets.

GENERAL INFORMATION

The forms are listed in the sequence in which they are discussed in this publication.

- Input Specifications
- Calculation Specifications
- Output-Format Specifications
- File Description Specifications
- File Extension Specifications

Figure 28 shows the five specifications sheets.

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS
 IBM System/360 Form X24-3347
 Printed in U. S. A.

Date _____
 Program _____
 Programmer _____

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

Line	Form Type	Filename	File Type	File Designation	Sequence	File Format	Block Length	Record Length	LF	Mode of Processing	Length of Record Address Field	Record Address Type	Type of File Organization	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Name of Label Exit	Extent Exit for DAM	Comments
3																					
4																					
5																					
6																					
7																					
8																					
9																					
10																					
11																					
12																					
13																					
14																					
15																					

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR FILE EXTENSION SPECIFICATIONS
 IBM System/360 Form X24-3348
 Printed in U. S. A.

Date _____
 Program _____
 Programmer _____

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

Line	Form Type	From Filename	To Filename	Table Name	Number of Table Entries Per Table	Length of Table Entry	Table Name (Alternating Table)	Length of Table Entry	Comments
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS
 IBM System/360 Form X24-3350
 Printed in U. S. A.

Date _____
 Program _____
 Programmer _____

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

Line	Form Type	Filename	Record Identification Codes	Field Location	Field Name	Field Indicators	Sterling Sign Position
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS
 IBM System/360 Form X24-3351
 Printed in U. S. A.

Date _____
 Program _____
 Programmer _____

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

Line	Form Type	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Resulting Indicators	Comments
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS
 IBM System/360 Form X24-3352
 Printed in U. S. A.

Date _____
 Program _____
 Programmer _____

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

Line	Form Type	Filename	Output Indicators	Field Name	Constant or Edit Word	Sterling Sign Position
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						

● Figure 28. RPG Specification Sheets

INPUT SPECIFICATIONS: This form 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.

CALCULATION SPECIFICATIONS: This form specifies the operations to be performed upon the input data and upon data obtained as the result of previous calculations.

OUTPUT-FORMAT SPECIFICATIONS: This form 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.

FILE DESCRIPTION SPECIFICATIONS: This form provides additional information about input and output files that is not included on the input or output sheets.

FILE EXTENSION SPECIFICATIONS: This form provides additional information about files used by the object program, such as tables, chaining files, and record-address files.

COMMON FIELDS

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

PAGE NUMBER (COLUMNS 1-2)

Each specification page of the source program must be numbered. The pages are numbered beginning with 01 for the file-description form, and continuing in the following sequence:

- File Extension
- Input
- Calculation
- Output-Format

LINE NUMBER (COLUMNS 3-5)

Each specification line must be identified by a line number. The first two digits of the line number are preprinted on the form. The third position (column 5) is used when, after all specifications have been written on the form, 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 column 5).

The cards must be in numeric sequence by columns 1-5 when they are read by RPG. This means that 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 (COLUMN 6)

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

- I - Input
- C - Calculation
- O - Output-Format
- F - File Description
- E - File Extension

COMMENTS (COLUMN 7)

This feature enables the programmer to insert an identifying comment in the specification sheets. 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 column 7. During the generation of the object program, the asterisk in column 7 identifies the comment so that it is not considered a specification.

PROGRAM IDENTIFICATION (COLUMNS 75-80)

This entry identifies the specification cards for a particular program or for a specific section of a large program.

are utilized in the RPG program. Therefore, it is important that the first seven (or five) characters provide a unique filename.

Numeric Entries

If there is more than one record type within an input file, the records may appear in some predetermined sequence. To process the input data records in a predetermined sequence, the input records are specified on the Input Specification sheet in the same sequence in which they are to be read by the object program. For example, assume that the input records appear as shown in Figure 31. The Name Record is assigned 01 because it is the first type. The Street Record is the second type. The Street Record is assigned 02. The City-State Record is assigned 03, and the Item Number Record is assigned 04. (In this case, there may be several items-number records). If the input records do not appear in this predetermined sequence, program processing stops (information is out of sequence).

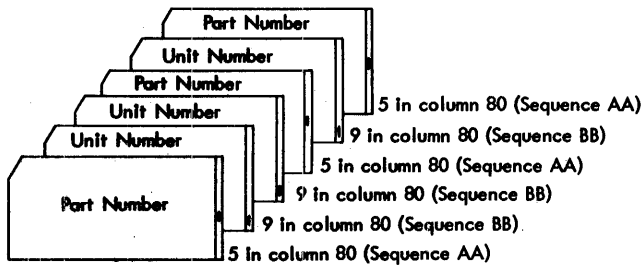
SEQUENCE (COLUMNS 15-16)

This entry designates the sequence of input record-types.

Alphabetic Entries

If the input records are not required to be in a predetermined sequence within a control group, or if it is not necessary to terminate processing when the input records are not in sequence, the entry in columns 15-16 should be a two-character alphabetic code.

In Figure 30, the input data records, which contain part numbers and unit number, do not appear in a predetermined sequence within a control group. Consequently, each input record-type is assigned an alphabetic code because the order of processing is not important to the program.



Line	Form Type	Filename	Sequence Number (1-9)	Option (0)	Resulting Indicator	Record Identifier		
						Position	Not (N) C/Z/D Character	Position
1	I	ILISTING	AA	1	4	8	D	5
2	I							
3	I		BB	1	5	8	D	9

● Figure 30. Input Data Records

- The entries in a predetermined sequence must be in ascending order in columns 15-16.
- Header cards or other cards that are not in sequence must be specified on the Input Specification sheet before specifications that are to be sequence-checked. Enter alphabetic sequence record-types first, followed by numeric-sequence record-types for each file.
- A maximum of 99 record-types is allowed.

NUMBER (COLUMN 17)

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

The entries for Number (column 17) are:

1 - Only one record of a type may be present in a control group.

N - One or more records of a type may be present in a control group.

Blank - The records do not appear in a determined sequence.

As shown in Figure 32, there can only be one Name Record, but there can be multiple Item Number Records.

OPTION (COLUMN 18)

This specification is used only with numeric-sequenced record-types. If a specific record-type must be present to perform an operation, leave this column blank. If the presence of a record is optional, enter the letter O in column 18.

In Figure 32, the O (alphabetic) in column 18 indicates there may or may not be an Item Number for each group. In this example, Name, Street, and City-State records must be present.

RESULTING INDICATOR (COLUMNS 19-20)

The entry in these columns is used in conjunction with the next entry Record Identification Codes. This entry serves two functions:

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

To understand the first function of this entry, assume that an input record is identified by all of the following codes:

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

By assigning one two-digit resulting-indicator code to represent all these codes, it is easier to refer to this input record during the processing of the calculation and output specifications.

As an example of the second function of this entry, resulting indicators may be compared to selectors in punched-card machines or to internal and external switches on electronic data processing machines. The use of resulting indicators (like the use of selectors and switches) permits certain operations to occur only on specific conditions.

In Figure 32, the four record-types have been assigned resulting indicators 12, 13, 14, and 15. When one of the records is present, the appropriate resulting indicator is turned on, and those specifications pertaining to the record are performed. Specifications associated with other record-types are not performed.

Resulting indicators are turned on and off during the processing of the object program, as the various record-types are read by the system. However, only one resulting indicator (for a record-type) can be on at one time. When a resulting indicator is turned on, all other resulting indicators are turned off. There is one exception to this condition (see Chaining).

Other indicator conditions that can be established in the program are field indicators (columns 65-70 of the Input Specification sheet) and resulting indicators (columns 54-59 of the Calculation Specification sheet). 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 from 01-99. The indicator codes can be assigned in any sequence. For example, four different record-types could be assigned codes 40, 62, 98, and 02.

RECORD-IDENTIFICATION CODES (COLUMNS 21-41)

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

These columns provide for the entry of one to three identifying codes as indicated by the numbers 1, 2, and 3 on the Input Specification sheet. It is possible to specify more than three 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 resulting indicator and sequence number must still be specified.

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

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

Position (Columns 21-24)

Enter in this column (or columns) the position in each data record of the character that contains the identifying code. The position must be right-justified in columns 21-24, and leading zeros may be omitted.

Not (Column 25)

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

C/Z/D (Column 26)

The object program 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 column 26 specifies whether the object program is to compare against the entire character (C), against only the zone portion (Z), or against only the digit (D). Enter a C, Z or D in column 26.

ZONE RECORD IDENTIFICATION: Testing a zone means that the zone of the character specified in columns 21-24 of the input sheet is used to identify the record. The ampersand, the minus, and the blank are exceptions. An ampersand will be identified as an 11-zone, and a blank will be identified as a no-zone. The four common zones are:

1. 12-zone or plus zone (0, A-I, and &).
+
2. 11-zone or minus zone (0, J-R, and minus).
-
3. 0-zone (S-Z).
4. No zone (0-9 and Blank).

When you use a blank, ampersand, or minus in the character portion of a zone test, other characters which contain the same machine zone (e.g. ?, \$, or %, respectively) will not satisfy the zone test.

Character (Column 27)

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

Examples of Record-Identification Codes

Examples of record-identification specifications are shown in Figure 33. The explanation of each entry is given here.



INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR INPUT SPECIFICATION
 IBM System/360

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic						
	Punch						

Line	Form Type	Filename	Sequence Number (1-N)	Option (O)	Resulting Indicator	Record Identification Codes									Field Location		File
						1			2			3			From	To	
						Position	Nat (N) C/Z/D	Character	Position	Nat (N) C/Z/D	Character	Position	Nat (N) C/Z/D	Character			
01	I				1	48	Z	K									
02	I				2	48	N	Z	K								
03	I				3	62	D	5									
04	I				4	49	C	T									
05	I				5	16	C	4	40	N	C	5	80	Z	A		

● Figure 33. Record Identification Codes

1. This entry specifies an 11-zone in record position 48. Any of the letters J-R, or the minus sign, could be placed in Character because they all contain an 11-zone. A Z must be placed in column 26 so that only the zone portion is checked. If a C were placed in this column, the object program would compare the letter K against the input-record code instead of a 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 column 16. Column 40 must not contain the number 5, and column 80 must contain a 12-zone. This is known as an AND relationship.

The last example illustrates the way to specify more than one code on more than one line. If an input record contained five different codes, they could be specified as shown in Figure 34. This is also known as an AND relationship. It implies that the card is identified by:



INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR INPUT SPECIFICATION
 IBM System/360

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic						
	Punch						

Line	Form Type	Filename	Sequence Number (1-N)	Option (O)	Resulting Indicator	Record Identification Codes												Field Location		File
						1				2				3				From	To	
						Position	Char (N)	C/Z/D	Character	Position	Char (N)	C/Z/D	Character	Position	Char (N)	C/Z/D	Character			
10	1	PAYREPT	AA		26	16	C4	40	NC5	80	ZK									
20	1		AND			20	C2	25	C3											
30	1																			

Figure 34. Specifying More Than Three Identification Codes

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

The use of this facility and its advantages will be easier to understand after the explanation of Records in an OR-Relationship.

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

OR Relationship

Three types of OR relationships are possible:

1. One or more record-types have the same fields in the same positions of the record. (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 these record-types in an OR relationship only if there are more fields that are alike than fields that are unlike.

Omitting Record Identification

When input records are to be processed alike without regard for their identifying codes, columns 21-41 may be left blank; however, sequence (columns 15-16) and a resulting indicator (columns 19-20) must be specified.

When only certain record-types within a group are to be read and processed, they must be 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 resulting indicator specified for them. Columns 21-41 of the Input Specification sheet are left blank.

Note: If the records are to be bypassed, they should not be referred to on the calculation or output sheets.

STACKER SELECT (COLUMN 42)

If the input record type is from cards that are to be stacker-selected, the number of the stacker into which the card is to be selected is entered in column 42. Leave column 42 blank if the cards are to go to the normal pocket.

The stacker pockets and their acceptable codes are:

IBM 1442	
Pocket	Code
1	1, or blank
2	2

IBM 2540	
Pocket	Code
Normal	Blank
1	1
2	2

The stacker-select entry should be made on the same line with the record identification. If different records in an OR relationship are to be stacker-selected, enter the codes on the same line with the record identification. Stacker select must be specified on each line in an OR-relationship if it is desired.

COMBINED FILES STACKER SELECT

When there are combined files to be stacker-selected, the user can determine the stacker pocket into which the cards will fall. The Stacker Select entries (column 42 of the Input Specification sheet and column 16 of the Output-Format Specification sheet) must be utilized in one of the following ways when using an IBM 1442 Card Read-Punch:

1. If stacker pocket 1 is selected, enter 1 in the appropriate column of either sheet. The remaining entry can be 1 or blank.
2. If stacker pocket 2 is selected, enter 2 in the appropriate column of either sheet. The remaining entry can be 1, 2, or blank.

Note: When there is to be punched input and punched output, stacker pocket 2 always takes precedence; i.e., it can never be overridden.

When an IBM 2540 Card Read-Punch is used, and the combined files are to be stacker selected, the last entry specified always takes precedence.

FIELD DESCRIPTION ENTRIES

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

On the record identification portion of the Input Specification sheet, one line represents the specifications for one record-type. On the field-description portion of the sheet, 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 Specification sheet. Columns 7-42 of the line must be blank.

PACKED (COLUMN 43)

Enter a P in this column if the input field is in the packed decimal format. Otherwise, leave this column blank.

Note 1: If a P is entered in this column and the input field specified is blank, an arithmetic check and an abnormal end-of-job will occur.

Note 2: If the field is in packed format, RPG will compute the unpacked field length using the following formula:

$$[(TO\ Position - FROM\ Position) * 2] - 1$$

FIELD LOCATION

Columns 44-51 of the Input Specification sheet 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 alphanumeric field is 256 characters.

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 column 53. The field name may be alphameric.

Two input files may have fields with the same field names. A field name is assigned only once by RPG. Two input files will 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 Specification sheet.

Certain restrictions apply to the selection of field names. The following names can only be used in certain instances, and cannot be used at any time other than that specified.

1. T A B _ _ _ . These three characters can only be used as a prefix to the name of a table.
2. P A G E _ _ . This prefix can only be used in reference to page numbering.
3. A L T S E Q. This name cannot be used as a field name.
4. C O N T D _ . This name can only be used in conjunction with the conversion routine.
5. I N n n _ _ . This name can only be used as an RLABL referencing an indicator.

Specifying the Same Data Field as Alphameric and 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.

Specifying Constants

The term constant is frequently used for 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 to get this information to the program is by defining a special record type and by assigning a field name that is not otherwise used. If the constants must be available before the first data record of a file is processed, it may be necessary to define an additional file. This technique will also permit entering of constants that are too large to be specified as literals in the source program.

Figure 37 illustrates field names that are easy to read and suggest the function of the fields they represent.

Field Location		Field Name	Control Level (L-19)	Matching Fields or Chaining Fields	Field-Record Relation	Field Indicators			Sterling Sign Position																						
From	To					Plus	Minus	Zero or Blank																							
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	4		0																											
	5	8																													
	9	14																													
	15	28																													
	29	31																													
	33	40		0																											
	41	48																													
	49	55																													
	56	60		0																											
	61	69																													
	70	74																													
	76	79		0																											
	28	31																													
	15	20		0																											
	50	52																													

● Figure 37. Field Names

CONTROL LEVEL (COLUMNS 59-60)

As the object program is processed, a change in a control field causes all processing indicated by this change to be initiated. Columns 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 L1, L2, L3, ... L9. An indicator similar to the resulting indicator is associated with each control-level designation. They are used to control functions specified on the calculation and output specification sheets. When the field specified in Field Location is a control field, its appropriate control level must be specified in columns 59-60. Figure 37 shows the entries for three control fields. (The control fields could have been listed on the sheet in any sequence.) When more than one file has control levels specified, the overall length of each control field must be the same length for all record types.

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 main 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 for arithmetic calculations).

For example, it may be necessary to use a field for controlling functions but without considering zone bits in the comparison of the record to the next record. If the zone bits are not to be used in a comparison, specify the field as numeric by making a decimal entry in column 52. If the zone bits are to be used in a comparison, leave column 52 blank. In Figure 37, L3 in columns 59-60 causes the field DIVNO to be placed into storage in one area with zone-bits removed from all positions of the field to be used for control functions and to be placed in another area with zone bits removed from all positions of the field except the units position. The RPG program

automatically performs the operations of storing the field twice and performing the appropriate zone elimination. No additional specifications need be written by the programmer.

SPLIT CONTROL FIELDS: Several fields in an input record can be specified as one control field. In the lower half of Figure 37, three fields, which are not in continuous record positions, 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 sheet 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 Specification sheet.

USING SPLIT CONTROL FIELDS WITH FIELD RECORD RELATION

Several rules must be observed when split control fields (columns 59-60) are specified in conjunction with Field-Record Relation (columns 63-64).

1. For each indicator, the sum of the length of the split fields must be equal to the length of the combined control fields as originally defined for a particular control level. In Figure 38, the sum of the lengths of FLDA and FLDB (L1 for blank indicator) must be equal to the sum of the lengths of FLDA1 and FLDB1 (L1 for indicator 01).
2. Within the specifications for a given indicator (in the Field-Record Relation columns), the control levels must be in ascending sequence. In Figure 38, the specifications for indicator 01 are entered in ascending sequence (L1, L1, and L4).
3. The split control levels specified with no entry in Field-Record Relation (blank resulting indicator) must be specified first. In Figure 38, the fields FLDA-FLDE (blank in columns 63-64) are entered before the remaining entries are specified.

Matching Fields

If a field specified in Field Location also has an entry in Matching Fields, the field is placed into another main storage area known as a matching-field holding area.

Up to three matching fields (designated by M1-M3) are allowed. This entry may be used to match records in different sequential files. The second sample program at the back of 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 may be found in Processing Multiple Input Files.

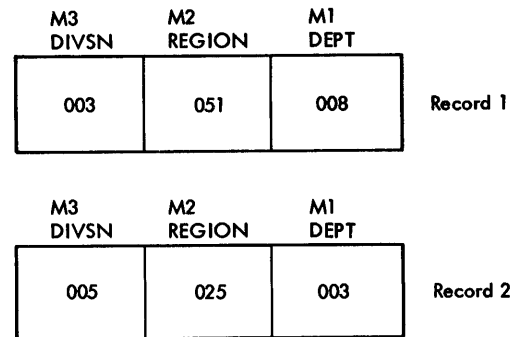
USING THE MATCHING FIELDS SPECIFICATION FOR SEQUENCE-CHECKING: If only one input file is specified, fields within the file may be sequence-checked by using the matching-fields specification. Up to three fields within the file may be checked. A chaining file may also be sequence-checked. If the chaining field is to be sequence-checked, it must be defined twice, using two different field names. In Figure 39, 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 holding area as shown in Figure 40. 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.

Data Contained in First Record

DEPT 008
REGION 051
DIVSN 003

Data Contained in Second Record

DEPT 003
REGION 025
DIVSN 005



● Figure 40. Comparing Matching Fields

In column 18 of the File Description Specification sheet the programmer must specify if the file is in ascending or descending sequence. In Figure 39 assume that the file has been specified in ascending sequence. The number 003051008 is lower than 005025003. Thus, the file is in ascending sequence.

EXIT TO A TRANSLATE SUBROUTINE

If the sequence of the matching fields is not the same as the collating sequence of the 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 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 ALT-SEQ. The register conventions for this subroutine are the same as those for the EXIT operation code. (See Exit to a Subroutine.) The address of the hold area containing the matching fields to be translated will be contained in Register 1. The subroutine must place the translated fields back into the matching-field holding area before it returns control to RPG.

CORPORATION Form X24-3350
Printed in U. S. A.

INPUT SPECIFICATIONS

)

										Page <table border="1" style="display: inline-table; width: 20px;"><tr><td>1</td><td>2</td></tr></table>	1	2	Program Identification <table border="1" style="display: inline-table; width: 100px;"><tr><td>75</td><td>76</td><td>77</td><td>78</td><td>79</td><td>80</td></tr></table>			75	76	77	78	79	80
1	2																				
75	76	77	78	79	80																

Packed (P)	Field Location		Field Name	Control Level (1-19)	Matching Fields or Chaining Fields	Field-Record Relation	Field Indicators			Sterling Sign Position						
	From	To					Plus	Minus	Zero or Blank							
	43-44	45-46					47-48	49-50	51-52		53-54	55-56	57-58	59-60	61-62	63-64
	1	10	NAME													
	11	15	NUM													
	17	19	DEPT		M1											
	20	22	REGION		M2											
	23	25	DIVSON		M3											

● Figure 39. Using Matching Fields in a Single Input File

The original data fields (to be used for controlling, printing, arithmetic calculations, etc.) are not translated by the subroutine.

Chaining Fields

The use of chaining files is explained in the section Chaining. Up to three chaining fields are permitted in a record. In these columns, enter the code that identifies the chaining field C1, C2, or C3.

FIELD-RECORD RELATION (COLUMNS 63-64)

This specification is used to specify records in an OR-relationship when the field records are not in the same location. Enter in columns 63-64 the appropriate resulting indicator which will be on when the field is used. An explanation of the use of this specification was contained in Records in an OR-relationship, and an example of this specification is in Figure 36.

Using Field-Record Relation with Chaining Files

These columns (63-64) can be used to condition chaining to a specific file. If the indicator entered in columns 63-64 is on when the chaining technique is used, the object program will then get the record from the designated file. When this column is blank, the chained record will be obtained whenever the specified record-type is present.

For example, if MR is entered in Field-Record Relation, the field will be used as a chaining field only if the matching record indicator is on.

FIELD INDICATORS (COLUMNS 65-70)

Entries in columns 65-70 are used to test the status of a field when it is read into the system. This entry is an indicator that will be turned on when the field specified on the line is plus, minus, or zero or blank.

Plus (Columns 65-66)

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

Minus (Columns 67-68)

A minus condition occurs when the value of the field is less than 0.

Zero or Blank (Columns 69-70)

A zero condition occurs if a field contains all zeros or all blanks.

Types of Indicator Codes for Plus, Minus, And Zero or Blank

If the status of a field is tested so that calculation and output specifications can be modified, a two-digit field indicator is used in the appropriate columns (65-66, 67-68, or 69-70). These codes, from 01-99, can be defined one or more times on the Input Specification sheet. If the same indicator is defined on more than one specification, the last specification executed will determine the status of that indicator.

Using Figure 37, if the input-field BALNCE is a plus number, field-indicator 01 will be turned on. Indicator 01 will remain on until a new BALNCE is read in and tested. If FIELD2 is a minus number, field-indicator 02 will be turned on. If AMTDUE is zero or blank, field-indicator 03 will be turned on.

Note: The indicator that is turned on remains on until the next time the field is tested by this particular specification.

Note that defining these indicators means that they have been entered in appropriate columns of the Input Specification sheet. Using these indicators means that they have been specified on the calculation or output sheets and that they are tested to determine their status. The field indicators will be discussed with the output and calculation sheets.

HALT INDICATORS

Nine additional indicator codes, known as halt indicators, can be used in the RPG program. These indicators, designated as H1 through H9, halt the processing of the object program when error conditions (as determined by the programmer) have been detected. These indicators can also be used in calculation and output specification sheets 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 will terminate the object program.

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

STERLING (COLUMNS 71-74)

Enter in these columns the position in the record that contains the sign of the Sterling field. If the sign is in the normal

position, enter an S in column 74. Leading zeros may be omitted. Leave these columns blank if the Sterling Specification is not used. Additional information on Sterling is in the section Sterling Routines for the Report Program Generator at the end of this publication.

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 Insert Files. The input specifications listed are used with the calculation and output forms. 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:

| Resulting Indicator
 Field Name
 Control Level
 Matching Fields
 Field Indicators.

2. What is the operation (add, subtract, multiply, etc.)?
3. What is done with the result?

3. A single quote symbol may be contained within a literal by entering an additional single quote symbol within the literal. For example, the literal o'clock would be coded as 'o'clock'.

FACTOR 1 (COLUMNS 18-27)

This specification can be a field name or a literal. If it is a field name it must have been defined on the Input Specification sheet, or it must have been defined in the result field of another calculation. The field name must be left-justified, and the first character must be either alphabetic or @, #, or \$.

Literals

A literal is the actual data to be operated on, rather than a name representing the location of data. Literals may be numeric or alphabetic. Numeric literals may be up to ten characters long. Alphameric literals may be up to eight characters long.

NUMERIC LITERALS: A numeric literal can consist of any combination of the numbers 0-9. One decimal symbol and/or one plus sign or one minus sign may also be used.

Rules for Forming Numeric Literals:

1. Blanks must not appear within a numeric literal.
2. The sign, if present, must be the left-most character. If a literal is unsigned, it is treated as a positive literal.
3. The decimal point can appear anywhere in the literal.
4. Numeric literals must not be enclosed in quote symbols.

ALPHAMERIC LITERALS: Any set of consecutive characters enclosed in a set of single-quote symbols is treated as an alphameric literal. Alphameric literals may be used for compare operations, but they may not be used in computations.

Rules for Forming Alphameric Literals:

1. Any character may be used in an alphameric literal. Blanks are treated as valid characters in the body of the literal.
2. Alphameric literals must be enclosed in a set of single-quote symbols.

Figure 44 illustrates entries for Factor 1.

INTERNATIONAL BUSINESS MACH																																							
REPORT PROGRAM GENERATOR CA																																							
IBM System/360																																							
																				Punching Instruction					Graphic														
																									Punch														
Indicators											Factor 1									Operation					Factor 2														
And		And																																					
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
							GROSS														1																		
							NETAMT														2																		
							12500.00														3																		
							'JANUARY'														4																		

● Figure 44. Factor 1

1. GROSS could be a field name specified on the input sheet.
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 decimal position must be indicated (if the number is not a whole number) but commas must not be used.
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, only upon records representing the month of January.

The discussion of Factor 1 also applies to Factor 2.

SUMMARY OF FACTOR 1 AND FACTOR 2

1. Enter either the name or the literal that is Factor 1 or Factor 2 in columns 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. Columns 53-58 (Field Name) of the Input Specification sheet.
 - b. Columns 43-48 (Result Field) of the Calculation Specification sheet.
3. A name cannot exceed six characters. Special characters and blanks must not be used.
 4. A literal cannot exceed ten characters.
 5. Entries in Factor 1 or Factor 2 must be left-justified.

OPERATION (COLUMNS 28-32)

Entries in these columns specify the operation to be performed using Factor 1, Factor 2, and the Result Field. Each operation is specified by placing the operation code in the operation field (Columns 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-Leftmost-Data-Characters	MOVEL
Move Low-to-High Zone	MLHZO
Move High-to-Low Zone	MHLZO
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 Go To)	GOTO
Providing a Label for GOTO ..	TAG
Exit to a Subroutine	EXIT
RPG Label	RLABL
User Label	ULABL

Turning Indicators ON or OFF	Code
Set Indicators On	SETON
Set Indicators Off	SETOF

Table Operations	Code
Table Lookup	LOKUP

Conversion Routine Operations	Code
External Conversion Routine ..	EXTCV
Record Key	KEYCV

ARITHMETIC OPERATIONS

The fields or literals involved in this operation may contain numeric characters only. All arithmetic operations are performed with automatic decimal alignment. 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.

ADD (ADD)

This operation causes the contents of the field or the literal in Factor 2 to be added, algebraically, to the contents of the field or literal in Factor 1. The result of the operation is placed in the result field specified in columns 43-48.

ZERO AND ADD (Z-ADD)

This operation causes the result field to be set to zeros and 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.

SUBTRACT (SUB)

This operation causes the contents of the field or literal in Factor 2 to be subtracted, algebraically, 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.

MULTIPLY (MULT)

This operation causes the contents of the field or literal in Factor 1 to be multiplied algebraically 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. 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 will result. Any remainder that results from this operation is lost unless the move-remainder operation is specified as the next operation.

Note: If a move remainder operation follows a divide operation, the result in the divide operation cannot be half-adjusted.

In this operation, RPG performs the adjustment of fields (divisor and dividend) when necessary. The divisor and dividend are adjusted by adding zeros to the right of the units position of that field. The following formula determines whether or not field adjustment is necessary:

$$A = (\text{Number decimal positions of Result Field}) + (\text{Number decimal positions of Factor 2}) - (\text{Number decimal positions of Factor 1})$$

If $A=0$, no adjustment takes place.

If $A>0$, the dividend will be adjusted.

If $A<0$, the divisor will be adjusted.

The amount of adjustment (the number of zeros added to the right of the units position) is determined by the absolute value of A in the preceding formula.

MOVE REMAINDER (MVR)

This operation is used to move the remainder from a divide operation to a separate field that has been set to zero by the RPG processor. If MVR is used, it must follow the divide operation. Figure 45 shows an example of the MVR operation. The remainder is placed in a field named STORE. The field that contains the remainder must be specified in the result field.

The value of the remainder can be determined by the following formula:

$$R = \text{Dividend} - (\text{Divisor}) (\text{Quotient})$$

The decimal positions of the remainder following the DIV operation are equal to the dividend decimal length (adjusted).

Note: A move remainder operation can specify resulting indicators.

MOVE OPERATIONS

For the MOVE and MOVEL operations, unpacked numeric fields may be changed to alphameric fields, and packed alphameric fields may be changed to numeric fields. To change a numeric field to an alphameric field, Factor 2 is numeric, and the result field is specified as alphameric. To change an alphameric field to a numeric field, Factor 2 is alphameric, and the result field is specified as numeric. No decimal alignment is performed when a move operation is used. When moving an alphameric field to a numeric field, a validity check is not performed on the sign of the result field.

Note: A move operation cannot specify resulting indicators.

IBM		INTERNATIONAL BUSINESS MACHINES CORPORATION	
REPORT PROGRAM GENERATOR CALCULATION SPECIFICAT			
IBM System/360			
Date			
Program			
Programmer			
		Punching Instruction	Graphic
		Punch	

Line	Form Type	Control Level (U, L, R)	Indicators					Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions Half Adjust (H)																																								
			Z	V	M	C	D																																														
														And	And																																						
3	4	5	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				
01	0	C			1	2																																															
02	0	C			1	2																																															
03	0	C																																																			

● Figure 45. Using the MVR Operation Code

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.

If Factor 2 is longer than the result field, the excess leftmost positions of Factor 2 are not moved.

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.

Factor 1 is not used in this operation.

MOVE LEFT (MOVE L)

This operation code causes (starting at the leftmost position) data characters to be moved from the field or literal contained in Factor 2 to the leftmost positions of the result field.

If Factor 2 is longer than the result field, the excess rightmost positions of Factor 2 are not moved.

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 sign of the result field is retained, except when Factor 2 is as long or longer than the result field. In this case, the sign of Factor 2 is assumed.

Factor 1 is not referenced in this operation.

Figure 46 shows a use of this operation code. FLDA contains a five-position number (10000). FLDB is a four-position field. The result of the MOVE L is 1000 in FLDB. In this example, the MOVE L operation code performs the same function as dividing FLDA by 10.

Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Position Half Adjust (H)	Resulting Indicators			Compare	Comm
						Plus	Minus	Zero		
						High	Low	Equal		
						1 > 2	1 < 2	1 = 2		

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									

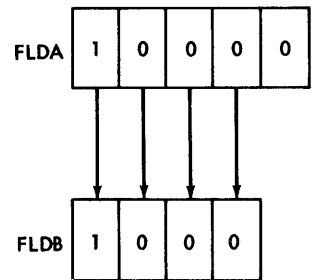


Figure 46. Using the MOVE L Operation Code

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. 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 on the Input Specification sheet. The result field can be numeric or alphameric. A result field specified as numeric always contains the standard plus or minus zone after this operation. Figure 47 illustrates this operation.

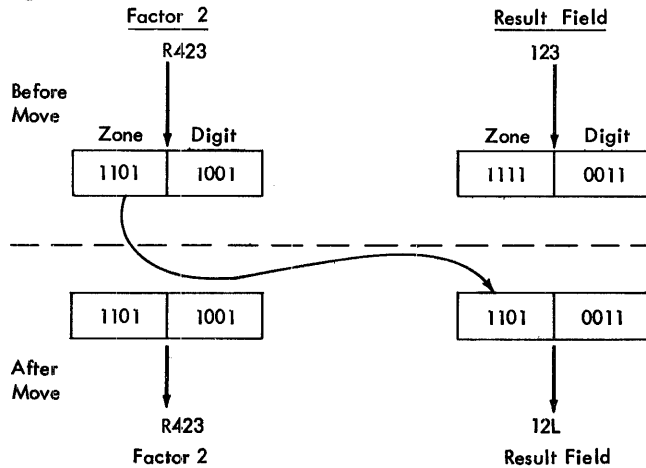


Figure 47. Move High-to-Low Zone Operation

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.

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.

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 always contains the standard plus or minus zone after this operation.

Figure 48 illustrates the movement of zones for all of the preceding four operation codes.

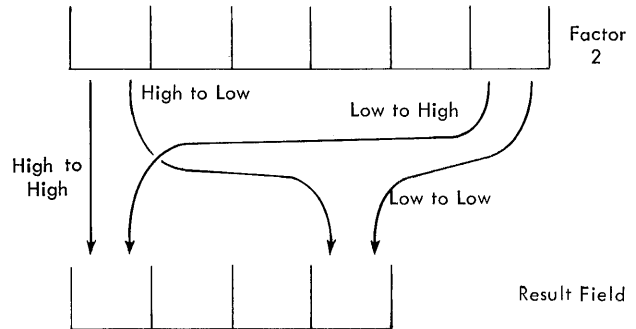


Figure 48. Move Zone Operation

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 turn on an indicator that has been specified in columns 54-59 (Result Indicators High, Low, or Equal). These indicators are turned on as follows:

- High Factor 1 is greater than Factor 2.
- Low Factor 1 is less than Factor 2.
- 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 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.

is identified by a name. This name is entered in Factor 2 and the code GOTO is entered in Operation (columns 28-31). For example, a routine (a group of specifications) 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 contains the name FICA and that the first operation in the routine be a TAG operation; that is, an operation that defines the name of the routine (refer to description of next operation code).

Branching can be performed within detail calculations or within total calculations, but not between detail and total calculations. Branching can be forward (skipping over specifications) or backward (going back to specifications previously skipped or processed). (For additional information, see Using the Calculation Specification sheet.)

PROVIDING A LABEL FOR GOTO (TAG)

The operation TAG (columns 28-30) provides a name to which the program can branch. Enter this name in Factor 1 and the code TAG in operation (columns 28-32). The name will be used as Factor 2 of the operation code GOTO. If the specifications in the TAG routine occur at total time, then the TAG operation must have L0 specified in Control Level. If they occur at detail time, a control specification is not required.

TURNING INDICATORS ON OR OFF

SET INDICATORS ON (SETON)

This operation code causes the indicators specified in columns 54-55, 56-57, or 58-59 to be turned on.

Note: The column headings of plus, minus, or zero and high, low, or equal have no meaning during this operation, and should be ignored. Columns 54 through 59 are used for this operation code merely to record three sets of indicator codes.

Specify the first indicator in columns 54-55, the second indicator in columns 56-57, and the third indicator in columns 58-59. One use of this specification is to turn on a halt indicator when input records are out of sequence. Any RPG indicator may be turned on. Figure 50 shows an example of this facility. The 01 is an indicator that is set on for the first record of a sequence. The L3 is a control level that occurs with the first record of the sequence. If such a situation (L3 and 01) does not occur, the halt indicator H1 is turned on.

SET INDICATORS OFF (SETOF)

This operation code causes the indicators specified in columns 54-55, 56-57, or 58-59 to be turned off. Any RPG indicators may be turned off. When an L3 break occurs, L1-L3 are turned on. If the user did not want L1 on, he could have turned it off by using the SETOF operation code.

Line		Form Type	Control Level (U, L, D, B)	Indicators		Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Resulting Indicators			Comments
				And	And						Plus	Minus	Zero or Blank		
				Not	Not						High	Low	Equal		
											1 > 2	1 < 2	1 = 2		
01	0	C		L3	N01		SETON					H1			
02	0	C													

● Figure 50. Using SETON for a Record Out of Sequence

TABLE OPERATIONS

TABLE LOOKUP (LOKUP)

This operation code causes the field name contained in Factor 1 to be used as the search argument in a table lookup operation. Factor 2 contains the name of the table to be searched, and the result field contains the table 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 user wants to determine if an argument is present in the table, but does not require the corresponding function.)

After the lookup operation is completed, the function that is retrieved is placed in a special holding area within the function table. The name of this area is the same as the name of the function table. The retrieved data remains in this area until the next table lookup operation.

To utilize the function in another operation (for example in an arithmetic 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 or Factor 2 refers to the special holding 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 holding area. In this case, the function table name in the result field refers to the location of the function within the function table and within the special holding area.

After each table lookup operation, the retrieved function should be used (or moved from the special holding 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 in the Object Program for additional information and examples of table lookup operations.

CONVERSION ROUTINE OPERATIONS

EXTERNAL CONVERSION ROUTINES (EXTCV)

This operation is used to indicate the point in the RPG program where the random address conversion routine is to be performed. Factor 1 contains the internal RPG label as specified in columns 27-32 of the File Extension Specification sheet. Factor 2 must contain the label of the external address conversion routine. The result field contains the name of the field that will contain the track address. The track-address length is always eight characters.

Note: All EXTCV operations must be specified as the initial entry(s) on the Calculation Specification sheet. Up to 4 of these operations are permitted.

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 (columns 28-32) and the name of the key field is placed in Result Field (columns 43-48). The field length and decimal positions must be specified.

If the records are retrieved by key field, this operation must follow its EXTCV operation. The name of the field that will contain the record key cannot be declared in the external conversion routine.

FACTOR 2 (COLUMNS 33-42)

The discussion of Factor 1 also applies to Factor 2.

RESULT FIELD (COLUMNS 43-48)

The name of the result field, which is entered in these columns, sets up a location in storage to contain the result of a calculation. The name of the result field must consist of alphameric characters, or @, #, or \$, and must be left-justified.

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.



INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS
 IBM System/360

Form X24-3351
 Printed in U. S. A.

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic Punch								
----------------------	---------------	--	--	--	--	--	--	--	--

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (UL, L, BL)	Indicators			Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions Half Adjust (H)	Resulting Indicators			Comments
			And	And	No							Plus	Minus	Zero or Blank	
01	C					GROSS	MULT	DRATE	DISCNT	02	H				
02	C					GROSS	SUB	DISCNT	NETAMT	02					
03	C					DISCNT	ADD	TOTDIS	TOTDIS	02					
04	C														

● Figure 51. Result Field Entry

Figure 51 shows the result-field entry. On the first line GROSS is multiplied by DRATE, and the result field is DISCNT. DISCNT is then used as Factor 2 on the next line to calculate the net amount. The same result field is then used as Factor 1 on the next line to calculate total discount.

FIELD LENGTH (COLUMNS 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 51, DISCNT is eight positions long. The unpacked length must be specified. The maximum numeric field length is 15 digits, and the maximum alphanumeric field length is 256 characters.

If the same field name is used for more than one calculation and the field length and the number of decimal positions are the same, the field-length specification need not be specified more than once.

DECIMAL POSITION (COLUMN 52)

An entry in this column indicates the number of positions to the right of the decimal point in the result field. An entry must be made in this column for all arithmetic operations if the field has not been defined previously. If the result field does not have any decimal positions, the entry must be a 0. A maximum of nine decimal positions can be specified. In Figure 51 each result field has two decimal positions.

If the result field is alphanumeric, this column must be left blank.

HALF-ADJUST (COLUMN 53)

This specification is used to half-adjust, or round, the result field. Enter an H in this column if the data in the result field is to be half-adjusted.

Half-adjusting is accomplished in the object program by adding a 5 to the right of the last position retained in the result field.

In Figure 51, DISCNT is half-adjusted.

This completes the description of the specifications required for determining the kind of calculations to be performed.

TESTING THE RESULTS OF A CALCULATION

The last category of the Calculation Specification sheet is used to test the results of the calculation.

RESULTING INDICATORS (COLUMNS 54-59)

An entry in columns 54-59 may be used to test the value of a result field after the completion of an operation. As a result of the test, an indicator may be turned on that can be used to control subsequent operations or to control output operations.

	The result is placed back in SAVE, and it is half-adjusted.	130	The contents of HOLD are com- pared with the literal 144.00, and if HOLD is less than, or equal to, 144.00, indicator 21 is turned on.
060	RATE is multiplied by 40, and the result is stored in GROSS. This operation is performed whether or not the employee worked any overtime. It is not performed if the employee has worked less than 40 hours.	140	If indicator 21 is on, the program branches to the label ADFICA.
070	GROSS is added to SAVE.	150	YDFICA is subtracted from the literal 144.00 and the result is placed in DFICA.
080	The program branches to the label FICA, if indicator 22 is off.	160	The operation <u>TAG</u> provides the label ADFICA to which the pro- gram can branch (either from the specification on line 140 or sequentially from the speci- fication on line 180).
090	Additional calculations.	170	DFICA is added to YDFICA and the result is stored in YDFICA.
100	The operation code TAG provides the label FICA. The RPG program branches to this label.	180	Additional calculations.
110	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.	190	The operation code <u>TAG</u> provides the label WHTAX to which the RPG program can branch.
120	DFICA is added to YDFICA and the result is placed in the field HOLD.		

OUTPUT-FORMAT SPECIFICATION SHEET

Entries on the Output-Format Specification sheet specify:

1. The kind of output that will be produced.
2. The location of the specific data fields in the output reports and records.

The specifications for this sheet can be divided into two categories as shown in Figure 54: file identification and control, and field description.

FILE IDENTIFICATION: These entries specify the output files and the records to be written on the file.

FIELD DESCRIPTION: These entries indicate the position of each field in the output record. Each field is written on a separate line of each sheet, and each field is written below its corresponding record entry.

FILE IDENTIFICATION AND CONTROL

FILE NAME (COLUMNS 7-14)

A file name is assigned to each output file. The file name must be left-justified, and it must begin with an

alphabetic character. The file name may be alphanumeric, but it must not contain special characters or blanks.

When writing the output specifications, the file name need only be entered once. Enter it on the first line to define the file as shown in Figure 55.

Line	Form Type	Filename	Type (W/D/T)	Space	Skip	Output Indicators			Field Name	Zero Suppress (Z) Blank After (B)
						Stacker Select	Before	After		
01	O	DAILYRPTH	201			OF				Heading Line
02	O		OR			1P				
03	O									
04	O		D	1		24				Detail Line
05	O									
06	O		T	12		L1				Level-1 Total
07	O									
08	O		T	22		LR				Final Total
09	O									

● Figure 55. Specifying Heading, Detail, and Total Lines

Line	Form Type	Filename	Type (W/D/T)	Space	Skip	Output Indicators			Field Name	Zero Suppress (Z) Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
						Stacker Select	Before	After						
01	O													
02	O													
03	O													
04	O													
05	O													

● Figure 54. Output-Format Specification Sheet

TYPE H/D/T (COLUMN 15)

The entry in this column identifies the type of record being specified. The following three entries are used for this specification:

- H Heading Record
- D Detail Record
- T Total Record

All heading records for the file must be entered first, followed by all detail records, and then by all total records (see Figure 55, column 15).

HEADING RECORDS: These records usually contain constant information. However, they 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.

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 total records that result from that control change. But heading and detail records can contain data from the input record.

STACKER SELECT (COLUMN 16)

If punched output occurs in the object program, a stacker number is entered in this column. The cards fall in the stacker that is indicated by this column. The stacker pockets and their acceptable codes are listed below:

IBM 1442	
Pocket	Code
1	1 or blank
2	2

IBM 2540	
Pocket	Code
Normal	1
4	4 or blank
8	8

Note: When punched output is specified, the last card will not be put out automatically.

For information concerning the stacker selection of combined files, refer to the section Combined Files Stacker Select.

SPACE (COLUMNS 17-18)

This specification (and the next specification Skip, columns 19-22) is used to provide the proper spacing of printed reports.

Note: If the record is to be printed, at least one entry is required in columns 17-22.

Space Before (Column 17)

Enter in this column the number of lines to be spaced before the line is printed. Specify zero, one, two, or three spaces before printing by placing the entry 0, 1, 2, or 3 in column 17.

Space After (Column 18)

Specify zero, one, two, or three spaces after printing by placing the entry 0, 1, 2, or 3 in column 18. Enter a zero in this column to specify no space after printing.

SKIP (COLUMNS 19-22)

This specification provides for the proper spacing of reports. It is directly related to the function of the printer carriage-control tape.

Skip Before (Columns 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.

Skip After (Columns 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. In Figure 55, before the heading line is written, the form skips to channel 1.

Note: The order in which spacing and skipping is performed is as follows:

- Skip Before
- Space Before
- Skip After
- Space After

If both Space After and Skip After are specified, only Skip After is performed.

Overflow Indicator

Carriage overflow, causes the setting of the indicator as specified by the user in the file-description specifications (Columns 33-34).

When a punch in channel 12 of the control tape is sensed the overflow indicator is turned on. It remains on for one complete processing cycle; it turns off after the heading and detail lines of the next record are printed. Because the overflow indicator is on during calculations, it can control calculation specifications.

A test for the overflow status is made by the object program 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 executed, 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, and then any overflow printing specified 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, and this is followed by any specified overflow printing.

Automatic Skipping

If an overflow indicator is not specified on the File Description Specification sheet for a file (and the Skip or Space columns on the Output-Format Specification sheet contain an entry) the RPG program provides automatic skipping from channel 12 to channel 1 for the object program.

If an overflow indicator is specified on the File Description Specification sheet for a file, and none of the records of the file are conditioned by overflow on the Output-Format Specification sheet, automatic skipping will also occur.

Printing Lines Conditioned by Overflow

If an output heading line is coded as in the upper half of Figure 56, the heading line will print whenever the OF or L1 indicator is on. If L1 and OF occur at the same time during processing, the line prints twice, thus possibly providing an error in the printed report. To prevent this situation the line is coded as shown in the lower half of Figure 56. By making the condition mutually exclusive, the line is not printed twice.

If more than one indicator is specified on one line, all indicators are considered in and AND relationship. That is, all conditions specified must be satisfied before the output condition can be executed.

If the object program requires that more than three indicators be in and AND relationship, the letters AND are entered in columns 14-16 of the following line, and the additional indicators are specified on that line. If one of the indicators is an overflow indicator (OA, OF, or OV), it must not appear on the same line as the AND.

If the output condition is executed in an OR-relationship (one or the other of two sets of conditions) the letters OR are entered in columns 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 either an AND or OR relationship as required by the object program. Each additional line must begin with AND or OR in column 14.

Permitted Entries for Output Indicators,
Columns 24-25, 27-28, and/or 30-31

Enter in these columns:

1. Resulting indicators (Columns 19-20 of

the Input Specification sheet or columns 54-59 of the Calculation Specification sheet).

2. Special indicators (LR, H1, MR, etc).
3. Field indicators (columns 65-70 of the Input Specification sheet).

Note: When an output record is not conditioned by any resulting indicators, the record is conditioned 00.

NOT (COLUMNS 23, 26, AND/OR 29)

If any of the output indicators must be off, enter an N in column 23, 26, or 29 (whichever is appropriate).

EXAMPLES OF OUTPUT INDICATORS

Figure 57 shows six examples of output indicators used with the output file. The numbers to the right refer to the following discussion:

In this example, the salesman field is group indicated.

2. The second example illustrates how to prevent the printing of just one field of a record. The field AMOUNT is printed only if indicator 16 is off. This indicator might be a resulting indicator 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 customer and invoice fields on succeeding forms are printed.

Printing of the entire line is controlled by indicators 04 or OF. In the field-description specifications, the OF indicator is also used to prevent printing of fields 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.

The field DIVSON is controlled by three AND indicators: 16, NH2, and NL3. The field 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 columns 14-15 of a field-description line.

FIELD NAME (COLUMNS 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 order in which they appear in the output record is determined by the entry in columns 40-43.

Enter in columns 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 sheet. If a constant is to be written, it is specified in Constant or Edit Word (columns 45-70), and Field Name is left blank.

If the value of the field is numeric, the sign of the field will appear in the units position unless the field is edited,

zero suppressed, or the zone of the field has been blanked out on the Calculation Specification sheet by a numeric literal.

Examples of field names are given in Figure 58.

ZERO SUPPRESS (COLUMN 38)

An entry of Z in this column causes zero-suppression of the field in the output record. Zero-suppression means that the zeros to the left of the significant digits in the field are not written on the output file. Fields to be zero suppressed must be specified as numeric.

An entry of Z in this column also provides another function. Zone bits in the units position of a field are removed in the output record before the field is written.

If an edit-control word is specified for the field, this column must be blank.

BLANK AFTER (COLUMN 39)

If a B is entered in this column, the output field will be reset to blanks or zeros after the field is moved to the output area. Alphameric fields are set to blanks, and numeric fields are set to zeros.

This specification has an additional feature. It allows the programmer to provide for resetting field indicators or resulting indicators that test for blanks or zero. If the output field being reset by the blank-after specification, is also being tested for zero or blank on the input specifications, or for zero on the calculation specifications, the corresponding indicator is turned on after the output field is reset. If multiple Zero or Blank indicators are specified for the same field name, only the last indicator specified will be turned on by the Blank After specification.

END POSITION IN OUTPUT RECORD (COLUMNS 40-43)

This specification indicates the exact location of the field in the output record. In columns 40-43, enter only the position in the output record where the rightmost (low-order) character of the field is to be located.

column 45 will be placed on the output record as defined in End Position in Output Record, columns 40-43.

Rules for Forming Alphameric Literals in an Output Record:

1. Any character in the character set may be used in an alphameric literal. Blanks are treated as valid characters.
2. Alphameric literals must be enclosed in a set of single-quote symbols. A single quote symbol may be contained within a literal by entering an additional single quote symbol within the literal. For example, the literal 5 o'clock would be entered as '5 o'clock' in columns 45-56 of the Output-Format Specification sheet.
3. Field Name (Columns 32-37) must be left blank when an alphameric literal is defined on the line.

Edit Words

An edit word provides for the punctuation of amount fields, including the printing of dollar signs, commas, periods, and sign status. An edit operation is shown in Figure 61.
IN STORAGE

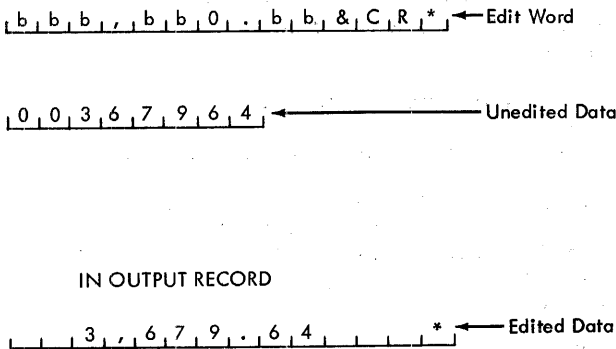


Figure 61. Edit Operation

When an amount field is to be edited, its edit word is placed in columns 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 62.

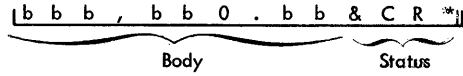


Figure 62. Two Parts of an Edit Word: Body and Status

The body of the edit word is the portion beginning with the leftmost blank, zero, or asterisk, and continuing to the right to the character that governs the transfer of the units position of the data field.

The status of the edit word is the portion continuing to the right from the body to the CR (credit), or - (minus) if present.

Rules for Forming an Edit Word:

1. An edit word must be enclosed in a set of single-quote symbols.
2. A blank in the edit word is replaced with the character from the corresponding position of the data field specified in Field Name.
3. An ampersand causes a space in the edited field.
4. A zero is used for zero-suppression. It is put in the rightmost position where zero suppression is to stop. It is replaced with the character from the corresponding position of the data field, unless that character is a zero. Column 38 (zero-suppression) must be left blank.
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 stop. It is replaced with the character from the corresponding position of the data field unless that character is a zero. 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. A dollar sign that is entered immediately after the initial single-quote mark will be fixed. That is, it will be printed in the same location each time. This is called the fixed dollar sign.

Field Name	Zero Suppress (Z) Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	VALUE IN DATA FIELD	CONTAINED IN OUTPUT RECORD AS:																																	
							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
①		96		'26.75'		26.75																																	
②		96		'DEPARTMENT TOTAL'		DEPARTMENT TOTAL																																	
③	AMOUNT	96		' , 0. &CR*'	000030 46-	30.46 CR*																																	
④	AMOUNT	96		' , \$0. CR*'	00030 46+	\$30.46 *																																	
⑤	AMOUNT	96		' , * CR**'	000030 46-	*****30.46CR**																																	
⑥	DEPT	96			001234	00123D																																	
⑦	DEPT	96		' '	001234	1234																																	
⑧	DEPT	96		' 0 '	001234	001234																																	
⑨	DEPT	Z 96			001234	1234																																	

● Figure 63. Using Constants, Edit Words, and Zero Suppression

- The decimal and commas are printed in the edited output field in the same relative positions 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.
- All other characters used in the body of the edit word are printed if they are to the right of significant digits in the data field. If they are to the left of high-order significant digits in the data word, they are blanked out, or if asterisk protection is used, they are replaced by an asterisk.
- The letters CR or the minus symbol in the status portion of the edit word are undisturbed if the sign in the data field is minus. If the sign is plus, or if the value of the field is minus zero, they are blanked out.
- Asterisks to the right of the status portion of the edit word are undisturbed. They are normally used to indicate a specific class of total.
- The constant 26.75 is in the output record ending in position 96. The field-name specification must be blank.
- The constant DEPARTMENT TOTAL is contained in the output record ending in position 96. The field name must be blank.
- This example illustrates zero-suppression to the left of significant digits. The letters CR are written because the amount field might be minus.
- In this example, the floating dollar-sign protection enters the \$ to the left of the first significant digit.
- 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.
- DEPT is a numeric field for which no editing or zero suppression is specified. The output field will contain a zone in the units position.
- An edit word containing blanks only will suppress all high-order zeros in the output field.
- The edit word has been specified as one

digit longer than DEPT, with a zero in the high-order position. This allows all zeros in DEPT to print, but the zone of the units position has been stripped.

9. This line notes the use of zero suppression. The output field will contain no high-order zeros as in line 7.

STERLING (COLUMNS 71-74).

Enter in these columns the position in the record that will contain the sign of the sterling field. Leading zeros may be omitted. Enter an S in column 74 if the sign is in the normal position. If the Sterling specification is not required, leave this column blank. Additional information on Sterling is in Appendix C: Sterling Routines for the Report Program Generator.

FILE DESCRIPTION SPECIFICATION SHEET

This sheet provides information to RPG about:

1. The input files, defined on the Input Specification sheet, from which the object program will obtain data records. A maximum of three input files may be specified, (excluding RAF files, table files, and chaining files).
2. The input files used by the object program, such as record address files, and table files.
3. The output files are defined on the Output-Format Specification sheet on which the object program will write data records. Each file used by the object program will be defined. Each line of the File Description Specification sheet is used to define one file.

MAXIMUM NUMBER OF FILES AVAILABLE

The maximum number of files that can be used in the program is 10. The following list 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
Primary	1
Secondary	2
RAF	1
Chained	3
Table	4
Output	
Printer	3
Other Output Units	9

$\sum \leq 10$

on the Output-Format Specification sheet (for output data files), or on the File Extension Specification sheet.

Note: The filename may be eight characters long but RPG will utilize only the first seven characters for all files except Indexed-Sequential files. For Indexed-Sequential files only the first five characters of the filename will be utilized in the RPG Program. Therefore, it is important that the first seven (or five) characters provide a unique filename.

FILE TYPE (COLUMN 15)

An entry in this column specifies the type of file defined on the line of the sheet. The following entries are allowed in this column:

Entry	Type of File	Explanation
I	Input	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	Identifies the file as an output file (it may be an updated table file to be written out). Column 39 must be left blank.

FILE NAME (COLUMNS 7-14)

Each file used in the program is identified by writing the name of the file in columns 7-14. File names must be alphameric and left-justified (the file name must begin in column 7). The file name entered in these columns must also be entered on the Input Specification sheet (for input data files),

U Update Identifies the file as an update file [DASD 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 without changing:

1. The nature of the data,
2. The length of the field in which the data is found, or
3. The location of the field.

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

C Combined Identifies the file as a combined file. A combined file is a card file containing cards read into the system and cards used for punching. It must reside on an IBM 1442 Card Read-Punch, or an IBM 2540 Card Read-Punch equipped with the Punch-Feed-Read Special Feature. 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.

A combined file differs from an update file because in the update file the input fields are revised or updated.

Figure 64 illustrates two examples of File Name and File Type specifications.

1. Detail cards are read into the system in one file, summary cards are punched into a separate 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.

IBM		INTERNATIONAL BUSINESS																																				
		REPORT PROGRAM GENERATOR																																				
		IBM Sy																																				
Date _____		Punching Instruction _____																																				
Program _____		Graphic Punch _____																																				
Programmer _____																																						
Line	Form Type	Filename	File Type															Mode of Processing																				
			File Designation															Length of Record Address Field																				
			End of File															Record Address Type																				
			Sequence															Type of File Organization																				
			File Format															Overflow Indicator																				
			I/O/U/C	P/S/C/R/T	E	A/D	F/V	Block Length	Record Length	L/R	K/T	L/D/T	Key Field Starting Location																									
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			
01	P	INPUT																																				
02	P	SUMCARD																																				
03	P	PRINT																																				
04	P	INPUTPRI																																				
05	P	INPUTSEC																																				
06	P	PRINT																																				

• Figure 64. Specifying File Name and File Type

FILE DESIGNATION (COLUMN 16)

Column 16 designates the type of input file being defined on this line of the File Description Specification sheet. The five entries permitted in this column 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 Program and Processing Multiple Input Files.

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

Entry	Explanation
P	The file defined is a primary file. <u>Only one primary file may be defined. at least one MUST be specified</u>
S	The file defined is a secondary file.
R	The file defined is a Record Address File which relates to a direct access storage file.
C	The file defined is a chained file (as defined under <u>Chaining</u>).
T	The file defined is a table file.

END OF FILE (COLUMN 17)

Enter an E in column 17 if the input file is a sequential file and it 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 turn on when the last record of the file has been processed.

For multiple input files, the end-of-job condition (LR) will occur when all the input files (defined on this sheet with an E in column 17) have reached the end-of-file.

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

SEQUENCE (COLUMN 18)

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

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

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, and M3) in columns 61-62 of the Input Specification sheet, sequence-checking with respect to these fields occurs. In column 18 either A or D must be entered to specify whether the file is ascending or descending.

FILE FORMAT (COLUMN 19)

This column is used to indicate the format of the input or the output records. Enter an F in this column if the records are fixed-length. Enter a V in this column if the records are variable in length. For more information, see the Variable-Length Records section.

not blocking factor
BLOCK LENGTH (COLUMNS 20-23)

This specification is used to indicate the block length of the input or output records. If the input records are unblocked, enter the length of the largest record in columns 20-23. If the input record is blocked, enter the size of the largest block. The block length may be from 18 to 4096. Leading zeros may be omitted.

RECORD LENGTH (COLUMNS 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.)

Note 1: If the block length equals the record length, RPG will consider 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 will consider the file to have blocked records for both fixed-length and variable-length records.

MODE OF FILE PROCESSING (COLUMN 28): DIRECT ACCESS STORAGE DEVICE (DASD)

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

<u>Entry</u>	<u>Explanation</u>
L	Enter an <u>L</u> in this column, if a segment of the file is to be processed. The upper and lower limits of the file (to be processed) will be supplied, in this case, by a Record Address File (RAF). The RAF will be supplied by the user.
R	Enter an <u>R</u> in this column, if the user's records are to be processed randomly. In this case, the records to be processed will be obtained by a record-address file or a chaining file.
Blank	If no entry is made in this column for the file, the entire file will be processed sequentially.

LENGTH OF RECORD ADDRESS FIELD (COLUMNS 29-30): RECORD ADDRESS FILE (RAF) ONLY

If the file defined on this line is a record-address file, enter the number of positions that each entry in the RAF occupies.

TYPE OF RECORD ADDRESS (COLUMN 31): (DASD OR RAF FILE ONLY)

If the records from a DASD file are to be retrieved by using record keys, enter a K in this column. The K indicates that the file defined on this line will be processed by use of the record key. If the records are to be retrieved by the record identification, enter an I in this column. If the file specified on this line is a RAF file, a K or an I must be entered in column 31. Otherwise, leave the column blank.

TYPE OF FILE ORGANIZATION (COLUMN 32): (DASD ONLY)

Leave this column blank if the file is organized sequentially, or if it is an output file. Enter an I if the file has an indexed-sequential organization. Enter a D in this column if the file has direct organization.

Enter a T in this column, if the file is the output from the ADDRUT (Address Output) option of the disk sort program. (See IBM Basic Operating System/360,

Sort/Merge Program (8K Disk), Form C24-3321, for an explanation of the ADDRUT option.) An example of using this option is contained in the section of this manual entitled Using the ADDRUT Option.

RULES FOR SPECIFYING MODE OF FILE PROCESSING, TYPE OF RECORD ADDRESS, AND TYPE OF FILE ORGANIZATION

1. If a direct access storage device is not used in the system, columns 28, 31, and 32 are left blank.
2. If the type of file organization is Sequential, then columns 28, 31, and 32 are again left blank.
3. If the type of file organization is Indexed-Sequential (I in column 32), column 31 must contain a K and column 28 can be either L, R, or blank.
4. If the type of file organization is Direct (D in column 32), then column 31 can contain either a K or I, and column 28 can only contain an R.

Note: If column 32 contains D and column 31 contains K, multiple tracks will be searched by the RPG object program.

Figure 65 illustrates the code combinations possible for these three specifications.

Type of File Organization (Col. 32)	Type of Record Addresses (Col. 31)	Mode of Processing (Col. 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. The limits to be processed are supplied by a Record Address File (RAF). (L)
		The records will be processed randomly. The addresses are supplied: (a) by a RAF file, or (b) by the data contained in the chaining field of an input record. (R)
Direct (Random) (D)	Track Address with Record Key, or (K) Track Address with Record Identification (I)	The records will be processed randomly. Conversion is required. The addresses to be converted are supplied by a RAF file or by a chaining field. (R)

*Entries in parenthesis indicate actual specifications to be used.

● Figure 65. Processing Direct Access Storage Files

OVERFLOW INDICATOR (COLUMNS 33-34)

If the file defined on the line is a printer file and overflow indicators are used, enter the overflow indicator associated with the file. A maximum of three overflow indicators is allowed. The following are permissible overflow indicators:

- 1. OA
- 2. OF
- 3. OV

VI

KEY FIELD STARTING LOCATION (COLUMNS 35-38)

Enter in these columns the starting location within the record of the field specified as the Key Field. (Leading zeros may be omitted.) This entry is used only if the records are blocked records for an Indexed-Sequential file.

E (COLUMN 39)

If the file defined on the line is a chaining file, a table file, or a record-address file, enter E in column 39. The E indicates that additional information about the file will be coded on the File Extension Specification sheet. If the file is defined as an output file (O in column 15), this column is left blank.

DEVICE (COLUMNS 40-46)

Enter in these columns the input or output unit associated with the file. If the output file is a printer, enter PRINTER in columns 40-46. 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:

READ42 for the IBM 1442

READ40 for the IBM 2540

If the file is an input or output file and it is associated with a tape unit, enter TAPE in these columns. If the file is associated with a Direct Access Storage Device (DASD), enter DISK11 in columns 40-46. The entry must be left-justified.

PRINTER

SYMBOLIC DEVICE (COLUMNS 47-52)

If the file defined on this line is an input or an output file, enter the symbolic name of the unit. (For example: SYSIPT in 47-52).

Valid entries in these columns are the following:

Entry	Examples
SYSRDR	Card input.
SYSLST	Printer.
SYSIPT	Card or tape input.
SYSOPT	Card, disk, tape, or printer output.
SYS001 SYS254	Any input or output unit.

Note 1: Input units containing files organized in indexed-sequential or random organization must be assigned Symbolic Device codes of SYS001-SYS254.

Note 2: If the logical file extends over more than one physical unit, the device codes must be adjoining. For example, if a logical file is contained in three IBM 2311's the device codes could be SYS019, SYS020, SYS021.

LABEL (COLUMN 53)

The entries for this specification are:

Blank = No labels exist for the file.

- 1. The RPG processor processes no labels and makes the first data record available.
- 2. RPG writes the first record.

N = The file has nonstandard labels.

- 1. For both input and output files, RPG always provides for an exit to a user's subroutine.
- 2. The user must provide for the checking of header and trailer labels.

PPDHQ

2. The record-address file defined on this line has a fixed format (column 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 column 39 indicates that the File Extension Specification sheet will be used. The record address file is read into the program by an IBM 2540 Card Read-Punch. The Device code is READ40 and the Symbolic Device code is SYS001.
3. The third file defined on this sheet is a table file. The T in column 16 indicates that it is a table file. It has a fixed format, a block length of 200, and a record length of 100. Column 39 (E) indicates that it will be used on the File Extension Specification sheet. The file is read in on magnetic tape, therefore the Device code is TAPE, and the Symbolic Device code is SYS006. The N in column 53 indicates that there are nonstandard labels on the tape. This means that the nonstandard labels will be processed by the user's subroutine LABEXT (as specified in columns 54-59).
4. MASTCUST is an input file that will be processed under the control of another file. It is a chained file (C in column 16). It will be processed randomly (R in column 28). The record addresses that will be referenced by the chaining file are record keys (K in column 31), and the file is organized indexed-sequentially (I in column 32). This file is located on a direct access storage device and is given the Device code of DISK11 and the Symbolic Device code of SYS008. The S in column 53 indicates that there are standard labels on the disk.
5. The update file DISKUPDT (U in column 15) will be used for input, and it will be updated after processing of each record has been completed. It is an indexed-sequential file, and it will be processed randomly. The C in column 16 indicates that the file is a chained file. In this example, the maximum record length is 200 with a maximum block length of 400. This file is located on a direct access storage device and is given the Device code of DISK11 and the Symbolic Device Code of SYS012. The key field starting location (columns 37-38) is position 15 within the data record.
6. The file CARDREC is a combined file (column 15). Assume that the file will be used as input, and additional information will be punched in the input cards during processing. Therefore, it is a Secondary file (Column 16). This combined file is read in and punched out on an IBM 1442 Card Read-Punch. The Device code is specified as READ42 and the Symbolic Device code as SYS002.
7. The file OUTPUT is a printed report in this example. It is variable in length, and the longest record is 132 characters. The overflow indicator for this file is OF (columns 33-34). The printed report in this case has a Device code of PRINTER and a Symbolic Device code of SYSLST.

FILE EXTENSION SPECIFICATION SHEET

Entries made on the File Extension Specification sheet provide information to the RPG about such functions as chaining files, tables used in the object program, and information about record-address files. In the sections Using Tables in the Object Program and Processing Multiple Input Files, detailed information and examples show how to use these functions. A maximum of 25 line entries may be made on the File Extension Specification sheet.

The entries allowed on the File Extension Specification sheet are discussed in the following sections.

RECORD SEQUENCE (COLUMNS 7-8)

Chaining File

If the file defined on the line is a chaining file, enter the sequence of the file in columns 15-16 of the Input Specification sheet. The file name of the chaining file has been entered in columns 7-14 of the Input Specification sheet.

Record-Address File or Table File

Leave columns 70-80 blank if a record-address file or a table file is defined on this line.

NUMBER (COLUMNS 9-10)

Chaining Field

Enter the number of the chaining field in columns 9-10. The number of the chaining field is taken from columns 61-62 of the Input Specification sheet.

Record-Address File (RAF) or Table File

Leave columns 9-10 blank if the file defined on the line is a record-address file or a table file.

FROM FILENAME (COLUMNS 11-18)

Enter in these columns:

1. The name of the chaining file,
2. The name of the RAF, or
3. The name of the table file as defined on the File Description Specification sheet.

Chaining File. This is the file that has the data record containing the chaining field. The name of the file is taken from columns 7-14 of the File Description Specification sheet. The entry must be left-justified.

Record Address File. If the file defined on the line is a record-address file, enter the name of the RAF file in these columns. The entry must be left-justified.

Table File. Enter the name of the file that contains the table if the file defined on the line is a table file. This file has been defined on the File Description Specification sheet. The entry must be left-justified. Table files are processed in the same order in which they appear on the File Extension Specification sheet.

TO FILENAME (COLUMNS 19-26)

Chained File Name. If the file is a chained file, enter its name. The data record is obtained from this file. The entry must be left-justified.

RAF File Name. If the From Filename entry (columns 11-18) is an RAF file, enter the name of the file that contains the data record to be processed. The entry must be left-justified.

Table File Name. If the From Filename entry (columns 11-18) is a table file, enter the name of the file to be used as an output file after the table has been updated during program execution. Leave this column blank if the table file is not to be written out. A table can be put out on only one device. For additional information on tables, see Using Tables in the Object Program. The entry must be left-justified.

TABLE NAME (COLUMNS 27-32).

Columns 27-32 contain the name of either the table that contains arguments or functions. If only one table name (argument or function) is read in on one device, then it is entered in this column, and columns 46-57 are left blank. The name must be of the form TABnnn. The entries nnn may be any alphanumeric characters. (The entry must be left-justified.)

Direct Organization

When a file that has direct organization is processed under control of a record-address file or a chaining file, the entry for these columns is the RPG internal label of the conversion routine. The concepts of chaining and record address files have not been discussed at this point in the publication. For a complete discussion of chaining, conversion, and record-address files, see Processing Multiple Input Files.

NUMBER OF TABLE ENTRIES PER RECORD (COLUMNS 33-35)

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

NUMBER OF TABLE ENTRIES PER TABLE (COLUMNS 36-39)

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

LENGTH OF TABLE ENTRY (COLUMNS 40-42)

Enter in columns 40-42 the length of each table entry. The maximum size of an entry is 248 bytes. The entry must be right-justified.

PACKED (COLUMN 43)

If the data in the table is in the packed-decimal format, enter P in this column. Otherwise, leave this column blank.

DECIMAL POSITIONS (COLUMN 44)

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

SEQUENCE (COLUMN 45)

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

Note: The next five specifications (columns 46-57) are used only if alternating functions and arguments are read in on one device.

TABLE NAME (COLUMNS 46-51)

If two table names are used, enter the second table name in these columns. It must be of the form TABnnn. This specification is used only if alternating arguments and functions are used. The entry must be left-justified.

LENGTH OF TABLE ENTRY (COLUMNS 52-54)

Enter in these columns the length of each table entry. The maximum size of a table entry is 248 bytes. The entry must be right-justified.

PACKED (COLUMN 55)

Enter a P if the data in the table is in the packed-decimal format. Otherwise, leave this column blank.



INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR FILE EXTENSION SPECIFICATIONS
 IBM System/360

Form X24-3348
 Printed in U. S. A.

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic								
	Punch								

Page

1	2
---	---

Program Identification

75	76	77	78	79	80
----	----	----	----	----	----

Line	Form Type	Record Sequence of the Chaining File																Number of Table Entries Per Record										Table Name (Alternating Table)	Length of Table			Comments
		Number of the Chaining Field																Per Table	Length of Table		Packed (P) Decimal Pos. Sequence (A/D)	Packed (P) Decimal Pos. Sequence (A/D)										
		From Filename	To Filename	Table Name	Per Table	Entry	Per Table	Entry	Per Table	Entry	Per Table	Entry	Per Table	Entry																		
01	E	AAC1	INPUT	MASTCUST													10	150	10	N	TABFUN	10	N		1							
02	E		RAFFILE	DISKUPDT																						2						
03	E		TABFIL	TABARG												10	150	10	N							3						
04	E		TABFIL	NEWTAB	TABREC											10	150	10	N							4						
05	E																															

● Figure 67. Entries for the File Extension Sheet

DECIMAL POSITIONS (COLUMN 56)

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

SEQUENCE (COLUMN 57)

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

COLUMNS 58-74

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

ENTRIES ON THE FILE EXTENSION SPECIFICATION SHEET

Figure 67 shows several examples of entries that can be made on the File Extension Specification sheet. 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 Specification sheet. 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 columns 61-62. A complete discussion of chaining may be found in the Chaining section.
2. In this example, RAFFILE is a record-address file that supplies 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 columns 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 (column 44).

The functions in the file are identified by the table name TABFUN. The function table is described in columns 52-57. Each function is 10 characters long, and each function is organized in

the form: argument-function. Therefore, TABARG was specified first.

4. This example shows the specification 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 (card or printer.)

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 columns 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. TABREC is the name used during updating or during a lookup operation on the Calculation Specification sheet.

This section of the publication contains information on:

1. How to create and use tables, and
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 PROGRAM

RPG enables the programmer to use tables in the object program. A table is nothing more than an arrangement of data that is searched and used by the object program. Tables are loaded into storage by the RPG object program before any files are processed. Tables can be loaded from card, tape, or disk (consecutive).

A table may consist of two parts: an argument and a function. In Figure 68 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 68 contains part numbers that have been ordered. The cards do not contain the prices of the parts. The part number is selected from the card by the RPG program, the table is searched, and the price is retrieved and made available for additional processing.

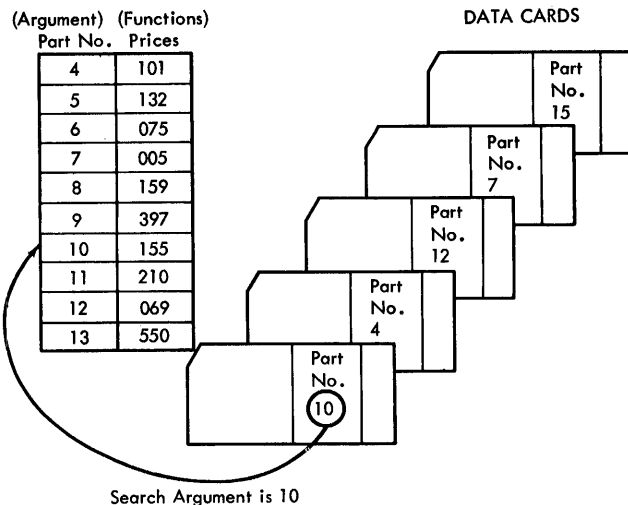


Figure 68. Using a Table

All entries in a table will be:

1. Arguments,
2. Functions,
3. Alternating arguments and functions, or
4. Alternating functions and arguments.

Figure 69 shows these four possibilities.

Arguments

Argument 1	Argument 2	Argument 3	Argument N
------------	------------	------------	------------

Functions

Function 1	Function 2	Function 3	Function N
------------	------------	------------	------------

Alternating Arguments and Functions

Argument 1	Function 1	Arg. 2	ment N	Fun. N
------------	------------	--------	--------	--------

Alternating Functions and Arguments

Function 1	Argument 1	Fun. 2	Function N	Argument N
------------	------------	--------	------------	------------

Figure 69. Four Types of Tables

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 arguments is assigned a name. The collection of all function entries is assigned a name. The names must be unique, and the names

of the tables must contain the letters TABnnn (nnn may be any alphanumeric entry). In Figure 73 the alternating table file called RATE-TABL is split into the collection of arguments (TABNUM) and functions (TABRAT). RATE-TABL is the name of the file containing the tables.

3. Although all the tables may be loaded from the same device, a unique file name must be assigned in columns 11-18 for each line entry on the File Extension Specification sheet. The tables will be loaded into storage before the object program is processed, and each line entry on the File Extension Specification sheet must have:
 - a. A unique file name (columns 11-18).
 - b. Entries in columns 27-45 if the table is only arguments or functions.
 - c. Entries in columns 27-45 and columns 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 70, the first card in the table file has ten table entries. All subsequent card records must have ten table entries. The second card could not contain six; the third could not contain seven, for example.

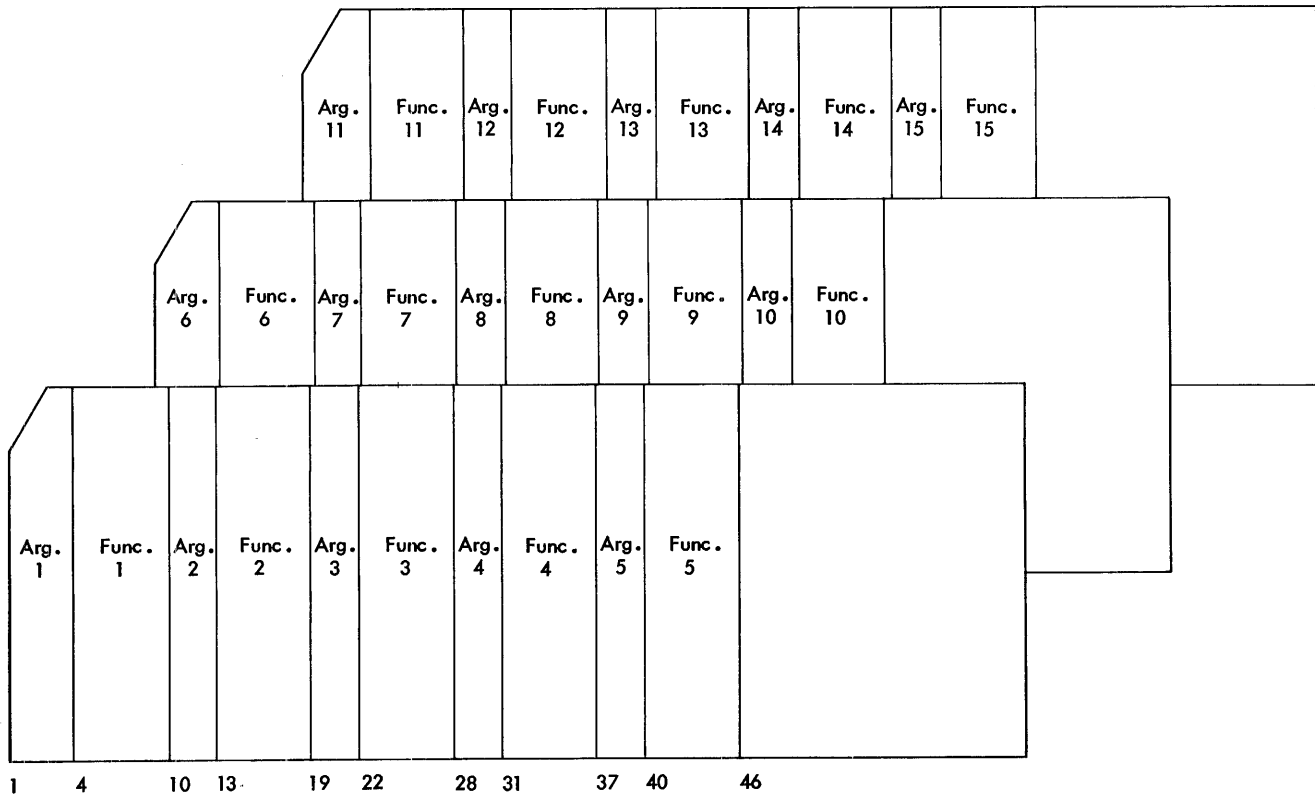


Figure 70. Table File Containing Arguments and Functions

- All entries must be continuous in every record. In Figure 70, the first entry begins in position 1 and the second entry begins in position 4. No blanks can be contained between the table entries.
- All entries belonging to a table must have the same length. In Figure 70, each argument is three positions long and each function is six positions long.
- 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 70.
- 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 other than its corresponding argument.
- 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 71 illustrates the correct way to specify records containing arguments or functions.

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 turned 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.)

If there is an equal compare, indicator 35 is turned on. Since the argument field of the table is vacant, the corresponding function field is also vacant. (It is the user's responsibility to make certain that the vacant table entries contain all zeros or blanks, depending on whether the fields are numeric or alphabetic.) The new argument (NEWARG) is inserted in the TABARG field, and the corresponding new function (NEWFUN) is inserted in the TABFUN field.

RETRIEVING UPDATED TABLES

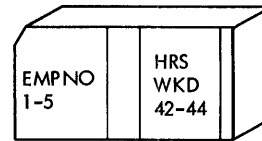
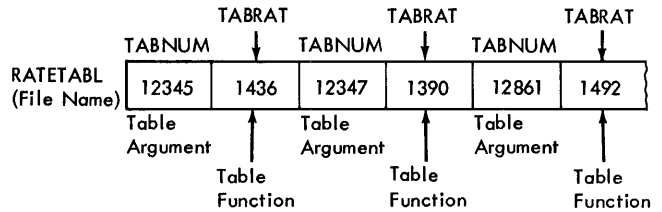
After a table has been updated, the table may be written out for later use. On the File Description Specification sheet, the programmer enters the name of the file that will contain the updated table. The file must be defined as an output file. On the File Extension Specification sheet, the programmer enters the name of the file on which the updated table will be written. He enters the name of the file in To Filename. 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 Specification sheet. 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 output table must have the same format as the input table. When using tape units as output devices for tables in blocked format, each output table file requires a separate tape unit.

EXAMPLE OF USING TABLES

Figures 73 and 74 illustrate an input data file, the way a table might appear, and the entries necessary on the RPG specification sheets in a program that uses tables. In this example, a card-input file contains the number of hours worked by each employee (columns 42-44) and the employee's number (columns 1-5). The RPG program takes the employee number and uses it as the search argument to find the salary rate for the employee. After the employee's rate has been determined, the rate is multiplied by the number of hours worked by the employee. The result of this operation is the amount earned for each employee.



(Input Data)

Figure 73. Using Alternating Arguments and Functions

In this example, the table consists of alternating arguments and functions. The way the table data might appear is shown in Figure 73) The name of the file that contains the arguments and functions is RATE-TABL. The collection of arguments is called TABNUM (the table number), and the collection of functions is called TABRAT (table rate).

Entries on the specification sheets follow.

FILE DESCRIPTION SPECIFICATION SHEET

The two files are defined on the File Description Specification sheet. The file containing the input card records is called TIMECARD. It is an input file (I in column 15); it is the primary file (P in column 16); and when the file is depleted, processing is terminated (E in column 17). The records in the file are in ascending order (A in column 18); they are fixed-in-format

(F in column 19). Each record has a block length of 80 (80 in columns 22-23), and each record is 80 characters long (80 in columns 26-27).

This file is read in on the IBM 2540 Card Read-Punch, so the Device code is READ40, and the Symbolic Device code is given as SYSRDR.

The table file is defined on the line below the card-input file. The name of the file (RATETABL) is entered in Filename (columns 7-14). It is an input file (I in column 15), and the records in the file are fixed in format (F in column 19). The file has a block length of 80, and each record is 80 characters long. The E in column 39 indicates that additional information about the file is coded on the File Extension Specification sheet.

This file is read in on the IBM 1442 Card Read-punch, and therefore it is assigned the Device code of READ42. This is the second card reading device on the system so it is assigned a Symbolic Device code of SYS001.

FILE EXTENSION SPECIFICATION SHEET

On the File Extension Specification sheet, the table file is further defined. The name of the file is entered in From Filename (columns 11-18). The collection of arguments (TABNUM) is entered in the first Table Name (columns 27-32). There are ten arguments per record (columns 34-35), and

there are 1500 entries in the table (columns 36-39). Each table entry is five positions long (5 in column 42), and there are no decimal positions (N in column 44). The table is ascending (A in column 45).

The collection of functions is described in columns 46-57. The name of the functions (TABRAT) is entered in columns 46-51. Each entry in the table is four positions long (4 in column 54), and there are three decimal positions specified (3 in column 56). The functions are in ascending order.

INPUT SPECIFICATION SHEET

The input file (TIMECARD) is described on the Input Specification sheet. The name of the file is entered in columns 7-14. The file is assigned a sequence of AA (columns 15-16), and resulting indicator 01 is turned 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 columns 1-5 of the input record, as specified by the entries in Field Location (columns 47 and 51), and the employee number is given the field name EMPNUM. The number of hours worked by the employee is found in columns 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.

CALCULATION SPECIFICATION SHEET

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 columns 28-32 causes the lookup operation to be performed. Factor 2 contains the name of the collection of arguments (TABNUM) which is searched by the search argument. The result field contains the name of the collection of functions (TABRAT). 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 columns 58-59 indicates that indicator 03 will be turned 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 is 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 EARN\$ 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. Columns 9-11 contain the specification N03.

The literal +000.00 is then moved to the field EARN\$, specifying that the employee does not have an entry in the table.

EXIT TO A USER'S ROUTINE

GENERAL INFORMATION

By use of the EXIT operation code on the Calculation Specification sheet, 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) or it might be a routine that performs a function not easily accomplished using RPG (such as a square-root computation). The subroutine, written in the Assembly language, is coded by the user, and entries made on the calculation sheet enable the programmer to:

1. Exit from the RPG program to the subroutine,

2. Execute the subroutine, and
3. Return to the main program after the subroutine has performed its function. Each subroutine must be assembled as a separate module.

HOW TO CODE EXIT

On the Calculation Specification sheet, the EXIT operation can be a conditional operation. 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. Columns 28-31 must contain the operation code EXIT, and Factor 2 must contain the label of the user's subroutine.

POSITION OF EXIT IN THE CALCULATION SPECIFICATION ENTRIES

The following results are obtained depending on the location of the EXIT code on the Calculation Specification sheet.

Calculation Entry	When the EXIT Will Occur
1. First Detail	At the end of the data routine (after the data is extracted from the input record).
2. Last Detail	Immediately before <i>detail</i> heading records are written.
3. First Total	At the end of the input routine (after the record-type has been determined and the control-field break has been tested).
4. 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 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.

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 the operation-code columns.
2. The label of the field in Result Field.
3. The length of the field in Field Length.
4. The decimal length in Decimal Position, if applicable.

If the user has defined a field in the RPG program and this field is to be used in the subroutine to which the EXIT will occur, he must code:

1. RLABL in the operation code.
2. The label of the field in Result Field. *in subroutine → EXTRN label*

When executing the subroutine, the user may need to reference the resulting indicators that are used in the RPG portion of this program. To do this, the user must code:

1. RLABL in the operation code columns,
2. INnn in the 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 the result field.

Indicators defined in the subroutine cannot be used or referenced in the basic operating system RPG program.

USE OF REGISTERS

The way in which registers are used by the programmer is strictly defined. These rules must be followed:

1. The register that contains the base address of the subroutine is register 15. The first instruction of the programmer's subroutine must load the base address of the subroutine in register 15.

Note: The base register for an exit to a user's subroutine for label checking is register 8, and not 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.
5. Before the subroutine transfers back to the RPG program, the registers must be restored to their original contents.
6. The general registers available for the user's purposes are registers 2, 8, 9, 10, and 11. Registers 3-7 can be used only if no RLABL's are specified in the user's subroutine. The remaining registers cannot be used.

USING INDICATORS AND FIELDS IN THE EXIT ROUTINE

If in the exit subroutine, the user sets on, sets off, or tests resulting indicators, he must observe the following rules:

1. To set on a resulting indicator, set the data located at INnn to the hexadecimal F0.
2. To set off a resulting indicator, set the data located at INnn to the hexadecimal 00.
3. To test resulting indicators:
 - a. If on, the data at INnn, will be the hexadecimal F0.
 - b. If off, the data at INnn will be the hexadecimal 00.

If numeric data from the RPG object program is used in the subroutine, it will be in the packed-decimal format. If the numeric data from the subroutine is supplied to the RPG object program, it must be in the packed-decimal format.

Figure 75 shows the coding steps necessary to implement the EXIT routine.

1. An input file of the name INPUT turns on resulting indicator 01 if an X is in position 80.
2. If the field AMOUNT is blank, field indicator 02 is turned on.
3. Whenever indicator 01 is on, the calculation specifications entry EXIT causes the program to exit to the user's subroutine called TAXROUTE.

The operation preceding EXIT defines (and sets off) an indicator that can be used in the subroutine.

4. Within the subroutine, the user wants three things: the AMOUNT field, and the indicators 02 and 14. RLABL enables the subroutine to reference the AMOUNT field, and IN02 enables the

subroutine to test indicator 02. IN14 enables the subroutine to utilize indicator 14 in the subroutine.

5. 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 turns on indicator 14. TAXAMT is referenced in the output specifications, and 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 enables the field TAXAMT to be referenced by the RPG program.
6. On the output sheet, TAXAMT is treated as a field.

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 a 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 2540 Card Read-Punch.

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 do 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 the 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 thousands of 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 disks. 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, the fixed length records would make inefficient use of file storage space. For example, a fixed record length of 850 positions would waste many storage positions, 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 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 interrecord 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 basic operating system RPG. The length of a variable-length block is

indicated by a block-length character-count field present in each block. (See Variable-Length Records).

The Basic Operating System handles the blocking and deblocking of records so the user need only design the most efficient blocking factor for his particular data file and equipment specifications. In the Basic 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 two types of file organization are: sequential and random.

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 provides 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. Another variation of sequential organization is known as Indexed-Sequential organization.

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 Basic Operating System) are analogous to the index card file in a library. If you know the name of a book, or the author, you can look in the index and find the location of the book in the book files.

For example, this might be an address (Catalog Number) of 426.25. You would then go to the book shelves, and (if it was your first time in the library) start at the first row of the book files and proceed through the rows until you 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. This, as you proceed through the rows, you would compare 426.25 with the numbers posted on each row. Assume that one row was labeled 300.88-550.00. You 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, you would scan the shelf numbers until you found something like "424.00-440.96". Then you would look at individual book numbers on that shelf until you found "426.25".

The RPG program uses index tables in much the same way to locate records organized in an Indexed-Sequential file.

RANDOM ORGANIZATION

In random 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.

FILE PROCESSING

For the three methods of file organization (sequential, indexed-sequential, and random) there are three methods of file processing:

1. Consecutive processing of sequentially organized files.

2. Sequential processing of indexed-sequentially organized files.
3. Random processing of indexed-sequentially organized files and randomly organized files.

CONSECUTIVE PROCESSING

In consecutive 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 14th card in the file could not be processed until after the 13th card had been processed.

SEQUENTIAL PROCESSING

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 the data is stored in the file. The data file could be organized in either a random 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 (there is no conversion).

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).

Random Files

To find a random record in a random 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 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. The program then searches the specified track for the record with the corresponding record key.

If an address is assigned to more than one record, these duplicate addresses (synonyms) will not be handled by RPG.

FILE PROCESSING IN RPG

RPG object programs process the following input file organizations:

1. Sequential.

2. Indexed-Sequential.

3. Direct.

This applies to update files as well. RPG will create sequential output files only.

SEQUENTIAL ORGANIZATION

RPG object programs process sequential files by processing the entire file from the beginning to the end. The records may be fixed or variable length and blocked or unblocked.

INDEXED SEQUENTIAL ORGANIZATION (DIRECT ACCESS STORAGE DEVICE)

RPG processes Indexed-Sequential files in three ways:

1. Sequentially, by processing the entire file.
2. Sequentially, by processing between the limits of the file.
3. Randomly, by processing records on the file in a random order.

The records may be fixed-length and blocked or unblocked.

DIRECT FILES (DIRECT ACCESS STORAGE DEVICE)

A direct file will be processed randomly. The records must be fixed-length and unblocked. The two major sections (Processing Single Input Files, and Multiple Input Files) describe disk storage processing in the RPG program. In these sections, processing methods are described in detail including specific examples of methods and of coding specifications.

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 column 16 of the File Description Specification sheet.

SEQUENTIAL ORGANIZATION

Sequential files may be considered as "tape-like". 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. At this point, an end-of-file condition occurs, which can be interrogated by RPG by specifying an E in column 17 of the File Description Specification sheet. In the case of DASD, the extent of the file is obtained from the operating system.

Figure 76 shows the coding on the File Description Specification sheet necessary to process a sequential file. The name of the file is MASTERIN. The records are fixed in length, and the block length is 200. Each record is 200 characters long. A blank in column 32 indicates that the file is a sequential file.

INDEXED-SEQUENTIAL ORGANIZATION

An indexed-sequential file can only be on a direct-access-storage device (such as disk or drum). For this organization method, an index table associated with the file contains a record key for every track of the file and the number of the track. For ascending files, the index table contains the highest record key on the track. For descending files, the index table contains the lowest record key on the track. An I must be entered in column 32.

For this organization, there are three methods of processing:

1. Processing the entire file sequentially.
2. Processing part of the file sequentially.
3. Processing the file randomly.

PROCESSING THE ENTIRE FILE SEQUENTIALLY

The object program obtains the limits of the file from the operating system, and the entire file is processed sequentially by record key in ascending or descending record-key sequence. Column 28 of the File Description Specification sheet must be blank.

PROCESSING PART OF THE FILE SEQUENTIALLY

If only a part of the entire 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. Column 28 must contain L.

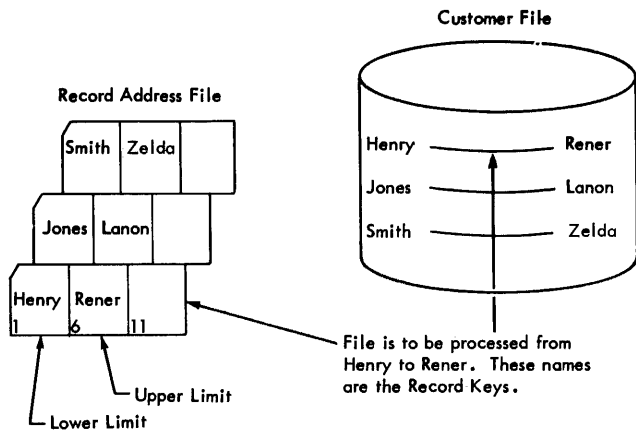
An auxiliary file is used to supply these limits. This file is called a Record Address File (RAF). The RAF does not contain data to be processed. It contains the record keys (in this case the limits) of the data records which will be processed.

The object program obtains the limits to be processed from a record contained in a RAF and then processes all the data between those limits. The object program then reads another record in the RAF, and the procedure is continued until the RAF is depleted.

IBM			INTERNATIONAL BUSINESS													
			REPORT PROGRAM GENERATOR													
			IBM Sys													
Date _____																
Program _____																
Programmer _____																
			Punching Instruction	Graphic Punch												
Line	Form Type	Filename	File Type	File Designation	End of File	Sequence	File Format	Block Length	Record Length	U/S	Mode of Processing	Length of Record Address Field	Record Address Type	Type of File Organization	Overflow Indicator	Key Field Starting Location
1	F	MASTERIN	I			F		200	200							
2	F															

● Figure 76. Coding a Sequential File

In Figure 77, the first card of the record-address file shows that the object program is to process from HENRY to RENER. The second card shows that the object program is to process from JONES to LANON. The third card shows that the third part of the file to be processed ranges from SMITH to ZELDA. Thus, the data records that the program processes are contained within these limits. In Figure 77, the record keys are customer names.



● Figure 77. Contents of the Record Address File

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 publication.

Figure 78 illustrates the coding required on the File Description and File Extension Specification sheets when limits of an indexed-sequential file are to be processed. In Figure 78, on the File Description Specification sheet 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 column 28. The K in column 31 indicates that the record key will be used to obtain the records from the file.



INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS
 IBM System/360

Form X24-3347
 Printed in U. S. A.

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic						
	Punch						

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	File Type		File Designation		Sequence		File Format		Mode of Processing		Device	Symbolic Device	Name of Label Exit	Extent Exit for DAM	Comments
			L/O/U/C	P/S/C/R/T	End of File	File Designation	Block Length	Record Length	L/R	Length of Record Address Field	Record Address Type	Type of File Organization					
01	F	DISKIN											DISK11	SYSIPTS			
02	F	RAFLIMIT											EREAD40	SYSRDR			
03	F																



INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR FILE EXTENSION SPECIFICATIONS
 IBM System/360

Form X24-3348
 Printed in U. S. A.

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic						
	Punch						

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Record Sequence of the Chaining File		Table Name	Number of Table Entries Per Record		Table Name (Alternating Table)	Length of Table Entry	Comments
		From Filename	To Filename		Number of Table Entries Per Table	Length of Table Entry			
01	E	RAFLIMIT	DISKIN						
02	E								

● Figure 78. Processing a Part of the Indexed-Sequential File

The record-address file (RAFLIMIT) is also an input file. The R in column 16 indicates that it is a RAF 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 5 characters.

Because the File Extension Specification sheet is required to relate the RAF file to the DASD file, an E is entered in column 39. On the file extension sheet the coding illustrates that RAFLIMIT is to provide the addresses for DISKIN.

sequential processing), the RAF contains the record key of each record of the file to be processed.

Figures 79 and 80 illustrate the specifications for this type of organization. In this example, the first record from the record-address file is read. The first RAF 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 THE FILE RANDOMLY

A K in column 31 indicates that record keys are used to obtain the records. An indexed-sequential file may be processed randomly by supplying a RAF. Instead of supplying the limits (as in the case of

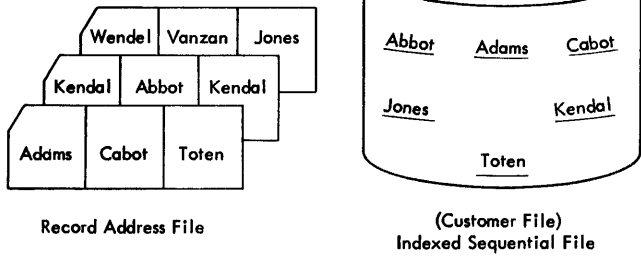


Figure 79. Randomly Processing an Indexed-Sequential File

Direct organization is specified by a D in column 32 of the File Description Specification sheet. The mode of processing is random, and an R in column 28 indicates random processing. If the record key is used to obtain the records, enter a K in column 31. If record ID is used, enter an I in column 31.

When an RPG program processes an input file of this organization, an RAF must be supplied. The necessary steps must be supplied which convert the data fields contained in the RAF to the track address and record key or record ID to be retrieved. The conversion routine depends, of course, on the way a data field can be converted to the track address of the record.

DIRECT ORGANIZATION

A file with direct organization must reside on a direct access storage device. Records are retrieved from this file by using a track address and either a record key or record identification.

The conversion routine is unique, according to the needs of a particular installation, but a conversion routine must be provided. To retrieve records from the file, some data fields must be converted to produce the track address of the record.

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION Form X24-3347 Printed in U. S. A.

REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS
IBM System/360

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

Line	Form Type	Filename	File Type		Mode of Processing		Device	Symbolic Device	Name of Label Exit	Extent Exit for DAM	Comments
			File Designation	End of File	Length of Record Address Field	Record Address Type					
010	F	DISKIN	IP	F	150	150	R	KI	DISK11	SYSIPTS	
020	F	RAFRANDMIR	IR	F	80	80	06	K	EREADA0	SYSRDR	

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION Form X24-3348 Printed in U. S. A.

REPORT PROGRAM GENERATOR FILE EXTENSION SPECIFICATIONS
IBM System/360

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

Line	Form Type	Record Sequence of the Chaining File		Table Name	Number of Table Entries Per Record		Table Name (Alternating Table)	Length of Table Entry	Comments
		From Filename	To Filename		Number of Table Entries Per Table	Length of Table Entry			
010	E	RAFRANDM	DISKIN						

● Figure 80. Processing an Indexed-Sequential File Randomly, Using an RAF File

Supplying Data to be Converted

One entry in the RAF is supplied for each record to be retrieved. The data supplied by the RAF should be such that the track address and either record key or record identification can be derived. Because the RAF is not described on the input specifications, the entries in the RAF will be made available consecutively in a field called CONTD. This field is always alphameric, and it has the record length as specified in columns 29-30 of the File Description Specification sheet.

Associating a Particular Conversion Routine with RAF

The File Extension Specification sheet entry describing the RAF file contains a field name in columns 27-32. This name is an RPG internal label linking the file extension specification with the corresponding EXTCV operation on the Calculation Specification sheet. It appears there as an entry in Factor 1.

Note: There could be another direct organized file in the same object program, for which a different EXTCV routine would be necessary.

The entry of EXTCV in the operation field indicates the point in the RPG program where the random address conversion routine is to be performed. Factor 2 must contain the name (or label) of the external routine. This external conversion routine must follow the same conventions which are specified for the EXIT routine. See Use of Registers in this publication.

The result field must contain the name (or label) of the field which will contain the track address of the record to be retrieved. This is the result of the conversion. If the record is retrieved by key (specified by a K in column 31 of the File Description Specification sheet),

KEYCV has to be entered as the operation code of the next calculation specification.

The result field must contain the name (or label) of the field which will contain the actual record key used to locate the record. The external conversion routine is written (in Assembler language) as an independent routine that must be combined with the generated object program. The name of this routine is the label which appears as Factor 2 of the EXTCV operation on the Calculation Specification sheet.

Conversion Operation Codes

The operation codes which follow are used in conjunction with the conversion routines. If there are several conversion routines in the program, these codes are repeated. However, the conversion routine (or routines) must be written first on the Calculation Specification sheet, preceding all other calculations.

EXTCV. This entry is used to indicate that a conversion routine, which is external to the RPG language, is supplied by the user.

KEYCV. This entry declares that the name specified in the result field will be the field that will contain the record key of the record. This entry must follow the EXTCV entry if the record is retrieved by key.

Figure 81 shows how the calculation specifications are coded for an external conversion routine. In this case, if the records from the disk are retrieved from information in a RAF file, each record address is contained in a special hold area CONTD. The length of this field (CONTD) is specified in columns 29-30 in the File Description Specification sheet. CONTD need not be defined as an RPG Label (RLABL) when EXTCV is used, nor do CONVERT, TRKADR, and KEYFLD.



INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICA
 IBM System/360

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic						
	Punch						

Line	Form Type	Filename	File Type				Mode of Processing				Device	Symbolic Device	Label (S, N, or E)			
			I/O/U/C	P/S/C/R/T	A/D	E	Length of Record Address Field	Record Address Type	Type of File Organization	Overflow Indicator						
			File Designation	End of File	Sequence	File Format	Block Length	Record Length	L/R	E/I	I/D/T	Key Field Starting Location	Extension Code E/I			
010	F	FILETAG	I	R	F	F	96	8		I	T		E			
020	F	PRIMARY	I	P	A	F	240	80		R	I					
03	F															



INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR FILE EXTENSION SPECIFICAT
 IBM System/360

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic						
	Punch						

Line	Form Type	Record Sequence of the Chaining File		Table Name	Number of Table Entries Per Record		Table Name (Alternating Table)	Length Entry
		From Filename	To Filename		Per Table	Length of Table Entry		
010	E	FILETAG	PRIMARY					
02	E							
03	E							
04	E							

● Figure 82. Using the ADDRROUT Option

Figure 82 illustrates the File Description and File Extension Specification sheets for a DASD file containing a data file and an Address Output (ADDRROUT) file.

ADDRESS OUTPUT FILE

The file name Filetag is assigned to the ADDRROUT file. It is an input file (I in column 15); it is an RAF file (R in column 16) and the end-of-file condition is determined by this file (E in column 17).

The file format is fixed (F in column 19) as a requirement of the Disk Sort Program.

In this example, the block length is 96 and the record length is 8.

The length of the Record Address Field (columns 29-30) is 8. The 8 bytes are a fixed requirement of the Disk Sort Program.

The entry I in column 31 is required for this operation. It applies not to the address output file (FILETAG) itself, but to the file (PRIMARY) that is to be retrieved by the address output file.

The T in Type of File Organization (column 32) indicates that the file is the output of the ADDRROUT option of the Disk Sort Program. The E in Extension Code (column 39) means that a File Extension Specification sheet is required.

ORIGINAL DATA FILE

The file name Primary is assigned to the original data record file. It is an input file (I in column 15). It is a primary file (P in column 16), and it is in ascending sequence (A in column 18). The sequence refers to the sequence in which the file is maintained.

In this example, the file format is fixed (F in column 19). The block length is 240 and the record length is 80. The mode of file processing is random (R in column 28). It is random processing because the file is processed in sequence by social security number, but it is organized in sequence by man number and department.

The entry I must appear in column 31 regardless of how the file is organized, since the records are retrieved by record ID.

FILE EXTENSION SPECIFICATIONS

The file extension specifications are used to identify, for the RPG program, the relationship of these two files. The name FILETAG (essentially an RAF) is specified in From Filename. The name PRIMARY is specified in To Filename because this is the file the RAF is linked to.

Note that Table Name (columns 27-32) is left blank. This field would normally contain the label of the EXTCV specification (Factor 1) that links to the external conversion routine, which converts the data in an RAF to an address for a DASD. When the ADDRROUT option is used, no data conversion is necessary because the data in FILETAG is already in disk address format.

PROCESSING MULTIPLE INPUT FILES

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 Field entry, (entered in columns 61-62 on the Input Specification sheet -- see Figure 83) determines which records of a secondary file are to be processed. Both files are processed sequentially. The following rules apply to the matching-field 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.
5. If M1, M2, and M3 are specified in the primary file, M1, M2, and M3 must be specified in the secondary files. Incorrect results will be obtained if the same number of matching fields is not specified. For example, Figure 83 shows three record-types. Record-type AA has two matching fields, and record-type BB has two matching fields. If record-type BB had three fields, incorrect results would be obtained. Record type CC has no matching fields (refer to step 4).

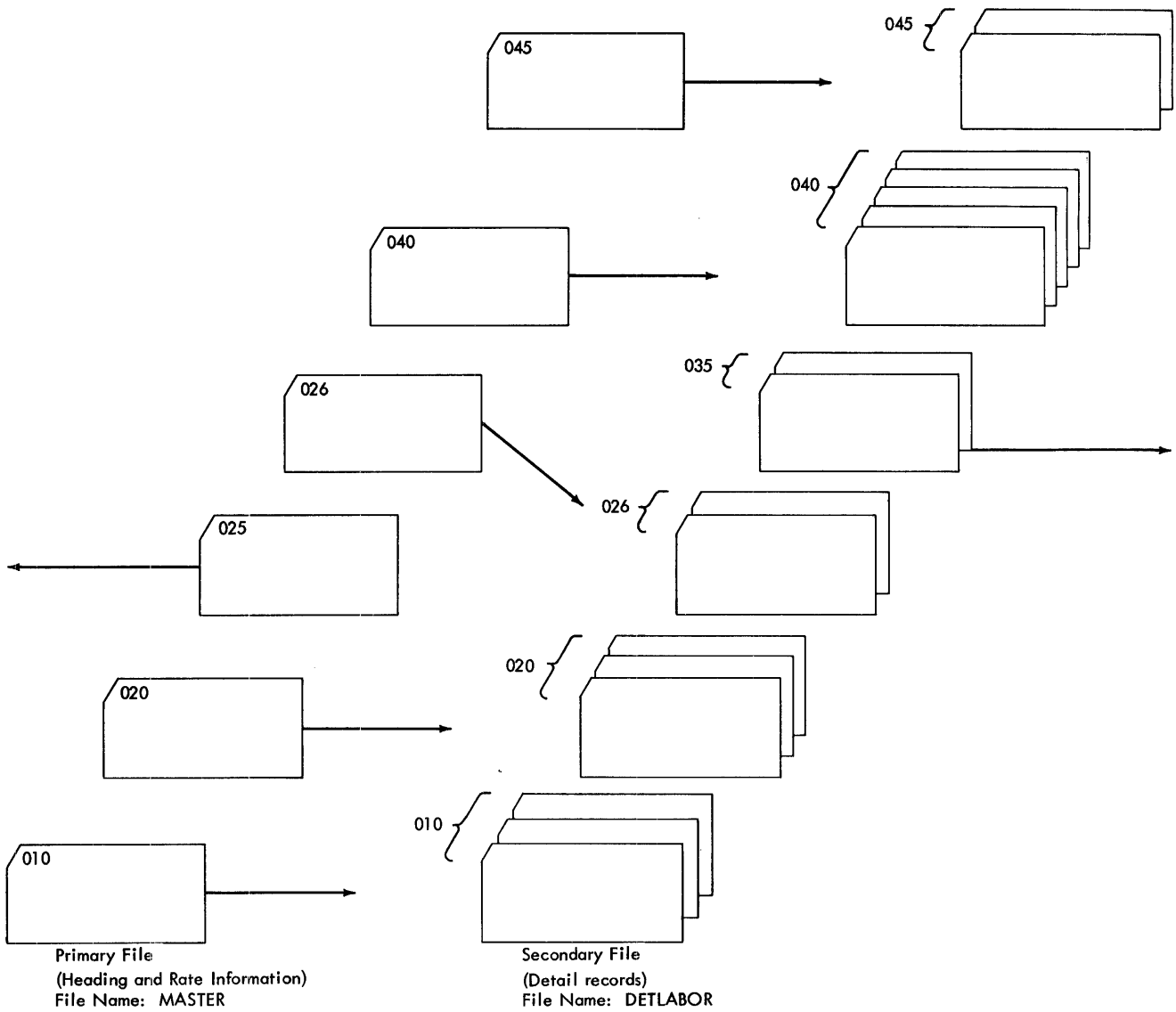


Figure 84. Two Input Files with Matching Records

The input specifications required to match these records are shown in Figure 85.

The 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.

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 85 only two fields are matched. M1 is assigned to DEPT which is the low-order field, and M2 is assigned DIVSON, which is the high-order 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 85, 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 86 illustrates a primary and a secondary file. The records in the two files will be 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 86, indicator 01 is turned on whenever there is a primary record. Indicator 02 is turned on whenever there is a secondary record. Thus,

1. A matching primary record will be coded 01 and MR.
2. A matching secondary record will be coded 02 and MR.
3. An unmatched primary record will be coded 01 and NMR.
4. An unmatched secondary record will be coded 02 and NMR.

The order in which the primary file and the secondary file are processed is listed here. Sample Program Two uses the matching field specifications.

Primary File		Secondary File	
Matching Field in the Record	Proc-essed	Matching Field in the Record	Proc-essed
01001	1st	001	3rd
01001	2nd	001	4th
02002	5th	002	6th
03004	9th	003	7th
03004	10th	003	8th
05006	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 File Description Specification sheet.

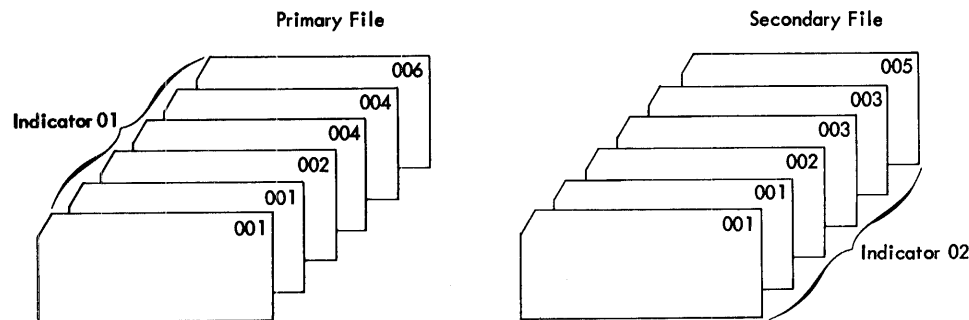


Figure 86. Order of Processing Matched Records

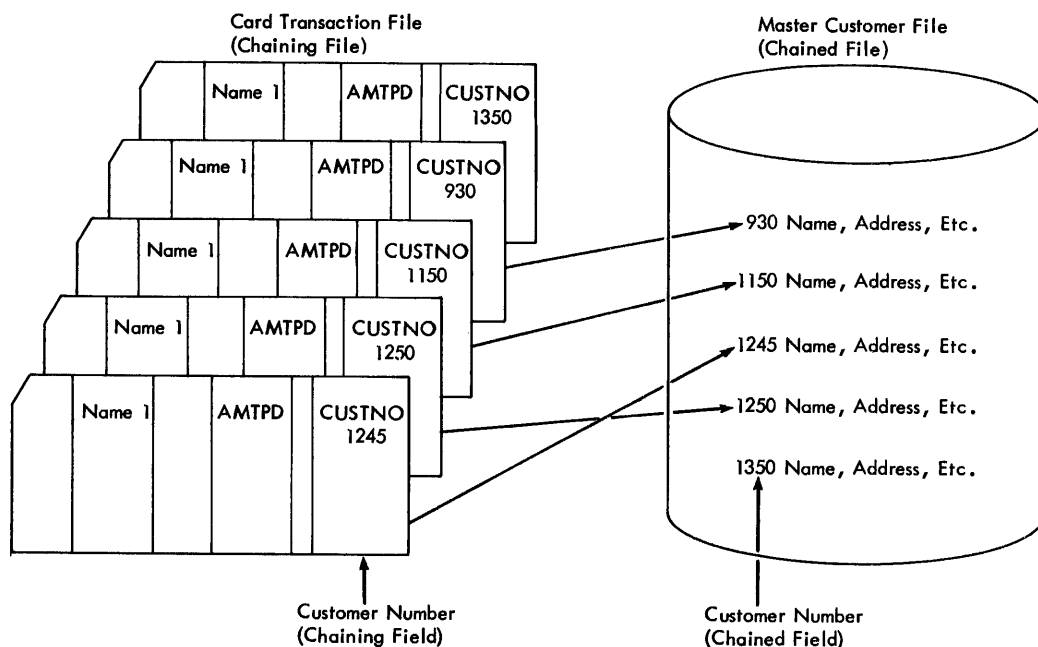


Figure 87. Using Chaining to Process an Indexed-Sequential File

RANDOMLY PROCESSING MULTIPLE INPUT FILES (CHAINING)

To understand chaining, assume that a sequential input file, as shown in Figure 87, 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 MASTCUST (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 customers' 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, and 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 because it links to the chained file. The master customer file (MASTCUST) is the chained file because it is linked to the transaction file.

Up to three chaining fields may be specified for a record. The chaining fields are designated by entering C1, C2, or C3 in columns 61-62 of the Input Specification sheet.

Note: There is no specific relationship between levels C1, C2, or C3 other than specifying the three possibilities for chaining fields.

Figure 88 illustrates coding used for chaining the transaction file CARDIN to the customer file MASTCUST. On the File Description Specification sheet, the card file is considered the primary file. When CARDIN is depleted, the program goes to the end-of-job routine (E in column 17).

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION Form X24-3347
REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS Printed in U.S.A.
 IBM System/360

Date _____
 Program _____
 Programmer _____

Punching Instruction: Graphic Punch

Page 1 2
 Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	File Designation		Mode of Processing		Device	Symbolic Device	Name of Label Exit	Extent Exit for DAM	Comments
			End of File	Sequence	Length of Record	Address Field					
010		CARDIN	IP	F	80	80	EREAD44	SYSDR			
020		MASTCUST	IC	F	1000	2000	DISK11	SYSD07S			

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION Form X24-3348
REPORT PROGRAM GENERATOR FILE EXTENSION SPECIFICATIONS Printed in U.S.A.
 IBM System/360

Date _____
 Program _____
 Programmer _____

Punching Instruction: Graphic Punch

Page 1 2
 Program Identification 75 76 77 78 79 80

Line	Form Type	Record Sequence of the Chaining File		Table Name	Number of Table Entries Per Record	Number of Table Entries Per Table	Table Name (Alternating Table)	Length of Table Entry	Comments
		From Filename	To Filename						
010		AA	CARDIN	MASTCUST					

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION Form X24-3350
REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS Printed in U.S.A.
 IBM System/360

Date _____
 Program _____
 Programmer _____

Punching Instruction: Graphic Punch

Page 1 2
 Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Record Identification Codes			Field Location		Field Name	Field Indicators			Sterling Sign Position
			Position	Character	Position	Character	From		To	Plus	Minus	
010		CARDIN	AA	01	80	CX						
020							75	80	CUSTNO			C1
030							7	20	NAME1			
040							21	30	AMTPD			
050		MASTCUST	AA	02	2000	D1						
060							1	6	CODE			
070							7	20	NAME2			
080							21	30	BALANC			
090							31	36	CUSNO1			
100							37	50	ADDRS			
110							51	60	CITY			
120							61	65	STATE			
130							66	80	SHIPER			

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION Form X24-3351
REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS Printed in U.S.A.
 IBM System/360

Date _____
 Program _____
 Programmer _____

Punching Instruction: Graphic Punch

Page 1 2
 Program Identification 75 76 77 78 79 80

Line	Form Type	Indicators		Factor 1	Operation	Factor 2	Result Field	Field Length	Resulting Indicators			Comments
		Control Level (1-3)	And						Plus	Minus	Zero or Blank	
010		01	02	BALANC	SUB	AMTPD	BALANC					
020		01	02	SETON							H1	

● Figure 88. Specifying Chaining

The File Extension Specification sheet contains the record sequence of the CARDIN file and the number of the chaining field. On the Input Specification sheet, the two files are defined. CUSTNO is the field which chains the files. CUSNO1 is the key field on disk, having a starting location of 31 (as denoted in columns 37-38 of the File Description Specification sheet).

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 chaining file (01N02), halt indicator H1 is turned 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 resulting indicators (representing two different record types) can be on at the same time.

Figure 89 and 90 show two additional uses of chaining. In Figure 89, 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.

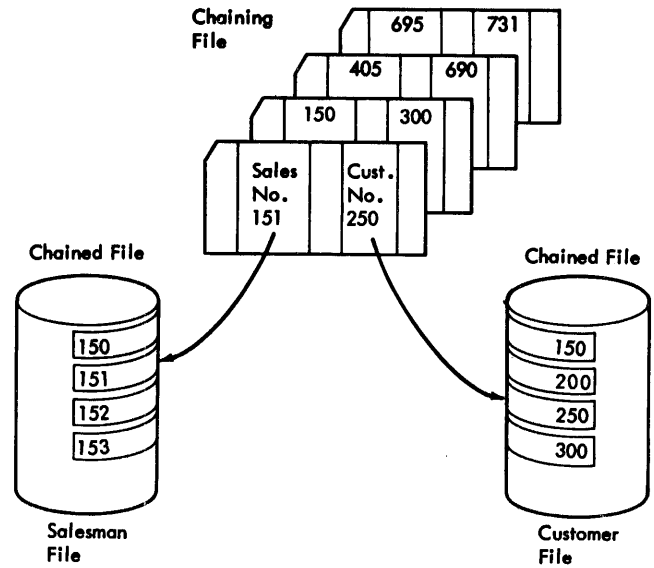


Figure 89. Chaining to Two Files

In Figure 90, 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.

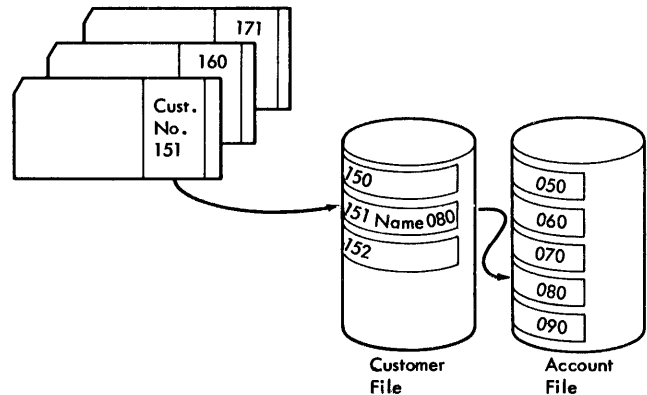


Figure 90. Using a Chained File as a Chaining File

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 Specification sheet, and the fields are then used as one chaining field. The first field

defined on the input sheet is placed in the high-order position, and the last field is placed in the low-order position. The fields are placed in the same sequence as they are defined on the Input Specification sheet.

CREATING RECORD ADDRESS FILES (RAF)

GENERAL INFORMATION

A record-address file is one of the ways by which the necessary information to retrieve records from nonsequential 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 a file with direct organization.
2. For sequential processing between limits of a file with Indexed-Sequential organization.

Only one RAF may be specified for an RPG program. An RAF is processed sequentially, and it must be on a file with sequential organization. An RAF is described on the file description sheet and the file extension sheet, but it is not described on the input sheet.

RANDOM PROCESSING OF INDEXED OR DIRECT ORGANIZATION

These rules must be followed when creating a RAF for random processing:

1. For an Indexed Sequential Organization, the field will contain the record key. For a direct organization, each entry in the RAF must consist of a field to be converted to the track address, and either the record Key or the record ID.
2. The record addresses must begin in position 1 of the record.

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, will cause the RPG program to read the next record in the RAF.

PROCESSING LIMITS OF AN INDEXED SEQUENTIAL ORGANIZATION

When a RAF is used to indicate what limits of an 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 will process 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.

MULTIPLE FILE PROCESSING

RPG will process multiple files two ways:

1. Sequentially - by using the matching record technique
2. Randomly - by using the chaining technique

Figure 91 shows the processing possibilities for files which have Sequential, Indexed Sequential or Direct Organization.

From \ To	Sequential Organization	Indexed Sequential Organization (DASD)	Direct Organization (DASD)
Sequential Organization	Sequential Processing (Matching)	Sequential Processing (Matching) or Random Processing (Chaining)	Random Processing (Chaining)
Indexed-Sequential Organization *	Sequential Processing (Matching) □	Sequential Processing (Matching) or Random Processing (Chaining) □	Random Processing (Chaining)
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 91. Processing Multiple Input Files

A sequential organization is processed sequentially and controls the processing of records in:

1. Another Sequential Organization. Both files are processed sequentially, using matching records to govern processing.
2. 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 Files are to be processed.
3. A Direct Organization. A direct organization will be processed randomly, under control of the sequential file. The sequential file will contain chaining fields which will be converted to the record locations on the Direct File.

Note: In all three cases, the Sequential Organization will be processed sequentially.

An Indexed-Sequential Organization may be processed sequentially or randomly, and it controls processing of records in:

1. A Sequential Organization. The records in both files will be processed sequentially, using matching records to control processing.
2. 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 will be used to control processing.

3. A Direct Organization. Chaining fields in the indexed file are converted to supply the record locations in the direct file which will be processed. The direct file is processed randomly.

A direct organization is processed randomly and controls processing of records in:

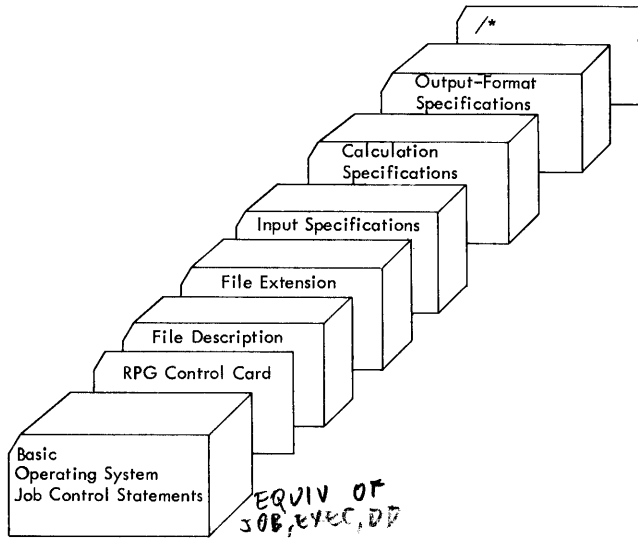
1. An Indexed-Sequential Organization. The indexed file is processed randomly, and chaining fields in the direct file control processing of the indexed file.

In both cases, the direct organization is processed randomly.

2. 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.

RPG SOURCE DECK ARRANGEMENT

The source deck prepared by the programmer will be arranged as shown in Figure 92. The contents of the basic operating system Job Control statements are listed in the programmer's guide listed on the front cover of this publication.



● Figure 92. RPG Source Deck Arrangement in the Basic Operating System Input Stream

The order in which the programmer places his source deck is as follows:

1. Basic Operating System Job Control Statements
2. RPG Control Card (Processor Control Card)
3. File Description Specifications
4. File Extension Specifications
5. Input Specifications
6. Calculation Specifications
7. Output-Format Specifications
8. End-of-File Card

The contents of the RPG control card follow.

<u>Processor Control Card</u>		<u>Explanation</u>
<u>Column</u>	<u>Contents</u>	
1-5	XXXXX	Page and Line numbers.
6	H	Identifies the card as a header card.
7-9	008, 016, 032, 064	Size of Generating Machine (8K, 16K, 32K, or 64K). This entry must not exceed the main storage capacity.
10	P, R, or blank	Enter a <u>P</u> if the object program deck is to be punched in cards (Compile-Only option). Enter an <u>R</u> if the object program is to be written in the relocatable library (permanently) and executed immediately following compilation (Compile-and-Execute option). Leave <u>blank</u> if the object program is to be written in the relocatable library (temporarily) and executed immediately following compilation (Compile-and-Execute option).
11	D, B, or blank	Type of RPG run. Punch a <u>D</u> if a listing is to be produced and no compilation. Punch a <u>B</u> if a listing is to be bypassed and the compilation is to be performed. (Bypass Listing option). Leave this column blank for both a listing and a compilation.
12-14	008, 016,	Size of Object

	032, 064	Machine (8K, 16K, 32K, or 64K).	21	I or blank	<u>Inverted Print.</u> If numeric literals and edit words use the European conventions of punctuation (commas for decimal point and vice-versa), enter an <u>I</u> in this column. Otherwise leave blank.
15-16	Blank	Leave these columns blank.			
17	1, 2, or blank	If the Sterling Shillings Field on Input is in the IBM format, punch <u>1</u> in this column. If it is in the BSI format, punch <u>2</u> . Otherwise leave blank.	22-25	Blank	Leave these columns blank.
			26	A, or blank	<u>Alternate Collating Sequence.</u> Enter an <u>A</u> in this column, if an external subroutine is used to translate the sequence of a matching field to the collating sequence of the System/360. If an external translating subroutine is not used, leave this column blank.
18	1, 2, or blank	If the Sterling-Pence Field on Input is in the IBM format, punch 1. If it is in the BSI format, punch <u>2</u> . Otherwise, leave blank,			
19	0, 1, 2, or blank	If the Sterling-Shillings Field on Output is in the printer format, punch 0. If it is in the IBM format, punch <u>1</u> . If it is in the BSI format, punch <u>2</u> . Otherwise, leave blank.	27-74	blank	Leave these columns blank.
			75-80	XXXXXX	<u>Program Identification.</u> Enter in these columns the program identification that will appear in the object program identification field. (See <u>Note.</u>)
20	0, 1, 2, or blank	If the Sterling-Pence Field on Output is in the printer format, punch 0. If it is in the IBM format, punch <u>1</u> . If it is in the BSI format, punch <u>2</u> . Otherwise leave blank.			

Note: Object program identification and the sequence number will be punched in each card of the object program deck. If the user specifies the program identification in the RPG Processor control card, XXXXnnnn will be punched in each card (XXXX = the first 4 characters of the program identification; nnnn = sequence number of the card). If no program identification is specified, RPGOnnnn will be punched in each card.

Although only the first 4 characters of the program identification appearing in columns 75-80 will be punched in the object deck, all 6 characters constitute the module name if the object program is cataloged in the relocatable library.

VARIABLE-LENGTH RECORDS

UNBLOCKED RECORDS

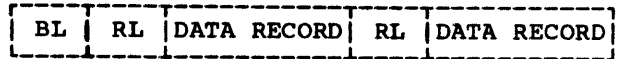
The format of a variable-length unblocked record is as follows:



where BL is the block length and RL is the record length. Each physical record contains one data record and 2 four-character fields, one containing the block length and one containing the record length. Figure 93 illustrates a segment of a file of variable-length unblocked records (VARUBLK). Data Record 1 contains 53 characters, has a record length of 57, and a block length of 61. However, when specifying these lengths, the user is concerned only with the length of the data record. The four-character record-length field and block-length field are not considered. In the File Description Specification sheet, the data record length of the largest physical record in the file is specified for both record length and block length. In Figure 95, 100 is specified because the largest data record in VARUBLK is Data Record 2, which contains 100 characters. Figure 96 illustrates the coding required in the Output-Format Specification sheet for the variable-length unblocked record file VARUBLK.

BLOCKED RECORDS

The format of a variable-length blocked record is as follows:



where BL is the block length and RL is the record length. A physical record contains one or more data records, each having a corresponding record-length field. Figure 94 illustrates one physical record in a file of variable-length blocked records (VARBLK). This physical record contains three data records, having lengths of 53, 100, and 31 characters. (Assume that this physical record is the largest in the file VARBLK.) The total number of characters is 184, which is placed in Block Length (columns 20-23) in the File Description Specification sheet as illustrated in Figure 95. The maximum length of any data record within the file is 100, which appears in Record Length (columns 24-27). If the filename in Figure 96 were changed to VARBLK, the coding illustrated would be equally applicable to the blocked file.

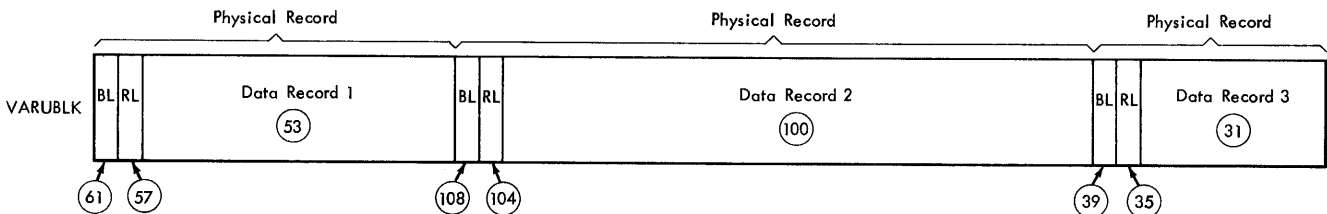


Figure 93. Variable-Length Unblocked Records

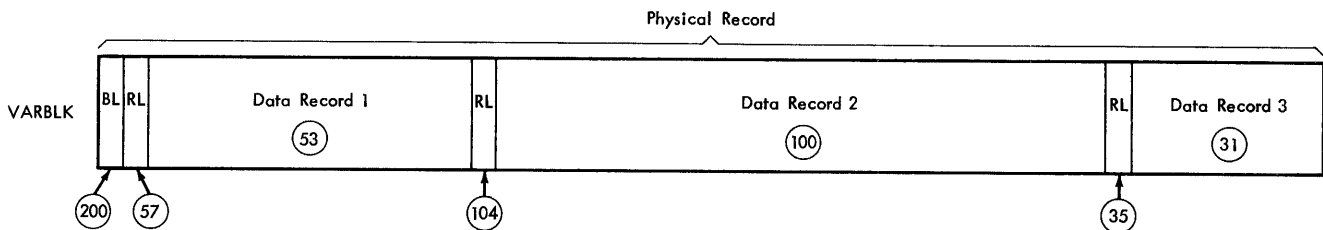


Figure 94. Variable-Length Blocked Records

The Basic Operating System RPG diagnostic messages to the programmer appear in the following listing. The abbreviated specification types appearing within parentheses (FDS, FES, IS, CS, and OF) refer to the RPG specification sheet on which the erroneous entries were made (File Description, File Extension, Input, Calculation, and Output-Format, respectively).

NOTE	CAUSE OF ERROR	RESULT
1	The <u>File Type</u> entry (FDS, column 15) is not I,O,U, or C, or is blank.	<u>I</u> is assumed.
2	The <u>Mode of Processing</u> entry (FDS, column 28) is invalid for disk device.	<u>Blank</u> is assumed.
3	The <u>Length of Record Address Field</u> entry (FDS, columns 29-30) is invalid or is blank.	<u>08</u> is assumed.
4	The <u>Record Address Type</u> entry (FDS, column 31) is not K or I.	<u>I</u> is assumed.
5	The <u>File Designation</u> entry of I (FDS, column 16) requires an entry in <u>Extension Code</u> (FDS, column 39).	<u>E</u> is assumed.
6	The <u>File Designation</u> entry (FDS, column 16) is not P,S,R,C, or T for the input file.	<u>S</u> is assumed.
7	The <u>Overflow Indicator</u> entry stated (FDS, columns 33-34) is invalid without a printer device.	<u>Blanks</u> are assumed.
8	The <u>File Type</u> entry stated (FDS, column 15) is invalid with a printer device.	<u>O</u> is assumed.
9	Multiple primary files are stated.	Secondary file is assumed.
10	The <u>Label Indicator</u> entry (FDS, column 53) is not N,S,E, or blank for this tape file.	<u>S</u> is assumed.
11	The <u>Type of File Organization</u> entry (FDS, column 32) is not I,D,T, or blank.	<u>Blank</u> is assumed.
12	The <u>Extension Code</u> entry (FDS, column 39) is not blank for an output file.	<u>Blank</u> is assumed.
13	The <u>End of File</u> entry (FDS, column 17) is not E or blank.	<u>E</u> is assumed.
14	The <u>Sequence</u> entry (FDS, column 18) is not A, D, or blank.	<u>A</u> is assumed.
15	The <u>File Designation</u> entry (FDS, column 16) is not blank for an output file.	<u>Blank</u> is assumed.
16	The <u>C</u> entry in <u>File Type</u> , (FDS, column 15) requires a card device entry in <u>Device</u> (FDS, columns 40-46).	<u>READ42</u> is assumed.
17	A Record Address File requires an entry in <u>Extension Code</u> (FDS, column 39).	<u>E</u> is assumed.

18	The <u>File Format</u> entry (FDS, column 19) is not F or V.	V is assumed.
19	The <u>Block Length</u> entry (FDS, columns 20-23) is not a numeric entry or is not right-justified.	The entry in <u>Record Length</u> (FDS, columns 24-27) is assumed.
20	The <u>Record Length</u> entry (FDS, columns 24-27) is invalid or is not right-justified.	0080 is assumed.
21	The U entry in <u>File Type</u> (FDS, column 15) requires a disk device entry in <u>Device</u> (FDS, columns 40-46).	DISK is assumed.
22	An <u>End of File</u> entry (FDS, column 17) and a <u>Sequence</u> entry (FDS, column 18) are invalid with a printer device.	Blanks are assumed.
23	Columns 28-32 (FDS) are not used with an output file.	Blanks are assumed.
24	More than two secondary files are specified.	Additional card(s) not processed.
25	The <u>Overflow Indicator</u> entry (FDS, columns 33-34) invalid.	Blanks are assumed.
26	The <u>Labels</u> entry (FDS, column 53) is incompatible with the device specified.	Blank is assumed.
27	The <u>End of File</u> entry (FDS, column 17) is not blank for an input file.	Blank is assumed.
28	The disk device entry in <u>Device</u> (FDS, columns 40-46) requires an S or E entry in <u>Labels</u> (FDS, column 53).	S is assumed.
29	The <u>Extension Code</u> entry (FDS, column 39) is not E or blank for an input file.	E is assumed
30	The <u>From Filename</u> entry (FES, columns 11-18) is not present or is not left-justified.	Card not processed.
31	The <u>From Filename</u> entry (FES, columns 11-18) is not defined.	Card not processed.
32	The <u>From Filename</u> entry (FES, columns 11-18) is improperly defined.	Card not processed.
33	<u>Number of the Chaining Field</u> entry (FES, columns 9-10) is not C1, C2, or C3.	Card not processed.
34	The <u>Record Sequence</u> entry (FES, columns 7-8) is invalid. The positions are neither both numeric nor both alphabetic.	Card not processed.
35	The <u>To Filename</u> entry (FES, columns 19-26) is not present or is not left-justified on RAF or Chaining type specifications.	Card not processed.
36	The <u>To Filename</u> entry (FES, columns 19-26) is not defined on RAF or Chaining type specifications.	Card not processed.
37	The <u>To Filename</u> entry (FES, columns 19-26) is improperly defined on RAF or Chaining type specifications.	Card not processed.

38	Columns 33-57 are not all blank on the Chaining type File Extension entry.	Columns 33-57 are ignored.
39	Columns 7-10 are not all blank on the RAF type File Extension entry.	Columns 7-10 are ignored.
40	Columns 33-57 are not all blank on the RAF type File Extension entry.	Columns 33-57 are ignored.
41	Columns 7-10 are not all blank on the Table type File Extension entry.	Columns 7-10 are ignored.
42	The <u>To Filename</u> entry (FES, columns 19-26) is not left-justified on a Table type specification.	<u>Blanks</u> are assumed.
43	The <u>To Filename</u> entry (FES, columns 19-26) is not defined on a Table type specification.	<u>Blanks</u> are assumed.
44	The <u>To Filename</u> entry (FES, columns 19-26) is improperly defined on a Table type specification.	<u>Blanks</u> are assumed.
45	The <u>Table Name</u> entry (FES, columns 27-32 and/or columns 46-51) is not present or is not left-justified.	Card not processed.
46	The letters <u>T A B</u> are not present in columns 27-29 and/or columns 46-48.	Letters <u>T A B</u> are assumed.
47	The <u>Number of Table Entries Per Record</u> entry (FES, columns 33-35) is not numeric.	<u>010</u> is assumed.
48	The <u>Number of Table Entries Per Table</u> entry (FES, columns 36-39) is not numeric.	<u>0100</u> is assumed.
49	The <u>Length of Table Entry</u> entry (FES, columns 40-42 and/or columns 52-54) is not numeric.	<u>008</u> is assumed.
50	The <u>Packed</u> entry (FES, column 43 and/or column 55) is not P or blank.	<u>Blank</u> is assumed.
51	The <u>Decimal Pos.</u> entry (FES, column 44 and/or column 56) is not blank or a numeric digit.	<u>0</u> is assumed.
52	The <u>Sequence</u> entry (FES, column 45 and/or column 57) is not A, D, or blank.	<u>Blank</u> is assumed.
53	The <u>Form Type</u> entry (FES, column 6) is not E.	Card not processed.
55	The <u>Name of Label Exit</u> entry (FDS, columns 54-59) is invalid.	<u>Blanks</u> are assumed.
56	Columns 60-65 (FDS) are not all blank.	<u>Blanks</u> are assumed.
57	The <u>Name of Label Exit</u> entry (FDS, columns 54-59) must be blank unless the <u>Labels</u> entry (FDS, column 53) is N or E.	<u>Blanks</u> are assumed.
60	No RPG Processor Control Card.	
61	Column 11 of RPG Control Card does not contain D, B, or blank or is invalid.	<u>Blank</u> is assumed.
62	The machine size (RPG Control Card, columns 7-9 and/or columns 12-14) is not 008, 016, 032, or 064, or is invalid.	<u>008</u> is assumed.
63	Columns 17-20 of RPG Control Card contain invalid entries.	<u>Blanks</u> are assumed.

64	Column 21 of RPG Control Card does not contain I or blank.	<u>Blank</u> is assumed
65	Column 26 of RPG Control Card does not contain A or blank.	<u>Blank</u> is assumed.
66	Column 10 of the RPG Control Card does not contain P, R, or blank.	<u>P</u> is assumed.
67	Columns 70-80 of the RPG Control Card is blank or invalid.	Program name <u>RPGO</u> is assumed. If <u>R</u> is specified in column 10, <u>blanks</u> are assumed.
71	The <u>Result Field</u> entry (CS, columns 43-48) is required but is not present.	<u>*****</u> in columns 43-48. <u>015</u> in columns 49-51, <u>0</u> in column 52, and <u>blank</u> in column 53 are assumed.
72	The <u>Result Field</u> entry (CS, columns 43-48) should be blank for this operation.	<u>Blanks</u> are assumed.
73	The <u>Factor 1</u> entry (CS, columns 18-27) is required but is not present.	Literal <u>1</u> is assumed.
74	The <u>Factor 2</u> entry (CS, columns 33-42) is required but is not present.	Literal <u>1</u> is assumed.
75	The <u>Resulting Indicators</u> entry/entries (CS, columns 54-59) is/are invalid.	<u>00</u> is assumed.
76	The <u>Factor 1</u> entry (CS, columns 18-27) should be blank for this operation.	<u>Blanks</u> are assumed.
77	The <u>Factor 2</u> entry (CS, columns 33-42) should be blank for this operation.	<u>Blanks</u> are assumed.
78	Column 7 of <u>Control Level</u> (CS, columns 7-8) does not contain L or blank.	<u>Blank</u> is assumed.
79	All <u>Detail</u> calculations must precede all <u>Total</u> calculations.	<u>L0</u> is assumed.
80	The <u>Factor 1</u> entry (CS, columns 18-27) is not left-justified.	Literal <u>1</u> is assumed.
81	The <u>Factor 2</u> entry (CS, columns 33-42) is not left-justified.	Literal <u>1</u> is assumed.
82	The entry in <u>Factor 1</u> (CS, columns 18-27) is an improperly stated literal or field name.	Literal <u>1</u> is assumed.
83	The entry in <u>Factor 2</u> (CS, columns 33-42) is an improperly stated literal or field name.	Literal <u>1</u> is assumed.
84	The <u>Factor 1</u> entry (CS, columns 18-27) is a field name of more than six characters.	The <u>first</u> six characters are assumed.
85	The <u>Factor 2</u> entry (CS, columns 33-42) is a field name of more than six characters.	The <u>first</u> six characters are assumed.
86	The <u>Operation</u> entry (CS, columns 28-32) is not present or is invalid.	The <u>MOVE</u> operation code is assumed.
89	The <u>Result Field</u> entry (CS, columns 43-48) is invalid.	<u>*****</u> in columns 43-48 <u>015</u> in columns 49-51 <u>0</u> in column 52,

		and <u>blank</u> in column 53 are assumed.
94	The <u>Field Length</u> entry (CS, columns 49-51) is neither blank nor numeric, or is not right-justified.	<u>015</u> is assumed.
95	The <u>Decimal Position</u> entry (CS, column 52) is not blank or numeric.	<u>0</u> is assumed.
96	The <u>Half Adjust</u> entry (CS, column 53) is not H or blank.	<u>H</u> is assumed.
97	A <u>resulting indicator</u> is required for this operation.	<u>00</u> in columns 58-59 are assumed.
98	The <u>Indicators</u> (CS, columns 9-17) entry(s) is invalid.	<u>00</u> in columns 10-11 is assumed.
109	Input and/or output specifications are missing or are invalid.	Job is terminated.
110	<u>Form Type</u> (column 6) is not I, C, or O and column 7 does not contain *.	Card not processed.
111	The <u>Filename</u> entry (IS, columns 7-14) is not in the filename table.	Card not processed.
112	The <u>Filename</u> entry (IS, columns 7-14) is not correctly defined on the File Description Specification sheet.	Card not processed.
113	<u>AND</u> card out of sequence (first in deck, after field name, or invalid file type).	Card not processed.
114	No entries in <u>Record Identification Codes</u> (IS, columns 21-41) in card preceding the <u>AND</u> card.	Card not processed.
115	<u>OR</u> card out of sequence (first in deck, after field name, or invalid file type).	Card not processed.
116	The <u>Filename</u> entry (IS, columns 7-14) is not left-justified.	Card not processed.
117	The entry in <u>Filename</u> (IS columns 7-14) begins with a numeric character.	Card not processed.
118	Entries in <u>Filename</u> (IS, columns 7-14) and in <u>Field Name</u> (IS, columns 53-58) are both present.	The <u>Filename</u> entry is assumed.
119	The <u>Sequence</u> entry (IS, columns 15-16) is blank.	<u>AA</u> is assumed..
120	The sequence is <u>descending</u> and alphameric sequence is found after numeric sequence.	The numeric sequence equal to the previous numeric sequence is assumed.
121	Descending numeric sequence is found.	Sequence equal to <u>previous</u> numeric sequence is assumed.
122	Sequence is numeric and <u>Number</u> entry (IS, column 17) does not contain <u>N</u> or <u>1</u> .	<u>N</u> is assumed.
123	Sequence is numeric and the <u>Option</u> entry (IS, column 18) is not <u>0</u> or blank.	<u>0</u> is assumed.
124	The input record <u>Resulting Indicator</u> entry (IS, columns 19-20) is blank.	<u>00</u> is assumed.

125	The <u>Stacker Select</u> entry (IS, column 42) is not 1, 2, or blank.	<u>Blank</u> is assumed.
126	An invalid input file has been specified.	No immediate action taken.
127	The entry(s) in <u>Position</u> (IS, columns 21-24, 28-31, or 35-38) contains a nonnumeric character.	<u>0</u> is assumed.
128	The <u>Not</u> entry (IS, column 25, 32, or 39) is not N or blank.	<u>N</u> is assumed
129	The <u>C/Z/D</u> entry (IS, columns 26, 33, or 40) is not C, Z, or D.	<u>C</u> is assumed.
130	Field name type specification out of sequence (first in deck, after invalid file name, or AND/OR specifications).	Card not processed.
131	The <u>Field Name</u> entry (IS, columns 53-58) is not left-justified.	Card not processed.
132	The <u>Field Name</u> entry (IS, columns 53-58) begins with a numeric character.	Card not processed.
133	The <u>From</u> entry (IS, columns 44-47) and/or the <u>To</u> entry (IS, columns 48-51) is blank.	<u>0001</u> is assumed.
134	The <u>From</u> entry (IS, columns 44-47) and/or the <u>To</u> entry (IS, columns 48-51) contains a non-numeric character.	<u>0</u> is assumed.
135	The input field length (<u>From</u> and <u>To</u> columns 44-51) is less than 1.	Field length of <u>1</u> is assumed.
136	Packed input field length (<u>From</u> and <u>To</u> , columns 44-51) is greater than 8.	Field length of <u>8</u> is assumed.
137	The <u>Packed</u> entry (IS, column 43) is not P or blank.	<u>P</u> is assumed.
138	The <u>Decimal Positions</u> entry (IS, column 52) is not numeric or blank.	<u>0</u> is assumed.
139	The numeric field is greater than 15 bytes in length.	Field length of <u>15</u> is assumed.
140	The <u>Control Level</u> entry (IS, columns 59-60) does not contain an <u>L</u> in column 59.	<u>L</u> in column 59 is assumed.
141	The <u>Control Level</u> entry (IS, columns 59-60) does not contain a numeric character in column 60.	<u>1</u> in column 60 is assumed.
142	Column 61 (IS) does not contain <u>C</u> or <u>M</u> .	<u>M</u> is assumed.
143	The <u>Matching Fields or Chaining Fields</u> entry (IS, columns 61-62) is not numeric.	<u>01</u> is assumed.
144	The <u>Matching Fields</u> entry (IS, columns 61-62) contains a matching value greater than <u>3</u> .	<u>03</u> is assumed.
145	Columns 65-68 (IS) contain numeric indicators for a nonnumeric field.	Indicators are ignored.
146	The alphameric field is greater than 256 bytes in length.	Length of <u>256</u> is assumed.
147	A <u>Sterling</u> field is indicated with a decimal position greater than <u>3</u> .	<u>3</u> is assumed.

148	A <u>Sterling</u> field is indicated without a decimal position specified.	<u>0</u> is assumed.
149	Record identification specification is out of order, or no fields are indicated for a given record.	No immediate action taken.
150	No <u>Decimal Position</u> is indicated for a packed input field.	<u>0</u> is assumed.
151	Column 0 has been indicated.	Column <u>1</u> is assumed.
159	A field name beginning with <u>TAB</u> is not a table name.	Card not processed.
160	The <u>Form Type</u> entry (Column 6) should be 0.	Card not processed.
161	The specification type cannot be determined.	Card not processed.
162	The <u>Filename</u> entry (OF, columns 7-14) is invalid, not present, or undefined.	Card not processed.
163	The <u>H/D/T</u> entry (OF, column 15) is out of order.	<u>H</u> is assumed.
164	The <u>Type H/D/T</u> entry (OF, column 15) is not H, D, or T.	<u>H</u> is assumed.
165	The <u>Output Indicators</u> entry (OF, columns 23-31) is not present in an OR-relationship.	<u>00</u> is assumed.
166	The <u>Output Indicators</u> entry (OF, columns 23-31) is not present in an AND-relationship.	Card not processed.
167	Columns 32-70 (OF) contain entries, but should be blank.	Columns 32-70 are ignored.
168	The <u>Field Name</u> entry (OF, columns 32-37) is not left-justified.	Card not processed.
169	The <u>Output Indicators</u> entry (OF, columns 23-25, 26-28, or 29-31) contain an invalid indicator.	Blanks are assumed.
170	The <u>OR</u> card or the <u>AND</u> card is out of order.	Card not processed.
171	Field type specification, when column 15 is blank, is not preceded by a valid record identification specification.	Card not processed.
172	Columns 7-22 (OF) contain entries, but should be blank.	Columns 7-22 are ignored.
173	Leading and/or closing quote signs in column 45 and 70 (OF) are not present.	No editing is performed.
174	The <u>Zero Suppress</u> entry (OF, column 38) contains an invalid or incorrectly used entry.	<u>Blank</u> is assumed.
175	The <u>Blank After</u> entry (OF, column 39) contains an invalid or incorrectly used entry.	<u>Blank</u> is assumed.
176	The <u>Packed Field</u> entry (OF, column 44) contains an invalid or incorrectly used entry.	<u>Blank</u> is assumed.
177	In the AND-relationship, columns 17-22 (OF) contain entries, but should be blank.	Columns 17-22 are ignored.
178	The <u>End Position in Output Record</u> entry (OF,	Card not processed.

	columns 40-43) is not present, not numeric, or incompatible with literal or edit word.	
179	The <u>Constant or Edit Word</u> entry (OF, columns 45-70) does not contain the required leading and/or closing quote signs.	Card not processed.
180	The numeric <u>Result Field</u> entry (CS, columns 43-48) has a length greater than 15 bytes.	Card not processed.
181	Incorrect alphameric or numeric field(s) have been specified for this <u>Move Zone</u> operation.	Card not processed.
182	In the <u>Alphameric Compare</u> operation, field lengths greater than 40 must be equal.	Field length of 40 is assumed.
183	The <u>Result Field</u> entry (CS, columns 43-48) contains an undefined field name.	Card not processed.
184	The <u>Field Name</u> entry (IS, columns 53-58, or OF, columns 32-37) contains an unreferenced field name.	No immediate action taken.
185	The <u>Field Name</u> entry (IS, columns 53-58; or OF, columns 32-37) contains a multidefined field name.	No immediate action taken.
186	An arithmetic operation is specified in <u>Operation</u> (CS, columns 28-32) with an alphameric factor in <u>Factor 1</u> (CS, columns 18-27), <u>Factor 2</u> (CS, columns 33-42), or <u>Result Field</u> (CS, columns 43-48).	Card not processed.
187	A <u>Compare</u> operation is specified in <u>Operation</u> (CS, columns 28-32) comparing an alphameric field with a numeric field.	Card not processed.
188	The result field length specified in <u>Field Length</u> (CS, columns 49-51) may be too small to contain the true result.	No immediate action taken.
189	The entries in <u>Factor 2</u> (CS, column 33-42) and <u>Result Field</u> (if specified) must be table names when the LOKUP operation code is used.	Card not processed.
190	EXTCV is not followed by the required operation code (KEYCV) or is not specified properly.	KEYCV card not processed.
191	In the TESTZ operation, a numeric field has been specified as the entry in <u>Result Field</u> (CS, columns 43-48).	Card not processed.
192	The label of the TAG operation and the corresponding GOTO are not in <u>Detail</u> or <u>Total</u> calculations.	No immediate action taken.
193	The <u>Half-Adjust</u> entry (CS, column 53) is incompatible with the result field decimal length.	<u>Half-Adjust</u> entry is ignored.
194	The lengths of Factor 1 and Factor 2 are not equal for the LOKUP operation.	Card not processed.
195	EXTCV operations must be the first to appear in the calculation specifications and no more than 4 are allowed.	Card not processed.
196	The <u>MVR</u> operation is not preceded by <u>DIV</u> operation.	<u>MVR</u> operation is ignored.
197	The <u>DIV</u> operation containing an entry in <u>Half-</u>	<u>MVR</u> operation is ignored.

	<u>Adjust</u> (CS, column 53) is followed by an <u>MVR</u> operation.	
198	In the <u>LOKUP</u> operation, Factor 1 and Factor 2 are not both alphameric or both numeric fields.	Card not processed.
199	HIGH and LOW resulting indicators are specified for the LOKUP operation.	<u>LOW</u> indicator is ignored.
200	No primary file is stated.	Job is terminated.
201	The <u>Form Type</u> entry does not contain F, E, C, or I.	Card not processed.
202	The <u>Filename</u> entry (FDS, columns 7-14) is invalid.	Card not processed.
203	More than 10 filenames specified.	Additional cards not processed.
204	The <u>Filename</u> entry (FDS, columns 7-14) is not referenced.	No immediate action taken.
205	The <u>Symbolic Device</u> entry (FDS, columns 47-52) is invalid.	Job is terminated.
206	The <u>Device</u> entry (FDS, columns 40-46) is invalid.	Job is terminated.
207	The <u>Filename</u> entry (FDS, columns 7-14) is multi-referenced.	No immediate action taken.
208	The file name is multidefined.	Card not processed.
212	Undefined Resulting Indicator.	No immediate action taken.
213	Unreferenced Resulting Indicator.	No immediate action taken.
214	The maximum number of indicators has been exceeded.	Job is terminated.
219	Undefined field name.	No immediate action taken.
220	Conversion label unreferenced.	No immediate action taken.
221	Sum of Matching Field lengths incompatible with other record type(s).	No immediate action taken.
222	Table Name multidefined.	No immediate action taken.
223	Matching field hierarchy violated.	No immediate action taken.
224	Split Chaining Fields improperly specified.	No immediate action taken.
225	Split Control Fields improperly specified.	No immediate action taken.
226	Split Matching Fields are not permitted.	No immediate action taken.
227	The Matching Field lengths are incompatible with other record type(s).	No immediate action taken.
228	The Control Field lengths are incompatible with other record type(s).	No immediate action taken.
229	The Chaining Field lengths are incompatible with other record type(s).	No immediate action taken.
230	The sum of the Control Field lengths is greater than 256.	No immediate action taken.

231	Field area exceeds record size.	No immediate action taken.
232	Printer file record is blank in columns 17-22 (OF).	<u>Space After</u> of 1 is assumed.
233	The <u>Stacker Select</u> entry (OF, column 16) is invalid.	No immediate action taken.
234	The <u>Space Before</u> entry (OF, column 17) is invalid.	<u>Space Before</u> of 1 is assumed.
235	The <u>Space After</u> entry (OF, column 18) is invalid.	<u>1</u> is assumed.
236	The <u>Skip Before</u> entry (OF, columns 19-20) is invalid.	<u>Blanks</u> are assumed.
237	The <u>Skip After</u> entry (OF, columns 21-22) is invalid.	<u>Blanks</u> are assumed.
238	The field to be packed is not numeric.	No immediate action taken.
299	The field to be zero-suppressed is not numeric.	No immediate action taken.
240	The Sterling option is requested on a nonnumeric field.	No immediate action taken.
241	Edit word is too small for the field.	No immediate action taken.
242	The field to be edited is not numeric.	No immediate action taken.
243	Both fixed and floating dollar sign have been specified.	No immediate action taken.
244	Both <u>CR</u> and <u>_</u> used for CREDIT.	No immediate action taken.
245	The output specifications are not present or are invalid for this program.	Job is terminated.
246	Page indicators are not numeric.	No immediate action taken.
247	The field length is greater than the field <u>End Position</u> indicates.	No immediate action taken.

Three complete sample programs are included in this section.

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 97. Figure 98 gives the complete listing resulting from the processing of this sample program.

The labels assigned to the fields into which the object program will place the information are as follows:

Field	Label
Customer Name	NAME
Invoice Date - Month	MONTH
Invoice Date - Day	DAY
Invoice Number	INVNO
Customer Number	CUSTNO
Customer Location	STATE
Customer Location	CITY
Invoice Amount	INVAMT

To produce an object program capable of writing the report shown in Figure 98 (Part 3), the programmer must prepare a source program as shown in Figure 99. The entries in the RPG specifications sheet are described here.

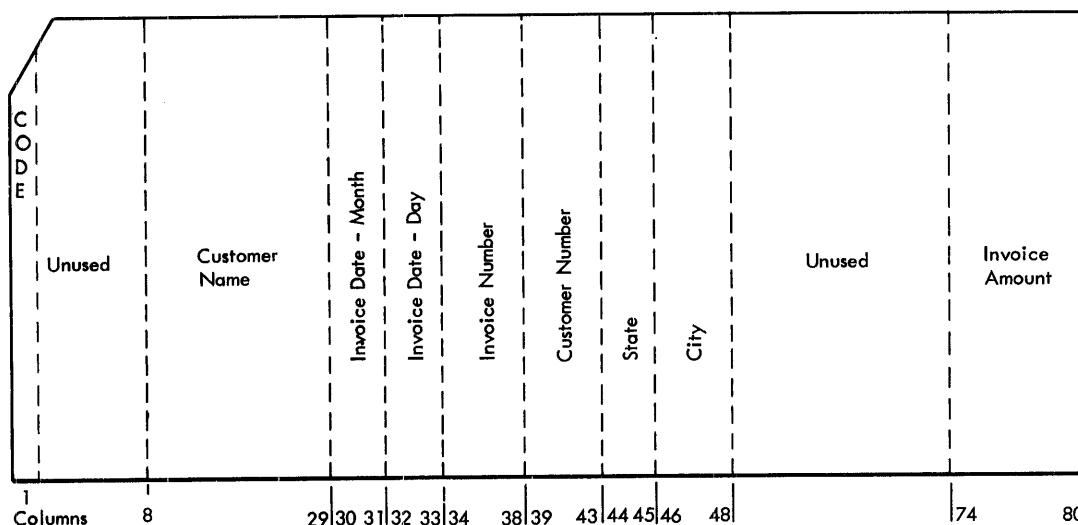


Figure 97. Input File-Card Format

SYSTEM/360 BOS RPG REPORT

SEQ.NO	PG	LIN	SPECIFICATIONS	COL. 6 - 74	ERRORS
00	000		H008	008	REPORT
1	01	01	FINPUT	IFE F 80 80	READ40 SYSIPT
2	01	02	FOUTPUT	O F 132 132	PRINTERSYSLST
3	02	010	IINPUT	AA 01 1 Z-	
4	02	020	I		8 29 NAME
5	02	030	I		30 31MONTH
6	02	040	I		32 33DAY
7	02	050	I		34 380INVNO
8	02	060	I		39 430CUSTNDL1
9	02	070	I		44 450STATE
10	02	080	I		46 480CITY
11	02	090	I		74 802INVAMT
12	03	010	C 01	INVAMT	ADD TOTAL
13	03	020	C 01	INVAMT	ADD GRPTOT
14	04	010	OOOUTPUT	H 201 1P	TOTAL 82
15	04	020	O	OR OF	GRPTOT 82
16	04	030	O		
17	04	040	O		53 @ ACCOUNTS R@
18	04	050	O		77 @ RECEIVABLE R E @
19	04	060	O	H 1 1P	88 @ R E G I S T E R @
20	04	070	O	OR OF	
21	04	080	O		75 @CUSTOMER@
22	04	090	O		80 @LOCATION INVOICE@
23	04	100	O		109 @INVOICE DATE INVOICE@
24	04	110	O	H 2 1P	
25	04	120	O	OR OF	
26	04	130	O		42 @NUMBER CUSTOMER @
27	04	140	O		46 @NAME@
28	04	150	O		79 @STATE CITY NUMBER@
29	04	160	O		108 @MO DAY AMOUNT@
30	05	010	O	D 2 01	
31	05	020	O		CUSTNDZ 22
32	05	030	O		NAME 53
33	05	040	O		STATE Z 59
34	05	050	O		CITY Z 67
35	05	060	O		INVNO Z 79
36	05	070	O		MONTH Z 89
37	05	080	O		DAY Z 97
38	05	090	O		INVAMT 109 @ \$, 0. @
39	05	100	O	T 2 L1	
40	05	110	O		GRPTOT B 109 @ \$, 0. @
41	05	120	O		110 @**@
42	05	130	O	T 2 LR	
43	05	140	O		TOTAL 109 @ \$, 0. @
44	05	150	O		111 @***@

IND. B/DIS IND. B/DIS IND. B/DIS INDICATORS IND. B/DIS IND. B/DIS IND. B/DIS

● Figure 98. Complete Listing of Sample Program One (Part 1 of 3)

```

MR      3 221  00      3 222  0A      3 223  0F      3 224  0V      3 225  1P      3 226  1O      3 227
L1      3 228  L2      3 229  L3      3 22A  L4      3 22B  L5      3 22C  L6      3 22D  L7      3 22E
L8      3 22F  L9      3 230  LR      3 231  H1      3 232  01      3 233

FIELD NAMES
FIELD  B/DIS  L  T  D  FIELD  B/DIS  L  T  D  FIELD  B/DIS  L  T  D  FIELD  B/DIS  L  T  D
NAME   3 234  22  A  D  MONTH  3 24A  2  N  0  DAY   3 24C  2  N  0  INVNO  3 24E  5  N  0
CUSTNO 3 251  5  N  0  STATE  3 254  2  N  0  CITY   3 256  3  N  0  INVAMT 3 258  7  N  2
TOTAL  3 25C  8  N  2  GRPTOT 3 261  8  N  2

LITERALS
LITERAL          LEN.  TYPE  B/DIS  LITERAL          LEN.  TYPE  B/DIS
A C C O U N T S R      24    A   3 266  E C E I V A B L E R E      24    A   3 27E
R E G I S T E R       15    A   3 296  C U S T O M E R           8     A   3 2A5
LOCATION              INVOICE      22    A   3 2AD  I N V O I C E   D A T E   I N V O I C E      23    A   3 2C3
NUMBER              CUSTOMER      24    A   3 2DA  N A M E           4     A   3 2F2
STATE              CITY          NUMBER  24    A   3 2F6  M O   D A Y   A M O U N T      21    A   3 30E
-----/-----      12    E   3 323  *                   1     A   3 32F
**                   2     A   3 330

EXTENDED CALC DIAGNOSTICS
SEQ. NO.  ERRORS
12        NOTE 188
13        NOTE 188

DIAGNOSTIC MESSAGE EXPLANATIONS
NOTE 188 RESULT FIELD LENGTH, COL. 49-51, MAY BE TOO SMALL TO CONTAIN TRUE ANSWER.

KEY ADDRESSES OF OBJECT PROGRAM
NAME OF ROUTINE      HEX ADDRESS      NAME OF ROUTINE      HEX ADDRESS
CLOSE FILES          01EA             H # D LINES          086E
TOTAL LINES          087C             DETAIL CALCS         06F8
TOTAL CALCS          073A             INPUT SPECS          05E0
CONTROL FLD          0658             LOW FIELD            06A2
FILE SEQ. 1          0358             FILE SEQ. 2          0460

CORE SIZE OF MACHINE IN DEC      08000      CORE STORAGE USED IN DEC      02678

```

● Figure 98. Complete Listing of Sample Program One (Part 2 of 3)

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 NUMBER	CUSTOMER NAME	LOCATION		INVOICE NUMBER	INVOICE DATE		INVOICE AMOUNT
		STATE	CITY		MO	DAY	
10712	AMALGAMATED CORP	33	61	11603	11	10	\$ 389.25
							\$ 389.25*
11315	BROWN WHOLESAL	30	231	12324	12	28	\$ 802.08
11315	BROWN WHOLESAL	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	12263	12	17	\$ 121.75
							\$ 121.75*
29017	GENERAL MFG CU	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*
							\$ 5,784.56**

● Figure 98. Complete Listing of Sample Program One (Part 3 of 3)

FILE DESCRIPTION SPECIFICATION SHEET

Two files (Input and Output) are described on this sheet. The input file INPUT (columns 7-11) is the primary file (column 16). It causes the end-of-job condition when it is depleted (column 17). The input records are fixed in length (column 19); the block length is 80 (columns 22-23); and the records are 80 characters in length (columns 26-27).

The output file OUTPUT is also defined on the file description sheet. The format is fixed length; the block length is 132 and the records are 132 characters in length. The entry OF in columns 33-34 indicate that the Output File defined on the line is to cause the overflow condition.

INPUT SPECIFICATION SHEET

The input file has a sequence of AA, and if column 1 contains the zone of a minus, resulting indicator 01 is turned 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 columns 46-47. The names of the input fields are entered in columns 53-58. Whenever a new customer is read in, control level 1 is turned on (columns 59-10).

CALCULATION SPECIFICATION SHEET

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 eight positions, and two positions are reserved for the decimal portion. The field GRPTOT is added to INVAMT, and the result is stored in GRPTOT which is eight positions long, and has two decimal positions.

OUTPUT-FORMAT SPECIFICATION SHEET

OUTPUT is the name of the file to which the records defined on the line belong, and OUTPUT is entered under Filename on the first line of the sheet. In column 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 columns 14 and 15 of the second line allows for printing on the first page or overflow. The other heading lines are to print on these conditions.

When output indicator 01 is on, the field entered in Field Name will print in the positions indicated in columns 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 96 of the previous program. Another input file (Master Customer File), which is on tape, contains information about the firm's customers (Figure 100). The fields contained in the two input files follow:

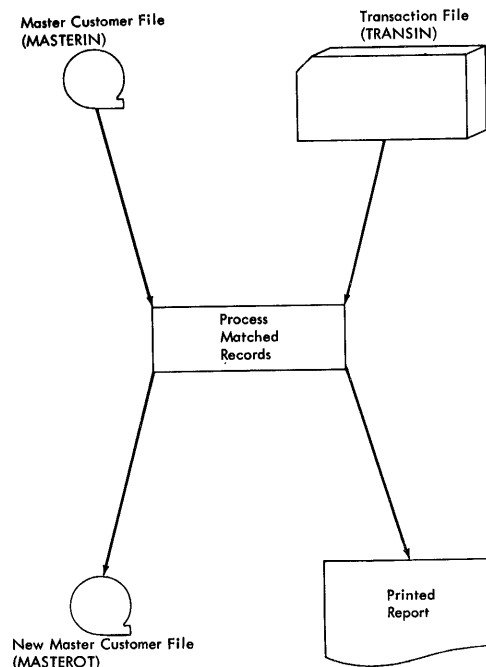


Figure 100. Sample Program Two

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	INVAMT	74-80

Master Customer File

Field	Label	Card Columns
Customer number	MASNUM	1-5
Customer name	MASNAM	6-27
Street	MASTRT	28-46
City	MASCTY	47-57
State	MASTAT	58-62
Customer balance	MASBAL	63-70
Date of last payment	PAYDAT	71-76
Date of last purchase	PAYPUR	77-82

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 101.

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION Form X24-334Z-7
REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS
 IBM System/360 Printed in U. S. A.

Date _____

Program _____

Punching Instruction: Graphic Punch

Page 1 2

Program Identification: **REPORT**

Line	Form Type	Filename	File Type	File Designation	Sequence	Block Length	Record Length	Mode of Processing	Length of Record	Record Address Type	Device	Symbolic Device	Name of Label Exit	Extent Exit for DAM	Comments
01	0	TRANSIN	IS	AF	02	80	80				READ40	SYSRDR			
02	0	MASTERINI	PEAF	300	100						TAPE	SYS001			
03	0	MASTERCOTO	F	300	100						TAPE	SYS002			
04	0	MASTLISTO	V	132	132			OF			PRINTERS	SYSLS			

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION Form X24-3350
REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS
 IBM System/360 Printed in U. S. A.

Date _____

Program _____

Punching Instruction: Graphic Punch

Page 1 2

Program Identification: **REPORT**

Line	Form Type	Filename	Sequence Number (1-9)	Reading Indicator	Record Identification Codes				Field Location		Field Name	Control Level (1-10)	Merging Fields or Changing Fields	Field Indicators		Sterling Sign Position
					Position	Character	Position	Character	From	To				Plus	Minus	
01	0	TRANSIN	AA	02	I	Z	-		2							
02	0	OR	03	I	Z	b										
03	0								8	29	NAME					
04	0								30	31	MONTH					
05	0								32	33	DAY					
06	0								34	38	INVNO					
07	0								39	43	CUSTNOLIM1					
08	0								44	45	STATE					
09	0								46	48	CITY					
10	0								74	80	INVTMT	02				
11	0								74	80	AMTPD	03				
12	0	BB	04	I	CD											
13	0								2		DATE					
14	0	MASTERINAA	01													
15	0								1	100	RECORD					
16	0								1	50	MASWMLIM1					
17	0								63	70	MASBAL					
18	0								71	76	PAYDAT					
19	0								77	82	PAYPUR					

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION Form X24-3351
REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS
 IBM System/360 Printed in U. S. A.

Date _____

Program _____

Punching Instruction: Graphic Punch

Page 1 2

Program Identification: **REPORT**

Line	Form Type	Control Level (1-10)	Indicators		Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Resulting Indicators		Comments
			And	And							Plus	Minus	
01	0		MR	02	MASBAL	ADD	INVTMT	MASBAL					
02	0		MR	03	MASBAL	SUB	AMTPD	MASBAL					
03	0		MR	02		MOVE	DATE	PAYDAT					
04	0		MR	03		MOVE	DATE	PAYPUR					

● Figure 101. Specifications for Sample Program Two (Part 1 of 2)



INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS
 IBM System/360

Form X24-3332
 Printed in U. S. A.

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic						
	Punch						

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Space				Skip				Output Indicators						Field Name	Zero Suppress (Z) Blank After (B)	End Position in Output Record	Punched Field (P)	Constant or Edit Word	Sterling Sign Position
			Type (R/D/T)	Standard Select	Before	After	Before	After	And	And	And	And	And	And	And	And						
0 1	0	MASTLISTH																				
0 2	0																					
0 3	0																			66		'DAILY TRANSACTION REPO'
0 4	0																			68		'RT'
0 5	0																					
0 6	0																					
0 7	0																			25		'CUSTOMER'
0 8	0																			80		'LOCATION INVOICE'
0 9	0																			109		'INVOICE DATE INVOICE'
1 0	0																					
1 1	0																					
1 2	0																			42		'NUMBER CUSTOMER'
1 3	0																			46		'NAME'
1 4	0																			79		'STATE CITY NUMBER'
1 5	0																			108		'MO DAY AMOUNT'



INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS
 IBM System/360

Form X24-3332
 Printed in U. S. A.

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic						
	Punch						

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Space				Skip				Output Indicators						Field Name	Zero Suppress (Z) Blank After (B)	End Position in Output Record	Punched Field (P)	Constant or Edit Word	Sterling Sign Position
			Type (R/D/T)	Standard Select	Before	After	Before	After	And	And	And	And	And	And	And	And						
0 1	0																					
0 2	0																					
0 3	0																					
0 4	0																					
0 5	0																					
0 6	0																					
0 7	0																					
0 8	0																					
0 9	0																					
1 0	0																					
1 1	0																					
1 2	0																					
1 3	0																					
1 4	0	MASTEROTD																				
1 5	0																					
1 6	0																					
1 7	0																					
1 8	0																					
1 9	0																					
2 0	0																					

E076510MSP

Card Electro Number _____

● Figure 101. Specifications for Sample Program Two (Part 2 of 2)

FILE DESCRIPTION SPECIFICATION SHEET

The four files are defined on this sheet. 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 sheet. 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 sheet) the master file is processed. Processing continues until all the records in the master customer file have been processed (indicated by the E in column 17). The input records contained in the master customer file are ascending in order, fixed in length, and have a block length of 300. Each record is 100 characters in length.

Note: A blank in column 53 indicates that there are no labels in the input file.

The file MASTLIST is the printed report. The format is variable. The length of the records can be 132. The overflow entry in columns 33-34 indicate 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 column 53 indicates that there are no labels in the output file.

INPUT SPECIFICATION SHEET

The two input files TRANSIN and MASTERIN are defined on the Input Specification sheet.

TRANSIN: The input records may be obtained from three types of cards. Sequence AA has been assigned to two types. If card column 1 contains the zone of a minus, resulting indicator 02 is turned on. If card column 1 contains the zone of a plus, indicator 03 is turned on. The cards that have a minus in column 1 are selected into the 2 pocket

(Column 42). The locations of the input records and their labels are defined in columns 46-58 of the sheet.

The field CUSTNO (customer number) has entries in columns 59-60 (Control Level) and columns 61-62 (Matching Fields) of the Input Specification sheet. Whenever a new customer number is read in, control level 1 (L1) is set on. This condition is tested on the output-format specification sheet 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 (MASNUM) in the MASTERIN file.

The fields INVAMT and AMTPD contain entries in column 63-64. Entering 02 in this column indicates that whenever resulting indicator 02 is on, the field INVAMT will be used for input data. Whenever 03 is on, the field AMTPD will be used.

The first card in the transaction file is a data card. It is assigned sequence BB. Whenever column 1 contains a D, indicator 04 is turned on. The date is contained in columns 2-5 of the card.

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 Specification sheet. MASNUM corresponds to CUSTNO in the transaction file. Whenever a new master number is read in, L1 is set. The entry M1 indicates that the master number will be matched with the customer number in the transaction file.

CALCULATION SPECIFICATION SHEET

Whenever the matching record indicator MR is on and indicator 02 is on, the contents of the field INVAMT are added to the MASHBAL. The result is stored in MASHBAL. The date is moved to the field PAYDAT.

Whenever the matching record indicator MR is on and indicator 03 is on, AMTPD is subtracted from MASHBAL, and the result is stored in MASHBAL. The date is moved to PAYPUR.

OUTPUT-FORMAT SPECIFICATION SHEET

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 (resulting 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

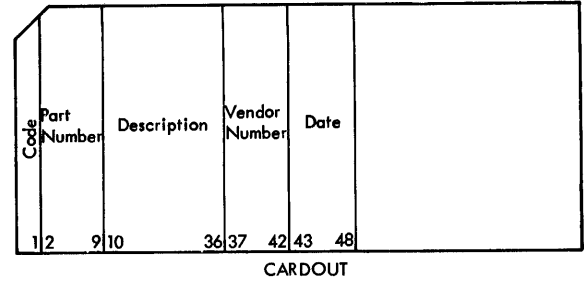
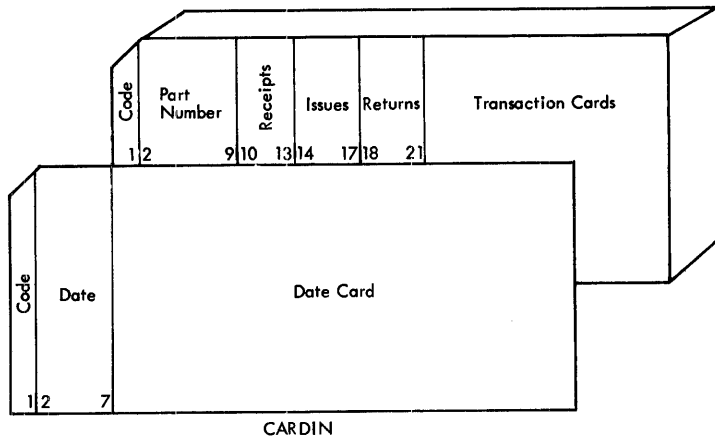
on the master customer file, but the customer's name and address are retained.

The specifications for the printed report are listed under the name of the output file MASTLIST.

SAMPLE PROGRAM THREE

The third sample program (Figures 102 and 103) illustrates some of the more complex capabilities of PRG.

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.



Code	Part Number	Description	Vendor Number	Mini mum	On Hand
12	9	10	36	37	42
			50	53	54
			58		
					125

INVFIL

INVENTORY LISTING								11/26/64	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	2,000	500	1,000		1,500	BELOW MINIMUM	
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	255	100	245		110	BELOW MINIMUM	
0 0 1 0	NO DISK RECORD FOR THIS PART								

PRINTOUT

● Figure 102. Input and Output Formats for Sample Program Three



INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS
 IBM System/360

Form X24-3347
 Printed in U. S. A.

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic Punch								
----------------------	---------------	--	--	--	--	--	--	--	--

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	File Type				Mode of Processing				Device	Symbolic Device	Name of Label Exit	Extent Exit for DAM	Comments	
			File Designation	End of File	Sequence	File Format	Length of Record Address Field	Record Address Type	Type of File Organization	Overflow Indicator						
			I/O/U/C	P/S/C/R/T	A/D	F/V	Block Length	Record Length	L/R	K/I	I/D/T	Key Field Starting Location	Extension Code E/L			
01	F	INVFIL	UC	F	125	125	R	KD						DISK11	SYS004S	
02	F	CARDIN	IP	F	80	80								EREAD40	SYSIPT	
03	F	CARDOUT	O	F	80	80								READ40	SYSOPT	
04	F	PRINTOUT	.V		132	132			OF					PRINTER	SYSLS	
05	F															



INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR FILE EXTENSION SPECIFICATIONS
 IBM System/360

Form X24-3348-1
 Printed in U. S. A.

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic Punch								
----------------------	---------------	--	--	--	--	--	--	--	--

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Record Sequence of the Chaining File		Table Name	Number of Table Entries Per Record		Table Name (Alternating Table)	Length of Table Entry		Comments
		From Filename	To Filename		Number of Table Entries Per Table	Length of Table Entry		Packed (P)	Decimal Pos. Sequence (A/D)	
01	F	CARDIN	INVFIL	CONRTN						
02	F									

● Figure 103. Specifications for Sample Program Three (Part 1 of 4)

INPUT SPECIFICATIONS

The input card and DASD files are described on the Input Specification sheet. Each field that will be used is defined. The C1 in columns 61-62 of the transaction card file designates the field labeled DTLPAR as the chaining field. The conversion routine CONRTN, indicated on the Calculation Specification sheet, will make the DTLPAR field available to the user's external conversion routine.

CALCULATION SPECIFICATIONS

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. Note that the address can be up to eight positions long. The external routine may be in the relocatable library so that it can be automatically retrieved by the Basic Operating System program. The routine could also be punched in cards that follow the RPG cards.

The operation code KEYCV states that the record key can be obtained directly from the field DTLPAR. This is the first entry on the Calculation Specification sheet.

The calculations in lines 030-070 require the presence of matching chaining (CARDIN) and chained (INVFIL) records. This condition is indicated by the simultaneous setting of resulting indicators 01 and 03. The ONHAND field from the INVFIL record is reset added to a work area (SAVE). Data fields RECPTS and RETURN from CARDIN records are added to SAVE; ISSUES are subtracted. When each input card for an INVFIL record is processed, the resulting new ONHAND field in SAVE is compared to the minimum balance. If ONHAND equals the minimum, resulting indicator 05 is set on. If ONHAND drops below the minimum, resulting indicator 04 is set on.

OUTPUT SPECIFICATIONS

The PRINTOUT file's heading information can be printed under control of either resulting indicator 02, which is set for the date card (the first card in the CARDIN file), or overflow (OF). The OF indicator will govern all heading printing after the first page. The entry PAGE in line 030 causes the page number to be updated automatically for each new page.

The detail line described by the entries in lines 150 of Figure 103 (Part 3) through 050 of Figure 103 (Part 4) requires the presence of both the CARDIN and INVFIL records (resulting indicators 01 and 03 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 of a CARDIN record, and no corresponding INVFIL record occurs. This condition is identified by resulting indicator 01 being off when indicator 03 is on.

The exception file CARDOUT will have a card punched for each transaction that results in a below-minimum or at-minimum stock level. These cards will contain the part number and description, vendor number, date, and the E or B code for EXPEDITE or BELOW MINIMUM. Below-minimum cards, identified by the simultaneous ON settings of indicators 01, 03, and 04, will be selected to stacker 4. At-minimum cards, with indicators 01, 03, and 05 on, will be selected into stacker 8.

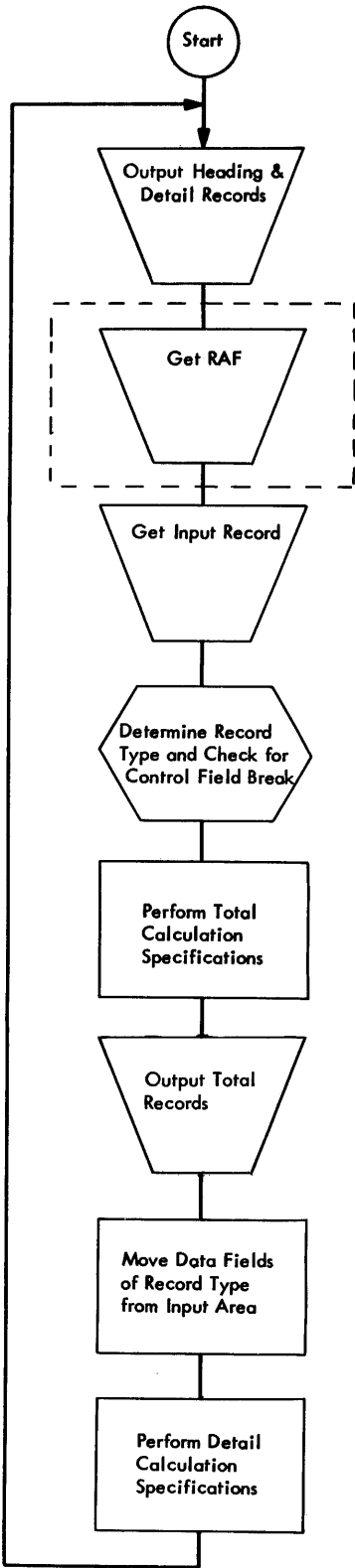
Lines 180 and 190 provide for the updating and writing out of the INVFIL records. If indicators 01 and 03 are both on, indicating that the INVFIL record has been updated by a CARDIN transaction, the new ONHAND is moved into its INVFIL location from the work area SAVE, and the record is written.

APPENDIX A: INDICATOR CHART

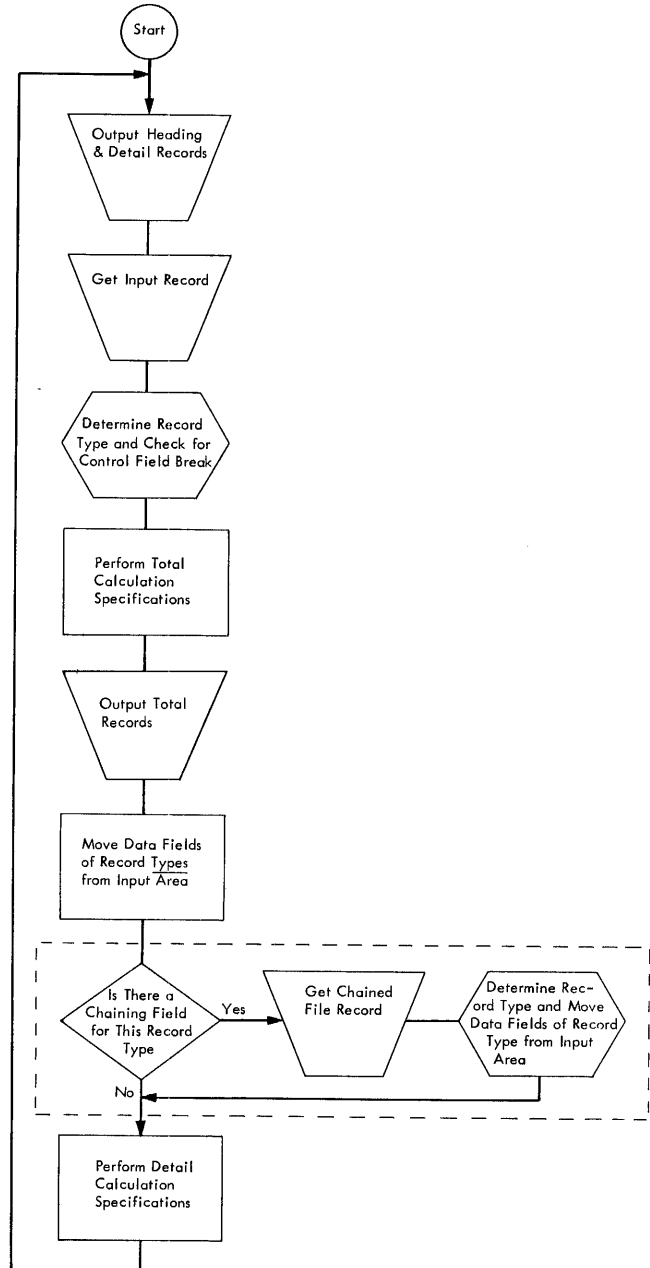
Indicators	Where Specified	Where Used	Turned On	Turned Off	Notes
Resulting Indicator 00	Internal	Output Indicators on Output Specifications (Columns 24 - 25, 27 - 28, 30 - 31)	This indicator is always ON	Can never be turned OFF	
Resulting Indicators (01-99)	Input Specifications (Columns 19-20)	Indicators on Calculation Specifications (Columns 10-11, 13-14, 16-17) Output Indicators on Output Specifications (Columns 24-25, 27-28, 30-31)	When specified record type has been read and is ready for processing.	Before the first record is read on the next processing cycle.	Turning OFF and ON can also be accomplished by using SETON and SETOF operation codes.
	Calculation Specifications (Columns 54-55, 56-57, 58-59)	Same as above.	Whenever the specified field status condition is satisfied.	The next time that this field status is to be tested	Same as above
Field Indicators (01-99)	Input Specifications (Columns 65-66, 67-68, 69-70)	Same as above	Same as above	Same as above	Same as above
Halt Indicators (H1 - H9)	Input Specifications (Columns 65-66, 67-68, 69-70) Calculation Specifications (Columns 54-55, 56-57, 58-59)	Same as above	Same as above	Can only be turned OFF by SETOF operation code. See Note.	If these indicators remain ON, the object program will terminate before reading the next record.
Control Level Indicators (L1 - L9)	Input Specifications (Columns 59-60)	Same as above and Calculation Specifications (Columns 7-8)	When the value in a control field changes. All indicators of the lower levels are also turned ON.	Before the first record is read on the next processing cycle.	Turning OFF and ON can be accomplished by using SETON and SETOF operation codes.
L0	Internal	Indicators on Calculations Specifications (Columns 10-11, 13-14, 16-17) Output Indicators on Output Specifications (Cols. 24-25, 27-28, 30-31)	This indicator is always ON.	Can never be turned OFF.	
LR	Internal	Same as above	After processing the last record of the last file.	Same as above	All Control Level Indicators (L1-L9) are also turned ON when the LR is turned ON.
MR	Internal	Same as above	When multiple input files and the matching fields specification are used, this indicator is turned ON if a secondary file record matches the primary file record.	Before the first record is read on the next processing cycle.	
Overflow Indicators OA, OF, OV	File Description (Columns 33-34)	Same as above	When Channel 12 of the carriage control tape is sensed.	After the detail and heading records are written.	These indicators remain ON for one complete processing cycle.
IP	Internal	Output Indicators on Output Specifications (Columns 24-25, 27-28, 30-31)	This indicator is ON at the beginning of processing before any records are read.	Before the first record is read.	This indicator is used to govern printing of the first page of the report.

All indicators specified in columns 69 - 70 of the Input Specification sheet and columns 58 - 59 of the Calculation Specification sheet are ON at the beginning of processing before any records are read. This is always the case except when the operation codes COMP, LOKUP, SETON, and SETOF are specified.

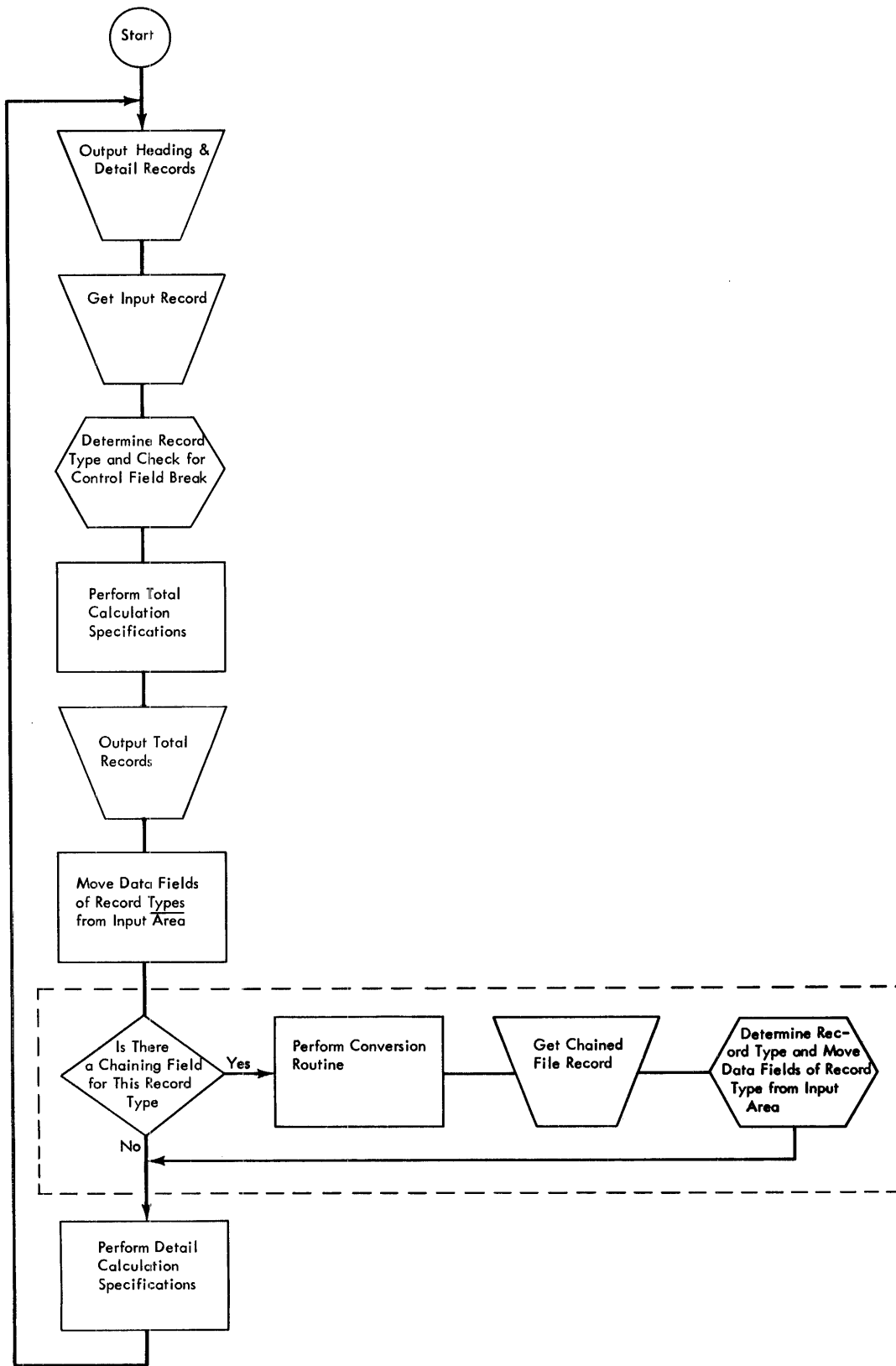
APPENDIX B: RPG LOGIC FLOW CHARTS



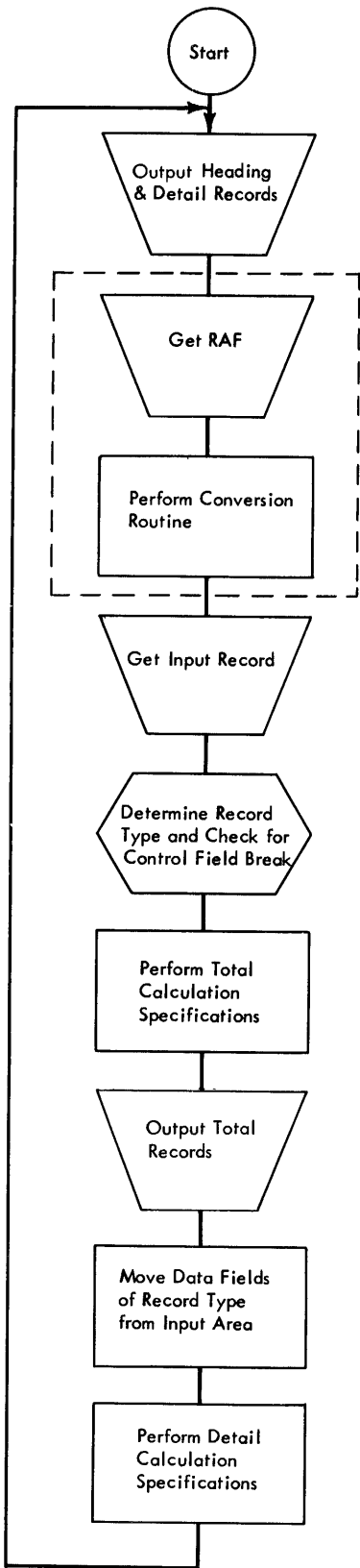
1. General Logic for RPG Object Program with RAF File



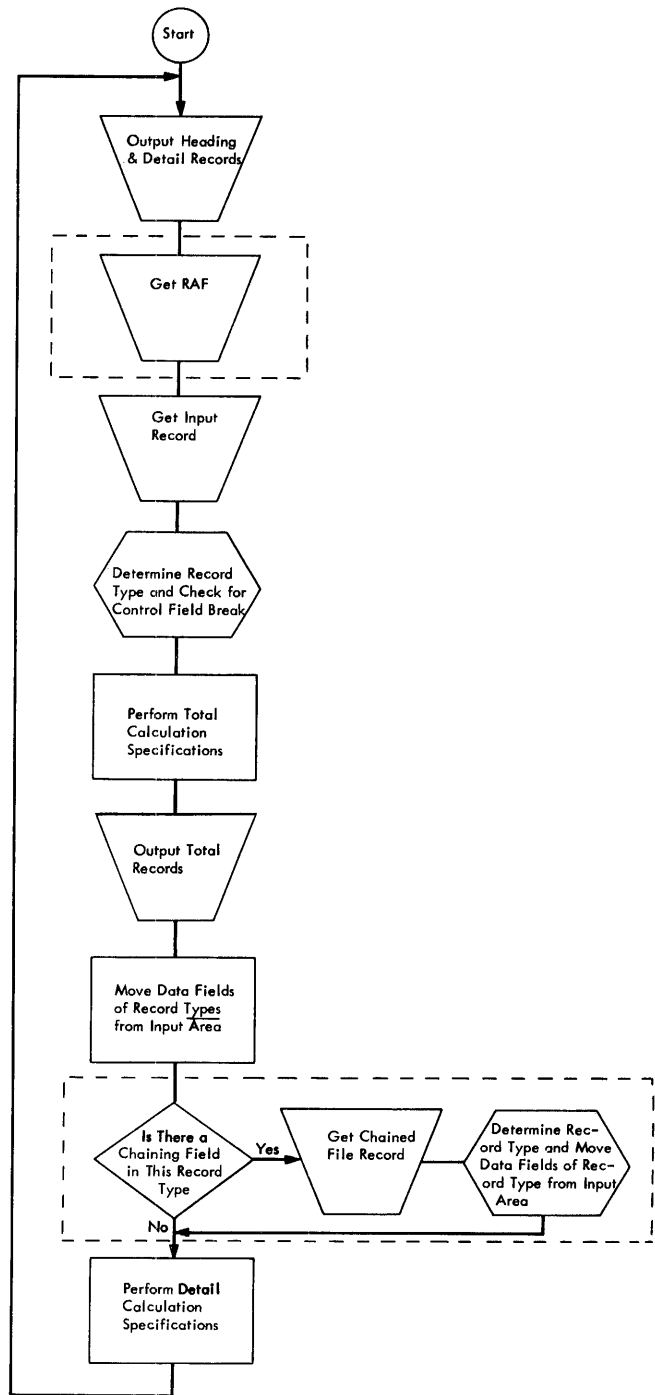
2. General Logic for RPG Object Program with a Chaining File



3. General Logic for RPG Object Program with Chaining File which Requires Conversion



4. General Logic for RPG Object Program with RAF File which Requires Conversion



5. General Logic for RPG Object Program with RAF and Chaining Files

APPENDIX C: STERLING ROUTINES FOR THE REPORT PROGRAM GENERATOR

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, file extension, and calculation 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 BSI or IBM input files of one program must use the same code combination throughout.

INPUT SPECIFICATIONS

The position of the sign must be specified in columns 71-74. Enter an S in column 74, if the sign is in the normal position. If the pence field has decimal positions, the normal position of the sign is in the rightmost decimal position of the pence field. If the pence field has no decimal positions, the normal 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 column 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 WHICH IS NOT PRINTED: Sterling output fields are specified in the same manner as sterling input fields.

PRINTED OUTPUT: Insert the letter S in column 74 of the Output-Format Specification sheet.

Note 1: Shillings and pence are printed in two positions each, with zero suppression in effect in the tens positions of each field.

Note 2: If a field is defined as a sterling field in the input but not in the output specifications, the output will be in pence format.

Note 3: The rules governing the use of edit control words are the same as those for decimal fields.

The following features are available:

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 penny 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.

Note: If column 38 of the Output-Format Specification sheet contains Z, the entire pounds field will be zero-suppressed. The ten's position of the shillings and pence fields will be zero-suppressed, and the sign of the field will be blanked out.

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 Format-1 field, the highest amount it is permitted to contain is 2,399,999,999,999,999.

POUND STERLING FORMATS

RPG will support, on the input and output fields, two standards for pence and shilling portions of sterling fields: IBM or BSI. Columns 17-20 of the RPG Processor Control Card indicate either the IBM or BSI formats. The formats for IBM and BSI are listed here.

COLUMN 17 (STERLING-SHILLING FIELD ON INPUT) IBM FORMAT: Two positions are allowed for the shilling option in the input field: 00-19 for 0 to 19 shillings.

BSI FORMAT: The shilling option in the input fields is indicated as listed here:

- 1-9 Shillings by a 1-9 punch.
- 10 Shillings by a 12-punch.
- 11-19 Shillings by a A-I punch.

COLUMN 18 (STERLING-PENCE FIELD ON INPUT) IBM FORMAT: The pence option on the input field is as listed here:

- 1-9 Pence by a 1-9 punch,
- 10 Pence by an 11-punch,
- 11 Pence by a 12-punch.

BSI FORMAT: The pence option on the input field is as listed here:

- 1-9 Pence by a 1-9 punch,
- 10 Pence by a 12-punch,
- 11 Pence by an 11-punch.

COLUMN 19 (STERLING-SHILLINGS FIELD ON OUTPUT) IBM FORMAT: Two positions are allowed for the shilling option on the output field:

- 00-19 for 0-19 shillings.

BSI FORMAT: The shilling option on the output field is as listed here:

- 1-9 Shillings by a 1-9 punch,
- 10 Shillings by a 12-punch,
- 11-19 Shillings by A-I punch,

COLUMN 20 (STERLING-PENCE FIELD ON OUTPUT) IBM FORMAT: The pence option on the output field is as listed here:

- 1-9 Pence by a 1-9 punch,
- 10 Pence by an 11-punch,
- 11 Pence by a 12-punch.

BSI FORMAT: The pence option on the output field is as listed here:

- 1-9 Pence by a 1-9 punch.
- 10 Pence by a 12-punch,
- 11 Pence by an 11-punch.

EDITING OF STERLING FIELDS

RPG provides for the editing of sterling fields, with columns 45-70 of the Output-Format Specification sheet containing the edit words. Each edit word is composed of three portions, or fields: the pounds field, the shillings field, and the pence field. The RPG program enables the user to make use of the following:

1. Floating and fixed pound signs.
2. Zero suppression of the pounds, shillings, and pence fields, or in any combination.
3. CR and minus (-) symbols.
4. Asterisk protection.
5. An ampersand causes the insertion of a blank in the edit word.

Rules for Forming Edit Words for a Sterling Field

1. An edit word must be enclosed by a set of single-quote symbols.
2. At least one separator character, or delimiter, must be inserted between the pounds and shillings fields and the shillings and pence fields. Any character can be used as a delimiter except an asterisk (*), a zero (0), a decimal point (.), and a comma (,). However, a comma is permitted within the pounds field, and a decimal point is acceptable within the pence field.
3. Asterisk protection if desired must be specified in the pounds field. This provides asterisk protection in both the pounds field and the shillings field.
4. When specifying the floating pound sign, there must be at least one pounds field position preceding the shillings field and following the pound sign.
5. For each field in which zero suppression is desired, it must be specified as such. Zero suppression in a field

occurs only if a zero appears in that field.

6. No constant information can appear in an edit word.
7. The edit word must always be at least 2 positions larger than the input field on which the editing is to be performed.
8. The number of blank spaces supplied in the pence field of the edit word must be 2 greater than the number of decimal positions specified in column 52 of the Input or Calculation Specification sheets.
9. There must always be at least 2 positions allowed for the shillings field in every edit word specification.
10. When zero suppression is to be performed on the pence field, the user may wish to have the delimiter blanked out if the pence field is zero. To do this, the pence sign (d) must occupy the position in the edit word immedi-

ately following the two blank positions in the pence field.

The following examples illustrate valid edit words for sterling fields. £ denotes the pound sign, s the shilling sign, and d the pence sign. The output data to be edited is 005070L.

Edit Word	Result of Editing on Printout
No Editing	005_7_L
Zero Suppression (No Editing)	5_7_3
b£0b/0b/0bCR	£5/_7/_3CR
bb£0&bb&bb-	£5__7__3-
£bbb&bbSbbd-	£_5__7S_3d-
*b0b/bbSbbd	**5/*7S_3d
b£0b:bb:bb&.d	£5:_7:_3_.d

APPENDIX D: 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
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.	Any other RPG calculations.

- ADD (Add) 60
 Addition and Subtraction Example 10,11
 ADDROUT (Address Output) Option,
 Using the 116, 117
 Alphabetic Entries 41, 42
 Alphameric Literals 59
 Alphameric Literals in an Output Record,
 Rules for Forming 81
 Alternate Collating Sequence 131
 Alternating Tables 97
 AND Relationship 45
 Appendix A 163
 Appendix B 164
 Appendix C 167
 Appendix D 170
 Argument (Table) 95
 Arithmetic Operations 60
 Arrangement, RPG Source Deck 130
 Automatic Skipping 73

 Blank After 78
 Block Length 86
 Blocked Records (Variable-Length) 132
 Blocking Records 108, 132
 Branching and Exit Operations 64

 Calculation Specification Sheet 38,56,69
 Capabilities of Program 2
 Chaining 124,125,126
 Chaining Fields 54
 Chaining File 91
 Character 44
 Collating Sequence, Alternate 131
 Combined File 85
 Combined Files Stacker Select 47
 Comments 38,69
 Common Fields on RPG Specification Sheets
 38,39
 COMP (Compare) 63
 Comparison of Two Fields 23,24
 Compatibility with Model 20 and BPS RPG
 Programs 5
 Configuration, System 6
 Consecutive File Processing 109
 Constant or Edit Word 80
 Constants 50,80
 CONTD 115
 Control Card, RPG Processor 130,131
 Control Fields 13
 Control Fields, Split 51
 Control Level 51,56,57
 Conversion of Data Fields 115
 Conversion Routine Operations 66,115,170
 Correlation of RPG Specification Sheets
 29, 30
 C/Z/D 44

 DASD File Processing 110,111,114
 DASD Layout Chart 34
 Data Conversion 115
 Decimal Point Location 11
 Decimal Positions 49,67,92,93
 Detail Calculations 56
 Detail Labor File Problem 8,9,10
 Detail Printing 11,12,14
 Detail Records 72,103

 Detail Time 33
 Device 88
 Diagnostic Messages 135
 Direct Organization 92
 Disk Storage Concepts 107
 DIV (Divide) 61

 Edit Words 81,82
 Editing of Sterling Fields 168
 End of File 86
 End Position in Output Record 78,79
 Exit, General Rules for Using 103,104
 Exit, How to Code 103
 Exit Entries, Position of 103
 Exit Operations 64
 Exit Routine, Using Indicators and Fields
 in the 104
 Exit to a Translate Subroutine 53
 Exit to a User's Routine 103, 105
 EXTCV (External Conversion Routine)
 66,115,170
 Extent Exit for DAM 89

 Factor 1 59
 Factor 2 59,66
 Field Description 40,47,71,76
 Field Indicators 54
 Field Length 67
 Field Location 47
 Field Name 49,78
 Field Name Restrictions 50
 Field-Record Relation 54
 Field-Record Relation, Using Split Control
 Fields With 51,52
 File Description Specification Sheet
 38,84,89
 File Designation 85
 File Extension Specification Sheet 38,91,93
 File Format 86
 File Identification 71
 File Name 40,71,84,85
 File Type 84,85
 Files, Maximum Number Permitted 84
 Fixed-Length Records 109
 Form Type 38
 From 48
 From Filename 91
 Function (Table) 95
 Function of RPG 5
 Fundamentals of RPG Programming 8

 GOTO (Branching or Go To) 64
 Group Indication 18
 Group Printing 16

 Half-Adjust 67
 Halt Indicators 55
 Heading Records 72

 Indexed-Sequential File Organization 108,110
 Indexed-Sequential File Processing
 109,111,113,128
 Indicator Chart 163
 Indicator Codes for Plus, Minus, and Zero,
 or Blank, Types of 54
 Indicators 57,58

Indicators and Fields in the Exit Routine, Using	104	Overflow Indicators	73,88
Input Files	84,86	Overflow Printing	19
Input Files, Processing Multiple	119	Output File	84,85
Input Specification Sheet	38,40	Output-Format Specification Sheet	38,71
Inverted Print Option	131		
		Packed	47,79,92
Key Field Starting Location	88	Page Numbering	38,79,80
KEYCV (Record Key)	66,115	Plus	54
		Plus Balance, Testing for	20,21
L0 (Level Zero) Indicator	57,163	Position	44
Label	88	Pound Sterling Formats	168
Label Exit, Name of	89	Preface	2
Last Record Indicator (LR)	57,163	Printer Spacing Chart	34,35
Length of Record Address Field	87	Printing Lines Conditioned by Overflow	73,74
Length of Table Entry	92	Problem Definition	34
Level Zero Indicator (L0)	57,163	Program Capabilities	2
Limits of an Indexed-Sequential Organization, Processing	127	Program Identification	39,131
Line Identification Codes	35	Program Logic	32,33
Line Number	38		
Literals, Alphanumeric	59	Random File Organization	109
Literals, Numeric	59	Random File Processing	109,110,127
Logic Flow of an RPG Object Program	33	Record Address File (RAF)	91,111,114,127
Logical File Vs Physical Unit	107	Record Identification Codes	44,45
LOKUP (Table Lookup)	66	Record Key (KEYCV)	66,115
LR (Last Record) Indicator	57,163	Record Length	86
		Record Sequence	91
Machine Features, Required and Supported	6,7	Records in an OR Relationship	48
Matched Records, Order of Processing	123	Registers, Use of	104
Matching Fields	53,119,121,122,123	Result Field	66,67
Matching Fields or Chaining Fields	52	Resulting Indicators	22,43,67,68
Matching Record Indicator (MR)	122	Retrieving Updated Tables	100
MHHZO (Move High-to-High Zone)	63	RLABL (RPG Label)	64,104
MHLZO (Move High-to-Low Zone)	63	RPG Logic Flow Charts	164
Minus	54	RPG Processor Control Card	130,131
Minus Balance, Testing for	20,21,23	RPG Programming Fundamentals	8
MLHZO (Move Low-to-High Zone)	63	RPG Source Deck Arrangement	130
MLLZO (Move Low-to-Low Zone)	63	RPG Specification Sheets	36,37
Mode of File Processing	86,87		
MOVE (Move)	62	Sample Program 1	145
Move Operations	61,62	Sample Program 2	151
MOVEL (Move Left)	62	Sample Program 3	156
MR (Matching Record) Indicator	122	Sequence	41,86,92,93
MULT (Multiply)	61	Sequence Checking	27,53,86
Multiple Printers	74	Sequence of Assigning Matching Fields	123
Multiplication and Division Example	25	Sequential File Organization	108,110
MVR (Move Remainder)	61	Sequential File Processing	109,111,128
		SETOF (Set Indicators Off)	65
Name of Label Exit	89	SETON (Set Indicators On)	65
Nonstandard Labels	88	Skip After	72
Not	44,75	Skip Before	72,73
Number	43,91	Skipping Automatic	73
Number of Table Entries Per Record	92	Source Deck Arrangement, RPG	130
Number of Table Entries Per Table	92	Space After	72,73
Numeric Entries	41,42	Space Before	72,73
Numeric Literals	59	Specifying the Same Data Field as Alphanumeric and Numeric	50
		Split Chaining Files	126
Operation	60	Split Control Fields	51
Option	43	Stacker Select	46,72
OR Relationship, Records in an	48	Stacker Select, Combined Files	47
OR Relationship, Types of	46	Standard Labels	89
Output Indicators	74,75,76,77	Sterling	55,83
Overflow, Printing Lines Conditioned by	73,74	Sterling Field, Rules for Forming Edit Words for a	168
		Sterling Routines for RPG	167

SUB (Subtract) 61
 Symbolic Device 88
 System Configuration 6

 Table Example, Coding Sheets for 102
 Table File 91
 Table Lookup 66,98
 Table Name 92
 Tables, Methods of Processing 98
 Tables, Retrieving Updated 100
 Tables, Rules for Forming 96
 Tables in the Object Program, Using 96
 Tables Using Alternating Arguments and
 Functions 100
 TAG (Providing a Label for GOTO) 65
 Tape Layout Chart 34
 Testing for Zero, Plus, and Minus Balance
 20,21
 Testing or Compare Operations 63
 Testing the Results of a Calculation 67
 TESTZ (Test Zone) 64
 To 48
 To Filename 91

Total Calculations 14,56
 Total Printing 14
 Total Records 72,103
 Translate Subroutine, Exit to a 53
 Turning Indicators On or Off 65
 Type H/D/T 72
 Type of File Organization 87
 Type of Record Address 87

 ULABL (User's Label) 64,104
 Update File 85
 Use of Registers 104
 Using Resulting Indicators 22
 Using RPG 5

 Variable-Length Records 108,132

 Z-ADD (Zero and Add) 60
 Z-SUB (Zero and Subtract) 61
 Zero Balance, Testing for 20,21,22
 Zero or Blank 54
 Zero Suppress 78
 Zone Record Identification 44



International Business Machines Corporation

Data Processing Division

112 East Post Road, White Plains, N. Y. 10601

READER'S COMMENT FORM

IBM Basic Operating System/360
Specifications
Report Program Generator (8K Disk)

Form C24-3387-2

- Your comments, accompanied by answers to the following questions, help us produce better publications for your use. If your answer to a question is "No" or requires qualification, please explain in the space provided below. All comments will be handled on a non-confidential basis. Copies of this and other IBM publications can be obtained through IBM Branch Offices.

- | | Yes | No |
|--|---|--------------------------|
| ● Does this publication meet your needs? | <input type="checkbox"/> | <input type="checkbox"/> |
| ● Did you find the material: | | |
| Easy to read and understand? | <input type="checkbox"/> | <input type="checkbox"/> |
| Organized for convenient use? | <input type="checkbox"/> | <input type="checkbox"/> |
| Complete? | <input type="checkbox"/> | <input type="checkbox"/> |
| Well illustrated? | <input type="checkbox"/> | <input type="checkbox"/> |
| Written for your technical level? | <input type="checkbox"/> | <input type="checkbox"/> |
| ● What is your occupation? _____ | | |
| ● How do you use this publication? | | |
| As an introduction to the subject? <input type="checkbox"/> | As an instructor in a class? <input type="checkbox"/> | |
| For advanced knowledge of the subject? <input type="checkbox"/> | As a student in a class? <input type="checkbox"/> | |
| For information about operating procedures? <input type="checkbox"/> | As a reference manual? <input type="checkbox"/> | |
| Other _____ | | |

COMMENTS:

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

Fold

Fold

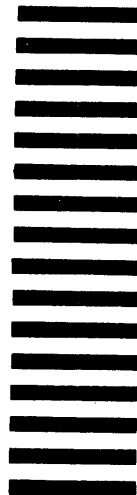
FIRST CLASS
PERMIT NO 170
ENDICOTT, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY

IBM Corporation
P. O. Box 6
Endicott, N. Y. 13764

Attention: Programming Publications, Dept. 157



Cut Along Line

IBM S360

Printed in U. S. A.

C24-3387-2

Fold

Fold



International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N. Y. 10601

Additional Comments: