File No. S360(Mod.20)-28
Order No. GC26-3600-7

CPS

IBM Systems Reference Library

IBM System/360 Model 20
Card Programming Support
Report Program Generator

This publication presents the complete programming specifications for the Model 20 Card Report Program Generator (RPG) -- a problem-oriented programming language.

The reader is assumed to have some understanding of punched-card data processing, but need not have any experience in programming or electronic data processing methods.

This manual also includes performance specifications, a list of machine features and units used by the program, numerous illustrations, four complete programming examples, and appendixes that amplify explanations and provide helpful programming hints.

PREFACE

This publication describes the Report Program Generator (RPG) for programming punched-card data processing applications on an IBM System/360 Model 20 computer. RPG is an easy-to-use programming language that does not require any prior programming or data processing experience. In conjunction with the Model 20, RPG combines into an integrated operation the functions performed separately by the following IBM unit record equipment:

- Reproducing punches
- Collators
- Printers
- Summary punches
- Interpreters
- Calculators.

The user is expected to be primarily familiar with his applications and his Model 20, rather than with the technical aspects of machine-oriented programming languages. Experience with unit record or data processing systems equipment and procedures will be helpful, but is not a prerequisite to an understanding, or utilization, of RPG.

However, familiarity with the concepts of punched-card records and procedures is assumed: programming by any language and for any data processing system always presupposes problem definition, and can do no more than instruct the system to execute the data processing steps previously planned by the user.

This manual contains the information necessary for programming jobs for the Model 20 with the RPG language for punched cards. It is intended as a reference text. Extensive explanatory and illustrative material, as well as programming tips and technical data, is also included to minimize the need to consult additional sources.

For a list of associated publications and their abstracts, see IBM System/360 Model 20 Bibliography, Order No. GA26-3565. Readers without previous data processing systems experience may find particularly useful information in IBM System/360 Model 20, Introduction and System Summary, Order No. GA26-5889.

Eighth Edition (October, 1970)

This is a reprint of GC26-3600-6 incorporating changes released in the following Technical Newsletters:

GN33-8593 (dated February 28, 1969)
GN33-8601 (dated April 15, 1969)
GN33-8611 (dated April 6, 1970).

This edition applies to version 2, modification 7, of IBM System/360 Model 20, Card Programming Support, Report Program Generator, and to all subsequent versions and modifications until otherwise indicated in new editions or Technical Newsletters.

Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest SRL Newsletter, Order No. GN20-0361, for the editions that are applicable and current.

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

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Laboratory, Publications Dept., P.O.Box 24, Uithoorn, Netherlands.

CONTENTS

PROGRAMMING

Programming consists essentially of writing instructions that can be understood by a data processing system. Before programming is attempted, the data processing problem must have been analyzed, and the step-by-step procedural requirements determined. The nature of the source data (input), the manipulations (calculations) to be performed on it, and the nature and form of the results (output) desired must have been defined.

THE NATURE OF RPG

The Report Program Generator (RPG) utilizes the abilities of the Model 20 system itself to convert data processing instructions written in natural quasi-English (RPG-language) statements to the language in which the central processing unit accepts its instructions. In many instances, one RPG-language statement will automatically be translated to several machine-language instructions.

The programmer using RPG writes statements in a sequence that comes naturally once the problem has been defined and the procedure determined. The expressions used largely consist of terms the programmer himself may coin, or of easily recognized mnemonics. The user must merely follow relatively simple rules. He need not be familiar with machine language, nor with "programming" in the technical sense.

OTHER PROGRAMMING LANGUAGES

The RPG language is easy to learn and to apply, and capable of handling almost every punched-card job requirement for the IBM System/360 Model 20. However, IBM currently provides two additional programming languages to satisfy special conditions:

1. Basic Assembler Language (B.A.L.) (Refer to SRL publication IBM System/360 Model 20, Card Programming Support, Basic Assembler Language, Order No. GC26-3602.

   B.A.L. provides for programming in the symbolic equivalent of actual Model 20 machine language. Effective utilization of B.A.L. requires some familiarity with the actual machine language (see IBM System/360 Model 20 Functional Characteristics, Order No. GA26-5847), and involves considerable experience with programming for electronic data processing systems as well as the component units of the system and their time relationships.

   As an adjunct to B.A.L., IBM also provides an Input/Output Control System (IOCS) for Model 20 card systems (IBM System/360 Model 20 Card Programming Support, Input/Output Control System, Order No. GC26-3603). IOCS provides tested input/output routines that programmers can use by means of macro-instructions, to control the input and output of data by programs written in the Basic Assembler Language.

   The vast majority of Model 20 users will never have to concern themselves with B.A.L. or IOCS, because the flexibility of the RPG and PCU (see below) will allow them to accomplish their tasks with these convenient and easy-to-learn languages. In a few installations, there may be occasional unusual requirements which cannot be directly satisfied by RPG or PCU. Frequently, a minor modification of the procedure will then permit RPG to handle the job; but there may be a few problems that are best solved by B.A.L., with or without IOCS. Even then, it will often be practical to write most of the program in RPG, merely inserting a brief B.A.L. routine to overcome the particular limitation. This approach is briefly covered in this manual.

   B.A.L. can, if efficiently applied, sometimes reduce the amount of core storage required for a program and may, on occasion, improve throughput. However, the much greater effort called for to program in B.A.L., and to debug the program, is usually out of proportion to the minor benefits derived.

2. Punched-Card Utility Programs (PCU) (Refer to SRL publication IBM System/360 Model 20, Card Programming Support, Punched-Card Utility Programs, Order No. GC26-3601.)

   The PCU performs on Model 20 the equivalent of IBM unit record machine functions. No knowledge whatsoever of programming is required. The user designates his job requirements by simple entries in pre-printed boxes--many of them multiple-choice pre-coded--on self-explanatory specification sheets

---

• Note: Sections delineated by upper- and lower-left right-angle brackets contain supplementary details--often of a technical nature.

from which matching specification cards are punched. For example, only a single specification card (in conjunction with the IBM-supplied program deck) is needed to perform a collating operation. The specification sheets correspond, in effect, to the control panel of a unit record machine, but are simpler and quicker to complete than plugboard wiring.

The PCU programs provide most of the functions of IBM collators, reproducers, gangpunches, summary punches, accounting machines, interpreters, and sorters (the latter practical with PCU only for large sort fields).

The PCUs are best used

- For jobs that correspond to unit record functions; i.e., where little is to be gained from the "systems" approach of processing an integrated series of jobs. For instance:

  To list and balance a keypunched deck of cards;
  To cross-foot fields in the same card;
  To sequence-check a master file;
  To interpret a keypunched deck;
  To reproduce a file of cards.

  A few of the applications that are easy to perform with PCU, are difficult or impossible with RPG. For example:

  Selection of the last card of each control group;
  Selection of single-card groups;
  Sorting.

- For one-time jobs, where it may not be worthwhile to design an integrated systems procedure; i.e., the "quick and dirty" solution.

- To continue getting the work out, during switch-over from unit record equipment to Model 20, for those jobs which the user has not yet had time to redesign and program to take full advantage of his Model 20.

RPG FUNCTIONS AND CHARACTERISTICS

PURPOSE OF RPG

RPG provides a quick and easy method for writing programs to accomplish most commercial data processing jobs with the IBM System/360 Model 20, taking full advantage of the Model 20 system's potential. It combines the attributes of flexibility, capability, and efficiency, with simplicity and absence of any requirement for prior programming or data processing experience.

Among the full spectrum of Model 20 data processing that can be programmed with RPG are the following common functions that can be performed individually or in any combination:

- Report Writing
  Listings and group-printed reports containing up to nine control and total levels, plus a final total.

- Summary Punching
  Up to nine levels of control, and final total level.

- File Matching and/or Merging
  With or without selection of cards.

- Card selection
  Based on card type and/or results of calculations.

- Gangpunching
  Direct, offset, interspersed, major-minor.

- Reproducing

- Card Document Printing (Interpreting)
  Feature available only for the 2560 MFCM, Model A1.

- Calculating
  Add, subtract, multiply, divide, cross-foot, compare.

- Table Look-up

STEPS IN UTILIZING RPG

1. Problem definition

   The nature of the source data, the processing to be performed upon it, and the type and format of the resulting output data must be determined. This encompasses such details as card-type identification codes, source and output card fields, calculations to be performed on the data, types of report totals desired, and arrangement of the data on a printed report. Printer Spacing Charts, IBM Form X24-3115, can facilitate the report layout (see Figure 1).

2. Programming

   The programmer writes RPG specifications on preprinted IBM forms. These forms guide the entries into the appropriate relative positions. The entries define his input and output data, the operations to be performed on the data,

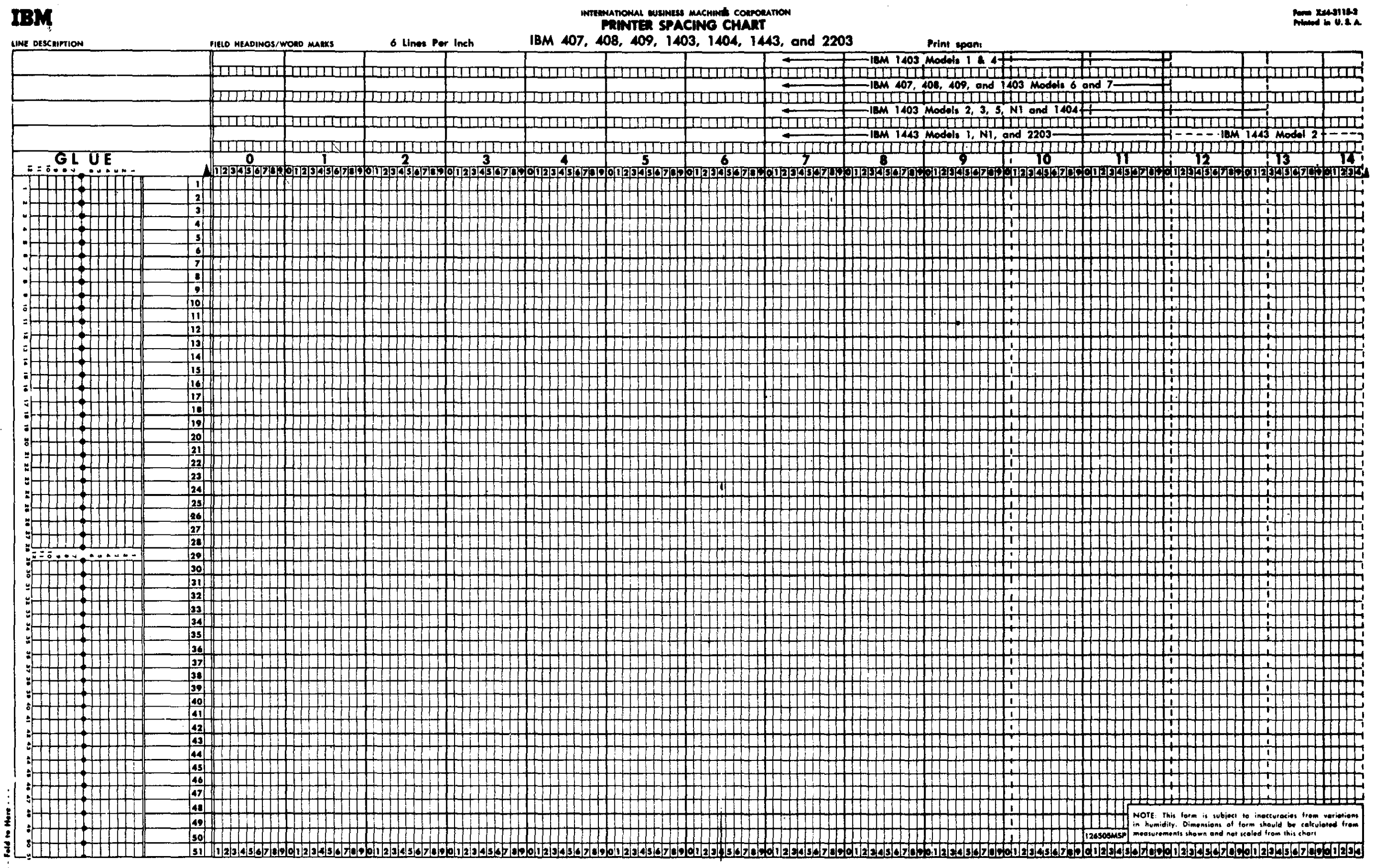


Figure 1. Printer Spacing Chart

and the input and output devices to be utilized.

The programmer is given wide latitude in the assignment of symbolic names to data files and fields, and most of the RPG-language operation codes are mnemonic. Much of the coding, therefore, approaches the use of meaningful English, combined with accustomed use of card-column numbers and print positions.

3. Punching Specification Cards

The program codes previously recorded on the specification sheets are keypunched, one card per specification line. The positions on each line of a specification sheet correspond to the appropriate columns in the specification cards.

4. Generating the Program

The specification cards now become the program "source deck". The source deck and the IBM-supplied RPG Generator deck are then read into the System/360 Model 20. Based on the program contained in the Generator deck, the central processing unit (CPU) of the Model 20 acts upon the specifications in the source deck to generate a machine language "object program." The object program contains all the necessary instructions to perform the job as designated by the RPG programmer on the specifications sheets. At the conclusion of the generation run, the object program is in core storage, ready for execution. The user has the option of also having the object program punched into cards so that, the next time the same job is to be run, the object program is ready to be loaded without the need to generate it again with the RPG.

5. Data card files are placed in appropriate card feeds, forms and carriage control tape are inserted in the printer, and the job is ready to run.

Figure 2 is a graphic representation of these steps.

MACHINE REQUIREMENTS

Input and Output Files (See also "File," under Definition of Terms, below.)

Input Files

The Model 20 card RPG can handle a maximum of three input files--one per card reading device attached to the system. The possible input devices are:

IBM 2560 MFCM Hopper 1, IBM 2560 MFCM Hopper 2, IBM 2501 Card Reader — or — IBM 2520 Card Read-Punch

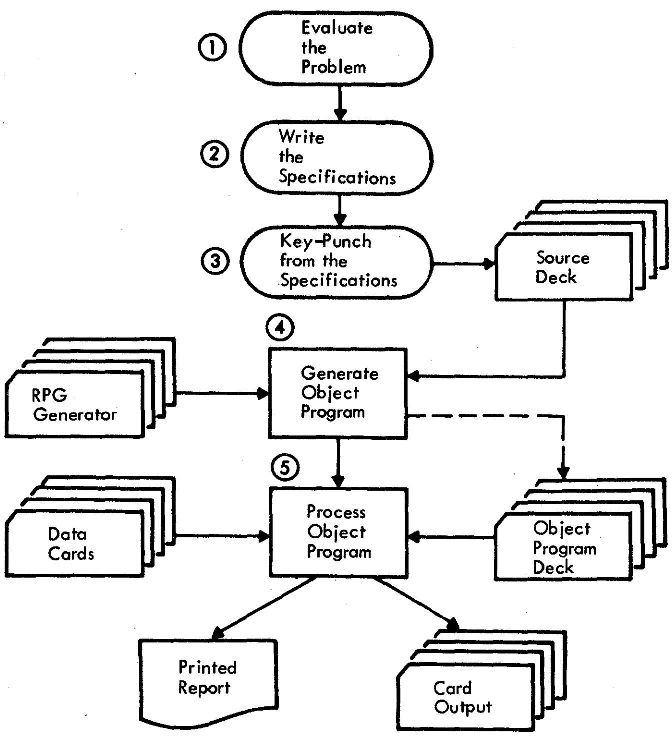


Figure 2. RPG Operations

Output Files

Up to five output files can be used--one per card punch device attached to the system, and one or two for the printer. The possible output devices are:

IBM 2560 MFCM Hopper 1, IBM 2560 MFCM Hopper 2, IBM 1442 Card Punch — or — IBM 2520 Card Punch or Read-Punch

IBM 2203 Printer, Lower Feed, IBM 2203 Printer, Upper Feed — or IBM 2203 Printer (standard carriage) or IBM 1403 Printer

Note: Each device listed above for both input and output files may serve to treat a single file as both input and output. That file is then designated a "combined" file. The MFCM permits cards from either or both hoppers to be read and/or punched and/or card-printed (interpreted). (The card document-printing special feature is available only for the 2560 MFCM, Model A1.)

Figure 3 is a schematic presentation of possible system configurations.

Note: With the IBM System/360 Model 20, Submodel 3 or 4, the 2560 MFCM Model A2 and the 2203 Printer Model A2 are the only I/O devices permitted. If an IBM System/360 Model 20, Submodel 5 is used, the following I/O devices may be attached: the 2560 MFCM, Model A1, the 1403 Printer, Model 2, 7, or N1, or the 2203 Printer, Model A1. For details regarding machine units and features required and supported, see Appendix B.

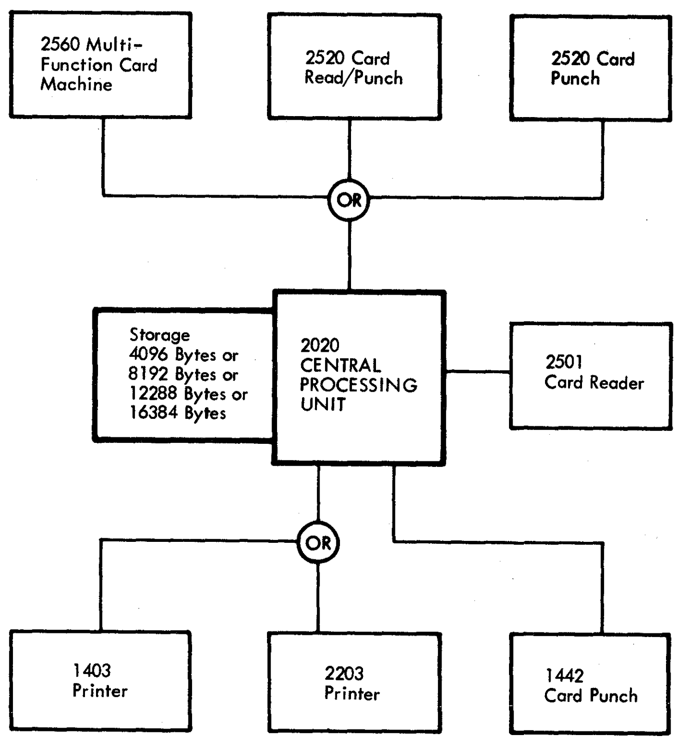


Figure 3. System/360 Model 20: Card RPG System Configurations

PERFORMANCE CHARACTERISTICS

The Model 20 can perform input, output, and internal processing operations concurrently. The RPG makes optimum use of this capability. Figure 4 shows which Model 20 operations can be carried out concurrently. In the case of concurrent card-punching and printing on the MFCM Model A1, this refers to the printing on one card while the next card is being punched.

Details for estimating core storage requirements and timing are given in Appendix A.

DEVICE	OPERATION	POSSIBLE COMBINATIONS													
2560 MFCM	Read	+	+												
	Punch			+		+				+					
	Card Print				+	+									+
2520 Card Read-Punch	Read						+	+	+						
	Punch								+					+	
2520 Card Punch	Punch											+	+		
1442 Card Punch	Punch	+						+	+		+		+		+
2501 Card Reader	Read					+				+	+	+	+	+	+
2203 or 1403 Printer	Print	+	+	+	+	+	+	+	+	+	+	+	+	+	+
2020 CPU	Processing	+	+	+	+	+	+	+	+	+	+	+	+	+	+

<u>Note</u>: Each vertical column shows a set of operations that may be carried out concurrently.

In the case of concurrent card-punching and printing on the MFCM Model A1, this refers to the printing of one card while the next card is being punched.

Figure 4. System/360 Model 20 Concurrent Processing Operations

ORGANIZATION OF THIS PUBLICATION

A summary of the functions of each of the five types of RPG specifications introduces the main portion of the manual. This abbreviated summary appears here only to facilitate relating subsequent sections to the specifications forms.

The specifications-types summary is followed by an example of specifications written for an RPG program, annotated with broadly-generalized explanations of the entries. The purpose of the section is merely to offer the novice an illustration of what a program written for RPG looks like. Full details are given in subsequent sections, which also incorporate any explanations given in the introductory example. This initial example does not fully cover the significance of, or limitations on, each entry. It should not be used as a reference for precise knowledge. Readers familiar with the concepts of RPG can bypass it.

The main portion of the manual is devoted to the detailed information needed by the user to write programs for his jobs that can then be converted to machine language by the Report Program Generator.

The information is presented in the following sequence:

1. Definition of recurring terminology

2. Graphic presentation and discussion of program logic flow

3. Indicators

4. Control fields

5. Matching of files

6. Sequence checking

7. Possible entries for every field of each specifications sheet (or card), including normal and unusual functions of each entry; warnings, where appropriate, about improper coding; interspersed illustrations for clarification of possibly abstruse points.

Limitations of the RPG Program are explicitly stated, where appropriate and not obvious.

Lengthy descriptions of rare, yet valid, uses of a code, or a specifications field, are marked off by corner brackets, so as not to detract from emphasis on the principal topic. It is suggested that the reader unfamiliar with RPG bypass these passages until he has a clear understanding of the basics. Where extensive or technical supplementary explanations are deemed of value only in exceptional situations and to a small segment of users, they have been relegated to an appendix when this was practical.

Three complete and realistic application examples are included, in addition to the Introductory Programming Example, to illustrate a large proportion of the program functions and codes. Each specification is explained.

The examples are:

1. Sales commission calculation and report.

2. An order-entry pre-billing application, with updating of the inventory file prior to invoicing.

3. The subsequent invoicing operation, with creation of accounts receivable invoice summary cards. Three lines of customer name and address printed from a single card, with ship-to name and address printed parallel from another card. A simple table look-up operation is included.

A number of technical appendixes follow (see Contents). Included is an appendix containing programming tips, and a summary of RPG specifications sheet entries laid out for convenient use if removed from the manual. In the appendixes, separate series of figure numbers have been assigned. Each figure number is preceded by the letter of the relevant appendix. This has been done to simplify subsequent additions and deletions without the necessity of making changes throughout the rest of the manual. The index, which concludes the manual, attempts to reference every informative mention of a relevant subject.

FUNCTION OF RPG SPECIFICATIONS SHEETS AND CARDS--SUMMARY

The RPG specifications sheets supplied by IBM (in pads) represent a convenient means for the programmer to record the information (instructions) to be keypunched as input to the RPG program, so that it will generate the appropriate machine-language program to perform the desired job.

The format and column headings of these sheets assist in guiding the programmer's entries. The forms are so designed that one specification card is to be punched per line, with each column on the sheet corresponding to a card column, in the same order. Card supplies with the appropriate RPG specification fields delineated can be purchased from IBM.

The RPG specifications sheets can also serve as documentation of the source program.

There are five types of specifications sheets and cards, each serving a different purpose, as outlined below. The forms are presented in the order in which they are most likely to be used by the programmer --not the order in which the different types of specification cards are entered for program generation. The details concerning the entries for the specifications sheets are covered in subsequent major sections of the manual, where pictures of each type of sheet are also reproduced.

In addition to the punched specification cards, the user must supply an RPG Control Card (Card H). This control card must be the first card of the source program. The format of the RPG control card is described in Appendix I. The control card specifies:

1. Core storage capacities of the systems used to generate and to execute the object program

2. Whether, and on which machine type, the object program is to be punched

3. Whether a generation listing is to be printed, and whether minor--as well as major--source deck errors are to cause a halt during generation of the object program

4. A typical MFCM input and output card stacking sequences

5. Additional IBM 2501 input core buffer storage, if desired

6. The number of print positions utilized by the object program

7. The format of any Sterling-currency fields (British monetary system)

8. Substitution of decimal comma for decimal point in numeric literals (i.e., European notation).

Types of Specifications Sheets and Cards

File Description Specifications (Required) (Sheets: Form X24-3347. Card electroplate: Form 3347)

Used to assign a symbolic name and, when appropriate, card sequence (ascending or descending) to each file; to associate each file name with a specific input and/or output device; and to define whether the file is to serve as input, as output, or both. For multiple input files, entries on this form also establish which file or files control end-of-job routines.

Input specifications (Required) (Sheets: Form X24-3350. Card electroplate: Form 3350).

Used to describe the input files: identification of card types within each file; stacker selection of cards, based on card type; specification of card-type sequence within each group of a file; assignment of symbolic names and decimal positions to input card fields; "tagging" of (i.e., setting indicators for) card fields with positive, negative, or zero/blank contents; assignment of control fields, and of fieldsto be matched between cards in different input files; file sequence-check instructions. For multiple input files, the order of precedence of the files is also established by the sequence in which the files are entered on this form.

Calculation Specifications (Optional) (Sheets: Form X24-3351. Card electroplate: Form 3351)

Used to describe the processing (calculating, comparing, etc.) to be performed on the data.

File Extension Specifications (Optional) (Sheets: Form X24-3348. Card electroplate: Form 3348)

Needed to describe the tables to be used with the Table-Lookup feature. Unless the Table-Lookup (LOKUP) instruction is used in the program, the File Extension form is not used.

Output-Format Specifications (Optional) (Sheets: Form X24-3352. Card electroplate: Form 3352)

Used to specify the arrangement of the data on printed reports and/or in output cards. Also includes such functions as editing, stacker selection of output- or combined-file cards, and forms-carriage spacing and skipping.

Note: A limited number of applications can be performed with only File Description and Input Specifications. For example: sequence checks, and/or stacker selection based on card type.

PROGRAM COMPATIBILITY

All functions that can be specified in the Model 20 card RPG can also be specified in other IBM System/360 Report Program Generators provided that an adequate I/O configuration is available.

Specifications which are presently unique to the Model 20 RPG are those supporting the IBM 2560 Multi-Function Card Machine (card printing, on the MFCM Model A1, and collator-type operations) and dual-feed carriage feature.

For further details, refer to the relevant SRL publication for other versions of IBM System/360.

MACHINE UNITS AND FEATURES REQUIRED AND SUPPORTED

Appendix B lists the machine units and features required and supported for the Model 20 card RPG.

INTRODUCTORY PROGRAM EXAMPLE

This chapter can be bypassed by users familiar with the concept of RPG. Its sole purpose is to give the novice a general insight into the approach to solving a simplified problem with RPG specifications. The explanations given are in broad terms only and are repeated in greater depth in subsequent sections. The example is not suitable as a reference for a full understanding of the specifications employed--while all specifications entries made here are valid, greater detail is necessary before the codes can be applied in all other circumstances.

THE JOB REQUIREMENTS

Given

1. Customer Name cards--one per customer.

   Name, in cols. 1-20; address in cols. 21-40 and 41-60
   Salesman No., in cols. 73-74
   Account No., in cols. 75-79
   Card identification (3-8-9), in col. 80

2. Daily Sales Summary cards--at least one per customer

   Account No., in cols. 1-5
   Amount, in cols. 7-13. (Cols. 12-13 are decimal positions.)
   X-punch (11-punch) over col. 13 for credit (returns)
   Gross profit percent for product group, in cols. 16-17
   Date, in cols. 75-79 (day, month, last digit of year)
   Card identification (1), in col. 80 --may have 11- or 12-overpunch.

Results Desired

1. Punch Monthly Summary cards--one per account

   Account No., in cols. 1-5
   Total amount, in cols. 6-13
   X-punch (11-punch) over col. 13 if negative
   Total gross profit, in cols. 14-21
   Salesman No., in cols. 73-74
   Date (month and year only), in cols. 77-79
   Card identification (9), in col. 80.

2. Printed Report

   Month and year only (slash between) --print on first detail line of each account. Eliminate leading zero in month only.

   Account No.--print only from first card for each account, and on forms overflow. Do not eliminate zeros.

   Customer Name--print from first card of each account, on same line as first detail card.

   Amount--list, but positive and negative amounts in separate columns. Eliminate leading zeros to decimal. Edit with comma and decimal point. Do not print sign for negative amounts.

   Amount--Net total by account, and grand total at end of report, with CR if negative. Eliminate leading zeros to decimal point.

   Gross profit--Total by account and grand total at end of report, with minus sign if negative. Eliminate leading zeros to decimal point.

   Amount of returns as percent of sales amount, for final total only. Eliminate first two leading zeros. Suppress line if positive sales are zero.

   Print suitable headings over columns on first page.

3. a. Select negative-amount summary cards to a different stacker.
   b. Separate Customer Name cards from Daily Sales Summary cards.

4. Stop program if first card of control group is not Customer Name card.

THE RPG SPECIFICATIONS

Figures 5A-5F show the printer layout and RPG specifications needed to produce the printed report shown in Fig. 5G. Explanations of the entries follow. Of necessity --since this example was deliberately inserted ahead of treatment of specifications entries--the discussion deals with items not yet covered, but will serve to illustrate the general approach. Obviously, with a language as flexible as RPG, the same results could be achieved by several alternate methods.

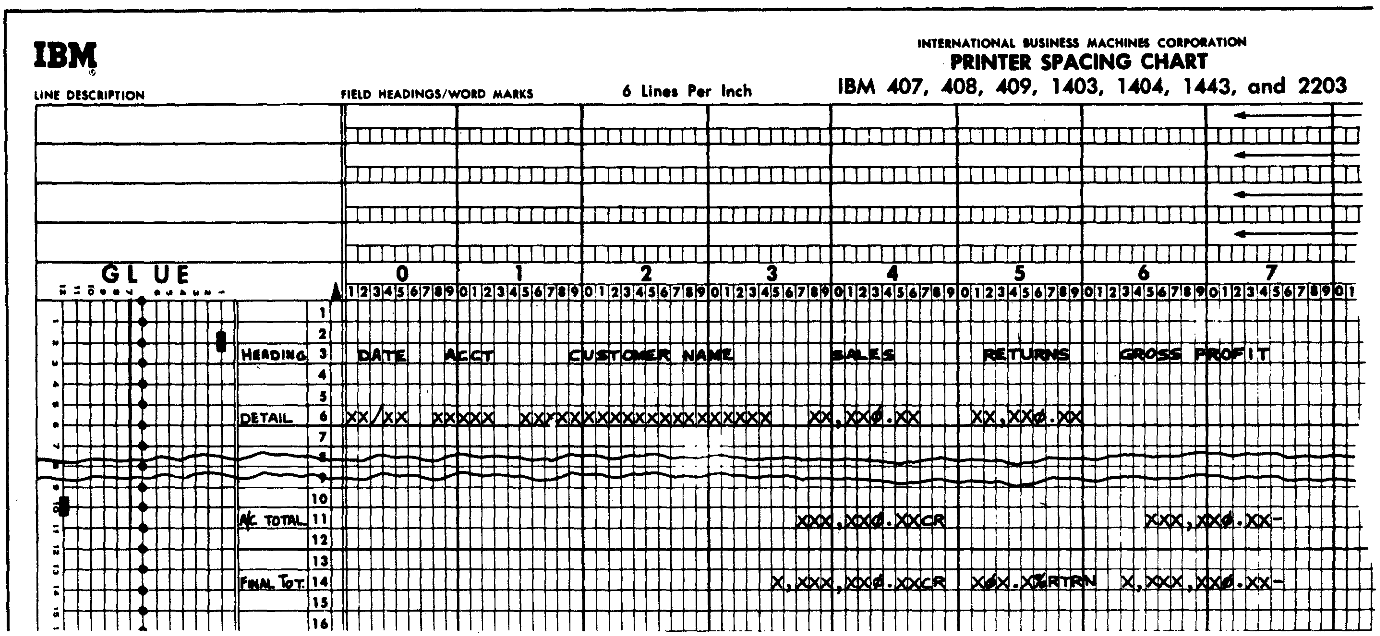


Figure 5A. Introductory Program Example, Printer Spacing Chart

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS

IBM System/360

Form X24-3347-3
Printed in U. S. A.

Date 7-15-66

Program SALES SUMMARY

Programmer K.B.

Punching Instruction: Graphic / Punch

Page 01 — Program Identification SLSSUM

Line	Form Type	Filename	File Type / File Designation / End of File / Sequence / File Format / Block Length / Record Length	Device	Symbolic Device
01	F	SLSDETL	INPUT	MFCM1	HOPPER
02	F	SLSSUMRY	OUTPUT	MFCM2	
03	F	REPORT	O	PRINTER	
04	F		COMBINE		
05	F				
06	F				
07	F				
08	F				
09	F				

Figure 5B. Introductory Program Example, File Description Specifications

IBM

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

Form X24-3350-1
Printed in U.S.A.

Date 7-15 66
Program SALES SUMMARY
Programmer K. B.

Punching Instruction: Graphic / Punch

Page Ø2 Program Identification SLSSUM

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Resulting Indicator	Rec. Id. 1 Position	Not (N)	C/Z/D	Character	Rec. Id. 2 Position	Not (N)	C/Z/D	Character	Rec. Id. 3 Position	Not (N)	C/Z/D	Character	Stacker Select	Packed (P)	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field-Record Relation	Field Indicators Plus	Field Indicators Minus	Field Indicators Zero or Blank	Sterling Sign Position
01	I	SLSDETL	Ø1	1		Ø1	8Ø		C	%									2												
02	I																				1	2Ø		NAME						1Ø	
03	I																				73	74		SLSMAN							
04	I																				75	79		ACCTNO	L1						
05	I		Ø2	N		Ø6	8Ø		D	1																					
06	I																				1	5		ACCTNO	L1						
07	I																				7	13	2	AMOUNT					Ø7		
08	I																				16	17	2	PCTPRF							
09	I																				77	79	Ø	MOYR							
10	I																														

Figure 5C. Introductory Program Example, Input Specifications

IBM

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

Form X24-3351-1
Printed in U.S.A.

Date 7-15-66
Program SALES SUMMARY
Programmer K.B.

Punching Instruction: Graphic / Punch

Page Ø3 Program Identification SLSSUM

Line	Form Type	Control Level (L0-L9, LR)	Indicators	And Indicators	And Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Plus (High 1 > 2)	Minus (Low 1 < 2)	Zero or Blank (Equal 1 = 2)	Comments
01	C		Ø6			AMOUNT	ADD	TOTAMT	TOTAMT	8	2			Ø8		
02	C		Ø6			AMOUNT	MULT	PCTPRF	GRSPRF	7	2	H				
03	C		Ø6			GRSPRF	ADD	TOTPRF	TOTPRF	8	2					
04	C		Ø6	Ø7		FTOTRT	SUB	AMOUNT	FTOTRT	8	2					
05	C		Ø6	NØ7		AMOUNT	ADD	FTOTSL	FTOTSL	9	2				Ø9	
06	C	L1	NØ1				SETON						H1			
07	C	L1				TOTAMT	ADD	FTOTAM	FTOTAM	9	2					
08	C	L1				TOTPRF	ADD	FTOTPR	FTOTPR	9	2					
09	C	LR					Z-ADD	FTOTRT	RTRDVD	13	5					
10	C	LR	NØ9			RTRDVD	DIV	FTOTSL	PCTRTR	4	3	H				
11	C															

Figure 5D. Introductory Program Example, Calculation Specifications

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System/360

Form X24-3352-1
Printed in U.S.A.

Date T-15-66

Program SALES SUMMARY

Programmer K. B.

Punching Instruction — Graphic: ' — Punch: 5-8

Page Ø4 — Program Identification SLSSUM

Line	Form Type	Filename	Type (H/D/T)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators	And	And	Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
01	O	REPORT	H			3			1P									
02	O								FIRST PASS						5		'DATE'	
03	O														12		'ACCT'	
04	O														31		'CUSTOMER NAME'	
05	O														44		'SALES'	
06	O														58		'RETURNS'	
07	O														74		'GROSS PROFIT'	
08	O																	
09	O		D			Ø	Ø1		OF	NL1								
10	O											ACCTNO			12			
11	O																	
12	O		D			1			Ø6									
13	O								N1Ø			MOYR			5		' / '	
14	O								N1Ø			ACCTNO			12			
15	O								N1Ø			NAME		B	34			
	O																	

Figure 5E. Introductory Program Example, Output-Format Specifications (Part I of II)

Page Ø5 — Program Identification SLSSUM

Line	Form Type	Filename	Type (H/D/T)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators	And	And	Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
01	O								NØ7			AMOUNT			46		' , Ø. '	
02	O								Ø7			AMOUNT			59		' , Ø. '	
03	O		T		1	3			L1									
04	O											TOTAMT			48		' , Ø. CR'	
05	O											TOTPRF			75		' , Ø. -'	
06	O																	
07	O		T					Ø1	LR									
08	O											FTOTAM			48		' , , Ø. CR'	
09	O								NØ9			PCTRTR			6Ø		' Ø . %RTRN'	
10	O											FTOTPR			75		' , , Ø. -'	
11	O																	
12	O	SLSSUMRY	T						L1	NØ8								
13	O	OR				3			L1	Ø8								
14	O											ACCTNO			5			
15	O											TOTAMT		B	13			
16	O											TOTPRF		B	21			
17	O											SLSMAN			74			
18	O											MOYR			79			
19	O														8Ø		'9'	
	O																	

Figure 5F. Introductory Program Example, Output-Format Specifications (Part II of II)

DATE	ACCT	CUSTOMER NAME		SALES	RETURNS	GROSS PROFIT
9/7	05944	ARTSON	H V		125.40	
				943.75		
				378.15		
				1,196.50		418.77
9/7	09772	BANKS	V H	649.50		
				649.50		227.32
9/7	02916	GILES	R D	278.00		
				149.50		
				427.50		149.62
9/7	07431	HARDING	C M	585.40		
					244.00	
				341.40		119.49
9/7	03349	KING	H R	278.95		
					125.00	
				400.00		
				450.60		
				1,004.55		351.59
9/7	01147	PAXTON	J M		350.50	
				50.50		
				155.00		
				145.00CR		50.75-
9/7	07728	STAUBER	A D	75.00		
				75.00		26.25
				3,549.45	19.2%RTRN	1,242.29

Figure 5G. Introductory Program Example, Printed Report

EXPLANATION OF SPECIFICATIONS. (FIGURE 5)

File Description Specifications--Figure 5B

Line 01 arbitrarily assigns the name SLSDETL to the input (I) data file consisting of the Customer Name cards and Daily Sales Summary cards. The DEVICE entry specifies that this file will be placed in hopper 1 of the IBM 2560 MFCM.

Line 02 assigns the file name SLSSUMRY to the deck of blank cards, to be placed in hopper 2 of the MFCM, which will become the Monthly Summary output (O) cards.

Line 03 specifies that printer output will be referred to by the file name REPORT.

These entries serve two basic purposes:

1. To associate a specific input and/or output unit with a file name that will subsequently be referenced in the program; and

2. To specify whether a given file is to serve as input for data, output, or both.

Input Specifications--Figure 5C

The input file--labelled SLSDETL in the File Description Specifications--consists of two types of cards.

Line 01 of the Input Specifications arbitrarily assigns "Indicator 01" (cols. 19-20 in the specifications) to the Customer Name card. The Customer Name card is identified by the punches 3-8-9 (a common unit record MLP or MLR code) in col. 80 (see specifications entries in cols. 23-24, 26, 27).

Line 05 assigns indicator 06 to the Daily Sales Summary card, identified by digit 1 in col. 80. D (digit), rather than C (character), was entered in col. 26, to eliminate a possible 11- or 12-overpunch in col. 80 from affecting the comparison with digit 1.

"Indicators" are discussed in detail in the next chapter. Briefly: the RPG Program provides for a large number of indicators which are either set by the RPG Program itself, or may be set by the programmer, to identify a condition. They may then be specified elsewhere in the program to condition the execution of a specification on the setting (ON or NOT ON) of the indicator.

Indicator 01, in this example, will be on when a card with 3-8-9 in col. 80 (Customer Name card) is being processed. Execution of certain instructions can then conveniently be associated with "Customer Name card", or "not Customer Name card", as desired.

When stacker selection is not specified, cards enter the normal stacker for the particular hopper of the I/O unit used. For hopper 1 of the MFCM, this is stacker 1. The card type identified by indicator 06 therefore enters stacker 1. The card type with indicator 01 (Customer Name card) is directed to stacker 2 by the entry in col. 42.

The entries in cols. 15-16 specify that the proper order of card types is Customer Name card (01 in cols. 15-16) followed by Daily Sales Summary card(s) (02 in cols. 15-16), which in turn are followed by the next Customer Name card. The 1 in col. 17 for the Customer Name card specifies that there must be exactly one such card before the Daily Sales Summary card. The N in col. 17 for the Daily Sales Summary card specifies that there must be at least one such card, but that any quantity of such cards greater than 0 is correct. If the card-type sequence does not conform to these specifications, an error stop occurs. Note, however, that the absence of a Customer Name card would not be detected--this would be treated as more than one Daily Sales Summary card. This contingency is guarded against by the specifications on line 06 of the Calculation Specifications.

Lines 02-04, and 06-09, contain the names the programmer has arbitrarily assigned to the fields he will subsequently utilize from the two input card types, respectively. They are preceded by their column numbers in the input cards. Col. 52 in the Input Specifications assigns the location of the decimal point of input fields, for automatic alignment in calculations. Use of a field in calculations or numeric comparison, or editing its output, requires a decimal specification even if no decimal point is relevant. This explains the 0 for MOYR. Note that field names start one line below identification of their record types.

The entry in cols. 59-60, next to ACCTNO, specifies that the Account No. field in both card types (note that it may be in different card columns in the two card types) is to be a control field, at the lowest level. (Nine control levels, L1-L9, are available.) Whenever there is a change in the contents of the Account No. field between successive cards, the L1 indicator turns on for one program cycle. The ON or NOT ON status of the L1 indicator can be used to control operations.

The entry in cols. 69-70 makes the status of indicator 10 dependent on the NAME field of each Customer Name card (Resulting

Indicator 01). Since that field is never blank in that card, indicator 10 will turn off each time a Customer Name card is processed. (It would be turned on by a blank NAME field.) In this program example, indicator 10 is turned on by another method, described later.

The requirements of the job call for printing the amount of returns (negative sales amounts) in a separate column. An indicator is needed to identify such cases. Indicator 07 (in cols. 67-68) will turn on when a Daily Sales Summary card with a negative amount is being processed, and will be off when the amount is positive or zero.

Calculation Specifications--Figure 5D

Calculations occur at detail time unless an L-indicator (control level) specification appears in cols. 7-8 (Control Level)--in which case that calculation takes place at total time. (Detail and Total times are discussed in the next chapter.) Thus, the calculations specified on lines 01-05 are executed at detail time; those on lines 06-10, at total time. All detail-time entries must precede all total-time entries. Within this grouping, calculations are performed in the order in which the specifications appear. A summary of the functions of the entries, by line, follows.

Line 01. Executed only when processing card type 06 (Daily Sales Summary card), because the indicator for that card type is designated as a condition.

The contents of the AMOUNT field are added (Operation code ADD) to the contents of TOTAMT field, and the result is stored as the new contents of TOTAMT field. TOTAMT field has not been previously specified; it is created by the entry in Result Field. (Field Length is specified as 8 digits, of which 2 are decimal positions-- the same number of decimals as in the source (AMOUNT) field. If the number of decimals specified here were to be different from those in the source field, alignment would be automatic.) This is the normal method for accumulating detail-card amounts for group totals. When object-program execution begins, the user may assume that the fields are all set to zero. Thereafter, each detail card amount is algebraically added to the previous total in the TOTAMT field, because TOTAMT is the addend (Factor 2) and the new result replaces the former TOTAMT contents.

Negative amounts (11-punch over low-order position) are automatically subtracted. An indicator (08) is specified for the identification of a negative amount in the TOTAMT field, so that summary cards (one punched at each control break) with a negative sales amount can be selected to a separate stacker. The status of indicator 08 can change after each algebraic addition. Its status is, however, only used in this example at the end of a control group, when it correctly reflects the sign of the total.

Line 02. The amount (with 2 decimal positions) in each detail card is algebraically multiplied by the gross profit percentage (2 decimal positions only, to transform percentage to ratio) for that product group. The resulting amount of profit (GRSPRF) contains four decimals, of which only two are desired. Specifying "2" automatically causes dropping of the two excess low-order positions. The "H" in col. 53 causes half-adjustment before the third decimal is dropped. The previous contents of the Result Field are replaced each time by the new result.

Line 03. The latest gross profit amount (GRSPRF) is algebraically added to the previous cumulation (which is zero if this is the first detail card) to provide a total for the control group.

Lines 04 and 05. These entries provide the final total of returns (negative sales) and of positive sales, so that the ratio of returns (FTOTRT) to positive sales (FTOTSL) may be calculated before the final total is printed.

Line 05 causes adding of the amount from each detail card (indicator 06) to FTOTSL--provided AMOUNT is positive (indicator 07 not on = N07 in cols. 12-14), as determined by indicator 07 in the Input Specifications. Indicator 09 is set on for zero results--see line 10 for its application.

Line 04 similarly provides for cumulating FTOTRT for negative amounts (indicator 07 on). Since a positive total is desired, and all amounts for this line--by definition--are negative, these negative amounts are subtracted from FTOTRT. (Subtracting a negative amount yields positive result.) This entry also illustrates absolute addition.

Line 06. Indicator H1 is set on--which will cause the system to stop after processing of the new card--if a control break (change in contents of ACCTNO field) occurs (L1 on) and the new card is not a Customer Name card (N01).

Line 07. When a control break has occurred (L1 on), the total amount (TOTAMT), accumulated above (line 01) algebraically for each control group, is added algebraically to FTOTAM (which is zero in the case of the first control group) to provide a final amount total at the end of the report. The total transfer must occur at this time, because TOTAMT must be reset to zero before the amount field from the first detail card of the new control group is added to it. TOTAMT then correctly reflects the total for each control group. The FTOTAM field has been specified as larger than TOTAMT, to accommodate the sum of several TOTAMT group totals.

Line 08. Similar to line 07, but cumulates final total of gross profit (FTOTPR), based on group total from line 03.

Line 09. This adjusts the size of, and number of decimal positions in, the final-total-returns (FTOTRT) field (from line 04), so that the size and decimal alignment are suitable for line 10. Operation code Z-ADD resets the Result Field to zero prior to adding in the data from the Factor-2 field. Since the operation is performed only once per job--after processing of the last data card (LR indicator = Last Record) --ADD could have been used equally well as the operation code.

Line 10. Before the final total is printed, the specification on this line causes the calculation of the ratio of total returns (RTRDVD, based on FTOTRT in line 04 and shifted left in line 09) to total positive sales.

The calculation is only performed after processing of the last data card (LR is then on), and provided there was a positive sales total (N09). Indicator 09 is set on in line 05 for a zero final total of positive sales. Conditioning the instruction on N09 is required because a divisor must not be zero.

A dividend (RTRDVD) with 5 decimal positions, and a divisor with 2, yield a quotient with 3 decimal positions (col. 52). The H in col. 53 causes half-adjustment of an extra decimal position (automatically provided for by the RPG Program) before it is dropped.

Output-Format Specifications--Figures 5E and 5F

Printed Report

The file name REPORT was designated an output file, and associated with the printer, in the File Description Specifications. Thus, its entry here thereby calls for printer output for all specifications below, until a different file name appears. H (heading) or D (detail) in column 15 specifies that the ensuing entries apply to detail- (rather than total-) time processing. (H and D may be used interchangeably.) T in col. 15 specifies total-time output.

Specification line 01 on page 04. Indicator 1P in cols. 24-25 determines that the output entries in lines 01-07 apply to the first page only. The 1P indicator is set on by the RPG Program itself at the beginning of program execution, and is turned off before the first card is processed. This output, therefore, occurs only once, before processing of the first card. It is used to print headings. After the heading line, the form advances 3 spaces (col. 18).

Specification lines 02-07 specify the heading data to be printed. The data within apostrophes is printed as shown (without the apostrophes, which merely identify the entries as constants). The numbers in cols. 41-43 designate the rightmost print positions for the respective constants to be printed.

Specification lines 09-10. The job requirements call for printing Account No. (ACCTNO) on the same line as the first detail card of a control group, and to repeat the Account No. as the only identification on overflow pages. The Account No. is to be printed with its rightmost position in print position 12 (see entry in cols. 41-43). Indicator OF in cols. 24-25 confines this output to overflow time.

Because ACCTNO is also printed on the first line of a new control group (see explanation for line 14)--and overflow time is separate from regular detail or total time (see next chapter)--the line must also be conditioned not to print if a control break has occurred (NL1 in cols. 26-28); otherwise, Account No. will print twice in that situation.

When any overflow indicator is used in the output specifications, forms-advance to channel 1, after a channel-12 punch has been sensed, is not automatic; therefore, Skip-Before to channel 1 (01 in cols. 19-20) is specified. No space (0) or skip is specified to follow the overflow indication, because the data from the next detail card is to be printed on the same line.

Specification line 12. Indicator 06 in cols. 24-25 conditions line 12 on page 04 through line 02 on page 05 to apply only to detail cards (Daily Sales Summary); i.e., all printing takes place when detail cards are being processed.

The job requirements stated that Account No. and Name are to be printed on the same line as the first detail-card data, although Name is available only from the Customer Name card. This can be accomplished in several ways. The method chosen here utilizes the fact that any field retains its data until read into again, or reset. Thus, the Customer Name is still in the NAME area in core while detail cards of the same group are being processed--until the next Customer Name card is processed, or the field is blanked by a program instruction. The name can, therefore, be printed at detail-card time. Col. 18 specifies single spacing after each detail line.

Line 13, page 04, through line 02, page 05. Throughout, the Field Name in cols. 32-37 specifies which input or calculation-Result Field is to be printed. The size of each of these fields was determined in the Input or Calculation Specifications. Cols. 41-43 specify the right-hand print position where the field, as edited and including edit-word constants, is to end. The printing of items on the print line may be further conditioned (besides the indicator-06 condition applicable to the entire print line), and the format may be edited (see below).

Specification lines 13-15 on page 04. Indicator 10 turns off whenever the NAME field is not blank--see Input Specifications, line 02. It is, therefore, off when a Customer Name card has been read. Thus, MOYR, ACCTNO, and NAME fields are printed with data from the first detail card (Daily Sales Summary), because the condition N10 (indicator 10 not on) in cols. 23-25 then still obtains. If indicator 10 is not turned on by a program specification, they will be printed on every detail line. By specifying B (Blank-After) in col. 39 next to NAME, the NAME field is blanked after it has been transferred to the output area. Indicator 10 then turns on. Therefore, the printing called for in specification lines 13-15 is not performed again after the first detail card, until a new Customer Name card has been read. (See also Program Logic Flow, Blank-After.)

Specification lines 01-02 on page 05. The printing specified in line 01 is performed when the detail-card amount is positive (N07), and that in line 02 when it is negative (indicator 07 on). (The setting of indicator 07 occurs in the Input Specifications.) Thus, positive amounts are listed to the left of negative amounts.

Specification line 03 on page 05. T in col. 15 designates execution at total time. The L1 indicator in cols. 24-25 conditions execution of the print line to occur on each Level-1 control break, i.e., a break on Account No. Col. 17 specifies a single space before this total line which, in conjunction with the single space after detail lines, leaves one blank line before the group totals. Three spaces after the group-total line (3 in col. 18) leave 2 blank lines before the next detail line.

Specification lines 04-05. Similar to previously explained field entries, but the fields are printed at Level-1 total time.

Specification line 07 on page 05. T in col. 15 designates execution at total time. The LR indicator in cols. 24-25 conditions execution of the print line further, to occur only after the last data card (Last Record) has been processed; i.e., for final totals. Cols. 21-22 contain the specification to skip the form to channel 1 after printing the final total.

Specification line 09. Besides being printed only at final total time, indicator 09 must be off (N09) to cause the PCTRTR (Percent Returns) field to print. The reason for this is that, if FTOTSL (Final Total Sales) was zero at the end of the report, PCTRTR was not calculated because a zero divisor would be meaningless (and division by zero is not allowed). (See also Calculation Specification lines 05 and 10 for establishment and application of indicator 09.)

Editing. When data appears between single quotes in cols. 45-70, on the same line as a Field Name, the entry modifies the format in which the data is printed. Only fields to which a decimal position (0-9) was assigned may be edited; that is, fields designated as purely numeric. Illustrations follow.

Specification line 13, page 04, cols. 46-50. Specifies that a slash is to appear between Month and Year digits. A zero in the first column of Month is automatically suppressed, because an edit word is used.

Line 14. All zeros in ACCTNO will be printed, because no editing or zero suppress is specified (it can only be specified for a field defined as numeric; i.e., a field for which Decimal Positions has an entry where the field is defined).

Lines 01-02, page 05. Leading zeros are eliminated, through the dollar position. Decimal point and two low-order positions are always printed in this example. Comma is printed between hundreds and thousands positions when there are significant digits to its left.

Lines 04, 05, 08, 10. Similar to editing of lines 01 and 02, but CR or minus symbol (respectively, as shown) is printed for negative amounts.

Line 09, page 05. Leading zeros are eliminated only in the tens and hundreds positions. The decimal point and a percent sign, followed by the letters RTRN, are always printed when the print line is printed. Zero percent is printed as 0.0% RTRN. Note that, by means of the decimal point in the edit word, the ratio with 3 decimals (Calculation Specification Line 10) is converted back to a percentage with 1 decimal position.

Punched Output

The file name SLSSUMRY was designated an output file, and associated with hopper 2 of the MFCM, in the File Description Specifications. Thus, its entry in the Output-Format Specifications calls for card output, the card source being MFCM hopper 2. The T in col. 15 specifies output at total (rather than detail) time. The entries (L1) in cols. 24-25 specify punching of the card at each control break of Level 1.

The OR in cols. 14-15 designates that the same punch data applies when the conditions for either specifications line 12 or 13 are met, subject to any further conditioning indicators. The difference in conditions is that line 12 applies if TOTAMT is positive, and line 13 if it is negative, at group control-break time. (See cols. 26-28 here, and line 01 in Calculation Specifications.) When positive, the card enters the normal stacker for hopper 2 of the MFCM; when negative, it is selected to stacker 3 (see entry in col. 16).

Lines 14-18 specify which fields--defined in Input or Calculation Specifications--are to be punched into the Monthly Summary cards, together with the low-order-position columns where the fields are to end in the output card.

Line 19 specifies that the constant 9 is to be punched in col. 80. (The absence of a Field Name designates cols. 45-70 as available for a constant rather than an edit word.)

The B in col. 39 (Blank-After) on lines 15 and 16 directs the program to reset the TOTAMT and TOTPRF fields to zero after they have been transferred to the output area, so that they are cleared to accumulate totals for the next control group.

This chapter deals with facts and functions that must be understood to derive the fullest benefits from RPG. In order to provide complete information on these subjects at one reference point, the chapter delves into considerable detail and, occasionally, complexities. This was considered preferable, from the user's viewpoint, to scattering related facets throughout the volume.

If the meaning or relevance of all statements made in this chapter is not apparent on a first reading, the user should not be concerned: they are illustrated as the manual proceeds. Both the ensuing itemized coverage of each specification field and the appended extensive applications examples clarify the contents of this chapter, and make frequent reference to them.

It is, therefore, suggested that the user read this chapter thoroughly once and, thereafter, expect to revert to appropriate portions of it repeatedly.

DEFINITION OF TERMS

Terminology that recurs throughout this publication is defined below, as it applies to Model 20 card RPG:

File

Note: A single card file can serve as both input and output. It is then termed a "combined file"--see definition for combined file, below.

Input File

One input file consists of all the cards that originate (i.e., enter the system) from one hopper of a card read or read-punch device, and fulfill all of the following conditions:

1. All the cards are to be read (i.e., punches in the card serve as input to the system). There must be an entry for them in the Input Specifications.
2. None of the cards are to be punched.
3. None of the cards are to be interpreted (card-printed).
4. None of the cards are to be stacker-selected on the basis of information not in the card itself; i.e., they can only be stacker-selected by designation on the Input Specifications sheet on the basis of card type.

In summary: There is no entry for an input file in the output specifications.

Output Card File

One card output file consists of all the cards that originate from one hopper of a card punch or read-punch device, and fulfill all of the following conditions:

1. All of the cards are to be punched and/or interpreted (card-printed) by entries on the Output-Format Specifications sheet.
2. None of the cards are to be read.

In summary: There is no entry for an output file in the input specifications. The cards in the file may be blank or prepunched, but they will not be read.

Combined File

One combined file consists of all the cards that originate from one hopper of a read-punch device to which the following condition applies:

All or some of the cards serve as input to the system and all or some of the cards--regardless of whether they are the same or different cards in the file --also serve as output; i.e., the file requires entries both on the Input and Output-Format Specifications sheets.

A combined file is a single file.

Output Printer File

All report (paper forms) printing performed by one program under control of a single forms carriage is designated as one output file.

For the IBM 1403 or standard 2203 Printer, this implies that all print lines are identified as belonging to a single file. If the Dual-Feed Carriage special feature is installed on the IBM 2203, and both carriages are to be used in one program , the lower and upper feeds are treated as two separate output files.

EBCDIC--EXTENDED BINARY-CODED-DECIMAL INTERCHANGE CODE

EBCDIC is the IBM System/360 machine code. It provides for 256 unique characters. For further details, see Appendix D, Figure D1, and the publication IBM System/360 Model 20, Functional Characteristics, Order No. GA26-5847.

Characters

Alphabetic Characters

The 26 letters of the English alphabet, plus these three characters:

Dollar Sign ($)	--card punch-combination 11-3-8
At-sign (@)	--card punch-combination 4-8
Pound or Number sign (#)	--card punch-combination 3-8

Numeric Characters

The digits 0 through 9.

Special Characters

The 217 EBCDIC characters not defined as alphabetic or numeric.

Alphameric Characters

Any of the 256 EBCDIC characters, including blank.

Fields

Alphameric Fields

All fields for which a Decimal Positions specification (0-9) has not been made in the appropriate column of any of the pertinent specification forms--regardless of whether the field contents are alphabetic, numeric, or alphameric. Zero and blank are distinguished.

Numeric Fields

All fields that have a Decimal Positions specification (0-9) in the appropriate column of any of the pertinent specification forms.

Numeric fields contain numeric characters, and possibly a 12-zone punch (+), or an 11-zone punch (-) sign over the rightmost position only. Blanks in digit positions of a numeric input field are converted to zeros. Zone punches in an input field, in other than the low-order position, are stripped.

Note: For other possible punches in numeric fields, see Packed and Decimal Positions, under Input Specifications.

Literals

A literal is the actual data to be operated upon, rather than a symbolic name representing the location of the data in core storage. The specifications sheet entry must be left-justified.

A literal is stored in storage only once, regardless of how often it is used--provided it is always the same size, and used in identical format (always alphameric; or always numeric, with decimal point position uniform; if numeric, always with the same sign designation or always without sign).

Alphameric Literals

Alphameric literals consist of any one or more of the 256 EBCDIC characters (see Appendix D, Figure D1, for the appropriate card punch-combinations). Initial and terminal apostrophe symbols (')--card punch-combinations 5-8--are required. They designate the literal as alphameric and define its extent.

If an apostrophe is required within the literal itself, it must be specified as two consecutive apostrophes (two card columns, each punched 5-8)--independently of the apostrophes needed to define and delimit the alphameric literal. Such an apostrophe within the literal consumes two of the number of positions allowed for the literal; but only one of the apostrophes is printed or punched (with any subsequent characters moved left one position) if the literal is to be used as output.

Numeric Literals

Numeric literals consist of any one or more of the digits zero through 9. One decimal point can precede or follow the literal, or can be contained within the literal; it effects automatic decimal alignment during calculations with the literal.
(If the literal does not include a decimal point, it is treated as an integer.) The literal can be preceded by a sign; if it is unsigned, it is treated as positive. Blanks are not allowed in numeric literals.

Numeric literals must not be enclosed in apostrophes.

Numeric literals can only be specified in the calculation specifications.

Note: If European notation is specified in the RPG control card (Card H), a decimal comma is allowed in numeric literals in place of a decimal point.

Control Fields

Fields that contain information to be compared from card to card for the purpose of detecting the end of a control group (a group of records having identical control fields). A control break is deemed to occur when information in a control field differs for two successively processed cards for which a control level is specified. When a control break occurs, the RPG program turns on the L-indicator (L1-L9) of the control level assigned to that control field, and all lower-level L-indicators.

Control Level

The significance level (L1-L9) assigned by the programmer to a control field.

SYMBOLS USED IN THIS PUBLICATION

Blank

For convenience, the symbol b is occasionally used to represent a blank column or the EBCDIC code for a blank.

Zero

Where confusion might otherwise result between the letter O and the numeral zero, the latter is either spelled out, or represented by 0.

Column

The word "column" is frequently abbreviated as "col.".

PROGRAM LOGIC FLOW

Each object program generated by RPG uses the same general logic, and for each card processed the program goes through the same general cycle of operations.

There are actually two different points (or times) in the RPG cycle where calculation and output operations may occur. These two major components are called detail time and total time. Detail time and total time may be split up into:

1. Detail (or heading) output time (1)
2. Detail calculation time (4)
3. Total output time (3)
4. Total calculation time (2)

Note: The numbers in parentheses refer to the numbers used in Figure 5H.

The RPG cycle shown in Figure 5H begins with the detail-output routine. This routine enables the program to write constant data (e.g. a heading line) at the top of the report.

Total-time calculations and output appear in the cycle before detail-time calculations and output. This time sequence is necessary for processing groups of records (e.g. for making group totals). Between READ and total time the program determines whether the record belongs to a new control group. If it does, a special indicator is set on and the final processing for the old record group is done during total time. At total time, the data from the last record of the old control group is still available. The contents of the record just read are not made available for processing until just before detail-time calculations. This new record thus begins being processed at detail time. It remains available during reading of the next record and until after the next total time.

Basically there is no distinction between the operations that can be performed at detail-time and at total-time; however, certain control information available (such as the status of indicators described below) differs, as well as the data available relating to the position of the card.

The END routines are performed after total time so that all required total processing can include the last record of the file.

Another component of the cycle is overflow-output time, with any overflow output designated T (for "total") preceding any designated D ("detail"). All overflow-time output operations are available after total-time output.

Figure 6 is a logic flow diagram of the RPG object program. Reference to points on the chart is made as the relevant subject matter is covered. Figure 6 is repeated as Figure G3, in appendix G, for convenient reference.

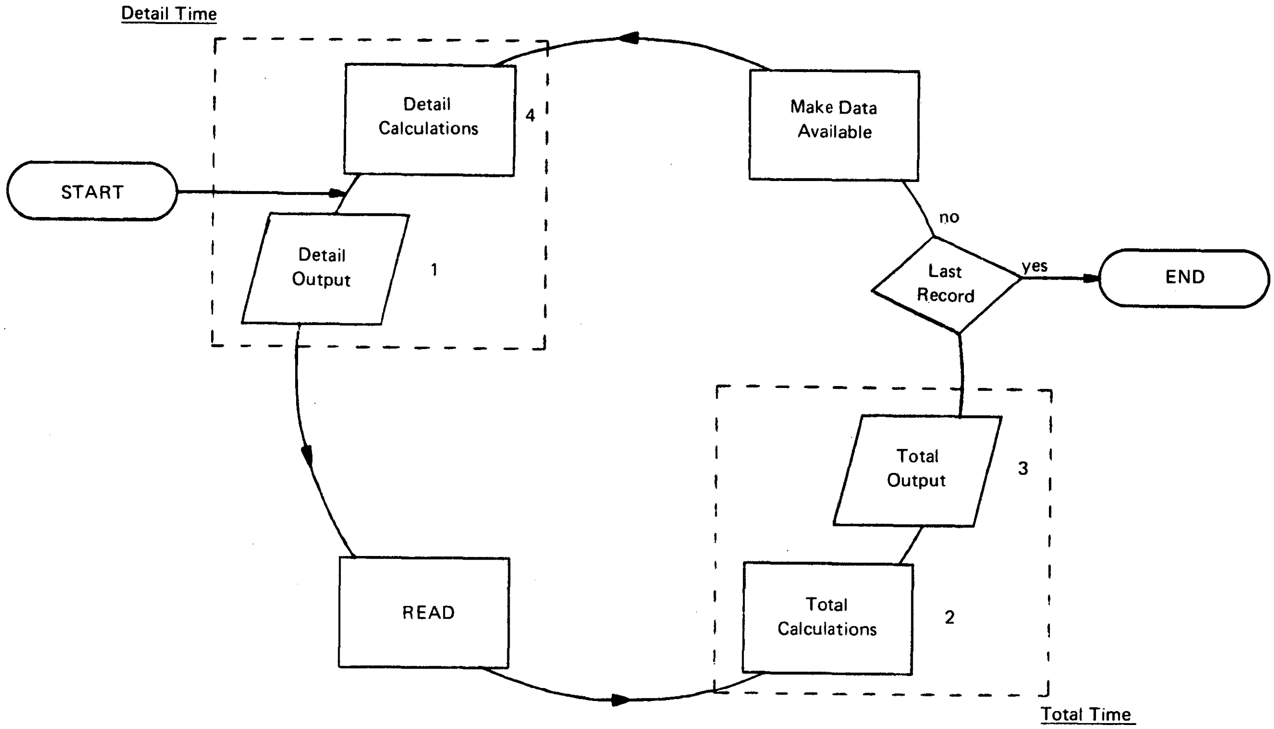


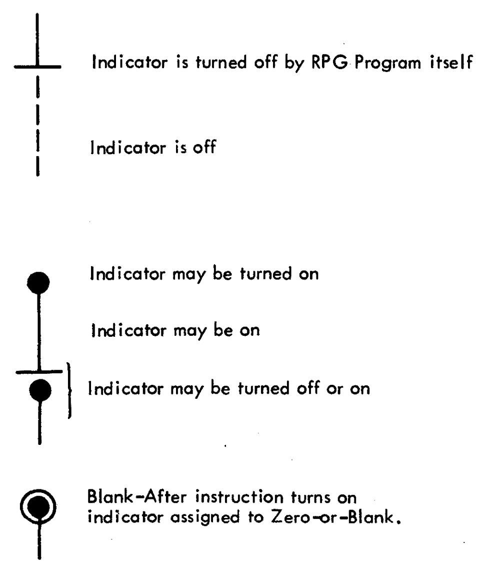
Figure 5H. RPG Object Program Cycle

Note: In referring to output operations other than total-time or overflow-time output, both detail (D) and heading (H) output are used. There is no distinction between D and H in the RPG program--the two codes are interchangeable, and are both available merely for the convenience of the user in identifying the purposes of different specification lines.

Indicator Settings

Vertical lines in the right margin of Figure 6 pertain to the possible setting or resetting of indicators at different points in the program cycle, when the indicators are used in a normal manner. (Greater detail is supplied in the next section, titled Indicators.)

The symbols shown in the indicator chart in Figure 6 have the following meanings:



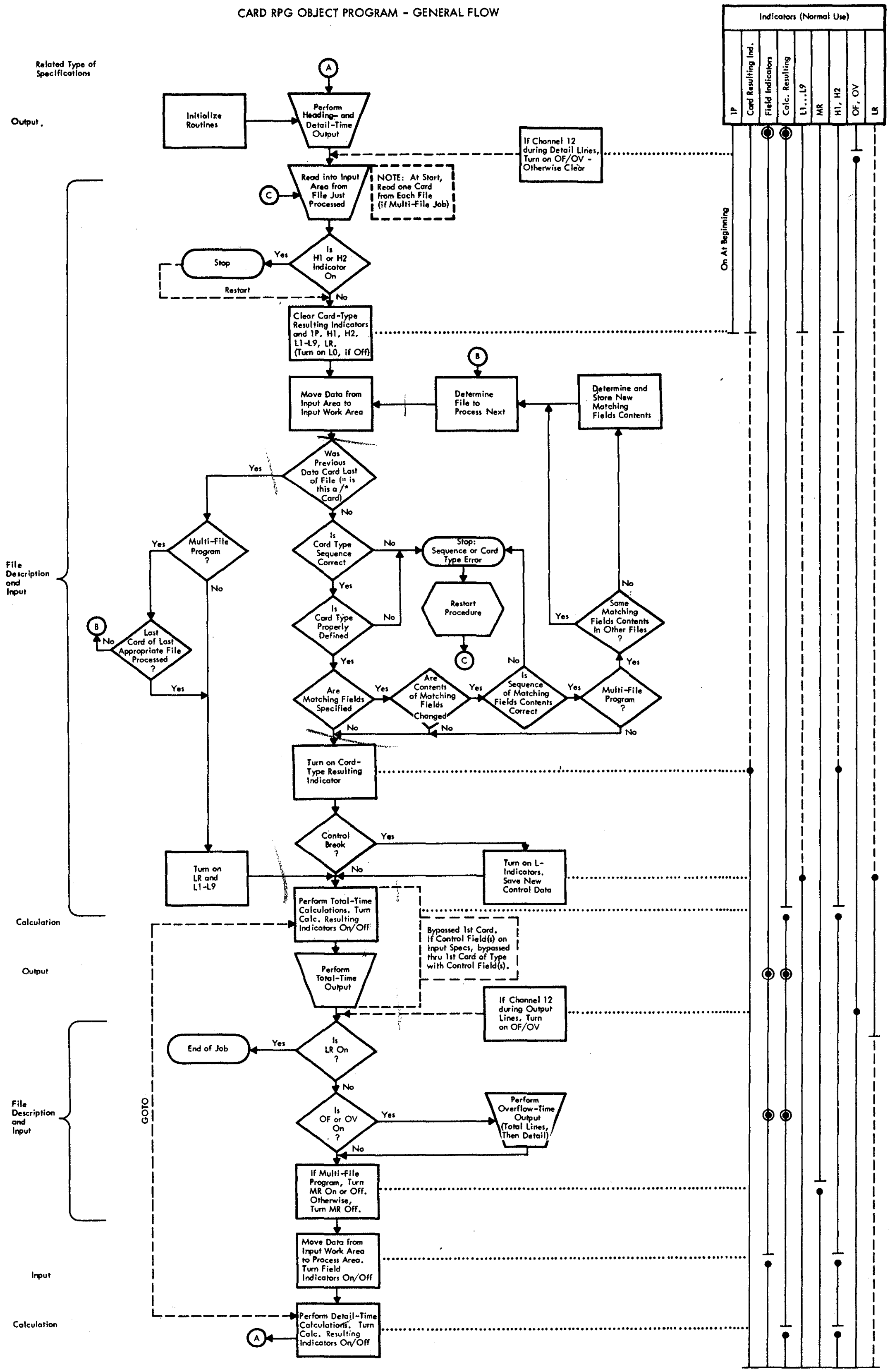


Figure 6. RPG Program Logic

Special Aspects of RPG Program Logic

Although the program logic chart is largely self-explanatory, and its entries are further explained in pertinent subsequent sections of the manual, a few points of overall significance to RPG programming are emphasized here:

1. Relationship of Total Time to Card Movement: Total-time calculations and total-time output occur after a new card has been read, and the previous card itself has been completely processed. Thus, any output to a card at total time is to the new card. However, the data available at total time is that from the previous card. (Figure 6 shows that the data from the new card is not transferred to the process area until just before detail time.)

   Because output is to the new card,
   a. a stacker-selection specification for total-time output causes selection of the new card, and

   b. card punching and document-printing at total time apply to the new card.

   Consequently, although it is known at total time whether the previous card was the last of a type or group--note in Figure 6 that L-Indicators and card-type Resulting Indicators for the new card have been set before total time--it is not possible

   a. to stacker-select the last card of a type or control group, with RPG; or

   b. to document-print the last card of a type or control group, with RPG; or

   c. to punch the last card of a type or control group, with any programming language

   on the basis of its being the last card of the type or control group.

   The PCU (Punched-Card Utility) program PLACE specification card provides a very simple method for selecting the last card of a control group. Alternatively, a Basic Assembler Language subroutine can be used with RPG to select the last card of a group (see Programming Tips, Appendix E).

   To punch a card only at the end of each group, that card must either be identified as a different card type (input specifications Resulting Indicator), or it must be in another file with its matching fields so coded that the card will advance at the appropriate point in a multiple-file operation. (See section on Matching of Files.)

2. Multiple-Time Output to Cards during One Program Cycle: Once an output operation (punching and/or card-printing and/or stacker selection) has taken place in a card, the card advances, and the next card assumes the equivalent position in relation to the punch or card-print station. Therefore, all output instructions for one card must be given under a single entry of File Name and Type (see Output-Format Specifications), for one point in the program cycle (total time or overflow time or detail time).

   One program cycle extends from entry point A at the top of the Program Logic Chart (Figure 6) through exit point A at the bottom; i.e., from detail-output time through total and overflow time and through detail-calculation time. It is permissible--if called for by an unusual job requirement--to give output instructions for more than one point in this cycle, and/or by separate groups of instructions for the same cycle-time segment, for the same card file. This is done by card-output entries for more than one output time (total-time output, overflow-time output, detail-time output), and/or by card output entries with repeated definitions of the same file for the same output time (and/or by branching (GOTO) from detail to total time). The user must then clearly understand the consequences:

   The first group of card-output instructions, for the earliest point in the cycle, results in operations on the card just read (or, if only an output rather than a combined file, the card just advanced to the equivalent position). Each additional group of card-output instructions for the same file--for the same point in the cycle or for subsequent points --performs the designated output operation on a next card from the same file. The data read (if a combined file) from the first of these several cards remains available for processing.

   The additional cards are not read; they are treated as though they were only output-file cards, even if they are part of a combined file; they do not enter the program logic cycle.

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System/360

Form X24-3352-1
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	Filename	Type (H/D/T)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators	And	And	Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
0 1	O	MULTIMPH	D						N1P									
0 2	O											DTLPRT			210			
0 3	O		D						N1P									
0 4	O											DTLPCH			15			
0 5	O	MULTIMPH	T															
0 6	O											TOTPCH			42			
0 7	O																	

Figure 7. Multiple-Time Output to Cards During One Program Cycle

For example (Figure 7): Output-Format Specifications for all cards of a combined file contain card punching instructions for total-time output, card-printing instructions for detail-time output, and a separate group (separate File Name and/or Type entry) of entries for card punching at detail time. For reference convenience, term successive cards in the file Ca, Cb, Cc, Cd, Ce, Cf, Cg, ...

Result: Ca is read, and its data is available for processing. It is card-printed at detail time. (Note total-time bypass on first card--described in item 4, below.) Cb is not read. It is punched at detail time. Cc is read, and its data is available for processing. It is punched at total time. Cd is not read. It is card-printed at detail time. Ce is not read. It is punched at detail time. Cf, Cg, and Ch repeat the sequence Cc, Cd, Ce; etc.

3. Overflow-Time Output: Regardless of whether a carriage-tape channel-12 punch was sensed during detail-time output or total-time output, all output operations conditioned by <u>on</u> status of the overflow indicator(s) (OF, OV) occur at overflow time, which follows total-time output. Three points should be noted:

   a. Since overflow-time output follows total-time output, all relevant totals are normally printed before page overflow, even when a carriage-tape channel-12 punch was encountered during detail output printing from the last detail card of a control group.

   Appendix E illustrates an RPG programming technique for implementing page overflow prior to total output, when a channel-12 punch was encountered during detail output for the last card of a control group. It is, however, not possible to create an overflow operation between successive detail-time or between successive total-time lines of the same program cycle. Thus, overflow cannot occur, say, between output for several total levels in the same program cycle, even if a channel-12 punch is sensed between these print lines.

   b. Because overflow output occurs at a separate time in the cycle, care must be taken that data does not print more often than desired, when a line is specified to print on an overflow condition as well as on some other condition (such as a control break)--both of which may occur in the same cycle, but at different times. This can be avoided by conditioning the overflow specification line not to apply when the condition for the other specification line applies in the same cycle. (The section on <u>Output-Format Specifications</u> illustrates the case, as does the preceding Introductory Program Example, Figures 5E and 5F.)

c. Overflow-time output is primarily intended for the printing of page headings on forms overflow; but card output operations are also possible--such as card punching, card-printing, and stacker selection. If any card operations are performed at overflow-time output, it must be understood that

   (1) output will apply to whatever card happens to have been read last, and

   (2) the next card may then advance, without being read, as explained in 2, above.

4. Total-Time Processing on "Run-In": In order to prevent undesired totals prior to detail processing of the first card, total-time calculations and total-time output are suppressed until the first card has been processed. Thereafter, further bypassing of total-time calculations and total-time output depend on other factors:

   a. If none or all of the card types in the output specifications have control fields (Control Level) specified, total-time processing is available on all cards after the first--including the portion of the cycle when the LR indicator is on, following the last data card of the pertinent file.

   b. If only some card types have control field(s) specified in the input specifications, total-time processing is bypassed until after the first card of a type with control field(s) specified in the input specifications has been processed.

      If the first card of the deck is of a type that has control field(s) specified in the input specifications, operation is tantamount to a, above.

      If no card of a type with control field(s) specified occurs in the data deck, total-time calculations and total-time output are bypassed for the entire job--including the portion of the cycle when the LR indicator is on, following the last data card of the pertinent file.

   Exception: See GOTO, under <u>Calculation Specifications</u>.

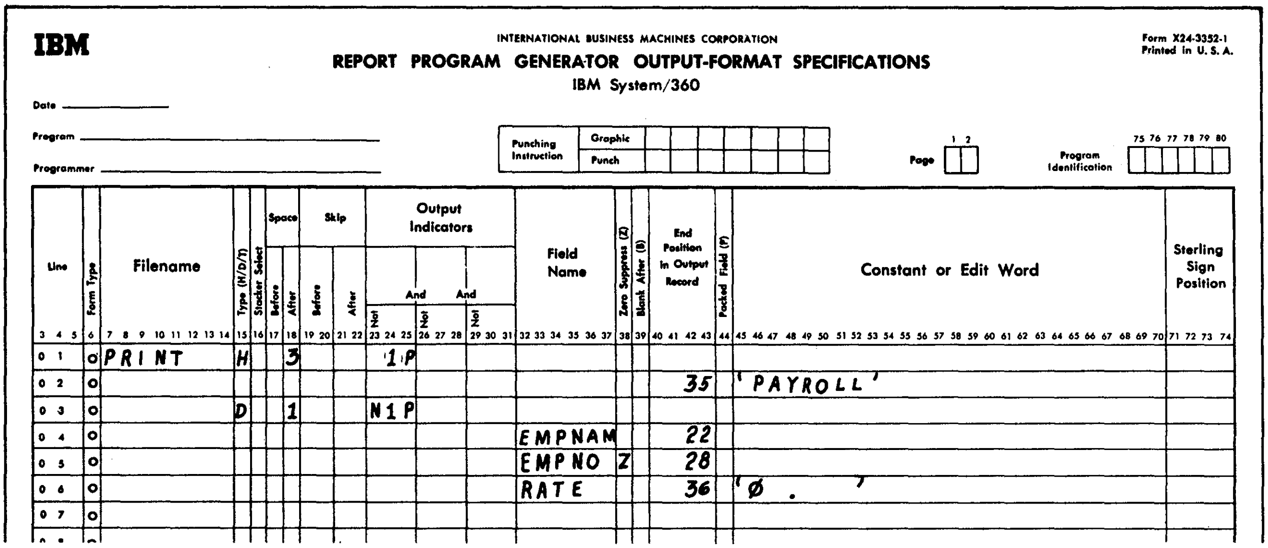
Whether total-time operations are bypassed or not is independent of the <u>status</u> of L-indicators--although L-indicators are normally also utilized to condition operations during total time, when total time is not bypassed. (See also <u>Indicators: L1-L9, L0, LR.</u>)

5. Output Before First Card is Read: As indicated by the first two input/output flow chart symbols at the top of Figure 6, detail (or heading) output takes place, at the start of the object-program run, before the first data card has been read. Therefore, all detail-output specifications--other than constant-data heading lines desired ahead of regular detail output--should be conditioned by an indicator that is not on initially but is on for the appropriate cards, or by the negative status of an indicator that is on initially but off thereafter. One simple method is to condition any initial heading lines of constant data by 1P, and the regular detail output by N1P or by card-type Resulting Indicators. (See Figure 8, and also the section <u>Indicators, 1P</u>.)

   If the detail output is not conditioned, spurious printing and/or card output (depending on the nature of detail output specified) occurs before the first input card has been read.

6. Blank-After (B in col. 39 of Output-Format Specifications): Figure 6 correctly indicates that Field Indicators are set on or off when the new input data is transferred to the process area, and that Calculation Resulting Indicators are set on or off during calculations. However, an indicator assigned to "Zero or Blank" in Input or Calculation Specifications (arithmetic operations or TESTZ) is on initially, and also turned on immediately when an output specification is executed for which "B" (Blank After) is specified in the Output-Format Specifications: as soon as the data for an output line with B in col. 39 is transferred to the output storage area, the field is blanked (if alphameric) or set to zeros (if numeric) and the Zero-or-Blank indicator for that field turns on--before any further output fields are processed. (This does <u>not</u> cause Field Indicators or calculation Resulting Indicators assigned to Plus or Minus to turn off, if they were on.)

   Fields for output at one point in the cycle, under one file-identification entry, are transferred to the output storage area in the sequence in which they appear in the Output-Format Specifications, with one exception:



IBM — INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS
IBM System/360
Form X24-3352-1
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	Filename	Type (H/D/T)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators Not		And Not		And Not		Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
3 4 5	6	7–14	15	16	17	18	19 20	21 22	23	24 25	26	27 28	29	30 31	32–37	38	39	40–43	44	45–70	71–74
0 1	O	PRINT	H			3				1P											
0 2	O																	35		'PAYROLL'	
0 3	O		D			1			N	1P											
0 4	O														EMPNAM			22			
0 5	O														EMPNO	Z		28			
0 6	O														RATE			36		'Ø . '	
0 7	O																				

Figure 8. Output Before First Card is Read

If card punching and document-printing are both specified for the same card, the program transfers all pertinent fields to the output storage for punching first. Therefore, if Blank-After is to be specified, it should be in the document-printing specification--otherwise the data is lost before transfer for document-printing.

The Introductory Program Example (Figure 5) illustrates Blank-After: As soon as NAME has been transferred to output storage for the first detail card of a group, the NAME field is blanked and, therefore, its data is no longer available (to be printed a second time or to be punched or card-printed). Indicator 10 also turns on immediately, and remains on until NAME is again read from a Customer Name card. This also explains why NAME was recorded in the Output-Format Specifications after MOYR and ACCTNO. If it were entered ahead, indicator 10 would be on before MOYR and ACCTNO are printed; they would then not print, being conditioned by N10.

7. Branching within RPG: The dotted line connecting the boxes representing total-time calculations and detail-time calculations indicates that it is permissible to skip between these points in the program cycle by a GOTO operation. It is thus possible, for example, to iterate the program sequence from total-time calculations to detail-time calculations any number of times within the same cycle.

If GOTO is specified from detail-time to total-time calculations, pertinent total-time calculations and output are performed following each such GOTO operation - even when total time would otherwise be bypassed as explained in 4b, above (Total-Time Processing on "Run-In"). This does not, however, prevent total-time bypass on subsequent cards on which it would normally be bypassed.

INDICATORS

Indicators are assigned--either by the programmer or, in some instances, by the RPG program itself--to identify conditions. They are represented in the specifications by two-character codes. (Both characters must always be recorded, even if the first one is the digit zero.) At the point on the specifications sheet where an indicator is assigned to become associated with a condition, it is termed a Resulting Indicator or a Field Indicator. Those indicators assigned by the program itself may also be thought of as resulting indicators, since they are associated with the occurrence of a specific condition or result.

The status (on or off) of a Resulting Indicator or Field Indicator assigned to a specification (a line on a specifications sheet, or a single specification card) reflects the condition resulting from execution or processing of that specification: if the resulting condition satisfies the criterion with which the indicator has been associated, the indicator turns (or remains) on; if the criterion is not satisfied, the indicator turns (or remains) off.

If the same indicator is again assigned to a criterion in the same program cycle, its status can change again. If, on the other hand, the specification (specifications sheet line) with which an indicator is associated is not executed or processed under certain conditions (say, certain card types), then the indicator does not change its status when these particular conditions exist.

Certain indicators are also set on or off, at particular points in the program cycle, by the RPG program itself. (Figure 6, RPG Program Logic, shows the normal relationship of indicators to periods in the program cycle.) In addition, any indicator may be set on or set off by a programmer's instruction (SETON or SETOF) independently of other conditions. Its status thereafter, however, is equally subject to revision by subsequent testing, subsequent SETON or SETOF instruction, or automatic setting by the RPG program.

Indicators assigned to Zero-or-Blank, in Input Field Indicators or in Calculation Resulting Indicators (arithmetic and TESTZ operations), are on at the beginning of program execution and as soon as an output Blank-After specification for the field has been executed. (Turning on a Field or Resulting Indicator--assigned to Zero-or-Blank--by Blank-After, does not cause indicators assigned to Plus or Minus to turn off if they were on.)

Internally in the central processing unit, the "off" or "reset" condition of an indicator is represented by hexadecimal code 00; "on", by hexadecimal F0. (See Appendix D for discussion of hexadecimal code.)

Different indicators may be assigned to any two, or to all three, of the possible resulting or field conditions for one specification line; or, the same indicator may be assigned to more than one condition in one specification line. In the latter case, the indicator turns on if any one of the conditions to which it is assigned has occurred.

The ON or NOT ON status of indicators can be used to condition the execution of instructions in specifications statements; i.e., performance of an operation can be stipulated to be contingent upon the status of particular indicators. An indicator used in this manner is referred to as a "Conditioning Indicator." The two-character indicator code is recorded in the two right-hand columns of a three-column (conditioning) Indicators field to condition execution of the instruction on that line upon ON status of the indicator. If the indicator code is prefixed by the letter N (in the leftmost position of the three-column Indicators field), the instruction is executed only if the indicator is OFF. It is possible to condition the execution of an instruction by a combination of the status of several indicators (termed an AND relationship), or by the acceptable status of one of several indicators (termed an OR relationship).

The same indicator may be assigned as both conditioning and resulting indicator in the same calculation specification line; its status then does not change until after execution of the instructions in that line, and provided the specifications in the line are executed.

The available indicators are itemized below, together with detailed information required for their use. Concurrent reference to Figure 6 is assumed. Immediately after the itemized discussion of all indicators, a brief section explains the priority of indicator settings when the setting of an indicator is controlled in a non-standard fashion. Further specifics, including limitations, are stated in the specifications sheets sections to which they apply.

Note: A particular indicator can have different assignments; i.e., it can be a card-type Resulting Indicator, Field Indicator, calculation Resulting Indicator, etc.--but it is always the same indicator. These different available assignments merely determine the conditions under which, and the points in the program cycle at which, the indicator setting occurs. Figure 11 (Indicator Hierarchy) summarizes the priority of indicator settings, and shows the status of each indicator at the beginning of program execution.

THE SPECIFIC INDICATORS

01-99 (General Indicators)

Normal uses: Identification of card types, of status of input fields, of results of calculations and comparisons; conditioning of calculations and output based on status of these indicators; determination of search in a table look-up operation.

Any of these ninety-nine numbers may be assigned by the programmer to be associated with the occurrence of a specific condition. When the criterion has been satisfied, the indicator turns on. It remains on until a criterion for that indicator has been tested again, and not satisfied.

These indicators are off at the beginning of object-program execution (unless assigned to Zero-or-Blank).

Four examples (Figures 9A and B):

1. Indicator 10 is assigned as Resulting Indicator, for a Minus result, on one line of the calculation specifications. Execution of that line itself is not conditioned.

   Effect: Indicator 10 turns on after the instruction of that line has been executed the first time, if the result was negative. (It remains off if the result was not negative.) It then remains on (or off) until tested next. The same test is performed at that point on every card, indicator 10 being turned (or left) on or off at that point in the cycle, depending on whether the result of that calculation is negative or not negative.

2. As (1), above, but the calculation specifications line on which indicator 10 is set, itself is conditioned to be executed only during detail processing of a card type with indicator 01.

   Effect: Indicator 10 turns on after the instruction on that line (the third line) has been executed the first time, if the result was negative. (It remains off if the result was not negative.) However, the instruction on that line is only executed if the card being processed is indicator 01 type.

IBM **9A** INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS
IBM System/360
Form X24-3350-1 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	Filename	Sequence	Number (1-N)	Option (O)	Resulting Indicator	Record Identification Codes 1: Position	Not (N)	C/Z/D	Character	2: Position	Not (N)	C/Z/D	Character	3: Position	Not (N)	C/Z/D	Character	Stacker Select	Packed (P)	Field Location: From	To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field-Record Relation	Field Indicators: Plus	Minus	Zero or Blank	Sterling Sign Position
01	I	PAYRLSUM	AA			05	80		C	P																					
02	I																				11	16	2	GRSEAR					10		
03	I																				1	5		EMPLNO							4
04	I																				20	25	2	NETEAR					10		
05	I																														

IBM **9B** INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS
IBM System/360
Form X24-3351-1 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	Control Level (L0-L9, LR)	Indicators: Not		And: Not		And: Not		Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Plus (High 1>2)	Minus (Low 1<2)	Zero or Blank (Equal 1=2)	Comments
01	C								SALES	SUB	RETRNS	NETSLS	9	2			10		1
02	C																		
03	C			01					SALES	SUB	RETRNS	NETSLS	9	2			10		2
04	C																		
05	C								SALES	SUB	RETRNS	NETSLS	9	2			10		
06	C								NETSLS	ADD	TOTSLS	TOTSLS	10	2					3
07	C								PROFIT	SUB	RTRPRF	NETPRF	8	2			10		
08	C																		
09	C								NETEAR	SUB	BONDS	PAY	6	2			10		4
10	C																		

Figure 9. Indicators 01-99

Therefore, once on (or off), indicator 10 remains on (or off) until that specification line instruction is encountered again while detail-processing a type 01 card.

The equivalent situation applies if a Field Indicator is assigned to Plus or Minus on the Input Specifications, and that field does not apply to all card types. The indicator then remains on (or off), until the field is tested again when the appropriate card type recurs.

3. As (1), above, but indicator 10 is assigned to two lines (say, lines 5 and 7) in the Calculation Specifications as Resulting Indicator. Execution of neither line itself is conditioned.

   Effect: Indicator 10 turns on after the instruction on line five has been executed the first time, if the result was negative. (It remains off if the result was not negative.) It is then on (or off) through the execution of the instruction on the seventh line. Its status thereafter, until the line-five instruction has been executed on the next card, is based on the result from line seven.

4. Indicator 10 is assigned as Field Indicator to a negative condition for field 1 and field 3 of every input card, and for a negative result on the ninth line of the calculation specifications (whose execution is not itself conditioned).

   Effect: Indicator 10 turns on prior to the detail calculations for a card, provided field 3 is negative--i.e., the fields are tested in the order in which they are written on the Input Specifications sheet. Therefore, the status of field 1 is ignored for indicator 10.

   Indicator 10 retains its status as determined by field 3 until completion of the instruction on line nine of the Calculation Specifications sheet. The result there determines its status until the beginning of detail calculations for the next card, when field 3 of that card determines the status of indicator 10.

Note: If any indicator 01-99 is set on or off by the special operation code SETON or SETOF, it remains in that status until again SETON or SETOF, or until after execution of a specification line where the indicator is a Resulting or Field Indicator, whichever occurs first. In the latter case, the resulting condition determines the status of the Indicator.

H1, H2 ("Halt")

Principal purposes: To cause a program halt after an unacceptable condition has occurred, and/or to bypass calculations and/or output on erroneous conditions.

These two indicators operate like indicators 01-99, with this difference: If H1 or H2 has been turned on (as a Resulting or Field Indicator, or by the instruction SETON), and has not been turned off again during the same program cycle (by SETOF, or as a Resulting or Field Indicator for a tested condition that was not satisfied), the system will halt after completion of the next detail-time output operations. (The halt actually occurs just after the next card has been read.)

The system can be restarted, and the job continued, at the operator's option, simply by pressing the START key on the CPU console twice (i.e., the halt is "non-abortive")--see the RPG Operating Procedures manual--halts EOF, EF0, EFF. If the system is thus restarted, the H1 and H2 indicators are set off by the program immediately upon restart.

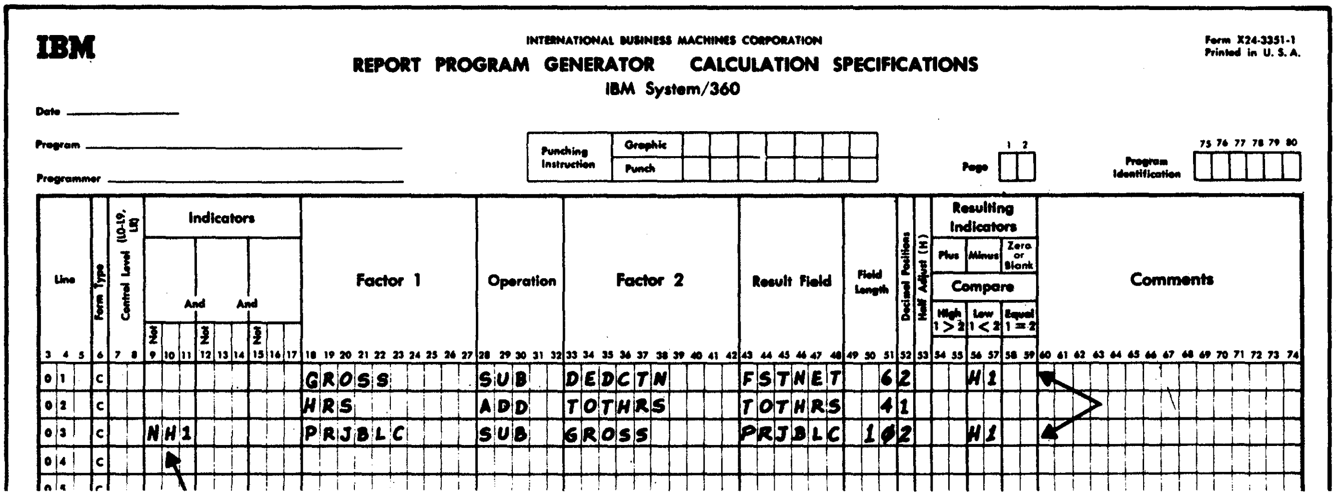
These indicators are off at the beginning of object-program execution.

Note:
1. If H1 or H2 is to cause a halt when any of several conditions exist, care must be taken not to turn the indicator off again by a later test which was not satisfied, if an earlier test turned it on.

   For example, if H1 is to be on when the result from the first and/or third line in the calculation specifications is negative, the test for the third line must be suppressed whenever the first line yielded a negative result. Otherwise, although the first-line result may have been negative, a non-negative third-line result will turn H1 off again before the system halt has occurred. (Figure 10 illustrates how to handle this problem.)

   This warning also applies if the same indicator is assigned to several fields, as Field Indicator on input specifications, since the fields are examined in the sequence in which they are written.

   Of course, this fact can be used to advantage deliberately to turn off H1 or H2 if a subsequent field or calculation meets a desired criterion.



Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust	Resulting Indicators Plus (High)	Minus (Low)	Zero or Blank (Equal)	Comments
01	C			GROSS	SUB	DEDCTN	FSTNET	6	2			H1		
02	C			HRS	ADD	TOTHRS	TOTHRS	4	1					
03	C		NH1	PRJBLC	SUB	GROSS	PRJBLC	10	2			H1		
04	C													

Figure 10. H1 Indicator On if Either or Both of Two Conditions Exist

2. Indicator H1 or H2 assigned to Zero-or-Blank is off at the beginning of program execution (see Indicator Hierarchy); but it will be turned on by a Blank-After output specification, like any other indicator. A system halt then occurs at the end of processing for that card.

1P ("1st Page")

Primary purpose: To condition fixed-data (constants) output to occur preceding the processing of the first card. It is normally used for report headings.

This indicator is set by the program itself. It is on prior to the processing of the first card, and turns off after detail-output time preceding the processing of the first card (see Figure 6). Figure 8, line 01 illustrates a common use of 1P.

The 1P indicator may also be assigned as a Resulting Indicator or Field Indicator, and SETON or SETOF, like indicators 01-99. Besides being on prior to processing of the first card, it will then operate like indicators 01-99--except that 1P always turns off after detail-time output operations, and does not turn on again unless SETON or used as Resulting or Field Indicator.

MR ("Matching Record")

Primary purpose: To identify cards in a file that match those in another file on control data, and to condition calculations and output based on the status of MR.

MR turns on when a primary-file card matches a secondary- or tertiary-file card, on the contents of a designated field or fields. If several fields are designated for matching, all of them must match. (See also section below titled Matching of Card Files.)

When the indicator turns on as a result of the matching of a primary- and a secondary- or tertiary-file card, or turns off as the result of a non-match, this occurs after overflow-output time (see Figure 6). The MR indicator is then on or off for complete cycles, beginning before detail-calculation time and through the next total and overflow-output time. If all primary-file cards match secondary-file (and tertiary-file, if present) cards on the designated field(s), and vice versa, the MR indicator remains on through the end of the job.

If card types for which no matching fields are designated intervene, they are treated--for purposes of feeding and processing--as belonging to the same matching-field(s) group as the preceding card in the same file; but the MR indicator is off for their processing cycle. However, the contents of the field(s) designated for matching are stored from the last preceding card for which matching was specified. Thus, when the next card for which matching fields are designated is processed, its matching-fields contents are first compared with those of the last preceding card for which matching fields were specified. The MR indicator therefore is on for the processing of all cards whose designated fields match, even if cards that are not to be matched occurred in the middle of a matching group.

The MR indicator may also be used as Resulting or Field Indicator, or SETON or SETOF, like indicators 01-99. It is, however, always turned on before detail-time processing of a card that satisfies the criteria, above, for matching records; it always turns off before detail processing of a card that does not meet these criteria--as though its status had never been subjected to any other criteria. When only a single input (or combined) file is being processed, MR is always turned off by the program before detail time.

The MR indicator is completely independent of any control levels (L1-L9--see below) that may be specified for control-group totals or group indication. It is common, but by no means necessary, that the matching fields are the same as the total-level control fields.

OF, OV (Overflow)

Primary purpose: To control the printing of identifying data on overflow pages.

Indicator OF refers to the standard paper-tape-controlled carriage, or the lower-feed carriage if the Dual-Feed Carriage special feature is installed on the IBM 2203 Printer. OV applies to the upper-feed carriage of the Dual-Feed Carriage feature.

If a line is printed after a punch in carriage-tape channel 12 for the relevant carriage is sensed, the appropriate indicator (OF or OV) is turned on. The indicator is turned on after all detail-time output for a program cycle has been completed, if the channel-12 punch was sensed during detail output. If the channel-12 punch was sensed during total output, the indicator is turned on after all total-time output for a program cycle has been completed. Regardless of whether the OF (or OV) indicator is on as a result of detail or total output, it remains on until conclusion of the next detail output time (see Figure 6). It then turns off, unless a channel-12 punch is sensed again during that detail-output time. Therefore, if calculations are to be conditioned by the status of the OF or OV indicator--and a channel-12 punch could be sensed during either detail or total output--these calculations should be performed at detail-calculation time. (See also Overflow Indicators - OF, OV, and Space, Skip, under Output-Format Specifications).

If OF (or OV) is used as a conditioning indicator in a file-identification output-specification line, it thereby designates that that output is to be performed at Overflow Time of the program cycle. This applies regardless of whether the specification line containing OF or OV is independent, is in an OR relationship with the line above or below, or is in an AND relationship with a contiguous line; however, it does not apply to NOF or NOV.

If the OF or OV indicator is specified as a conditioning indicator in any file-identification output-specification line, no forms skipping ever takes place for that file (standard or upper carriage, respectively) without an Output-Format forms-skip specification; otherwise, forms advance to channel 1 is automatic for the respective file (standard or upper carriage) after total-output time if the OF or OV indicator is then on. (Note that this refers only to OF or OV - not NOF or NOV.)

Note: If a channel-12 carriage-tape punch is passed while forms skipping from a higher point on the page to carriage tape channel 1 (e.g., skipping to a new page after total output above the overflow point of the page), the OF or OV indicator is not turned on as a result of having skipped past the channel-12 punch. The OF or OV indicator will, however, be turned on as a result of forms skipping past a channel-12 punch to any channel punch other than that in channel 1, unless this is past a channel-1 punch; it then remains on until conclusion of the next detail output time.

The OF and OV indicators may also be used as Resulting or Field Indicators, or SETON or SETOF, like indicators 01-99. The indicator setting resulting from such technique will, however, be superseded at the end of detail-time output and at the end of total-time output of each program cycle by the status that the indicator would assume if it were used only in its normal (channel-12 signal) manner.

L1-L9 (Control Levels)

Principal purposes: To recognize control breaks, produce totals at the desired levels, and permit group-indication or group printing.

However, these indicators can function in several ways, and are not limited to control-group identification:

1. If entered in the "Control Level" columns (cols. 59-60) next to a field name in the input specifications, such a field is thereby designated a control field of that level.

   This has the following effects: Whenever the contents of the designated control field of a pertinent card do not match the contents of the equivalent (same control level) field of the last preceding card with such control level designated, the specified L-indicator turns on--as well as all L-indicators of lower levels.

The L-indicators turn on prior to the total-calculations time that precedes detail processing of the new card (see Figure 6); they turn off after detail-output time of the new card.

If card types without the relevant control level designated intervene, they are treated as belonging to the same control group as the preceding card, and the L-indicator does not turn on. If the next card with such control level designated then contains the same control information as the last preceding card with that control level specified, the new card does not set the L-indicator on. (See Figure 13B.)

Note: See Definition of Terms (in General Information section) and Control Level (Input Specifications, cols. 59-60) for the relation of blanks, zeros, zone punches alone, and ± sign punches, to control fields.

2. Indicators L1-L9 also turn on when the LR indicator (described below) turns on, following the last data card.

3. The L1-L9 indicators may also be used as Resulting and Field indicators, or they may be SETON or SETOF. If an L-indicator is turned on or off in this manner, lower-level L-indicators do not automatically turn on or off also (e.g., L2 could be on and L1 off).

   If L-indicators are used in this fashion in the same program in which L-indicators are also assigned for Control Levels in input specifications, the Resulting or Field Indicator setting will supersede any prior control-break L-indicator status. The exact time relationship in the program cycle is apparent in Figure 6.

   In any event, L1-L9 are turned off after conclusion of detail-output time.

4. When any L-indicator is specified in the "Control Level" field (cols. 7-8) of a calculation specification, it thereby designates that the particular instruction is to be executed during total-time calculations--and simultaneously conditions it to be executed only if that particular L-indicator is on at that time; i.e., it normally serves there to confine calculations to total time after the end of a control group of that or higher level. (It should then also be considered in an AND relationship to indicators in cols. 9-17.) When Control Level (cols. 7-8) of calculation specifications is blank, the particular instruction is to be executed at detail time.

5. Any L-indicator may be used as a conditioning indicator, like other indicators:

   a. If any of the indicators L1-L9 (employed in the normal manner) appears in Indicators (cols. 9-17) of calculation specifications--and Control Level (cols. 7-8) is blank--the specifications are executed at detail time during the processing of the first card of a control group of that or higher level.

   b. If any of the indicators L1-L9 (employed in the normal manner) appears in Output Indicators, the output is performed only if a control break of that or higher level has occurred.

      (i) If the indicator is associated with total-time output (T in col. 15 and no OF or OV in Output Indicators of the file-identification line), the output occurs only at total time after processing of the last card of the control group.

      (ii) If the indicator is associated with detail-time output (D or H in col. 15 and no OF or OV in Output Indicators of the file-identification line), the output occurs only at detail time during processing of the first card of the control group--i.e., group indication is performed.

      (iii) If OF (or OV) is specified in Output Indicators of the file-identification line, the output is performed at overflow-output time (if OF or OV is on), but only if the overflow occurred at the end of the control group.

Special considerations for Indicators L1-L9 on "Run-In"

At the start of the job run, the core storage areas for all control fields contain zeros (hexadecimal F0--see EBCDIC table, Figure D1, Appendix D). The control-field contents of the first card with control level(s) specified in the input specifications are, therefore, compared with zeros. Furthermore, as previously stated, no control-break test is made when processing a card type for which Control Levels are not specified in the input specifications. Therefore, L-indicators (when used in the normal manner, to signal control breaks) operate as follows, at the beginning of object-program execution:

a. None of the indicators L1-L9 is on while processing a card type for which Control Levels are not designated in the Input Specifications.

b. The first card for which control levels are designated in the Input Specifications is tested: the card contents of the designated control field(s) are compared with zeros. If the card field contents are unequal to zero, the L-indicator of the level specified for that field--and for all lower L-levels --turns on. It remains on (unless set off by one of the methods described under 3, above) until completion of detail-time output for that card, when the L1-L9 indicators are set off by the program. (The L-indicators are thus available on the first card--if control field contents were unequal to zero--to control detail-time group-indication operations.)

Note: As pointed out under Definition of Terms, designation of a field as numeric (col. 52 of input specifications) causes conversion of blanks to zeros, and stripping of zones except in the low-order position. Furthermore, a low-order-position zone is also omitted from numeric data when it is stored in a separate location for control-level data. Therefore, numeric control fields containing only blanks, and /or zeros, and/or zone punches, in the first card with control fields, will result in an "equal" comparison with zeros, and will not set L-indicators on.

On the other hand, blanks, zones, and zeros are distinguished for alphameric fields. Therefore, while zeros in alphameric control fields (of the first card with control fields specified) also will not set the relevant L-indicator(s) on, blanks and zones in an alphameric field will, because they are unequal to the zeros contained initially in the control-field core storage areas.

See Programming Tips, Appendix E, for a technique that assures group indication for the first card even if the control fields contain only zeros.

The setting and status of L-indicators are independent of whether or not total-time processing is bypassed (see section titled Program Logic). Indicators L1-L9 are off at the beginning of object-program execution.

L0 (L Zero) (Universal Total)

Primary purpose: To facilitate calculations at Total Time even though no control break has occurred.

This indicator is on at the start of program execution and is never turned off by the RPG program itself. It is considered a total-level indicator (like L1-L9), because its entry in the Control Level field (cols. 7-8) of a calculation specification designates that calculation specification to be executed during total-time calculations (and provided L0 is on). This facilitates designating the execution of a calculation specification for total time, even though no control break--to set any of the L1-L9 indicators--may have occurred.

For example: Calculations during total-time processing of an unmatched card (say, a blank trailer card) if the preceding card was a matched record--note that, at total time, the MR indicator still reflects the matching-fields status of the preceding card.

Numerous other examples appear throughout the manual, particularly among Programming Tips (Appendix E). Notably, L0 makes it possible--without a control break --to perform calculations after processing a card when data, MR, and Field Indicator settings from that card are still available while the card-type Resulting Indicator for the next card is already on.

The L0 indicator may also be used as calculation Resulting Indicator, or SETOF or SETON; but it must not be assigned as card-type Resulting Indicator or as Field Indicator. Two points must be borne in mind:

1. The L0 indicator is on initially, until turned off by the result of a programmer's specification; and

2. If L0 is thus turned off, it is turned on again by the RPG Program after a new card has been read (see Figure 6--same point in the cycle where other L-indicators are turned off).

LR (Last Record)

Primary purpose: To provide for processing final totals at end of job, and to terminate job.

This indicator turns on, before total-time calculations, following the processing of the last data card of an input (or combined) file. When a multi-file program is being executed, entries in the File Description Specifications designate which file(s) must be completed before a Last

Record condition exists (i.e., before LR turns on). (Actually, it is the first End-of-File--/*--card in the pertinent file(s) that causes LR to turn on.)

When LR turns on as the result of the last record condition, indicators L1-L9 also turn on.

The LR indicator may also be used as Resulting and Field Indicator, or SETON or SETOF. Indicators L1-L9 do not, then, also turn on or off automatically.

The LR indicator is considered a total-level indicator (like L1-L9), because its entry in the Control Level field of a calculation specification designates that calculation specification to be executed only during total-time calculations (and provided LR is on).

If LR is on at the conclusion of total-time output, the program terminates after total-time output. It cannot be restarted. When LR is used in the normal manner--to recognize the end of the appropriate data file(s)--it can be utilized only to condition total-time operations; no further overflow-time or detail-time operations will occur. If the LR indicator is on at the conclusion of detail time (i.e., it was turned on by other than the last-record condition), it is turned off right after that point by the RPG Program.

In the rare situation--described in the section _Program Logic, Total-Time Processing on "Run-in"_--where only some card types have control field(s) specified in the input specifications, and no cards of these types ever occur throughout the job, total-time processing will be bypassed throughout the job. Therefore, even though the LR indicator will turn on before total-time processing, it cannot be utilized since no total-time processing will take place. The program will terminate as soon as LR has turned on.

INDICATOR HIERARCHY

The program classifies indicators in four priority groups. This is of concern to the user only when he chooses to employ an indicator in a non-standard fashion.

Any indicator may be designated as a resulting or field indicator, and used as a conditioning indicator, in any specifications fields provided for such entries (except that Control-Level fields are limited to L-indicators, and some restrictions apply to L0). However, for unconventional application of an indicator, Figure 11 may have to be consulted--in addition to Figure 6 (Program Logic)--to assure that the indicator will not be set on or off by the RPG Program at a point in the cycle not desired by the user. The hierarchy order in Figure 11 indicates the priority sequence applied by the RPG Program in setting indicators.

Examples:

1. MR is used only as card-type Resulting Indicator in Input Specifications. Only a single input file exists. (See Figure 12A, lines 01 and 13.)

   Effect:
   a. MR turns on before total-time calculations, if the card read was of the type to which MR is assigned. MR is turned off by the RPG Program itself before the input data is moved to the process area, preceding Detail-Time calculations for the card.

   b. Note that, if the MR indicator is used as a card-type Resulting Indicator in an OR relationship, the wrong input fields may be moved to the process area: MR has been turned off by the RPG Program itself before it can serve to implement Field-Record Relation.

   If MR is also set on during detail-time calculations, it remains on through total-time calculations of the next cycle--even if the new card is not the type to which MR is assigned as card-type Resulting Indicator.

   Reason: The MR indicator belongs to a higher group (hierarchy group 1) than card-type Resulting Indicators (group 2).

2. As (1), above, but MR is also used in the normal manner; i.e., the program is multifile with matching fields. (See Figure 12A, lines 03-13.)

   Effect: As (1), above; but MR is turned off or on (or remains on) before detail-time calculations, depending on the result of matching the contents of control fields between files. Since MR may be turned on also by card type in this example, it could be on during total-time calculations and output even if the preceding records did not match between the files. On the other hand, it could be on--as a result of matching records--and thus implement the wrong Field-Record Relation, when input fields are transferred to the process area.

   Since MR may be on, in this method of use, as both a card-type Resulting

STATUS AT BEGINNING OF OBJECT-PROGRAM EXECUTION	INDICATOR	HIERARCHY LEVEL (1=highest)	SET ON OR OFF BY RPG PROGRAM ITSELF: CRITERIA, and TIME IN THE PROGRAM CYCLE
OFF	MR	1	Set after overflow-output time: If multi-file program, and Matching Fields match - set ON; all other situations - set OFF.
OFF OFF	OF OV	1	Set ON after total-time output if printing occurred at or below carriage tape channel-12 punch during total-time output; set ON after detail-time output and after total-time output if printing occurred at or below channel-12 punch during detail-time output. Set OFF after detail-time output if no channel-12 punch encountered during detail-time output.
ON	1P	2	Set OFF each time a data card has been read. Never set ON again by the RPG Program itself.
OFF	H1,H2	2	Set OFF upon system restart following the Halt after a new card has been read. Never set ON by the RPG Program itself.
ON	L0	2	Set ON each time a data card has been read. Never set OFF by the RPG Program itself.
OFF	L1-L9	2	Set OFF each time a data card has been read. Set ON by the RPG Program itself only when a higher L-indicator or LR turns ON in normal use.
OFF	LR	2	Set ON before total time following last data card of appropriate file(s). If set ON by programmer during detail-time calculations, will be set OFF by the RPG Program itself after next card has been read.
OFF	CARD-Type RESULTING Indicators	2	Set OFF each time a data card has been read. Never set ON by the RPG Program itself.
ON	ZERO-or-BLANK Field and Calculation Resulting Indicators	3	Set ON or OFF based on contents of Input Specifications field (if Field Indicator), or Calculation Specifications Result Field (if Resulting Indicator) when tested. Also set ON by BLANK-AFTER specification on output. Never set OFF, or ON again, by RPG Program itself.
OFF	ALL OTHERS	4	Set ON or OFF based on contents of field when tested. Never set ON or OFF by RPG Program itself.

Figure 11. Hierarchy and Summary of Indicators

Indicator and for matching records, two card-type indicators could be on, for part of the cycle, during processing of one card.

3. Indicator 10 is used as card-type Resulting Indicator for a card type (say, CUSTMAST). It is also assigned as Zero-or-Blank indicator to an input field (say, GROSS) in another card type (say, TOTPURCH), and as Zero-or-Blank resulting indicator in a detail-time calculation (say, line 01) of TOTPURCH cards. (See Figure 12B.)

Effects: Indicator 10 is always set on or off--depending on the card type (on for CUSTMAST, off for others)--before total-time calculations, regardless of prior settings by its other uses.

IBM

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

Form X24-3350-1
Printed in U.S.A.

Date ____________
Program ____________
Programmer ____________

Punching Instruction	Graphic							
	Punch							

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

Line (3-5)	Form Type (6)	Filename (7-14)	Sequence (15-16)	Number (1-N) (17)	Option (O) (18)	Resulting Indicator (19-20)	Record Identification Codes 1: Position (21-24)	Not (N) (25)	C/Z/D (26)	Character (27)	2: Position (28-31)	Not (N) (32)	C/Z/D (33)	Character (34)	3: Position (35-38)	Not (N) (39)	C/Z/D (40)	Character (41)	Stacker Select (42)	Packed (P) (43)	Field Location From (44-47)	To (48-51)	Decimal Positions (52)	Field Name (53-58)	Control Level (L1-L9) (59-60)	Matching Fields or Chaining Fields (61-62)	Field-Record Relation (63-64)	Field Indicators Plus (65-66)	Minus (67-68)	Zero or Blank (69-70)	Sterling Sign Position (71-74)
01	1	MASTRFIL	BB			MR	8Ø		Z	J																					
02	1																														
03	1	MASTRFIL	Ø1	1		Ø1	8Ø		Z	J																					
04	1																				1	5		EMPNO		M1					
05	1																				6	1Ø	3	RATE							
06	1		Ø2	1	O	Ø4	8Ø	N	Z	J																					
07	1																				1	5		EMPNO		M1					
08	1																				21	25	2	BONDBL							
09	1	TIMECARD	AB			Ø5	8Ø		D	1																					
10	1	O	R			MR	8Ø		D	2																					
11	1																				4	8		EMPNO		M1					
12	1																				1	3	1	HOURS			Ø5				
13	1																				77	79	1	HOURS			MR				
14	1																														

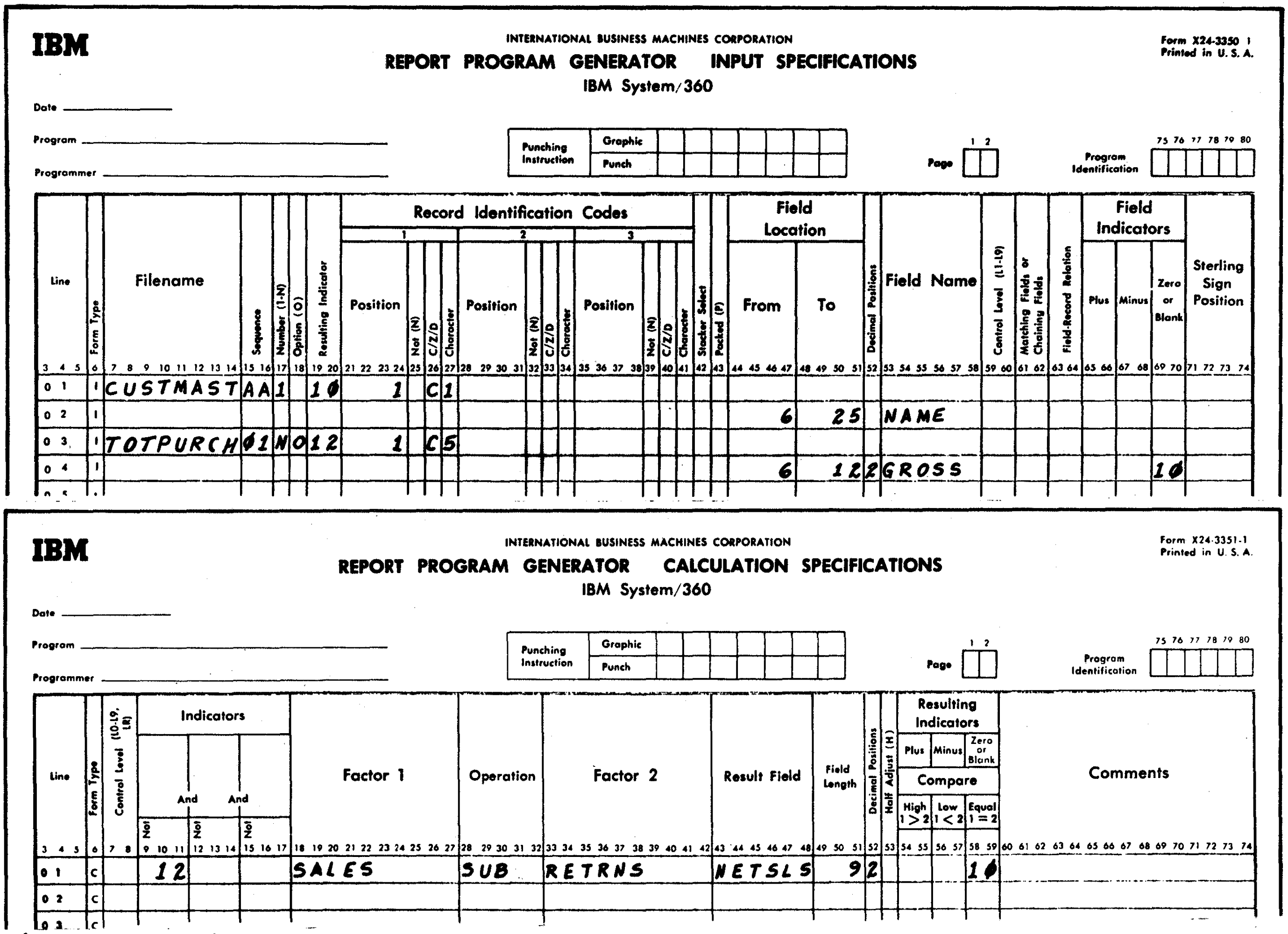
Figure 12A. Hierarchy of Indicators - Illustration of Examples 1 and 2

When TOTPURCH card is read, indicator 10 turns on when input data is transferred to the process area, if GROSS field is zero or blank; its status is again determined by the result of line 01 of the detail-time calculations of TOTPURCH card. If the next card read is not CUSTMAST, indicator 10 is set off before total time of the new card.

Reason: Card-type Resulting Indicators take precedence over Zero-or-Blank Indicators--hierarchy group 2 versus group 3. Therefore, Indicator 10 is turned off before total time if the card just read is not CUSTMAST type. It is turned on by blank or zero in GROSS field of TOTPURCH card before detail time, when input fields are transferred to the process area--because this occurs later than the resetting of card-type resulting indicators.

Thus, it is possible for more than one card-type indicator to be on at the same time, for part of the cycle--e.g., the card-type indicator assigned to TOTPURCH type plus indicator 10, serving to identify CUSTMAST card type but possibly turned on by GROSS field of TOTPURCH card.

Note: Initially, indicator 10 is off, because card-type Resulting Indicators take precedence over Zero-or-Blank in Field Indicators and calculation Resulting Indicators. However, a Blank-After output instruction for the field GROSS or NETSLS will turn it on; it is then turned off again before total time of a card of type 12.



IBM — INTERNATIONAL BUSINESS MACHINES CORPORATION — REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS — IBM System/360 — Form X24-3350 1 — Printed in U. S. A.

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Resulting Indicator	Position	Not (N)	C/Z/D	Character	From	To	Decimal Positions	Field Name	Zero or Blank
01	I	CUSTMAST	AA	1		10	1		C	1					
02	I										6	25		NAME	
03	I	TOTPURCH	01	N	O	12	1		C	5					
04	I										6	12	2	GROSS	10

IBM — INTERNATIONAL BUSINESS MACHINES CORPORATION — REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS — IBM System/360 — Form X24-3351-1 — Printed in U. S. A.

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Plus	Minus	Zero or Blank / Equal (1 = 2)	Comments
01	C		12	SALES	SUB	RETRNS	NETSLS	9	2				10	
02	C													

• Figure 12B. Hierarchy of Indicators - Illustration of Example 3

MATCHING OF CARD FILES, AND SEQUENCE CHECKING

A primary file can be matched against one or two other files, defined as secondary and tertiary file, respectively. A secondary file cannot be matched against a tertiary file.

In order for a file to be matched against another, its name must be entered in the Input Specifications; i.e., it must be either an input file or a combined file (see Definition of Terms). The order in which input or combined files are entered in the Input Specifications determines their relative priority: the first file defined thereby becomes the primary file; a next file entered becomes the secondary file; and if a third file is defined in the Input Specifications, it is thereby designated the tertiary file.

The criteria for matching of files are the contents of one, two, or three card fields, defined as Matching Fields (M1, M2, and M3)--see Input Specifications. The number of Matching Fields (one, two, or three) used must be the same for all files, and for all card types for which matching is specified. Card columns of different Matching Fields in the same card can overlap; but the total length of all Matching Fields for one file (each M1, M2 and M3 counted once) must not exceed 144 characters. (Note, however, that - even for overlapped fields - the program stores the data from the different levels of Matching Fields contiguously, without overlapping.) The length of a Matching Field (M1, M2, or M3) must be the same throughout (i.e., in all records to be matched). Matching Fields may be defined as alphameric or numeric (all zones stripped), and this designation need not be uniform for the several specifications entries of one Matching Fields level (i.e., it may differ between files and card types, provided the field name differs). However, if any Matching Field is defined as numeric for one card type, all Matching Fields of that level (M1, M2, or M3) in all card types are treated as numeric for the Matching Fields operation--i.e., all zones are ignored in the match, and blanks are converted to zeros.

Note: Contents of fields to be matched are stored separately for the Matching Fields operation. This is independent of storage of the data for other purposes (calculations, output, etc.).

When more than one input (or combined) file is specified, matching of the primary file to the other input (or combined) file(s) is mandatory; i.e., at least one Matching Field is then required for each such file. At least one card type in each input (or combined) file must have Matching Field(s) specified.

Whenever Matching Fields are specified, sequence checking on the Matching Field(s), of all card types being matched, is automatic. The files being matched are treated as a single sequential file. The sequence may be specified as ascending (A) or descending (D), but must be the same for all files being matched. The sequence is checked according to EBCDIC (see Appendix D, Figure D1); but any other sequence can be substituted by a translation table (see Appendix D). It is not possible, if there is more than one input (or combined) file, for the files to be in random sequence, even if the cards in all files are in the same order and would match on the Matching Field(s). An error in the card sequence stops the program. The program can be restarted--see IBM System/ 360 Model 20, Report Program Generator for Punched-Card Equipment, Operating Procedures (Form C26-3800). Only an error in the direction of the sequence is detected: a stop on duplicates is not part of the Matching Fields operation (it can, however, be accomplished by calculation specifications).

PROCESSING SEQUENCE OF MULTIPLE FILES

The sequence in which cards from multiple files are processed resembles a standard IBM Collator match or merge operation, except that there can be three files (if the I/O devices required for three files form part of the system).

> When primary-file cards match secondary-file cards, all matching primaries are processed first, followed by all matching secondaries. Refer to the shaded section of the processing flowchart, Figure 12.1.
> In the case of three files, all matching primaries are processed first, followed by all matching secondaries (if any), followed by all matching tertiaries (if any).
>
> When cards do not match, those with matching-field contents earliest in the sequence (ascending or descending), are processed first. Refer to block 6 of the processing flowchart, Figure 12.1. When matching-field values in secondaries and tertiaries are equal, the secondary-file cards are processed first.

Note: When processing of only one file is required, this condition can be satisfied by entering the appropriate resulting indicator, from cols. 19-20 of the Input Specifications, into cols. 9-17 of the Calculation Specifications, or cols. 23-31 of the Output-Format Specifications.

Figure 12.1 illustrates the processing sequence of cards in two files. The files are in ascending sequence. The shaded area shows the processing sequence when a matching-record condition exists, that is, the matching-field of a primary-file card equals the matching field of a secondary-file card. The MR indicator is turned on before the first card of the matched primary file is processed. The MR indicator remains on during the processing of all following primary and secondary-file cards that contain the same matching field. The MR indicator is turned off when all total calculations and output are completed for the last matching secondary-file card.

A card type for which no Matching Field is specified is processed immediately after the card it follows in the file, like a trailer card. Such cards at the front of a file are processed before cards in any other file. (If they appear at the front of more than one file, the normal priority applies: primaries, secondaries, tertiaries.)

Whenever a primary-file card matches a secondary- or a tertiary-file card, the MR (Matching Record) indicator turns on before detail-time calculations (see Figure 6 – Program Logic Flow); it remains on for the processing of all primaries with the same Matching Field(s) value(s). It also remains on for the processing of all secondaries and tertiaries that match the primary. If the MR indicator is turned off during the processing of one of these matched cards, by a programmer's specification, it turns on again, before detail-time calculations of the next card that belongs to the matched group. The MR indicator turns off (before detail time) for the processing of a card whose Matching Field(s) contents do not match those in the relevant other file.

The MR indicator also turns off during the processing of a card type for which no Matching Field is specified. However, such cards

1. are ignored in the sequence check,

2. do not destroy the value(s) stored for sequence checking from the last preceding card with Matching Field(s) specified, and

3. do not destroy the value(s) stored for file matching.

Cards continue to be processed from the same file until the next card is read for which Matching Fields are specified. The normal matching and sequence-checking operations then resume. The Matching Fields values in the new card are compared with matching and sequence values stored before the not-to-be-matched card(s) intervened.

The status of the MR indicator may be utilized to control calculations and output operations, including stacker selection (e.g., to direct unmatched cards to separate stackers).

Normally, stacker selection based on the status of the MR indicator should be at detail-time output--otherwise the next card is selected, since the MR indicator reflects the matched or unmatched status of a card beginning at its detail time and through the ensuing total time, when the next card is in output position.

In order to stacker-select cards, or control other output operations (i.e., punching and card-printing), on the basis of the MR indicator (in a multi-file operation), the file must be defined as a combined file (C in File Description Specifications).

The matching of files also makes it possible to punch and/or document-print data from primary-file cards into matched secondary- and/or tertiary-file cards, or from matched secondary-file cards into tertiary cards of the same matching group*. Similarly, contents (codes, data) from a card in higher-priority file can be used to condition operations for a matching card in a lower-ranking file (primary to secondary and/or tertiary, secondary to tertiary). The converse is not possible, because matching cards of a higher-priority file are completely processed before a matching card from a lower-priority file begins processing.

*<u>Note</u>: The reference to data from matched secondaries to tertiaries in the last paragraph means that both types matched a primary-file card--secondaries and tertiaries cannot be directly matched to each other.

Although Matching Fields are frequently used concurrently as control fields (indicators L1-L9), the MR and Lx indicators are independent--i.e., file matching and group control have no inherent connection. In considering the status of L-indicators (if control levels are specified), the files to be matched should be thought of as though they resulted in one continuous merged file--even if they are not being merged. (However, Matching Fields and Control Level are related to the extent that control-field comparisons are only performed when cards from pertinent files are processed; this, in turn, is based on the Matching Fields operation.)

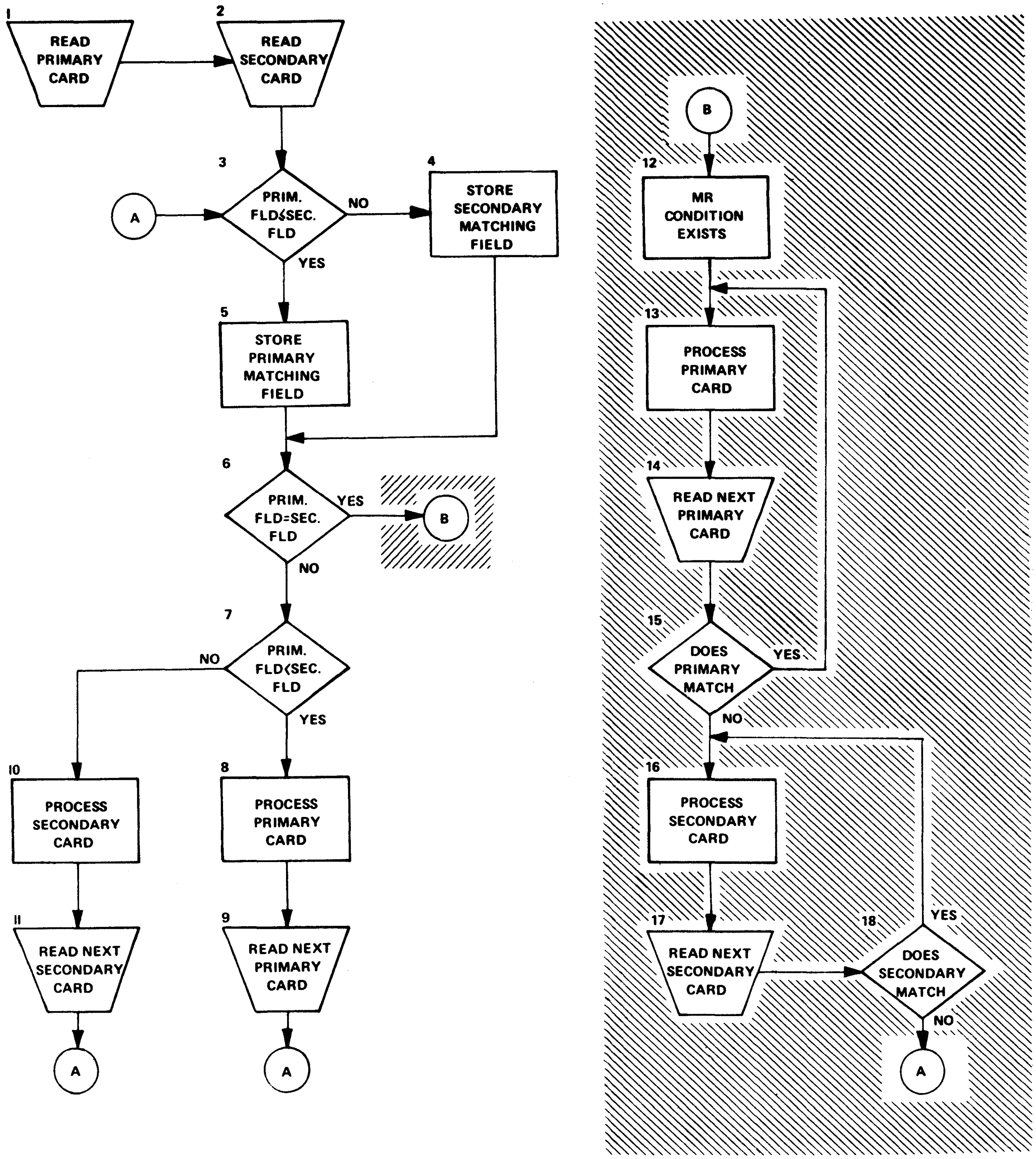
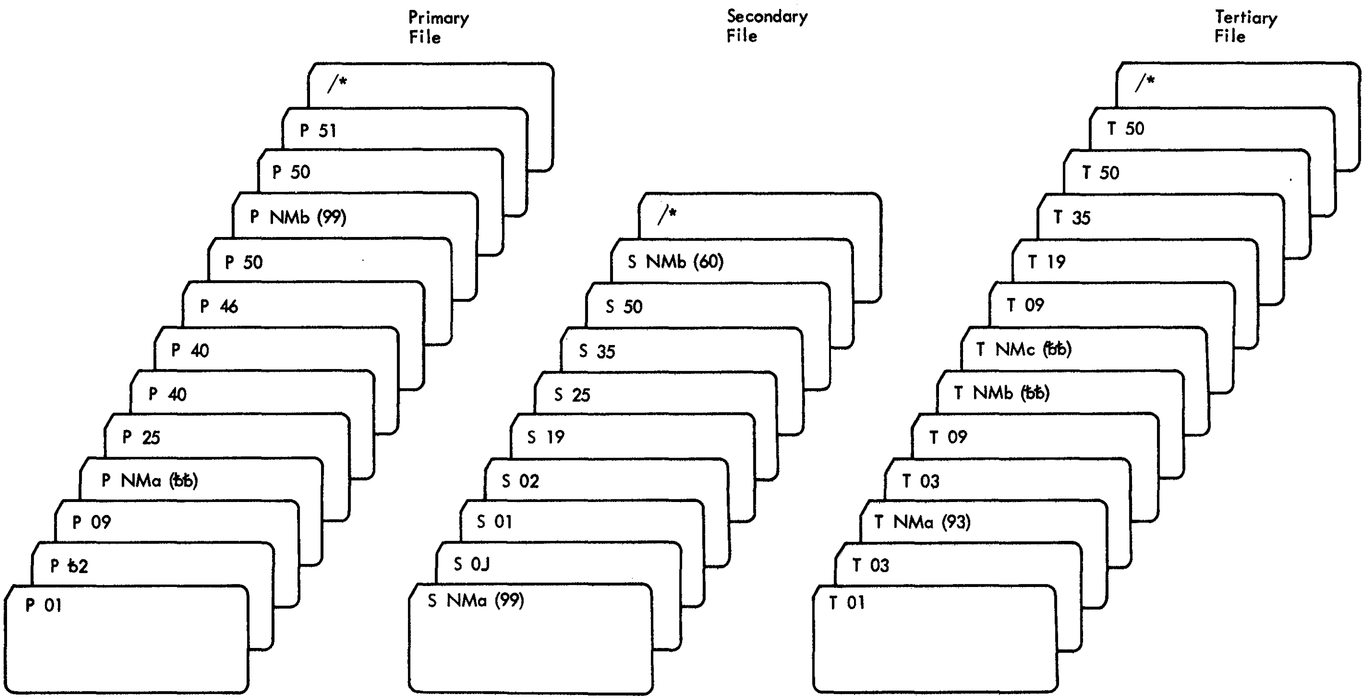


Figure 12.1. Processing Sequence of Cards in two Files

Processing Sequence for Three Files being Matched
(see CONDITIONS and LEGEND below)



CONDITIONS

GIVEN: Two single-column Matching Fields are used, and designated as numeric (0 in "Decimal Positions" in Input Specifications); ascending sequence is specified. Control levels are also designated, for all card types (including those not being matched): L2 for high-order (left) field, L1 for low-order field. L1 is specified for all files; L2 only for Primary and Tertiary. Assume col. 17 of File Description Specifications is blank for all three files, so that all files must be completed before LR turns on.

LEGEND: P = Primary-File card; S = Secondary-File card; T = Tertiary-File card. Arabic numerals = Contents of fields specified as Matching and Control-Level fields; ƀ = blank. NM = No Matching Field specified for this card type; lower-case letter = Specific card of such type. MR = MR indicator ON for processing of this card (Detail Time through ensuing Total Time); NMR = MR indicator OFF for processing of this card. L2, L1 = L2 and/or L1 (as shown) indicator on for this card (beginning with Total Time and through its Detail Time).

Figure 13A. Matching of Files - Input Files before Matching

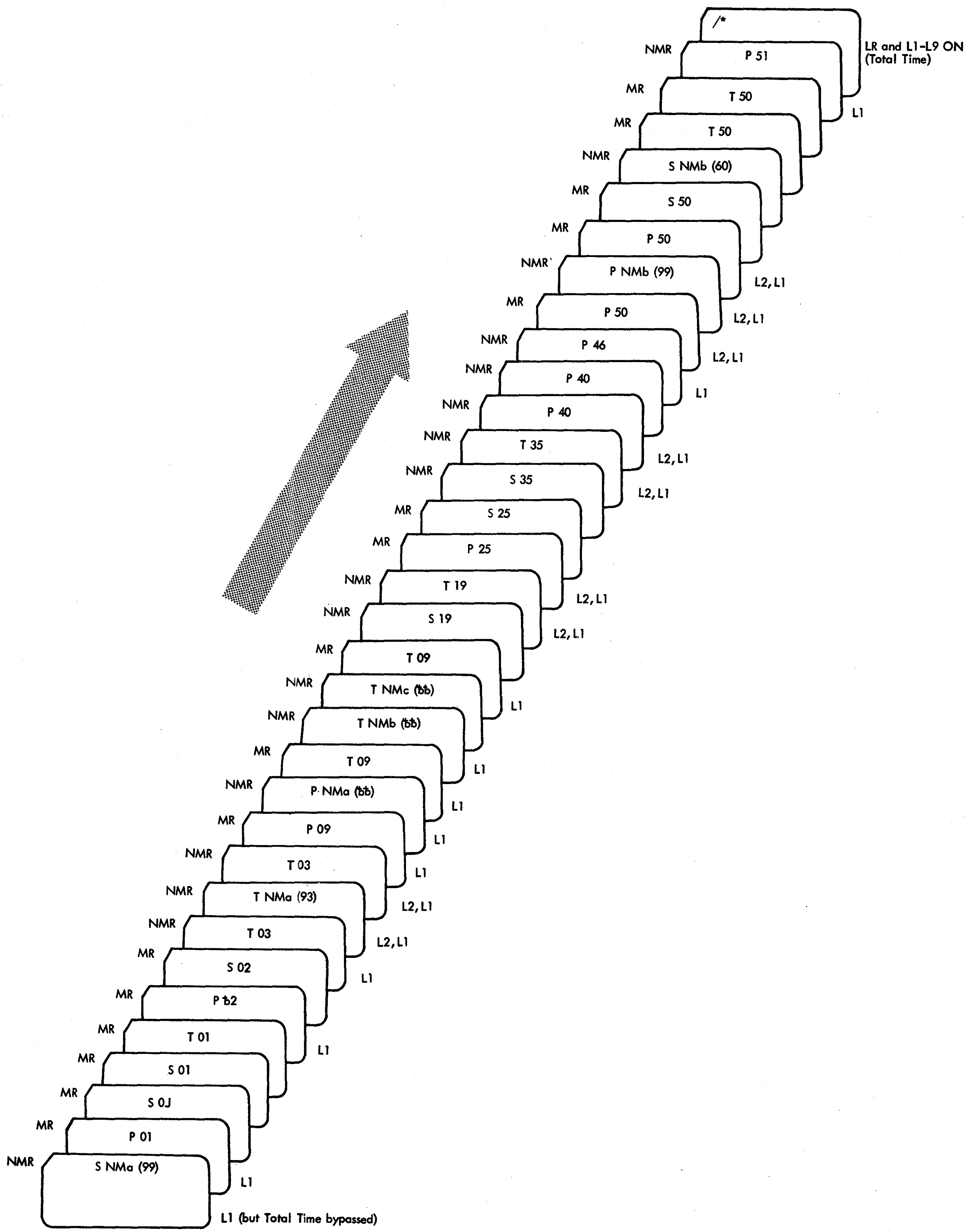


Figure 13B. Matching of Files - The three Files after Merging

Figures 13A and B illustrate the processing sequence for multiple files, the status of the MR indicator in the various situations, and the action of Lx indicators (L1 and L2) if assigned to the same fields used as Matching Fields. The example was deliberately constructed to show the effects of various unusual conditions in combination, and is therefore somewhat artificial. It should, however, make it possible for the user to predict how sequencing, the MR indicator, and Lx indicators will act under any combination of conditions he may set up.

For clarity in showing the sequence in which the cards are processed, the three original files are subsequently pictured merged into one file, although they need not be merged. Explanatory comments for Figure 13 follow. If the reader is only interested in the processing sequence and MR indicator--not the Control Levels--he can ignore the Control-Level entries: they have no effect on the file processing sequence or the status of the MR indicator.

Items to be Noted in Figure 13

1. Card types for which Matching Fields are not specified

   a. A card type for which Matching Fields are not specified is processed immediately after the preceding card from the same file. See position of cards T NMa, P NMa, T NMb, T NMc, P NMb and S NMb.

   b. If cards of such type are at the front of any file, they are processed first, even if they are neither in the primary file nor contain the lowest (if ascending sequence) value in the fields on which other cards are matched. See position of card S NMa. (If all files began with such cards, these cards would be processed in the file-priority sequence: primaries, secondaries, tertiaries.)

   c. None of these cards causes a sequence error. The card itself is omitted from the sequence check, and thus is never signalled as out of sequence. The core-storage sequence-data area retains its contents from the last preceding card with Matching Fields specified, and the next card to be matched is compared with this data--thus, the intervening not-to-be-matched card does not affect the sequence comparison of the ensuing card.

   d. The MR indicator is OFF for such cards; but it turns ON again for the next card if--without the intervening not-to-be-matched cards--it would be ON. See MR indicator for card following P NMa, T NMc, P NMb, and S NMb.

2. Zones and Blanks

   Since Matching Fields were designated numeric: ƀ = 0, and 11- and 12-overpunches are omitted by the program from match and sequence comparisons. See cards S 0J and Pƀ2.

3. No matching is performed between secondaries and tertiaries. Note that the MR indicator is off for cards S19/T19 and S35/T35.

4. During the processing of a matched primary-file card, there is no indication whether it matches only a secondary- or only a tertiary-file card, or both.

5. Control Levels

   a. Whether, and in what manner, control levels are specified has no bearing on the sequence in which cards of multiple files are processed.

   b. When a secondary-file card is processed, the high-order control-level field (L2) is not compared nor are its contents altered in core storage, from the last-preceding primary- or tertiary-file card--because, solely to illustrate the effect, L2 was not specified for the secondary file.

   c. Since control levels were specified for all cards in a file, the NM cards affect control breaks although they are not being file-matched.

   d. The Control-Level fields were designated numeric. Therefore ƀ = 0, and 11- and 12-overpunches are omitted by the program from group controlling.

   e. Control level operates as though the three files were one file, in proper sequence according to the RPG file-matching operation.

The L1 and L2 indicator status, as shown in Figure 13B, will now be discussed for each relevant card in the example, in the order the cards appear in the merged file. The reader should bear in mind that control levels L2 (high-order field) and L1 (low-order field) were specified (for the same fields used for matching) for all cards of files P and T, but only control level L1 was specified for file S.

Card Identification	
S NMa	9 in low-order field, compared with 0 stored in control field areas at start of program execution, sets L1 on. Note that total time is bypassed on first card with control fields specified (see sections on <u>Program Logic Flow</u> and <u>Indicators</u>). L2 is off because it is not specified for file S.
P 01	Low-order 1 is compared to preceding 9 and sets L1 on. L2 remains off because high-order 0 is equal to 0 initially in storage, and not changed by card S NMa (no L2 control on file S).
S 0J	Zones are removed from numeric fields for control-level operations. Therefore, 0J = 01.
Pƀ2	L1 set on for change from 1 to 2. L2 off because ƀ = 0.
T 03 (first)	L1 on for change from 2 to 3.
T NMa	L2 on for change from 0 to 9. L1 on only because L2 is on.
T 03 (second)	L2 on for change from 9 to 0. L1 on only because L2 is on.
P 09	L1 on for change from 3 to 9.
P NMa	L1 on for change from 9 to ƀ (0). L2 off because ƀ = 0.
T 09 (first)	L1 on for change from ƀ (0) to 9. L2 off because ƀ = 0.
T NMb	L1 on for change from 9 to ƀ (0). L2 off because ƀ = 0.
T 09 (second)	L1 on for change from ƀ (0) to 9.
S 19	L2 off because no control level on high-order field of file S. L1 off because low-order field contents unchanged.
T 19	L2 on because 1 is compared to 0 still in control-level 2 storage from card T 03 (second). L1 on only because L2 is on.
P 25	Normal control break. L2 on for change from 1 to 2. L1 on for change from 9 to 5, and because L2 is on.
S 35	L2 off because no control on high-order field of file S. L1 off because low-order field contents unchanged.
T 35	L2 on because 3 is compared to 2 still in control-level 2 storage from card P 25.
P 40	Normal control break, levels 1 and 2.
P 46	Normal control break, level 1.
P 50 (first)	Normal control break, levels 1 and 2.
P NMb	Normal control break, levels 1 and 2: 99 versus 50.
P 50 (second)	Normal control break, levels 1 and 2: 50 versus 99.
S NMb	L1 off because low-order field unchanged. L2 off because no control level on high-order field of file S.
T 50 (first)	L1 off because low-order field unchanged. L2 off because 5 is compared to 5 still in control level 2 storage from card P 50 (second).
P 51	Normal control break, level 1.
/*	LR on because end of data files. L1-L9 on because LR is on. Job ends (system halts) after total time of /* (in cols. 1-2) card. No detail-time processing takes place for this card.

<u>Sequence-Checking of Single Files</u>

When the application involves only a single input (or combined) file, specification of Matching Field(s)--M1, M2, M3--provides sequence-checking on the(se) field(s), without file matching.

The explanations for sequence-checking given above, under Matching of Card Files, apply--i.e., in regard to ascending, descending and special translation-table sequence; stopping on sequence errors; maximum aggregate size of sequence fields; and ignoring sequence of intervening card types for which no Matching Fields are specified.

Note: Programming a sequence check by entries in the calculation specifications may yield faster throughput, and usually consumes less core storage space, than utilizing the Matching Fields entry for that purpose alone; it also permits detection of duplicates, which is not possible with the Matching Fields operation. (Figure 68 - Part I illustrates sequence-checking and guarding against duplicates by calculation specifications.)

This chapter and the five chapters that follow discuss the specifications for every field in each of the five specifications forms. Where illustrations were thought desirable for clarification, they are given. (Solutions to some special applications problems, however, are presented in Appendix E, Programming Tips, rather than here.) Attention is drawn to limitations and potential trouble areas, where deemed of general interest and significance.

Functions treated extensively in earlier chapters will not be repeated in detail here. The user is assumed to have read the preceding material, and is asked to refer back to it for the fine points. The contents of the chapter Programming for RPG--General Information, particularly, will be an indispensable reference.

In a few instances, the Model 20 card RPG provides a latitude in specifications that does not conform to the requirements of other IBM System/360 RPGs. In these cases, a brief note will follow, indicating that differences exist between this and other System/360 RPGs. To obviate repetitive detailed explanations in each such note, the term "compatibility" will be employed as reason for the recommended approach.

FIELDS COMMON TO ALL SPECIFICATIONS FORMS AND CARDS

Columns 1-2: Page identification
Columns 3-5: Line identification

While Page and Line identifications will usually be numeric, any EBCDIC characters are valid. (Zero does not equal blank.)

The first two digits of line number are preprinted; the third is left blank to make it easy to assign line numbers for insertions, by writing specifications lines following line 15 and numbering for appropriate insertion. The cards within a specifications type must be in appropriate sequence when they are read by RPG. Proper sequential numbering facilitates sorting of specification cards, and checking.

Page and line identifications are read as one combined continuous value and checked for ascending sequence according to EBCDIC (See Appendix D, Figure D1). They may start at any value and gaps in the sequence are permitted. A step-down (descent in sequence) or repetition is identified by a printed symbol (S) during program generation, but generation is not interrupted. (The order in which the programmer writes and numbers the different kinds of specifications sheets will often differ from the order in which the specifications card types must be fed into the system. The symbol for sequence step-down will then be printed, and can be ignored if due to this reason.) No sequence symbol is printed for a step-down at the first card of calculation specifications.

Column 6: Form Type

This is a predetermined and constant letter for each type of specifications form and card. If cards of one specification type are followed by those of another type that may legitimately follow, and cards of the first type then recur, the latter group is ignored and an error message is printed; but generation continues.

Columns 75-80: User's Identification

May contain any EBCDIC characters. This field is not checked by the program, but the contents are printed during generation.

Comments Card: *(card punch-combination 11-4-8) in column 7.

The * in col. 7 designates that this is not a specification card. The program checks only columns 1-6 (see above). The card does not effect any program generation; but the contents of columns 1-80 are printed during generation.

This allows the programmer to insert notations at any points in the specifications.

Blank Specifications Lines or Cards

Blank <u>lines</u> may be left between specifications lines, for clarity; but intervening blank specifications <u>cards</u> (without * in col. 7) cause printing of a diagnostic message during generation, although they do not prevent proper generation of the object program.

Leading Zeros in Specifications Fields

The recording of leading zeros is optional only in specification fields for card-column numbers, forms skip (in this RPG), End Position in Output Record (Output-Format Specifications), field legths (Calculation Specifications), and for numbers and lengths of table entries (File Extension Specifications).

GENERAL INFORMATION

Each file used in the object program must be defined in one line of the File Description Specifications form (See Figure 14). The form serves the following purposes:

1. To assign a name to each file by which it is referenced in the Input and/or Output Specifications;

2. To associate each file name with a specific input and/or output device;

3. To indicate whether the file is to provide data input, serve for output, or both;

4. To specify--when cards within an input (or combined) file, or files, will be sequence-checked--whether sequence is ascending or descending; and

5. If more than one file provides data input, to indicate which file(s) must be completely processed before the job is terminated.

Maximum Number of Files Available

This is dependent on the input, output, and input/output devices attached to the system:

- One input file is available for each input or input/output device.

- One output file is available for each output or input/output device. (An IBM 2203 Printer with Dual-Feed Carriage special feature offers two output files.)

- One combined file is available for each device that can serve for both input and output.

File types (input, output, or combined) are mutually exclusive (i.e., one file can only be an input file, or an output file, or a combined file), and one input/output device can be assigned only to a single file.

Each of the two hoppers of the IBM 2560 MFCM can be independently assigned to an input file, or an output file, or a combined file.

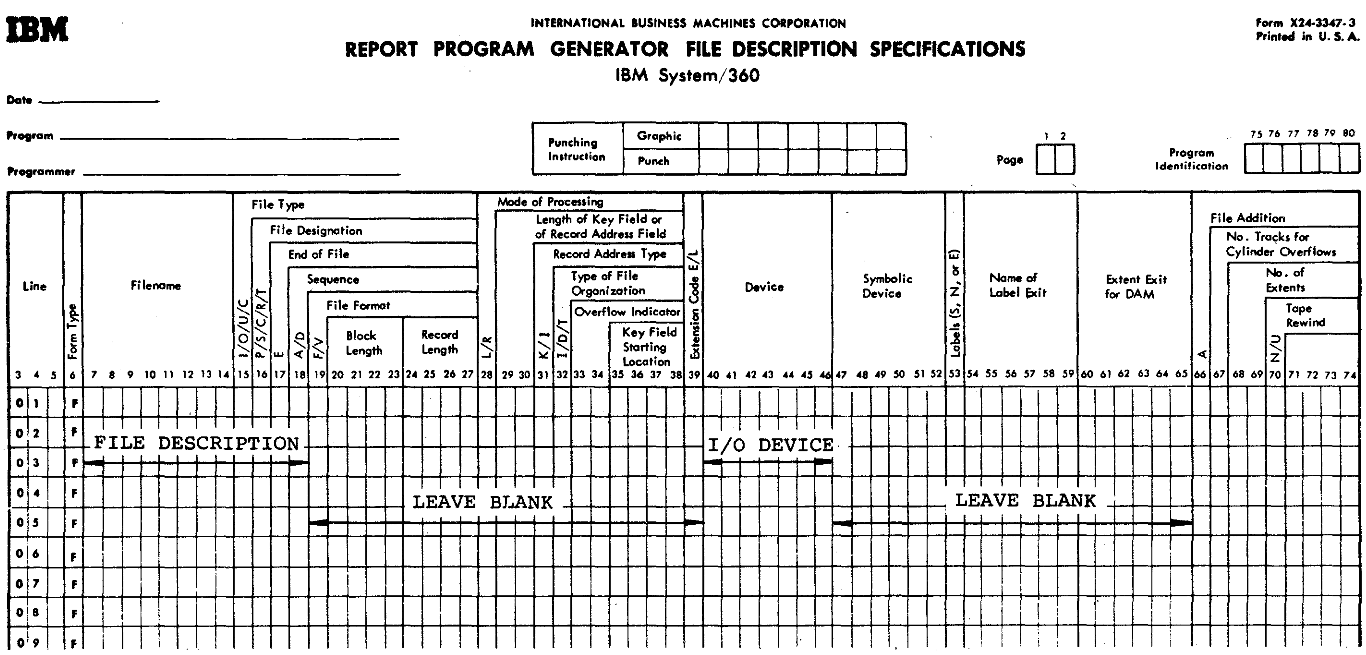


Figure 14. The File Description Form

Figure 3 shows which input, output, and input/output can be combined on the system.

FILE DESCRIPTION

File Name--Columns 7-14

Each file used in the program is assigned a name by the programmer. This name is recorded here on a separate line for each file. It must begin in column 7 with one of the 29 alphabetic characters, and may continue with alphabetic or numeric characters. It may be one to eight characters long. (See Definition of Terms, for "alphabetic" and "numeric" characters--neither permits embedded blanks.) The same name must not be assigned to several files.

Note 1: Throughout the publication, it is stated that File Names and Field Names must not contain embedded blanks, and only alphabetic or numeric characters are allowed. Actually, the program checks only that the first character is one of the 29 defined as alphabetic. Subsequent characters may be any of the 256 EBCDIC characters. For compatibility with other RPGs, however, the stated restrictions should be adhered to.

Note 2: Files may be entered in the File Description Specifications in any convenient sequence. For compatibility with other RPGs, the order of input and combined files should correspond to that in the Input Specifications. See also File Designation (col. 16), below.

File Type--Column 15

I = Input file.

The cards are read to provide input data.

There is no output to any of the cards.

The file name appears in the input specifications, but not in the output-format specifications.

O = Output file.

No information is read from the cards; they serve only to receive output. Or, this file name represents the printer.

The file name appears in the output-format specifications, but not in the input specifications.

C = Combined file.

Some or all of the cards in the file provide input information and some or all of the cards in the file receive output. The file name appears both in the input and the output specifications.

If input cards are to be stacker-selected in the output specifications (e.g., based on calculation results or on status of certain indicators, such as MR), they must belong to a combined file.

Note: If the user wishes to make certain that output cards are blank in certain or all columns before they are punched, he must designate them as belonging to a combined file. The fields that should be blank can then be read via input specifications, and indicators set for blank or not blank.

File Type U does not apply to Model 20 card RPG.

No check is performed by the program to assure that, if C is designated, the File Name appears in both the input and output specifications.

File Designation--Column 16

Leave blank for output files. No entry required for input (or combined) files.

Model 20 card RPG ignores entries in column 16 of the File Description Specifications form, and the order in which the files are listed on this form. The order of priority of input (or combined) files is established in the Input Specifications.

However, for compatibility with other RPGs, the order in which input (or combined) files are recorded in the File Description Specifications should conform to that in the Input Specifications, and column 16 should contain a specification:

P (Primary) =The only input (or combined) file, or the input (or combined) file recorded first on the Input Specifications form.

S (Secondary)=The second or third input (or combined) file on the Input Specifications form.

The C, R, and T entries do not apply to Model 20 card RPG.

Primary-file cards are processed ahead of matching secondaries; when secondary- and tertiary-file cards match primaries, the order of processing is: primary cards, secondary cards, tertiary cards (second file with S in col. 16).

It is permissible for output-file entries to intervene between the entry lines for the several input (or combined) files.

End of File--Column 17

If there are multiple input (or combined) files, this column determines which of these files must be exhausted before the LR indicator turns on and the job terminates.

E entered in column 17 } for all input (or or b̸ in column 17 } combined) files:

All input (or combined) files are exhausted before LR is turned on and job is terminated.

E entered in column 17 for some--but not all--input (or combined) file(s): The LR indicator turns on, and the job terminates, after processing of the last data card of all files for which E was entered--even if one or two other files (blank in column 17) are not yet exhausted.

For a single input (or combined) file, the LR indicator turns on, and the job terminates, after the last data card has been processed--regardless of whether column 17 is blank or contains E.

Leave column 17 blank for output files.

Sequence--Column 18

Leave blank unless sequence checking is called for in the Input Specifications (by entries in Matching Fields, columns 61-62).

If multiple input (or combined) files exist, or a single input (or combined) file is to be sequence-checked, the direction of the file sequence must be specified here. The sequence check operates according to the EBCDIC sequence, unless the user has modified the sequence by a translation table (see Appendix D). With multiple input (or combined) files, all must be in the same sequence, and the specification in col. 18 must be entered for all of them.

A = Ascending sequence
D = Descending sequence

Leave column 18 blank for output files.

Columns 19-39

Leave blank. These columns do not apply to Model 20 card RPG. (While comments may be recorded in these columns with this program, this would interfere with other RPG programs.)

I/O DEVICE ASSIGNMENT

Device--Columns 40-46

A mnemonic code is written in this field to assign a specific input, output, or input/output device to the file whose name was recorded in columns 7-14. Whenever a particular file name is then referred to in subsequent specifications, the system acts upon a card (or paper form) in the I/O device identified here with that file.

The Device code is written left-aligned, starting in column 40. A code for each device has been pre-determined by IBM, and must be written exactly as shown below.

Specification Entry	Input/Output Device
CRP20	IBM 2520 Model A1, Card Read-Punch
MFCM1	IBM 2560 MFCM, Hopper 1
MFCM2	IBM 2560 MFCM, Hopper 2
PRINTER	IBM 1403 Printer, or IBM 2203 Printer (Standard or Lower Feed)
PRINTLF	Same as PRINTER (see above)
PRINTUF	IBM 2203 Printer, Upper Feed of Dual-Feed Carriage
PUNCH20	IBM 2520 Model A2 or A3, Card Punch
PUNCH42	IBM 1442 Card Punch
READ01	IBM 2501 Card Reader

Note: If the Dual-Feed Carriage special feature is installed, and both carriages are used in the program, each carriage is assigned a separate file name and Device code (PRINTER or PRINTLF, and PRINTUF). Two printer output files then exist. The lower-feed carriage is the standard (or sole) carriage. (See IBM System/360 Model 20, 2203 Printer, Form A26-5926.)

For compatibility with other RPGs, PRINTER (rather than PRINTLF) should be used for the standard, or lower-feed carriage.

READ01 will function also as MFCM1, provided there is only a single input file, no MFCM output is involved, and no 2501 is attached.

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS

IBM System/360

Form X24-3347-3 Printed in U.S.A.

Date 7-15-66

Program A/R

Programmer K.B.

Punching Instruction: Graphic / Punch

Page 01 | Program Identification (75-80) ACTREC

Line (3-5)	Form Type (6)	Filename (7-14)	File Type I/O/U/C (15)	File Designation P/S/C/R/T (16)	End of File E (17)	Sequence A/D (18)	File Format F/V (19)	Block Length (20-23)	Record Length (24-27)	Mode of Processing L/R (28)	Length of Key Field or of Record Address Field (29-30)	Record Address Type K/I (31)	Type of File Organization I/D/T (32)	Overflow Indicator (33-34)	Key Field Starting Location (35-38)	Extension Code E/L (39)	Device (40-46)	Symbolic Device (47-52)	Labels (S, N, or E) (53)	Name of Label Exit (54-59)	Extent Exit for DAM (60-65)	File Addition A (66)	No. Tracks for Cylinder Overflows (67-68)	No. of Extents (69)	Tape Rewind N/U (70)	(71-74)
01	F	OLDBALC1	I	P		A											READ01									
02	F	$TRNSACT	C	S	E	A											MFCM1					ST2.PROBL				
03	F	NEWBAL1	O														MFCM2									
04	F	L5ARTRNS	O														PRINTER									
05	F																									

Figure 15. Example of File Description Specifications

Columns 47-65

Leave blank. Not applicable to Model 20 card RPG. (While comments may be recorded in these columns with this program, this would interfere with other RPG programs.)

Comments--Columns 66-74 (for card RPG only)

Available for any information the programmer wishes to have printed out during generation of the program. Except for this printout, entries in this field are ignored by the generator program, and do not take up any core storage in the object program.

EXAMPLE OF FILE DESCRIPTION SPECIFICATIONS --FIGURE 15

Job Requirements (Those Relevant to File Description Specifications)

Match a large file of monthly accounts receivable balance cards against transaction cards. When there are transactions on an account, punch a new summary card. Also list balance and transaction data. The data files are in ascending sequence. Terminate the run after the last transaction card, so that any remaining portion of the old-balance file is not unnecessarily processed. Select unmatched transaction cards.

Explanation of Specifications Entries

File Name

Arbitrary, but descriptive, file names were chosen to illustrate

a. That the first letter must be alphabetic

b. That there are three symbols that are defined as alphabetic--note $ in first position of second file name (see Definition of Terms.)

c. That numeric characters may be used except in first position of file name.

File Type and I/O Device

The old-balance file (OLDBALC1) serves as input only (I in col. 15), and is to be fed through the IBM 2501 Card Reader (Device code READ01 in cols. 40-45).

The transaction file ($TRNSACT) serves as data input only; but the $TRNSACT cards that do not match OLDBALC1 cards are to be selected. Stacker selection predicated on the status of the MR indicator must be specified in the Output-Format Specifications, which makes it an output operation. The $TRNSACT file thus appears in both input and output specifications, and must therefore be defined as a combined file (C in col. 15). The file is to be fed from hopper 1 of the MFCM (Device code MFCM1).

The new-balance file (NEWBAL1) serves only for output, and does not appear in the input specifications. Therefore, O is entered in col. 15. It is immaterial whether the cards are blank when placed in the hopper (as they would normally be in this application) or prepunched: they will not be read. The file is to be fed from hopper 2 of the MFCM (Device code MFCM2).

An accounts receivable transaction list (file L5ARTRNS) is to be printed on an IBM 1403 Printer, or on an IBM 2203 Printer under control of the lower feed. The Device code for either is PRINTER (or PRINTLF). The printer can only be output (O in col. 15).

File Designation, and Order of File Entries

The entries in column 16 designate that OLDBALC1 cards (P in col. 16) are processed ahead of matching $TRNSACT cards (S in col. 16). For the same reason, the OLDBALC1 file is entered ahead of the $TRNSACT file--and this corresponds to their order on the Input Specifications form.

Both the code in col. 16 and the order in which the files are listed on the File Description form are ignored in Model 20 card RPG. They are required only for compatibility with other RPGs.

End of File

The E in col. 17 of the $TRNSACT file specifies that the job is to be terminated after the last card of this file has been processed--regardless of whether the OLDBALC1 file was exhausted earlier, at the same time, or still has unprocessed cards left.

If column 17 were blank for both input (or combined) files, or contained E for both, the job would not be terminated until both files are exhausted.

Col. 17 is not used with output files, and is therefore blank for the NEWBAL1 file.

Sequence

The A in column 18 specifies that the input (or combined) files are in ascending sequence. Multiple input files must be in sequence, and all must be in the same sequence.

Comments

The entry in cols. 66-74 of the $TRNSACT file line will be listed at program-generation time, on the same print line as the File Description Specifications for this file. In card RPG, its only function is a comment to the programmer or operator (e.g., "remember to check stacker 2 during program execution: it will contain problem cards").

INPUT SPECIFICATIONS (MANDATORY)

GENERAL INFORMATION

The input specifications (see Figure 16) serve to

1. Establish the processing priority for matched cards from multiple input (or combined) files
2. Identify card types within an input or combined file
3. Specify card-type sequence within the file
4. Direct input- or combined-file cards to stackers on the basis of card type
5. Define input fields, and their data formats
6. Set indicators based on the status of individual input fields
7. Identify control fields
8. Designate fields to be matched between cards of multiple input (or combined) files
9. Specify card field(s) for sequence checking

Each input or combined file must be recorded on the Input Specifications form.

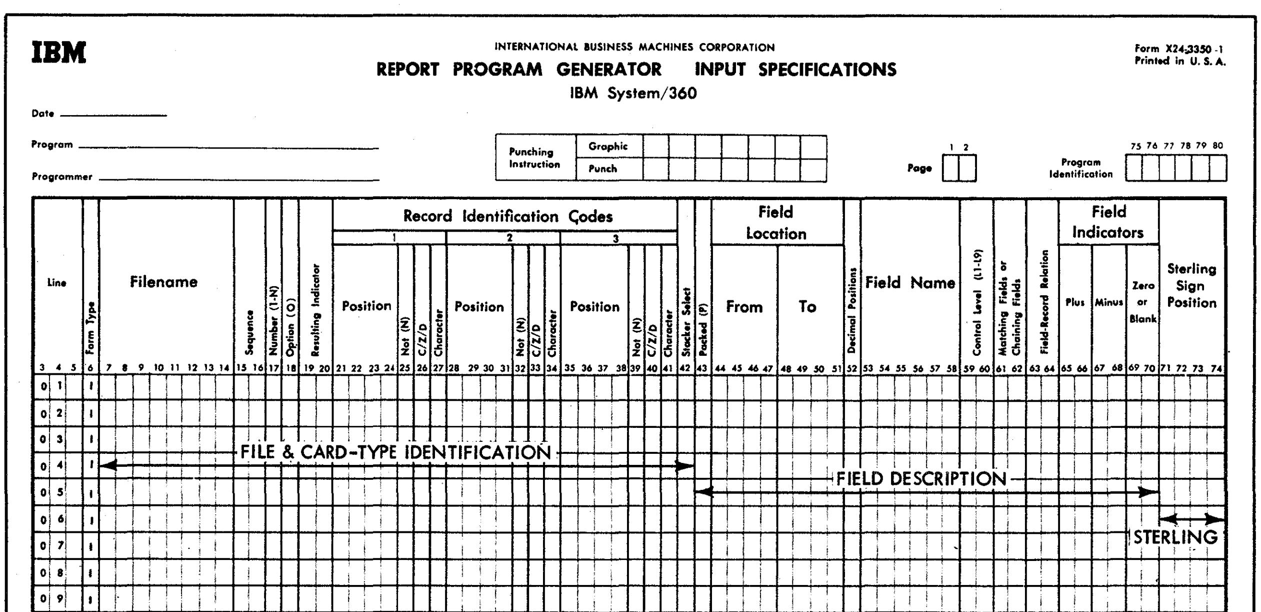
A file consists of one or more card types in one hopper.

Each card type that can exist in an input file must have at least a Sequence entry (cols. 15-16) in the Input Specifications--otherwise an error stop or perpetual program loop occurs when the card type appears during program execution.

In a combined file, any card type from which data is to be read, or which is to be processed on the basis of card-type identification, must also be entered in the input specifications. Normally, this means that all card types of a combined file must be identified on the Input Specifications form, just as for an input file; an exception is a combined-file card type which is never read or identified, but is punched and/or card-printed by multiple-time output instructions during one program cycle of another card (see Program Logic Flow, Multiple-Time Output to Cards during One Program Cycle).

At least one input or combined file is required for an RPG program. Input fields are defined only if they are to be read; it is possible to perform an RPG job without any input fields (e.g., stacker selection, or error halt, based on card type).

Output files must not contain an entry on Input Specifications forms.


IBM
INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS
IBM System/360
Form X24-3350-1
Printed in U. S. A.
Date
Program
Programmer
Punching Instruction: Graphic, Punch
Page
Program Identification

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

Figure 16. The Input Specifications Form

The entries for the Input Specifications form are divided into three categories, as shown in Figure 16.

1. File and Card-Type Identification--Columns 7-42

   This segment designates the file, identifies the card types it contains, determines the order in which card types within the file should occur, and permits stacker assignments by card type.

   Each card-type identification uses a separate specification line, or group of lines, which must not contain any field description.

   The order in which multiple input (or combined) files are entered in the input specifications determines the relative processing priority of matched files: the first file entered thereby becomes the primary; the next one becomes the secondary; and a third one becomes the tertiary (or second secondary) file. (Also see <u>Matching of Files.</u>)

2. Field descriptions--Columns 43-70

   In this segment, the input fields are defined, and their formats are described. Indicators may be set on the basis of positive, negative, or zero/blank contents of input fields. Control fields, and matching fields for multiple files, are assigned here. Sequence-check fields may be specified.

   Each field description uses a separate specification line. All field descriptions for one card type (or grouping of card types, if OR-relationships apply--see below) follow the specification of that card type (or card-type group), beginning on the line below the card-type specification.

3. Sterling Sign Position--Columns 71-74

   Applies to Sterling currency fields (British monetary system) only. Not covered in this manual. See <u>IBM System/360 Model 20, Sterling Currency Processing Routines,</u> Form C26-3605.

<u>FILE AND CARD-TYPE IDENTIFICATION</u>

<u>File Name--Cols. 7-14</u>

Each file is given a separate name by the programmer--the same name used for that file in the File Description Specifications, where the file name is associated with a particular I/O device.

The name of each input or combined file is entered on a separate line in this field. It must begin in column 7 with one of the 29 alphabetic characters, may continue with alphabetic or numeric characters, and may be one to eight characters long. (See <u>Definition of Terms</u>, for "alphabetic" and "numeric" characters--neither permits embedded blanks.) Field description must not appear in the same line.

The file name is recorded once per file, on the first line for that file. If desired, it may be repeated for additional card types within the same file; but this is unnecessary.

<u>Sequence, Number, Option--Cols. 15-16, 17, 18</u> (Card-Type Sequence Check)

When there are several card types within one file, the job may or may not require them to be in a particular sequence. The program can be directed to check that the sequence in which the types occur during object-program execution conforms to a specified sequence. An error results in a halt. The system may be restarted (see restart procedure in <u>Operating Procedures</u> manual).

This check has no connection with a sequence check on the values in fields of succesive cards in a file (see cols. 61-62) nor with control-level groups (see cols. 59-60). For instance, the correct sequential position of a card type among other card types--but in the wrong control group --is not detected by entries in these fields. These entries merely verify that a specified sequence of card types iterates within the file and--with limitations--that the quantity of each card type adheres to a criterion on each iteration. Therefore, not every kind of error in card-type sequence is detected; nor is the last group in a file checked for completeness, so long as no detectable card-type sequence error occurs up to the point of the last card in the file. However, Control-Level specifications (see cols. 59-60) provide for guarding against admixture of cards of the correct type but wrong control group; and simple Calculation Specifications entries can protect against most of the remaining card-type sequence errors not detected by entries in cols. 15-18 in the input specifications. (One such example appears in Figure 5D, Line 06.) Detailed explanations and illustrations of the card-type sequence-check operation follow later in this section.

The Sequence entries in cols. 15-18 in a specification line apply to the card type in that line, as identified by specifications in cols. 23-41.

Card types in an OR relationship (see below) have no card-type-sequence specifications. The specifications in the main line above the OR line(s) apply to the OR types, too. It is not possible to specify a different sequence position or quantity check to card types in an OR relationship.

Sequence--Cols. 15-16

Columns 15-16 must both have an entry in the first specification line of every card type (i.e., no Sequence entry is made in an AND or OR line--see below).

Note: AND and OR lines are identified by AND and ORb, respectively, in cols. 14-16--see below.

The entry in cols. 15-16 must consist of a two-digit number when the card-type position in relation to other card types in the same file is to be checked.

The entry in cols. 15-16 must consist of any desired combination of the 29 alphabetic characters (see Definition of Terms) if the relative position of that card type, among several card types in the file, is not to be checked. (If desired, the same alphabetic characters may be used for several such card types.) A card type with alphabetic entry in cols. 15-16 cannot be checked for number of such cards in a group, and its presence in a group is always considered optional.

For card types whose relative positions are to be checked, the card type that is first in sequence in a file is assigned sequence number 01 in cols. 15-16. The next types in sequence are each assigned any desired higher number, in ascending sequence, in the order in which the card types occur in the deck. Gaps in the card-type sequence numbers are permitted. The card-type specification lines for each file must be written in the same order in which the sequence of card types for that file is numbered.

When some but not all card types in a file are to be checked for relative position, the specifications for card types not to be checked for sequence (alphabetic in cols. 15-16) must precede those for all sequence-numbered card-types for that file --even though the cards themselves might be interspersed in the card deck among the sequence-numbered types, and may even occur between multiple cards of a single sequence-numbered type.

Col. 17 must contain an entry when cols. 15-16 contain a numeric Sequence number. Cols. 17-18 must be blank when cols. 15-16 contain an alphabetic Sequence code.

When there is only a single card type in a file (or all card types are in an OR relationship), cols. 15-16 may be alphabetic or numeric (if numeric, col. 17 must be coded 1 or N). Alphabetic entries are recommended in this case, because of the contingency described in Warnings, item 2(a), below.

Note: The rules for alphabetic and numeric Sequence entries stated above are compatible with other System/360 RPGs. The Model 20 card RPG Program actually distinguishes between a Sequence entry defined as numeric (i.e., relative card-type position to be checked) and one defined as alphabetic (i.e., card-type position not to be verified) on the basis of col. 15 alone; there is no restriction on the contents of col. 16. Specifically, the Sequence code is defined as

1. Alphabetic, if col. 15 contains any EBCDIC character other than blank (hexadecimal 40) and other than those in EBCDIC-table column F (upper half-byte hexadecimal F).

   Col. 16 may contain any of the 256 EBCDIC characters (including "blank").

2. Numeric, if col. 15 contains any character in EBCDIC-table column F (upper half-byte hexadecimal F).

   Col. 16 may contain any of the 256 EBCDIC characters (including "blank").

   The card-type sequence check is based on the EBCDIC sequence.

See Appendix D and Figure D1 for explanation of EBCDIC.

Number--Col. 17

When cols. 15-16 contain a numeric Sequence entry, col. 17 must also contain an entry to specify the number of cards of this type in each iteration of card types: 1 or N.

1 = If the card type is present, there must be exactly one card of this type.

N = If the card type is present, there must be at least one card of this type and there may be more than one.

Column 18 determines whether the card type must be present.

Column 17 must be left blank in AND and OR lines, and if cols. 15-16 contain an alphabetic code (no card-type sequence check). It is not possible here to verify the quantity of cards of a type whose sequential position in relation to other types is not consistent (i.e., cannot be checked).

Option--Col. 18

When col. 17 contains an entry (i.e., cols. 15-16 contain a Sequence number), col. 18 may be blank or contain the letter O.

b = This card type must be present in each iteration of card types.

O = Presence of this card type in each iteration of card types is optional.

Whether only one card or several cards may be present--if the type is present--is determined by the entry in col. 17; i.e., even if presence of a type is optional, a check is made to verify that--if the card type is present at all--only one card of the type is present if 1 is specified in col. 17. (As clarified further on, this verification is not effective if all card types in the file are optional.)

Column 18 must be blank if col. 17 is blank (no card-type sequence check, or AND or OR line). When cols. 15-16 contain alphabetic entries (i.e., no check on position of card type), the program assumes that the presence of such card type is optional.

WARNINGS:

1. No card-type sequence or quantity check is effectively performed at all if any of these conditions apply:

   a. All card types in a file are optional (O in col. 18) or have alphabetic Sequence code in cols. 15-16. See Figures 19A and D.

   b. One card type is non-optional (numeric entry in cols. 15-16, and blank in col. 18), but all others have alphabetic codes in cols. 15-16. See Figure 19B.

   c. One card type is required (numeric in cols. 15-16, blank in col. 18) and coded 1 or N in col. 17; only one other card type is specified with numeric sequence (cols. 15-16), and it is coded N and O in cols. 17 and 18, respectively. (In this case, however, the first card of the file is checked to verify that it is either of the first type specified or of the non-optional type.) See Figure 19C.

   While the program goes through the sequence-check steps when numeric specifications in cols. 15-16 exist, it cannot (in the above situations) distinguish between an incorrect card-type sequence and legitimate appearance of card types from successive groups, nor between erroneous duplication of a card type and two successive cards of that type from successive groups.

2. a. If the presence of all card types in a file is optional, and at least one of these types has a numeric Sequence specification in cols. 15-16, (the others either having alphabetic entries in cols. 15-16, or also numeric ones with O in col. 18), then--if a card of a type appears for which there is no entry in cols. 15-16--the program will neither advance nor error-stop: it remains in a perpetual loop, searching for the card-type sequence-check specifications of an unspecified type. See Figure 19D.

      For this and other control reasons, it is recommended that--for each input (or combined) file--a specification line always be included with Resulting Indicator and Sequence entry for "other" card types, possibly with halt (H1 or H2) specified in Resulting Indicator and, if desired, selection to a separate stacker. See next and later sections.

   b. However, if any card type with a numeric entry in cols. 15-16 is non-optional (no O in col. 18), or if all card types have alphabetic entries in cols. 15-16, then an unidentified card type automatically halts the system. See Figures 19A and C.

Figures 19A, B, C, and D illustrate these points.

Example of Card-Type Sequence Check

Figure 17A portrays an inventory file ready for updating. Figures 17B and C show alternative proper entries in cols. 15-18 for such a file. Some aspects of the example may appear artificial, but were selected to maximize clarification. Figure 18 illustrates the method the program follows to check card-type sequence, using the card arrangement in Figure 17A.

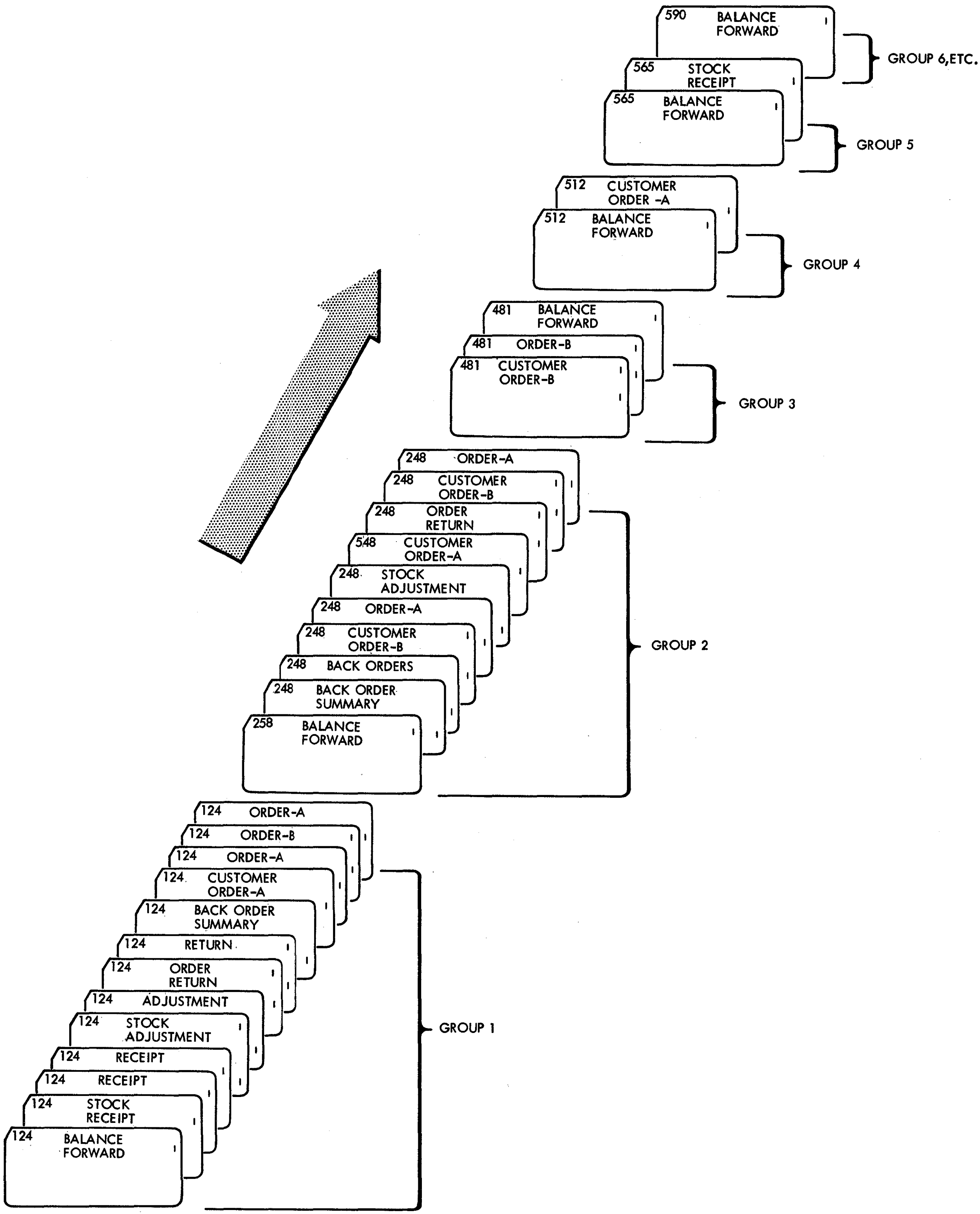


Figure 17A. Sequence Checking of Card Types within a File

Assumptions. For each stock number:

- There is one Balance Forward card, at the front.
- The Balance Forward card may be followed by one or more Stock Receipt cards.
- There may be any number of Customer Order Return cards next. In calculations, these are to be treated like Customer Order cards, type A.
- There may next be one Back Order Summary card.
- There must be one or more Customer Order cards last. There are two types, A and B, which are to be treated differently in pricing, but are treated as one group in sequence-checking. Either type may appear first, the two types may be intermixed, and one or both may be represented in a group.
- There may be one or more Stock Adjustment cards, positioned anywhere in the group.

Each card type is recorded in a separate specification line, and assigned an identifying Resulting Indicator number. (The next section explains how to accomplish the identification. It is included here merely so that the entries in cols. 15-18 do not appear to be the only ones in the line.)

The numbers that appear on the cards are stock numbers. They are irrelevant to the function of cols. 15-18. They have been included for the sake of reality, and to illustrate the limitations of the check performed by entries in cols. 15-18.

The first and fourth groups of cards in Figure 17A are correct. The second, third, and fifth groups contain some errors that would be detected as a result of the specifications in cols. 15-18, and some that would not.

Explanations--Figure 17B.

Note:
1. The entries in cols. 19-27 are explained in the next section, which references this figure again. It may be noted here merely that card-type Resulting Indicator numbers have no connection with card-type sequence numbers.
2. For convenience, little space has been left between specification lines in Figures 17B and C. It is, of course, assumed that--in actual use--the necessary number of lines are left to accommodate field descriptions (described later).

Line 01: The Stock Adjustment cards may appear anywhere in the group. Their position is therefore not to be checked, and (any) alphabetic characters must be assigned in cols. 15-16. (SA was selected as a mnemonic.) Cols. 17 and 18 must be blank when cols. 15-16 are alphabetic. All card types with alphabetic entries in cols. 15-16 must be specified ahead of card types, within the same file, to be checked for sequence position. The Stock Adjustment cards therefore are the first type specified, for the INVENTRY file, in the input specifications.

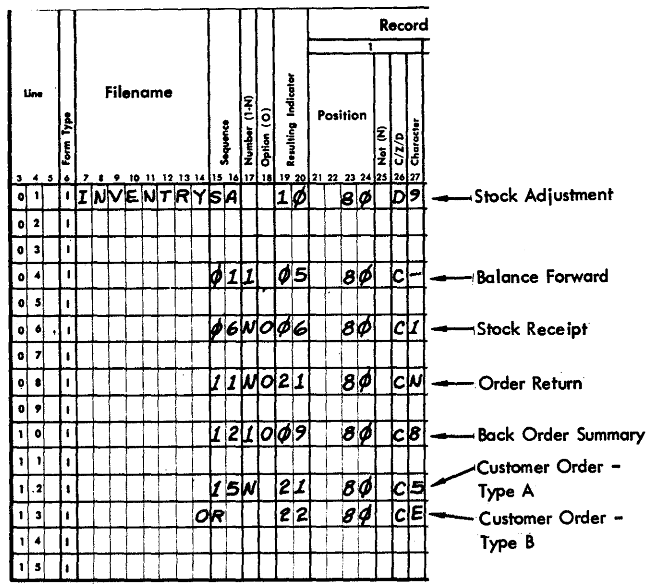


Figure 17B. Sequence Checking of Card Types within a File

The remaining card types in the file are to be checked for proper relative position among card types. They must therefore be recorded in the input specifications in the order in which they appear in the data file.

Line 04: Of the card types that can be checked for sequence, the Balance Forward card must be first. It must, therefore, be numbered 01 in cols. 15-16. It must also be the card type specified first of all types within the file to be checked for relative position. There is to be one Balance Forward card per group; therefore, a 1 is entered in col. 17. Col. 18 is blank, because the presence of this card is mandatory.

Line 06: The next card type whose relative position is to be checked is Stock Receipts. It is assigned any two-digit number higher than 01 (06 was arbitrarily selected). There may be any number of cards of this type in each group; therefore, N is specified in col. 17. Col. 18 contains the letter O, because presence of these cards in each group is not required.

Line 08: Any number higher than the preceding number (06) is assigned to the next card type in sequence. (Sequence number 11 is an arbitrary choice.) Again, there may be any number of Order Return cards, and their presence is optional. Cols. 17 and 18 are therefore coded N and O, respectively.

Line 10: The Back Order Summary card is assigned any higher sequence number than the Order Return cards. (The next number in sequence, 12, was chosen.) Its presence is optional (O in col. 18) but, if present, there must be no more than one (1 in col. 17).

Lines 12 and 13: Customer Order cards come next, and are therefore assigned any number higher than 12 (which was used for the preceding card type). Because there may be any number of Customer Order cards in a group, N is specified in col. 17. Because there must be at least one card of this type in each group, col. 18 (Option) must be blank.

These specification lines illustrate two further points:

1. No card-type sequence-check entry can be made for an OR line. (OR specification lines are discussed fully later.) For purposes of sequence-checking of card-type position, both card types--those defined in the basic specification line, and those in the OR line--are treated as one type:

   a. The presence in the proper position of either type satisfies any requirement for presence (if no O in col. 18);

   b. The presence of either type in the wrong position is regarded as an error;

   c. If 1 is specified in col. 17, and there is one card of each of the two types in a group, this is treated as an error as though there were two cards of one type.

   d. OR lines offer a method of checking the position of several card types in relation to others, when the several OR card types may occupy any relative position to each other.

2. Sometimes it is desired to treat two similar input card types uniformly in most calculations and/or for output; but they appear in different positions in the input file, and are to be checked for proper relative position.

   Order Return and Customer Order (Type A) cards fit this description.

   Note that these two card types were assigned different sequence numbers (11 and 15, respectively), but the same card-type Resulting Indicator (21). (The next section deals with Resulting Indicators in detail.)

Explanations--Figure 17A, and Sequence Check as Specified in Figure 17B

The first group of cards (Group 1: stock number 124) encompasses every type of card provided for in Figure 17B. It is correct in every respect, and no card-type sequence-check error stop will occur.

Group 4 contains the minimum number and types of cards allowed; all others are optional. No error stop occurs.

Groups 2, 3, and 5 contain various errors, introduced deliberately to illustrate the effectiveness and limitations of the card-type sequence check based on specifications in cols. 15-18:

Group 2 (stock number 248) is headed by a Balance Forward card with stock number 258. No error stop occurs. The criteria of cols. 15-18 are met: the Balance Forward card is present, it follows a Customer Order card, and there is exactly one Balance Forward card. Cols. 15-18 specifications do not cause checking of control-level fields.

There are two Back Order Summary cards in Group 2. An error stop occurs, because 1 is specified in col. 17.

A Stock Adjustment card is intermixed among Customer Order cards in Group 2. Our assumptions stated that Stock Adjustment cards may be located anywhere; they were coded alphabetic in cols. 15-16, and are not checked for position. Figure 17C presents a method for allowing a card type (e.g., the Stock Adjustment cards) to occupy any of several positions, and yet checking against its occupying any others. (Alternatively, specifying the Stock Adjustment cards to be in an OR relationship to Stock Receipt cards would check that they follow the Balance Forward card, or are among or behind Stock Receipt cards.)

One of the Customer Order cards in Group 2 (stock number 548) does not belong in this group. The card-type sequence check will not detect this.

In Group 2, an Order Return card is among the Customer Order cards, instead of being ahead of the first Back Order Summary card. This causes an error stop.

Group 3 commences without a Balance Forward card. This is not detected by the card-type sequence check. The Customer Order card at the front of Group 3 acts as a continuation of the Customer Order cards of Group 2.

When the Balance Forward card of Group 4 is read, an error stop occurs because two Balance Forward cards have been read consecutively, and 1 is specified in col. 17. Only for that reason is the erroneous position of the Balance Forward card in Group 3 detected. If the Balance Forward card in Group 4 were missing, neither its absence nor the erroneous position of the Balance Forward card in Group 3 would be detected.

Line (3-5)	Form Type (6)	Filename (7-14)	Sequence (15-16)	Number (1-N) (17)	Option (O) (18)	Resulting Indicator (19-20)	Record 1 Position (21-24)	Not (N) (25)	C/Z/D (26)	Character (27)	
01	I	INVENTRY	Ø1	1		Ø5	8Ø		C	-	Balance Forward
02	I										
03	I		Ø2	N	O	1Ø	8Ø		D	9	Stock Adjustment
04	I										
05	I		Ø6	N	O	Ø6	8Ø		C	1	Stock Receipt
06	I										
07	I		Ø8	N	O	1Ø	8Ø		D	9	Stock Adjustment
08	I										
09	I		11	N	O	21	8Ø		C	N	Order Return
10	I										
11	I		12	1	O	Ø9	8Ø		C	8	Back Order Summary
12	I										
13	I		15	N		21	8Ø		C	5	Customer Order - Type A
14	I	O	R			22	8Ø		C	E	Customer Order - Type B
15	I										
	I										
	I										
	I										
	I										
	I										

Figure 17C. Sequence Checking of Card Types within a File

Group 5 lacks any Customer Order card. An error stop occurs when the Balance Forward card of Group 6 is read, because no Customer Order card preceded it.

Explanations--Figure 17C

Figure 17C presents a method of permitting an optional card type to appear in several (i.e., two, in this example) acceptable positions, yet signalling an error if it were to appear in any other position. Otherwise it is identical with Figure 17B.

Change the assumptions for Stock Adjustment cards to read: they may directly follow the Balance Forward card and/or Stock Receipt cards only.

By entering the specifications for Stock Adjustment cards as shown in lines 03 and 07--instead of with alphabetic code in cols. 15-16, as in Figure 17B, line 01--presence of this card type is permitted in either or both of these positions, and limited to these two positions.

The position of the Stock Adjustment card in Group 2 of Figure 17A will now be signalled as erroneous.

Note that this technique requires assignment of different Sequence numbers (cols. 15-16) for the several permitted positions, but that the same card-type Resulting Indicator is assigned. The single Resulting Indicator number then always references that card type in calculation or output specifications, regardless of the position where the card appeared.

Nature of the Card-Type Sequence Check

A brief explanation of the method the program follows to verify card-type sequence will further clarify the preceding example. It is also necessary to proper specification of Record Identification Codes--the next section of the manual, which references this discussion.

a. If all card types in a file have alphabetic specifications in cols. 15-16, the program checks, as each card is read, for the first card type specified for the file just read, then the second, etc.--until a match is found, based on Record Identification Code (or absence of any identification specifications--see next section), or until all specifications for card type have been exhausted without encountering a match. (An error stop then occurs.)

b. If all card types in a file have numeric specifications in cols. 15-16, the program starts its check, as each next card is read, at the first card type that may legitimately have appeared next--it does not necessarily begin at the first specification line, except at the start of program execution.

If this does not match, it checks for the next card type specified, and so on, through the last card-type specification (for that file) in the input specifications, continuing in a circle up to the first one, etc.--until a match is found, or an error detected (illegal card-type sequence or quantity). An error stops the system--except in the particular undefined-card-type situation described above (Warnings, item 2(a)), when the search circle (loop) continues ad infinitum.

(Note: Card types in an OR relationship are checked consecutively after the main line, until a match is found or the OR lines are exhausted.)

c. If the specified card types are a combination of (a) and (b) above, the program first searches through the card-type specifications with alphabetic entries in cols. 15-16, as in (a) above. If no match is found, it continues--as in (b) above--at the first card type with numeric specification in cols. 15-16 that may legitimately have appeared next. If this does not match, it continues in a circle of the card-type specifications with numeric entries in cols. 15-16, until a match is found or an error detected. (But see Warnings, item 2(a), above.)

Note: When several or all card types have alphabetic Sequence codes in cols. 15-16, process time is minimized by recording the most-frequently occurring type with alphabetic Sequence code first, the second most-common type next, etc. The program then makes the least number of attempts to match a card-type definition to cards read.

Group	Card Position (entire file)	Verification Steps	Sequence-Check Result
1	1	Card type 10?No/05?Yes	OK
	2	10?No/06?Yes	OK
	3	10?No/06?Yes	OK
	4	10?No/06?Yes	OK
	5	10?Yes	OK
	6	10?Yes	OK
	7	10?No/06?No/21(11)?Yes	OK
	8	10?No/21(11)?Yes	OK
	9	10?No/21(11)?No/09?Yes	OK
	10	10?No/21(15) or 22?Yes	OK
	11	10?No/21(15) or 22?Yes	OK
	12	10?No/21(15) or 22?Yes	OK
	13	10?No/21(15) or 22?Yes	OK
2	14	10?No/21(15) or 22?No/05?Yes	OK
	15	10?No/06?No/21(11)?No/09?Yes	OK
	16	10?No/21(15) or 22?No	Error Stop
	17	10?No/21(15) or 22?Yes	OK
	18	10?No/21(15) or 22?Yes	OK
	19	10?Yes	OK
	20	10?No/21(15) or 22?Yes	OK
	21	10?No/21(15) or 22?No/05?No	Error Stop
	22	10?No/21(15) or 22?Yes	OK
	23	10?No/21(15) or 22?Yes	OK
3	24	10?No/21(15) or 22?Yes	OK - Note!
	25	10?No/21(15) or 22?Yes	OK
	26	10?No/21(15) or 22?No/05?Yes	OK - Note!
4	27	10?No/06?No/21(11)?No/09?No/21(15) or 22?No	Error Stop - Note!
	28	10?No/06?No/21(11)?No/09?No/21(15) or 22?Yes	OK
5	29	10?No/21(15) or 22?No/05?Yes	OK
	30	10?No/06?Yes	OK
6	31	10?No/06?No/21(11)?No/09?No/21(15) or 22?No	Error Stop

<u>Note</u>:
1. Card-type number used is Resulting Indicator number in Figure 17B. Because Indicator 21 is used twice, the Sequence number is shown in parentheses.

2. The illustration after each error proceeds as though the error card did not exist (dotted line)--merely so that illustration can be continued.

Figure 18. Example of Card-type Sequence-Check Action Based on Figures 17A and B

The action will now be illustrated in Figure 18, with reference to Figures 17A and B. (For convenience, card-type number used for reference is the Resulting Indicator number in cols. 19-20. Because indicator 21 is used twice, the sequence number is shown in parentheses.) In order to maximize the explanation, the illustrations in Figure 18 continue--after each error stop condition--as though the error card had not been present (dotted lines).

Figures 19A, B, and C highlight several potential trouble spots that can arise if the card-type sequence-check operation is not fully understood. The problems were mentioned under <u>Warnings</u>, above. The numbers in the upper left-hand corner of the cards are values in a potential control field which are ignored by the card-type sequence check.

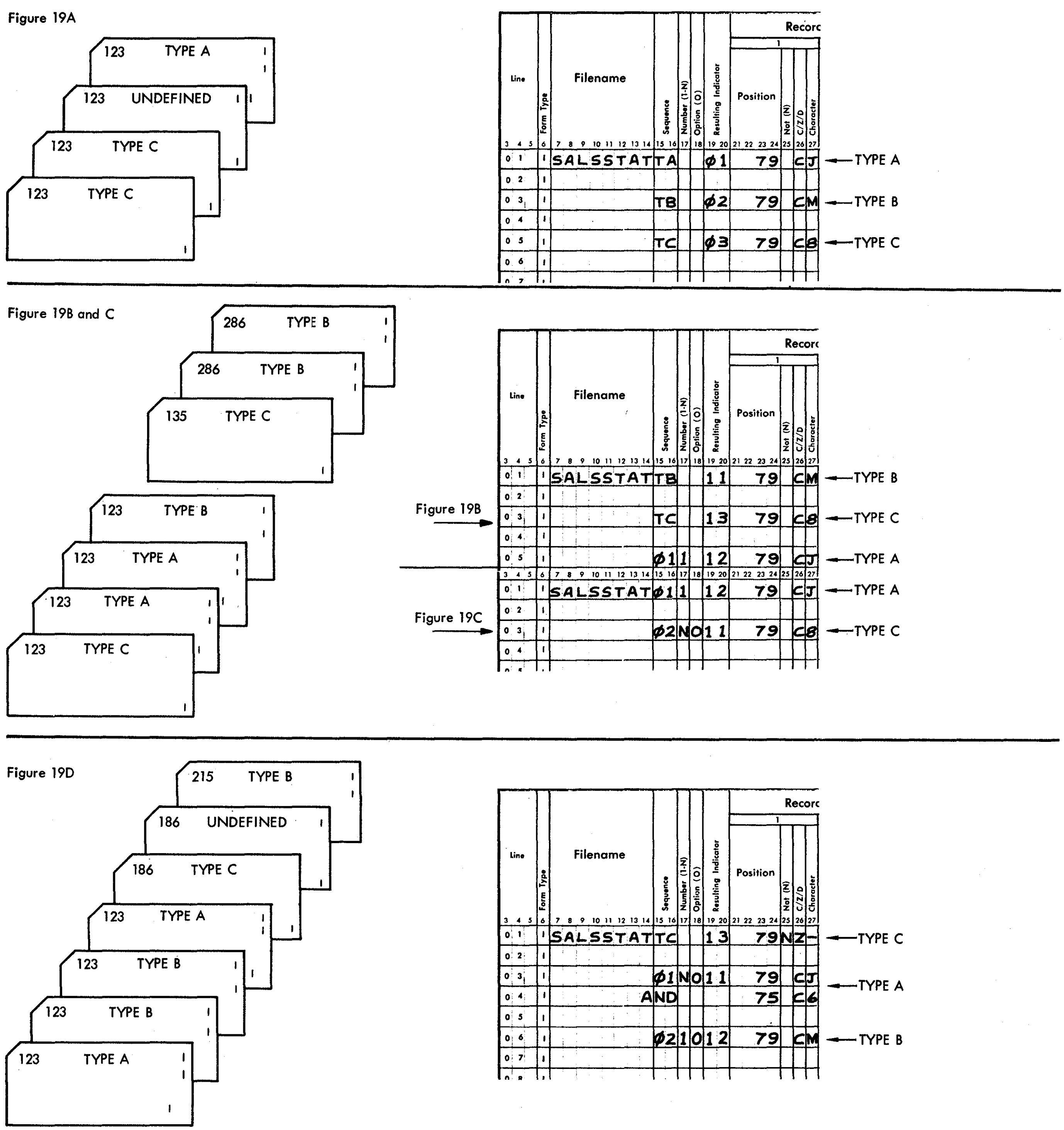


Figure 19. Potential Card-Type Sequence-Check Trouble Spots

Figure 19A

No card-type sequence check is performed, since all types are defined as optional. (In this particular illustration, all types are optional by virtue of alphabetic code in cols. 15-16; but the same applies to numeric specifications with 0 in col. 18.) Therefore, no error stop occurs for any arrangement of legitimate card types, and the absence of card type B is not signalled. (See Warnings, item 1(a), above.)

An error stop occurs when the unidentified card type is read, since all optional card types are coded alphabetic in cols. 15-16. (See Warnings, item 2(b), above.)

Figure 19B

The duplicate of card type A (with number 123) is not detected. All other types are optional and the program does not know that the two cards do not belong to two different groups. (See Warnings, item 1(b), above.)

The fact that a type-C card precedes types A and B is not signalled, because types with alphabetic specification in cols. 15-16 may appear in any relative positions.

Although a card of type A is required in every group (no 0 in col. 18), its absence from groups numbered 135 and 286 is not detected. The program has no means of recognizing that the card types C and B, numbered 135 and 286, respectively, are not part of the preceding group (No. 123), or of some following group of unknown size in which a type-A card might yet appear--because card types defined by alphabetic code in cols. 15-16 can appear in any order and quantity. (See Warnings, item 1(b), above.)

Figure 19C. Using same card arrangement as Figure 19B, but different specifications.

An error stop occurs at the very beginning, because a type-C card is read before a type A.

No stop occurs for the duplicate type-A cards: the program does not know that they do not represent two groups, since presence of type-C cards is optional. (See Warnings, item 1(c), above.)

An error stop occurs when any of the Type-B cards are read, because they are undefined--and there is at least one non-optional card type specified. (See Warnings, item 2(b), above.)

The absence of the type-A card for group 135 is not detected because, as far as the program is concerned, the type-C card numbered 135 could belong to group 123. (See Warnings, item 1(c), above.)

Figure 19D

No card-type sequence or quantity check of any kind is effectively performed, since all card types are optional. No error stop occurs for card sequence or quantity. (See Warnings, item 1(a), above.) The duplicate type-B card is not detected, nor the absence of the type-A card for group 215. (The erroneous location of the second type-A card (group 123) would never be detected by the card-type sequence check--even if neither type A nor B were optional--because it would be treated as belonging to the next group: the card-type sequence check ignores group-control values.)

When the undefined card type (group 186) is read, the program goes into a perpetual loop. (See Warnings, item 2(a), above.)

The entries in line 04 of the specifications are included only to illustrate that no card-type Sequence specifications are entered for an AND line.

Card-Type Definition--Cols. 19-20, 23-27, 30-34, 37-41

If different input card types are to be processed differently, or are to be checked for card-type sequence (see cols. 15-18), they must of course be distinguished for the program. The distinguishing entries in cols. 19-41 are made in the card-type identification lines, above the field-definition lines.

Columns 19-20 provide for the entry of a distinguishing reference code, termed a card-type Resulting Indicator, for each card type. The distinction between card types is based on the presence or absence of specific punches in each type of card, as designated by the programmer by entries in cols. 23-41.

The Resulting Indicator associated by the programmer with each card type makes it easy to condition calculation and output specifications to be executed only for certain card types (or predetermined sequences of card types).

When a new card has been read, the program checks the Record Identification Codes (cols. 23-41) of successive card-type identification lines, until it finds a match between the specifications in cols. 23-41 and the punches in the corresponding columns of the card just read. It then assigns the card-type Resulting Indicator that appears in cols. 19-20 of the line whose col. 23-41 entries match the card. (For the time in the cycle when the previous card's indicator turns off, and the new card's indicator turns on, see Figure 6: RPG Program Logic.)

Two important related points should be noted:

1. The program does not necessarily begin each time at the first card type entered in the input specifications, in its attempt to match the punches in the new card just read against the Record Identification Codes.

   The preceding section headed Nature of the Card-Type Sequence Check, and Figure 18, explain the starting point of the comparison and the order in which it is carried out. It is affected by the Sequence entries in cols. 15-16.

2. Once a match has been found between punches in the card just read and Record Identification Code entries (cols. 23-41, including possible AND lines--see below), no further card-type identification lines are searched.

   Therefore, unless card-type identification specifications in cols. 23-41 are mutually exclusive for different card types, an undesired identification may be made. Figure 20 illustrates the problem. (The possible entries in cols. 19-41, and their significance, are fully described in the next two sections.)

Explanation of Figure 20

a. Assume that only the Record Identification Codes in field 1 of Figure 20 were entered (cols. 23-27)--ignore cols. 30-41. Also assume that all three card types in Figure 20 contain a 1 in col. 80; that the second card type also has a 1 in col. 78; and that the third card type contains a 2 in col. 78, besides the 1 in col. 80. The following results occur:

   If the program begins its attempt to match the punches in the new card just read against the specifications in Record Identification Code line 01, the card will always match and indicator 01 is always assigned. No attempt is made to check for punches in col. 78, because a match has been found.

   If the program begins its attempt to match with the entries in line 03, any of the three card types is correctly identified.

   If the program begins its attempt to match with the entries in line 05, a card of the third type is correctly identified, and indicator 05 assigned thereto. A card of either the first

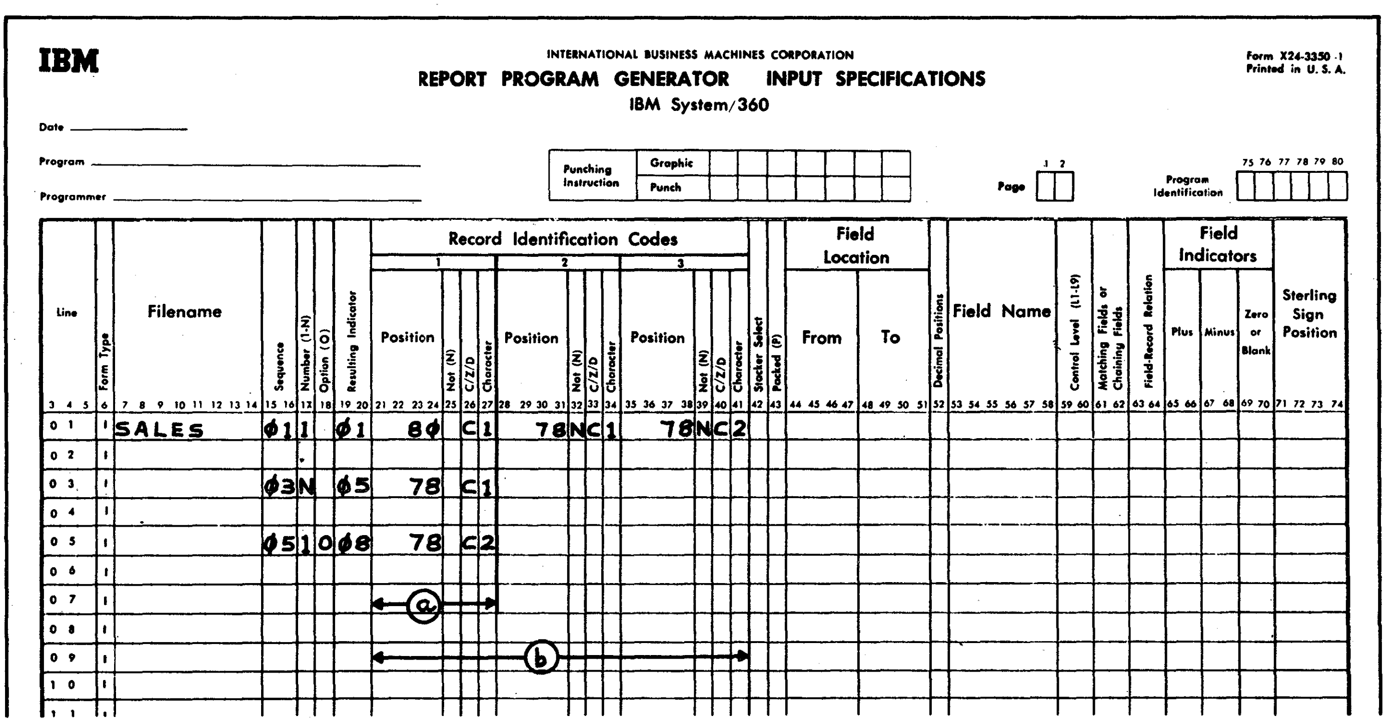


Figure 20. Making Card-Type Identifications Mutually Exclusive

or second type is identified as the first type, and assigned indicator 01. This occurs because, if the card does not contain a 2 in col. 78, it is next tested for a 1 in col. 80. This condition is satisfied by cards of all three types, and a match therefore occurs as soon as line 01 specifications are compared.

b. Assume all the entries shown in Figure 20. All three card types are then always correctly identified, because cards with 1 or 2 in col. 78 are excluded from matching the specifications for cards of the first type.

Resulting Indicator--Cols. 19-20 (Card-Type Indicator)

Any of the RPG indicators except L0 (see earlier section, Indicators) may be assigned by the programmer to each card type, and entered in cols. 19-20 in the first input specification line for that card type. The entries in cols. 23-41 associate the indicator in cols. 19-20 with a particular card-type.

The object program can then be directed, by use of the indicator code, to execute certain calculation and/or output specifications only when processing that card type--or, if desired, only when processing a card type other than that one.

Normally, any of the indicators 01-99 are assigned as card-type Resulting Indicators. The indicator on from the previous card is set off by the program before the indicator for the new card is set on. Thus, there is only one card-type Resulting Indicator on at any one time, if only indicators 01-99 (or H1, H2--see below) are used for card-type identification.

It is permissible to assign the same indicator to more than one card type. The same indicator, or two different indicators, also may be assigned to two card types in an OR relationship (see below).

Indicators H1 and H2 are suitable card-type Resulting Indicators to represent an erroneous card type. The system then halts after the card has been completely processed and before the next card is processed. (It can be restarted by depressing the CPU START key twice.)

Indicators may be assigned to the various card types in any order; numeric indicators (01-99) need not be in ascending sequence for successive card-type identification lines.

It is permissible not to assign any Resulting Indicator to a card type (i.e., to leave cols. 19-20 blank). When a card of this type is then processed, the program executes only those calculation and output specifications that are conditioned by the off status (Nxx) of card-type Resulting Indicators for other cards, and those not conditioned by any card-type Resulting Indicator. (If no card-type Resulting Indicator is assigned, care must be exercised to prevent spurious output before the first card has been read, at 1P time.) For compatibility with other RPGs, an indicator should always be assigned.

The use of indicators has already been illustrated in preceding sections, some dealing with other aspects of RPG (Figures 5, 9, 12, 17, 19, and 20). Further examples specific to indicators and card-type identification follow discussion of cols. 23-41.

Note: Card-type Resulting Indicators other than 01-99, H1 and H2 should not be assigned without a complete understanding of the sections Program Logic Flow, Indicators, and Indicator Hierarchy, in the chapter Programming for RPG--General Information.

Record Identification Codes--Cols. 23-27, 30-34, 37-41 (Card-Type Identification)

These fields provide for the identification of different card types on the basis of specific punches--or the absence of specific punches--in designated card columns.

When the punches in a card meet the criteria established in these fields for a card type, the indicator (if any) assigned (cols. 19-20) to that card type turns on before total-time processing, and remains on through detail-time processing of that card. During that time, all other card-type Resulting Indicators are off.

Exceptions: More than one card-type Resulting Indicator may be on during part or all of the processing of a card if

1. An indicator is assigned as a card-type Resulting Indicator that is not standard for that purpose (such as MR); or

2. The same indicator is assigned as both a card-type Resulting Indicator, and as Field Indicator and/or calculation Resulting Indicator; or

3. An indicator is assigned as a card-type Resulting Indicator, and the same indicator is turned on by a SETON instruction in the calculation specifications.

Similarly, although a Resulting Indicator may be assigned to every card type, all of them could be off for part or all of the processing of a card for the above reasons. (In item 3, above, SETOF would then apply, instead of SETON.)

See Program Logic Flow, Indicators, and Indicator Hierarchy.

If cols. 21-41 of a card-type identification line are left blank, all cards matched against the specifications in that line are considered to be of that card type. (See Nature of the Card-Type Sequence Check for explanation of the order in which card-type identification lines are matched.) If an indicator is specified in cols. 19-20, it is set on for the processing of that card. Leaving cols. 19-41 all blank could be a practical approach if either all input cards are to be processed identically, or multiple input files are to be merged without any need to recognize different card types, or all card types to be distinguished from the remainder are defined with alphabetic Sequence codes in preceding lines. (The user must then be certain that the deck contains no undesired cards.) Cols. 15-16 must, however, be coded.

Normally, identification of a card type must be made dependent on the presence or absence of a character in a single card column or on a combination of punches in several card columns. For convenience, space is provided on one line for three such criteria. If entries are made in two or three sets of columns, these two or three criteria are in a logical AND relationship; all of the stated criteria (specified presence or absence of certain punches) must be met for the card to be considered of that type. If more than three criteria in a logical AND relationship are required, additional lines may follow the first card-type identification line. Each additional line requires the word AND in cols. 14-16. Up to three Record Identification Code fields are again available in each AND line. Resulting Indicator (cols. 19-20) must be left blank in AND lines.

It is also possible to place any number of card-type criteria into an inclusive OR relationship; i.e., the card type is considered identified if one or more of the criteria are satisfied. Each OR criterion is then specified on a separate card-type identification line, with the word OR in cols. 14-15. The card-type Resulting Indicator number need not be repeated in the OR lines (but it may be). If no Resulting Indicator is specified in an OR line, the program assumes the indicator from the last preceding line for which a Resulting Indicator was specified (or it assumes that no indicator is assigned, if none of the identification lines for that card type has an indicator specified).

AND and OR relationships may both exist for one card type. Also, by using AND with negation of a criterion, together with an OR line, exclusive OR conditions can be specified.

There is no limit (other than the number of columns in a card, and core storage capacity) to the number of card-column characters that may be used as criteria in an AND or OR relationship to identify a card type.

There is a situation in which it is desirable to treat two or more different card types in an OR relationship. Different card-type Resulting Indicators are then assigned in the main line and the OR line(s). This application is described under Field-Record Relation (cols. 63-64).

In Model 20 card RPG, it does not matter, when using less than three Record Identification Code fields (cols. 23-27, 30-34, 37-41) in a line, which of the three fields are used. It is also permissible to use an AND line even though not all three fields are used in the main line. For compatibility with other RPGs, however, the first field should always be used, the second field should be used if two or more are needed in an AND relationship, and an AND line should not be used unless more than the three fields in the preceding line are needed.

The kinds of entries that can be made in each of the three Record Identification Code fields are identical. Therefore, only the first field (cols. 23-27) is described in detail. (Cols. 21-22, 28-29, and 35-36 do not apply to Model 20 card RPG. They may be left blank or coded with zeros.) Illustrations of all common types of entries for card-type identification follow. (Earlier illustrations appear in Figures 5, 9, 12, 17, 19, and 20.)

Position (cols. 23-24): The number of the card column (right-justified) to be checked for the identifying code punch. A leading (tens-position) zero need not be recorded.

Character (col. 27): The character to be matched against the contents of the card column specified in cols. 23-24. Any of the 256 EBCDIC characters, including blank, is a valid entry. (But see C/Z/D, below.)

Not (col. 25):

ƀ = The criterion is satisfied if the specified character (as per col. 27) appears in the designated (cols. 23-24) card column.

N = The criterion is satisfied if the specified character does not appear in the designated card column.

C/Z/D (col. 26): The programmer specifies here whether the entire character in col. 27 is to be matched against the entire character in the card column, or only the digit or the zone portions of both are to be considered. For complete flexibility in the use of the Z or D specification in col. 26, reference will be made to the EBCDIC table (Appendix D, Figure D1). Examples (Figures 21 and 22) follow the details below.

C = Character.
The entire character specified in col. 27 is compared with the entire character in the data-card column. Any of the 256 EBCDIC characters may be used.

Unless it is necessary to specify D or Z (see below), to eliminate the zone or digit portion of a character, C should be entered in col. 26. This conserves core storage and program execution time.

Z = Zone.
The zone portion of the character specified in col. 27 is compared with the zone portion of the character in the designated (col. 23-24) data-card column.

Considering first only the most common comparisons:

12-zone: If & (12-punch), any one of the letters A through I, character $\bar{0}$, or any one of the remaining six characters in the EBCDIC-table column labelled C is specified in col. 27, it will match as equal in zone to any of these 17 characters in the data-card column (specified in cols. 23-24). Any other characters in the data-card column are treated as unmatched.

11-zone: If - (11-punch), any one of the letters J - R, character $\bar{0}$, or any one of the remaining six characters in the EBCDIC-table column labelled D is specified in col. 27, it will match as equal in zone to any of these 17 characters in the data-card column (specified in cols. 23-24). Any other characters in the data-card column are treated as unmatched.

No-zone: If col. 27 is blank, contains any one of the digits 0-9, or contains any one of the remaining six characters in the EBCDIC-table column labelled F, it will match as equal in zone to any of these 17 characters (actually, 16 characters and blank) in the data-card column (specified in cols. 23-24). Any other characters in the data-card column are treated as unmatched.

Expressed more broadly, and generalized to the full EBCDIC (see Appendix D, Figure D1): Any one of the 256 EBCDIC characters may be specified in col. 27. It will match "equal" in zone to any data-card character that appears in the same column of the EBCDIC table, and be unmatched to any other data-card character, with three exceptions:

If & (12-punch) or any character in table column C is specified in col. 27, & is considered to be part of EBCDIC-table column labelled C (only). However, if one of the characters in the EBCDIC-table column labelled 5 is specified, other than & (12-punch), then & in the data card matches only any character shown in that column.

If - (11-punch) or any character in table column D is specified in col. 27, - (11-punch) is considered to be part of EBCDIC-table column labelled D (only). However, if one of the characters in the EBCDIC-table column labelled 6 is specified, other than - (11-punch), then - (11-punch) in the data card matches only any of the characters shown in that column.

If column 27 is left blank, or any character in table-column F is specified, ƀ is considered to be part of EBCDIC-table column labelled F (only). However, if one of the characters in the EBCDIC-table column labelled 4 is specified, other than ƀ, then ƀ in the data card matches only any character shown in that column.

D = Digit.
The digit portion of the character specified in col. 27 is compared with the digit portion of the character in the designated (cols. 23-24) data-card column. Any of the 256 EBCDIC characters (including blank) may be specified in col. 27.

Any character in col. 27 will match "equal" in digit to any data-card character that appears in the same row of the EBCDIC chart.

Figure 21 gives examples of C, Z, and D specifications, and the results of comparing various characters.

Col. 26	Col. 27	Contents of Data-Card Column being Compared (Column Specified in Cols. 23-24)	Result of Comparing Specified Character with Character Data-Card Column	Comments
C	S	S	Match	
C	&	&	Match	In each case, any other character in the designated data-card column would result in a non-match: C in col. 26 causes comparison of the entire character.
C	ƀ	ƀ	Match	
C	5	5	Match	
C	$	$	Match	
C	11-9-8-5	11-9-8-5	Match	
Z	A	D	Match	Both have same zone--EBCDIC-table column C
Z	H	+0 (12-0)	Match	Both in EBCDIC-table column C
Z	H	& (12-punch)	Match	When a character in column C is specified, (&) is assigned to column C
Z	12-0-9-8-6	F	Match	Both in EBCDIC-table column C
Z	& (12-punch)	12-0-9-8-5	Match	When (&) specified, it is assigned to EBCDIC-table column C
Z	& (12-punch)	$ (11-8-3)	Non-match	When (&) specified, it is assigned to EBCDIC-table column C - not 5
Z	$ (11-8-3)	& (12-punch)	Match	When any character, except (&), in EBCDIC-table column 5 is specified, (&) remains in column 5
Z	- (11-punch)	M	Match	When (-) is specified, it is assigned to EBCDIC-table column D
Z	J	R	Match	Both in EBCDIC-table column D
Z	11-0	P	Match	Both in EBCDIC-table column D
Z	11-0	12-11-9-8-7	Match	Both in EBCDIC-table column D
Z	12-11-9-8-2	- (11-punch)	Match	When a character in column D is specified, (-) is assigned to column D
Z	P	- (11-punch)	Match	When a character in column D is specified, (-) is assigned to column D
Z	- (11-punch)	% (0-8-4)	Non-match	When (-) specified, it is assigned to EBCDIC-table column D - not 6
Z	% (0-8-4)	- (11-punch)	Match	When any character, except (-), in EBCDIC-table column 6 is specified, (-) remains in column 6
Z	& (12-punch)	- (11-punch)	Non-match	Zone punches in different EBCDIC-table columns
Z	0̄ (11-0)	- (11-punch)	Match	When a character in column D is specified, (-) is assigned to column D
Z	2	4	Match	Same zone (no-zone), because in same EBCDIC-table column
Z	ƀ	0	Match	When ƀ specified, it is assigned to EBCDIC-table column F (no-zone)
Z	8	ƀ	Match	When a character in column F is specified, ƀ is assigned to column F
Z	ƀ	12-8-1	Non-match	When ƀ specified, it is assigned to EBCDIC-table column F - not 4
Z	12-8-1	ƀ	Match	When any character, except ƀ, in EBCDIC-table column 4 is specified, ƀ remains in column 4
Z	ƀ	S	Non-match	When ƀ specified, it is assigned to column F, and does not match zone in column E
Z	S	ƀ	Non-match	Zone in column F does not match zone in EBCDIC-table column 4
Z	0	T	Non-match	Zone of column F (no-zone) does not match zone of column E

Figure 21 (part I of II). Results of Comparing Various Data-Card and Record-Identification-Code Characters, with Specification of C, Z, or D

Z	T	0	Non-match	Zone of column E does not match zone of column F (no-zone)
Z	0-8-2	S	Match	Both are in EBCDIC-table column E and therefore have same zone
Z	W	Z	Match	Both are in EBCDIC-table column E and therefore have same zone
Z	- (11-punch)	V	Non-match	Column-D zone does not match column-E zone
Z	& (12-punch)	V	Non-match	Column-C zone does not match column-E zone
Z	12-11-0	@ (8-4)	Match	Both in EBCDIC-table column 7, and therefore same zone
Z	@ (8-4)	12-11-0	Match	Both in EBCDIC-table column 7, and therefore same zone
Z	12-11-6	12-11-1	Match	Both in EBCDIC-table column 9, and therefore same zone
Z	12-11-9-8-1	11-9-8-7	Match	Both in EBCDIC-table column 1, and therefore same zone
Z	H	Y	Non-match	EBCDIC-table columns C and E have different zones
Z	Q	Y	Non-match	EBCDIC-table columns D and E have different zones
Z	8	Y	Non-match	EBCDIC-table column F (no-zone) does not match zone of column E
Z	Y	8	Non-match	EBCDIC-table column E does not match in zone to column F (no-zone)
D	A	J	Match	Both in same row (1) of EBCDIC table
D	A	11-0-9-1	Match	Both in same row (1) of EBCDIC table
D	A	1	Match	Both in same row (1) of EBCDIC table
D	& (12-punch)	- (11-punch)	Match	Both in same row (0) of EBCDIC table
D	ƀ	0	Match	Both in same row (0) of EBCDIC table
D	C	12-11-8-1	Match	Both in same row (0) of EBCDIC table
D	$ (11-8-3)	12-9-8-3	Match	Both in same row (B) of EBCDIC table
D	12-0-9-8-3	12-0-9-8-2	Non-match	In different rows (B and A) of EBCDIC table
D	S	T	Non-match	In different rows (2 and 3) of EBCDIC table

Figure 21 (part II of II). Results of Comparing Various Data-Card and Record-Identification-Code Characters, with Specification of C, Z, or D

Figure 22 illustrates various correct card-type definition entries in cols. 19-41, including some uncommon ones. Explanations follow, lettered to correspond to the circled letters in the figure. Letters, rather than numbers, are used to stress that the order in which the program tests the specified codes against those in the input card does not necessarily correspond to the order in which card types are entered in the input specifications--see preceding section Nature of the Card-Type Sequence Check. (All card-type definitions should therefore be known to be mutually exclusive--one cannot assume that the type listed last will not be tested unless none of the other lines match. In Figure 22, we will assume that we know the specifications to be adequate for mutual exclusion.)

Explanation of Figure 22

a. The card type is assigned indicator 05 when col. 80 contains an & (12-punch), any of the letters A - I, or any of the remaining seven characters (12-0, and 12-0-9-8-2 through 12-0-9-8-7) in the column labelled C in the EBCDIC table (see Appendix D, Figure D1).

b. Indicator 10 is assigned to a card that meets all of these five conditions:

1. Col. 1 contains a 12-punch, and no other punch (the specification is C, not Z); and

2. Col. 80 does not contain a 12-punch, or any of the letters A - I, or any of the remaining seven characters in column C of the EBCDIC table; and

3. Col. 79 contains one of the 16 characters in column 5 of the EBCDIC table. (Note that a 12-punch is one of these 16 characters.); and

4. Col. 75 does not contain any of the characters in row 4 of the EBCDIC table (e.g.: 4, U, M, D, 12-11-0-4 etc., to 12-9-4); and

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS

IBM System/360

Form X24-3350-1
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	Filename	Sequence	Number (1-N)	Option (O)	Resulting Indicator	Position	Not (N)	C/Z/D	Character	Position	Not (N)	C/Z/D	Character	Position	Not (N)	C/Z/D	Character	Stacker Select	Packed (P)	From
01	I	COSTANAL	BA			∅5	8∅		Z	&											a
02	I																				
03	I		∅1	1		1∅	1		C	&	8∅	N	Z	B	79		Z	$			b
04	I	AND					75	N	D	M	∅5		Z	T							
05	I																				
06	I		∅5	N	O	∅8	1		D	$											c
07	I	OR					1		C	4	5	N	C	b̸							
08	I																				
09	I		1∅	1	O	25	1		Z	-											d
10	I	OR				26	5		Z	b̸											
11	I																				
12	I		12	N		12	1		C	1	75	N	C	2							e
13	I	OR					75		C	2	1	N	C	1							
14	I																				
15	I		99	1	O	H1	1		C	1	75		C	2							f

Figure 22. Examples of Card-Type Identification Entries

5. Col. 5 contains one of the 16 characters in column E of the EBCDIC table (e.g.: 0-8-2, 11-0-9-1, S, T, etc., to 11-0-9-8-7). Note that no match occurs if the data card is blank, or punched zero only (i.e., the unit record Hollerith code 0-zone for letters S - Z does not apply).

This example also illustrates AND lines. Note also that a leading 0 may be omitted from card-column number (e.g., col. 1) or recorded (e.g., col. 05).

c. Indicator 08 is assigned to a card that meets either of these criteria:

1. Col. 1 contains one of the characters in row B of the EBCDIC table (e.g.: $, 12-11-0-9-8-3, comma, 12-9-8-3, etc.); or

2. Col. 1 contains a 4 (and no other punch) and col. 5 contains any punch (i.e., is not blank).

This example also illustrates an (inclusive) OR relation, with either of two card types assigned the same Resulting Indicator. It also shows the combination of an AND relationship (two criteria in the OR line) with the OR relationship.

d. Indicator 25 is assigned when col. 1 of a card contains an 11-punch, any of the characters J - R, or any of the remaining seven characters in column D of the EBCDIC table (11-0, 12-11-9-8-2, etc.).

Indicator 26 is assigned when col. 5 is blank, contains any of the digits 0-9, or contains any of the remaining six characters in column F of the EBCDIC table (12-11-0-9-8-2, etc.)

The value of this type of OR relationship--where two card types are assigned different indicators, yet placed in an OR relationship--will become clear when Field-Record Relation (cols. 63-64 in the input specifications) is discussed later.

e. Indicator 12 is assigned when either one of two sets of criteria is met:

   1. Col. 1 contains a 1 (and no other punch) and col. 75 does not contain a 2 alone (other characters that incorporate a 2-punch are permitted, since the specification in col. 33 is C for an exact character match); or

   2. Col. 75 contains a 2 (and no other punch) and col. 1 does not contain a 1 alone.

   This illustrates an exclusive OR relationship: the criteria for indicator 12 are satisfied if <u>either</u> of two conditions applies (1 in col. 1 <u>or</u> 2 in col. 75), but not if <u>both</u> apply.

f. This assumes that the card is wrong if both of the conditions occur that were handled in entry (e) as mutually exclusive.

   If the card contains both a 1 (alone) in col. 1 and a 2 (alone) in col. 75, indicator H1 is assigned. Unless the H1 (or H2) indicator is reset by a programmer's specification before then, the system halts after the card has been completely processed.

   The H1 indicator may be used like any other indicator. It might logically be utilized to condition calculation and output specifications not to be executed when H1 is on (by specifying NH1 in the conditioning indicator fields).

Throughout, in Figure 22, notice that--while numbered Sequence entries (in cols. 15-16) must be in ascending-number order, and must start with 01--Resulting Indicators can be assigned in any order.

Previous sections stated that, when an input card of an undefined type is read, either an error stop or a perpetual program loop (see <u>Warnings</u>, item 2(a), above) results. To avoid a perpetual program loop or an error stop which requires card handling for restart, and to facilitate bypassing of calculation and output specifications for invalid cards, the user should make provision for invalid cards in the card-type definition specifications (cols. 19-41) for each file. <u>Figure 23</u> illustrates three approaches. For simplicity, only two legitimate card types are used in each example; and the assumption is made that one type contains a 1 (only) in col. 5, and the other a 2 (only).

Line (3-5)	Form Type (6)	Filename (7-14)	Sequence (15-16)	Number (1-N) (17)	Option (O) (18)	Resulting Indicator (19-20)	Record Identification Codes 1: Position (21-24)	Not (N) (25)	C/Z/D (26)	Character (27)	Record Identification Codes 2: Position (28-31)	Not (N) (32)	C/Z/D (33)	Character (34)	Record Identification Codes 3: Position (35-38)	Not (N) (39)	C/Z/D (40)	Character (41)	Stacker Select (42)	Packed (P) (43)
01	I	SALES	Ø1	1	O	H1	5	N	C	1	5	N	C	2					2	
02	I																			
03	I		21	N	O	45	5		C	1										
04	I																			
05	I		22	1	O	48	5		C	2										
06	I																			
07	I	COMMISSN	XY			99	5	N	C	1	5	N	C	2					2	
08	I																			
09	I		Ø1	1		15	5		C	1										
10	I																			
11	I		Ø5	1		24	5		C	2										
12	I																			
13	I	COSTSALS	XA			Ø1	5		C	1										
14	I																			
15	I		XB			Ø2	5		C	2										
16	I																			
17	I		YY																2	
	I																			

Figure 23. Examples of Protection Against Undefined Card Type

Explanation of Figure 23

<u>Example 1</u> assigns an H indicator (H1) to an illegal card (neither character 1 nor 2 in col. 5). A stop then occurs after the illegal card has been processed, unless the H indicator was reset (by a programmer's specification) before detail-time output. (The halt is non-abortive: the system can be restarted by pressing the CPU START key twice.) The H indicator can also be used to bypass specifications that should not be executed for invalid card types.

For illustrative purposes, even the legitimate card types were specified as optional (O in col. 18), which negates the card-type sequence check. However, if they were not designated as optional--and the entry for invalid card types has a numeric Sequence specification (cols. 15-16), as shown--a stop for card-type <u>sequence</u> error could occur on an invalid card, even though we chose to write the specifications for invalid cards on the first line. (Whether the card-type sequence-error stop or the invalid-card-type halt (specification line 01) would occur on an invalid card is dependent on the particular specification line which the program tests first against a card just read--see <u>Nature of the Card-Type Sequence Check</u>. In example 1, either specification line 01 or line 03 could be the first one compared against a data card.) A card-type sequence error stop --in contrast to the H1 halt--is more complex to restart, does not provide a unique indicator to condition execution of specifications for invalid card types, and does not offer a single method to stacker-select such a card (with or without stopping).

Example 2 shows an effective method of achieving the same flexible result as in Example 1, yet requiring a specific (non-optional) card-type sequence for the valid card types. Card-type specifications with alphabetic Sequence entries (cols. 15-16) are always tested before those with numeric Sequence specifications (see Nature of the Card-Type Sequence Check). Thus, in this example, the validity of the card type is always checked first--the user is assured of indicator 99 for an invalid card type, and can stacker-select invalid cards by a simple entry in specification line 07.

Indicator 99 does not cause a system stop; but otherwise it may be used like the H1 indicator in Example 1 to condition the execution of specifications. If a halt after an invalid card is desired, H1 or H2 can, of course, be assigned in place of a numeric indicator like 99.

Example 3 takes advantage of the fact that card-type identification specifications with alphabetic Sequence designation (cols. 15-16) are always tested in the order in which they are entered (see Nature of the Card-Type Sequence Check). The example, however, assumes that no card-type sequence check is required.

Specification line 17 will be tested against a data card only when neither the specifications in line 13 nor those in line 15 matched the data card. Since no Record Identification Codes appear in cols. 23-41 of line 17, it will always match against a data card when tested. Thus, whenever neither line 13 nor line 15 matches the data card, line 17 will be tested and it will match--therefore, all invalid cards will be associated with line 17.

No Resulting Indicator (cols. 19-20 blank) was assigned to invalid card types, merely to illustrate another possible approach (any indicator, including H1 or H2, could have been assigned). If specifications that are to be executed for valid card types (or before the first card) are all conditioned by indicators which are off for invalid card types, then the absence of an indicator during processing of invalid types suppresses execution of such specifications.

Example 3 illustrates a convenient technique for identifying invalid card types when specifications for invalid types would be complex. For example: If there are several valid card types, each with numerous AND and/or OR relations, it could become involved to specify the Record Identification Codes for an invalid type. Such specifications would require the negation of all possible valid card-identification punch combinations. The limitation of the approach in Example 3 is its requirement that the valid cards cannot be checked for card type sequence.

Note that, with this method, the entry for the invalid type must always be last for that file--if it is first, every card will be treated as invalid, since there are no specifications in cols. 23-41 to exclude valid cards from matching the line.

The stacker-select entry in col. 42 of all three examples is explained below (under Stacker Select).

Records in an OR relationship

Records in an OR relationship must be in the same file. The three types of OR relationships are described below.

1. Identical Fields and Similar Processing:

   The input fields of several card types are in the same columns, have the same format, and identical field names apply. No distinction between the card types is required in the field description entries--each field is described only once--and the several card types are treated throughout as though they were identical, with one possible exception available: they can--if desired--be selected to different stackers by entries in col. 42 of the input specifications (if no output operation is performed on them). Because the fields for two or more card types are described only once, core storage space is saved.

   The Resulting Indicator in cols. 19-20 of the main card-type identification line applies also to the OR line(s) where no Resulting Indicator is entered; alternatively, the same Resulting Indicator may be repeated in the OR line(s).

   This type of OR relationship was already illustrated in Figure 22: lines 06 and 07, and lines 12 and 13. (A different stacker could have been specified in col. 42 for the two card types in an OR relationship.)

2. Identical Fields but Different Processing:

   Two or more card-types differ only in their Record Identification Codes (cols. 23-41); but their input fields are in the same columns, have the same format, and identical field names apply. No distinction between the card types is required in the field description entries--each field is described only once--but the card types are to be processed differently in the calculation and/or output-format specifications. (They can, of course, also be directed to different stackers.) Because the fields for two or more card types are described only once, core storage space is saved.

Different Resulting Indicators in cols. 19-20 are assigned to the card-type specification lines in an OR relationship, to permit distinction between the card types in the calculation and/or output-format specifications.

Figure 17B, lines 12 and 13; Figure 17C, lines 13 and 14; and Figure 22, lines 09 and 10 represent this kind of OR relationship if cols. 63-64 of the field description lines (not shown) are blank.

3. Some Identical and Some Different Fields for Different Card Types:

   See Field-Record Relation, below.

OR Relationships are further illustrated in Figure 26, below.

Stacker Select--Col. 42

If no stacker-select entry is made in input or output specifications, the cards of that type enter the normal stacker for the particular card read and/or punch device. If the device contains more than a single stacker, cards can be program-directed to a non-normal stacker, by an entry in the input or output specifications. Figure 24 itemizes the normal and additional stackers for card input/output units with multiple stackers, and the pertinent stacker-select codes. For single-stacker I/O devices, stacker select should be left blank (however, any entry is simply ignored by the program).

Note: In the case of the IBM 2520 Card Punch or Read-Punch, cards with punch errors are automatically directed to stacker 2--the non-normal stacker--by the system.

INPUT/OUTPUT UNIT	STACKER SELECT CODE	STACKER NO.	
IBM 2520 CARD PUNCH or READ-PUNCH	blank or 1	1 (Normal)	
	2	2	
IBM 2560 MULTI-FUNCTION CARD MACHINE		Model A1	Model A2
	blank (from hopper 1)	1 (Normal Selection)	1 (Normal Selection)
	blank (from hopper 2)	5 (Normal Selection)	4 (Normal Selection)
	1	1	1
	2	2	2
	3	3	3
	4	4	4
	5	5	4

•Figure 24. Summary of Stacker-Select Specifications for Multi-Stacker Card I/O Devices

Rules for Stacker Selection

Output-file cards can only be stacker-selected in the output-format specifications.

Input-file cards can only be stacker-selected in the input specifications. This is accomplished by entering the number of the desired stacker in col. 42 of the card-type identification line for the pertinent card. If a card type is to enter the normal stacker for the I/O device that contains the file, column 42 may either be left blank or coded with the number of the normal stacker (which is always 1, except for the secondary hopper of the MFCM).
The preceding Figure 23 illustrated, in all three of its examples, how to select a particular card type--in this case, invalid cards--to stacker 2, while letting all other card types enter the normal stacker.

Stacker selection of input-file cards is possible, based on file matching and/or calculation results. In this case, however, the file must be defined as combined and the file name entered in the Output-Format specifications. (See also Rules for Stacker Selection under Output-Format Specifications.)

Note: It is also possible to perform stacker selection on input-file cards, based on file matching and/or calculation results, by means of the EXIT operation code and BAL subroutines (see Programming Tips, Appendix E).

Combined-file cards may be selected in the input specifications--when selection can be based on card type alone--or in the output-format specifications. It is permissible to select some card types within the file in the input specifications, and others in the output-format specifications. The criteria to be applied are:

1. A card type must not have stacker-select instructions in both the input and output-format specifications.

2. If any output operation (punching and/or card-printing) is to be performed on cards of a type, any stacker selection to be specified for this type must be in the output specifications. (If stacker selection is specified in the input specifications, but output operations are also specified, the output is to the next card and this next card is never read.)

3. If stacker selection is to be based on the results of calculation specifications, or on the result of matching between files (MR indicator), it must be designated in the output specifications.

4. While not necessary, it is recommended that a stacker-selection specification --even if it is the number of the normal stacker--be made in the input specifications for any combined-file card type on which an output operation is not to be performed--i.e., the card type will not be punched or card-printed (interpreted), nor will it be stacker-selected on the basis of calculation-specifications or Matching-Fields results. This expedites throughput.

The points itemized above are logically supported thus: For a combined file, the program makes provision to read each card, and then halts it at the pre-punch station, to await output instructions after completion of calculations. However, if a stacker number (even if that of the normal stacker) is given in the input specifications, the program uses this fact to eject the card immediately after reading and to read the next card. Processing (calculations) for one card is then overlapped with reading of the next card.

Note: When stacker 5 is designated, but the I/O device referred to is the 2560 MFCM Model A2, the card is directed to stacker 4.

AND Lines

When the number of Record Identification Codes requires AND lines, any Stacker-Select entry must be in the main (first) line--never in an AND line.

OR Lines

Stacker-selection is independent for the main line and each OR line, just as for different card-type identification lines.

To amplify:

1. If no Stacker-Select entry is made in the OR line, the card type enters the normal stacker, regardless of the stacker for the card type defined in the main line above the OR line.

2. The card type in the OR line may have a stacker-select specification different from that in the main line; or the stacker-select column in the main line could be blank, but the OR line could have a stacker specification; or both could be blank.

3. The rules for combined-file card types apply as though OR lines defined totally separate card types.
   However, if the main line and the OR line are not assigned separate card-type Resulting Indicators, nor is any distinguishing indicator assigned elsewhere--and one specification line (say, the main line) is designated as an input type only (by a stacker-select entry in the input specifications) and the other line (say, the OR line) is designated as a combined type that is to receive output (by virtue of the absence of an input stacker-select entry)--then output could be into the following, rather than the relevant, card.

FIELD DESCRIPTIONS

Field descriptions are required for each field of an input card that is to be used in the application (i.e., as data field for calculations, as Control-Level field, as Matching Field, to set Field Indicators, or to provide data for output). No field description is entered in the input specifications for a card field that serves solely to receive output data, nor for an input card-type field that is ignored in the application. (Entries for fields not needed for data input waste core storage space and process time.)

A separate line is used for each field description. Field descriptions for each card type begin on the line immediately below the line describing the card type.

If there are several lines describing one card type (AND lines) or related card types (OR lines), field descriptions begin immediately below the last of such card-type identification lines. The file and card-type identification area (cols. 7-42) must be left blank in field description lines.

Input fields are tested for setting of Field Indicators, and transferred to the internal process area, in the sequence in which they are entered. This need concern the user only when unconventional assignment of indicators, or multiple assignment of the same indicators, is involved.

If the application does not utilize any data from fields of a card type, no field description is required. This could be the case, for instance, when the only operation performed for a card type is stacker selection based on card type.

Packed--Col. 43

Leave col. 43 blank for normal (unpacked) input data.

Enter the letter P in col. 43 if the input data is (already) in packed-decimal format.

If the same input field appears more than once in the input specifications, with the same name, that input field must always or never be specified as in packed format (P in col. 43)--i.e., it cannot be designated as packed input for one card type and unpacked for another, with the same field name.

Packing is a data-storage technique whereby two digits (or one digit and sign) are stored in the space normally required by one alphameric character--i.e., one core storage byte or one card column. Packing removes all zones of a zoned decimal field, except in the low-order (rightmost) position, where inversion of the zone and digit occurs. At the same time, blanks from the source field are converted to zeros. For example, the input field (zoned decimal) bb123 is represented in storage as 4040F1F2F3. After packing it looks like this: 00123F. The RPG converts numeric input data to packed format, if the data is to be used in numeric compare, arithmetic, editing, or Zero Suppress operations (see Decimal Positions, col. 52, below). The P entered in col. 43 prevents RPG from packing numeric data again if it is already in packed format at time of input.

Numeric input data may, for example, be in packed format in order to get more information into one card. (This could reduce the number of cards to be processed by up to 50%.) The data might have been punched into the cards as output in packed format in a previous operation. Punching output data in packed format can save punching time on a serial punch (e.g., MFCM or 1442) if it thereby reduces the number of the last column to be punched. Incidentally--where data is required to be in packed format because arithmetic or editing operations are to be performed upon it--input in packed format saves the processing time for packing, and core storage for the packing routine.

When input data is already in packed format, the RPG program assumes that the low-order position of the field contains a punch combination whose bit equivalent for the lower half-byte represents a valid sign. This implies that the punch combination in the low-order position of the field must be represented in row A, B, C, D, E, or F of the EBCDIC table (Appendix D, Figure D1)--B and D are treated as minus, the others as plus. Hence, no blanks (X'40') are allowed in the low-order (rightmost) byte of a field specified as packed. Since 0 is an invalid sign, any arithmetic operation attempted with this field will result in a non-standard machine halt (specification error). If the field is to be used in numeric compare, arithmetic, or editing (including Zero Suppress) operations, the punch combinations for the low-order column are further confined to EBCDIC-table columns 0-9; the punch combinations for all columns, except the low-order column, are then confined to EBCDIC-table columns 0-9 and rows 0-9 (i.e., they must consist of two valid digits). Appendix D discusses data formats.

An input field specified as Packed (P in col. 43) is always considered by the program to be numeric; a specification must therefore be entered for such a field in Decimal Positions (col. 52)--see below.

Maximum field length for a packed input field is 8 columns (which corresponds to 15 digits, and sign).

In input specifications, Field Location (col. 46-47 and 50-51) must reflect the actual columns that contain the field in the input data card--not the number of digits these columns represent. Decimal Positions (col. 52) must reflect the number of digits--not the number of columns--to the right of the decimal point.

In calculation specifications, the field size must be considered exactly large enough to accommodate the same number of digits (and sign) in unpacked format. If n represents the number of card columns of the packed input field, the length of the field in calculation specifications becomes 2n-1. This also applies to output-format specifications for packed input fields, unless "Packed Field" is specified in the output-format specifications, too.

Input fields designated as packed (P in col. 43) cannot be used with Control-Level (cols. 59-60) or Matching-Fields (cols. 61-62) entries. The equivalent effect can, however, be achieved by a second field-description entry for the same input field, with a different field name, leaving col. 43 blank and treating the field this time as alphameric (no entry in col. 52). If Matching Fields (cols. 61-62) are used with this second entry, the user must realize that the sequencing operations are then based on the field contents as one EBCDIC character per column--not two digits per column. Appendix D explains the relative sequence position of each of the 256 EBCDIC characters. Note that, if the second definition of the same field (with a different field name) is used solely for Control-Level or Matching-Fields purposes, a diagnostic warning message ("unreferenced field names") is printed during generation of the object program; but generation proceeds properly.

Note: While, as discussed above, Packed format is available as a compaction technique for numeric input (and/or output) data, column-binary (card-image) input (or output) cannot be used with this RPG.

Field Location--Cols. 44-47 and 48-51 (Cols. 44-45 and 48-49 are not used in Model 20 card RPG--they may be left blank or coded with zeros.)

These columns define the location of each input field in the card.

The maximum length of an input field is:

a. For a standard (unpacked) numeric field: 15 columns.
   Fields to be used in numeric compare or arithmetic operations, and/or to be edited or zero-suppressed in output specifications, must be defined as numeric--by an entry in Decimal Positions (col. 52).

b. For a packed field: 8 columns. (See Packed, above.)

c. For an alphameric field: No limit, other than data input-card capacity (80 columns).

Input fields may be listed in any order, except when Control Levels are specified (see cols. 59-60, below) or Field-Record Relation is involved (see cols. 63-64).

It is permissible for input-field card columns to overlap, if the fields are given different names.

From--Cols. 46-47

The number of the leftmost (high-order) card column of the input field. The entry must be right-justified; a leading zero may be omitted.

To--Cols. 50-51

The number of the rightmost (low-order) card column of the input field. The entry must be right-justified; a leading zero may be omitted.

Note: A single-column field is defined by the same column number in cols. 46-47 and 50-51.

Decimal Positions--Col. 52

For alphameric fields, leave col. 52 blank.

An entry (0-9) in this column defines the associated field (as named in cols. 53-58) as a numeric field. An input field must be defined as numeric in the input specifications if any one or more of the following statements apply:

1. The input field is in packed format (P in col. 43)--see Packed, above.

2. The field will be used as a factor or result field in numeric compare or arithmetic operations in the calculation specifications. Arithmetic operations comprise: addition, subtraction, multiplication, and division--i.e., the operation codes ADD, Z-ADD, SUB, Z-SUB, MULT, DIV, MVR. (An input field cannot be defined as numeric--i.e., have Decimal Positions specified--in calculation specifications unless it was defined as numeric in the input specifications.)

3. The field will serve as search argument in a look-up (LOKUP) operation for an argument table defined as numeric.

4. Edit or zero-suppress operations are specified for the field in the output-format specifications.

5. Output is specified to be in packed format (see Output-Format Specifications).

For a field that is to be treated as numeric, enter in col. 52 a digit from 0-9 to represent the number of decimal places in the input data field. For standard (unpacked) numeric input fields, the Decimal Positions entry is synonymous with the number of card columns to be considered to lie to the right of the decimal point. For packed input fields, it applies to the number of digit positions to the right of the hypothetical decimal point (e.g., a 3 in col. 52 for a packed input field specifies 3 digits to the right of the decimal point, contained in the 2 right-hand columns).

The maximum number of decimal pcsitions that can be specified for a field is 9; but the number of decimal positions specified must not be greater than:

1. the length of the field--for standard (unpacked) numeric input fields; or

2. the digit capacity of the field--for packed input fields. (Digit capacity = 2n-1, where n = the length of the packed input field.)

If the entire field represents an integer, without any decimal places, enter 0 in col. 52.

An entry (0-9) in col. 52, besides designating that field as numeric, also serves three related purposes for the field specified in that line:

1. It assigns the location of the decimal point, so that the object program can perform automatic decimal-point alignment during numeric compare and arithmetic operations.

   Note : If a field must be defined as numeric, but will not be used in compare or arithmetic operations, any of the digits 0-9 (within field-size limit) may be specified--it need not conform to the number of decimals in the field.

2. It directs the object program to pack the field (see Appendix D)--unless input was already in packed format (P in Col. 43). Packing strips off all zones, except in the low-order (rightmost) position of the field, where packing causes inversion of the zone and digit. At the same time, blanks are converted to zeros.

   In numeric compare, arithmetic, and editing operations, the program treats an input field with a 12-overpunch or the absence of a zone overpunch in the low-order column as positive (+); and an input field with an 11-overpunch in the low-order column is treated as negative (-).

   Where zones are stripped (i.e., all but the low-order column) all punch combinations that appear in one row of the EBCDIC table (see Appendix D, Figure D1) take on the value that appears under the column heading F for that row (e.g.: 12-9-4, 12-11-9-4, D, M, U, 4, etc., all become 4; b, &, -, 12-11-8-1, etc., all become 0).

   For the low-order position, zones are handled by the program as follows:

   a. If the column contained any of the punch combinations in EBCDIC-table column E or F, the E- or F-zone remains and the field is treated as positive (or zero, if entire field is zero).
   b. If the column contained any of the punch combinations in EBCDIC-table column D, or an EBCDIC 60, D-zone is assigned and the field is treated as negative (or zero, if entire field is zero).
   c. All other punch combinations are assigned C-zone and the field is treated as positive (or zero, if entire field is zero and/or blank).

   Once the field becomes a result field for an arithmetic operation, it is signed C-zone (not F-zone) for plus or all zeros, or D-zone for minus.

   If the input field is to be used in numeric compare, arithmetic, or editing operations, the punch combinations in all card columns of the field must be represented in rows 0-9 of the EBCDIC table, to yield valid digits when packed.

3. It causes zones in any position of that field (including the low-order position) to be ignored from data comparisons effected by Control-Level (cols. 59-60) and Matching-Fields (cols. 61-62) specifications.

   Whenever Control Level or Matching Fields is specified for a field, the data from the field is stored separately in an additional core location for each of these two functions, besides its storage as a regular input data field. The data is stored for the Control-Level and/or Matching-Fields operations in standard (unpacked) format; however, if there is a specification in col. 52, all positions are stored as no-zone (hexadecimal F zone--see EBCDIC table)--specifically: each code in an EBCDIC-table column labelled 0-E is converted to the code in the same row under column heading F.

   If it is desired to treat the field as numeric for calculations and/or editing--but to retain zones for Control-Level and/or Matching Fields operations (e.g., to distinguish positive from negative control groups)--the field may be specified twice, with different field names in the two specification lines. The entries in one line then include a Decimal Positions specification (col. 52); the field name in this line is used with numeric compare, arithmetic, and editing operations. The other line is blank in Decimal Positions (col. 52), but includes the Control-Level (cols. 59-60) and/or Matching-Fields (cols. 61-62) specifications. This thechnique, of course, consumes additional core storage space.

Note also that, if the one field name is used solely with Control-Level or Matching-fields specifications, and not referred to for calculation or output data, a diagnostic warning message ("unreferenced field names") is printed during program generation. This does not prevent proper program generation.

> Note: Once a given Control Level (L1-L9) or a given Matching-Fields level (M1-M3) is specified in an input field-description line that contains a Decimal Positions entry (col. 52), Control-Level or Matching-Fields entries of the same level (L1-L9, M1-M3) in all card types are treated as numeric for Control-Level or Matching-Fields operations, respectively. This applies even if the field name is different in the several specification lines for that Control Level or Matching-Fields level. (A warning message is printed during program generation in the case of control fields for one level being defined as both alphameric and numeric.)
>
> If the field names are different, they may differ in format--alphameric versus numeric; if numeric, in position of decimal point--but the number of columns in the field must be the same. They are then treated as numeric for Control-Level or Matching-Fields operations, respectively (if one card type was specified as numeric for that Lx or Mx level); but for other operations, each field is treated in accordance with its own format specifications.

Note: The program does not perform automatic decimal alignment on numeric fields in Control Level or Matching Fields operations.

If there is no need to define the input field as numeric (i.e., it will not be used in numeric compare, arithmetic, edit, or zero-suppress operations)--even though the data is numeric--the programmer has the option of defining the field as alphameric (col. 52 left blank) or defining it as numeric (0-9 in col. 52), depending on the relative significance, to his program, of the factors itemized immediately below.

Defining an input field as numeric causes the program to pack it for input, and to unpack it for output (unless Packed Field is specified for output).

1. Packing and unpacking consume object-program process time, and core storage space.
2. Packed data occupies less core storage space than unpacked data.
3. A field that is to be packed cannot be longer than 15 columns before it is packed.

If the same input field appears more than once, with the same name, in the input specifications it must always be the same size, and defined in the same format: always standard data format or always packed; always alphameric or always numeric; if numeric (or packed), then always with the same number of decimal positions. This uniformity of size and format for one field name applies within and between different specifications forms (input, calculations, output). (However, since the format of input fields is fully defined in the input specifications, the number of decimal positions, together with field length, need not be repeated in the calculation specifications if an input field is also used as a calculation Result Field.)

Field Name-Cols. 53-58

Each input field delimited in Field Location must be given a Field Name by the programmer. Once a name has been assigned to a field, the field is referenced in calculation and output specifications by its name. The name is associated by the program with an address in core where the data for that field is stored; but the user need not concern himself with the actual core storage location.

The name must begin in column 53 with one of the 29 alphabetic characters, may continue with alphabetic or numeric characters, and may be one to six characters long. (See Definition of Terms, for "alphabetic" and "numeric" characters--neither permits embedded blanks.) Within these rules, any Field Name may be assigned to any field in the input specifications, with the exception of

> ALTSEQ, or a name beginning with CONTD or TAB, or PAGE followed by one or two characters.

Also, the name PAGE is reserved for a special purpose (see Consecutive Numbering).

The same field name may be used for any number of fields in different card types (and as Result Field in calculation specifications), provided all fields with the same name

1. are the same length; and
2. have the same data format: standard or packed, alphameric or numeric; and
3. if numeric (or packed), have the same number of decimal positions specified.

The same core storage location is used for fields with identical names. Core storage is thus conserved by assigning the same name to fields in different card types. This has the further advantage that, if the same processing is to be applied to the field for different card types, calculation or output specifications may be saved. The programmer must be careful, however, that data in storage from one card type is not superseded by that from another type until it is no longer needed: any time a card with the particular field name is read, its data replaces that previously stored at the location for that field name. (The actual data substitution occurs just before detail-time calculations--see Figure 6, RPG Program Logic.)

On the other hand, by making a field name unique to a card type, the data for that field is retained until a card of the same type is read again. This permits processing data from a previous card with data from a later card. (For exception, see Blank After, under Program Logic Flow, and under Output-Format Specifications.)

It is also possible to save core storage space, in specialized situations, by assigning the same name to different fields within the same input card. (See Field Indicators, "Points to Note.")

Note: While not recommended, because it would tend to confuse, it is permissible for a file name to be the same as a field name.

Defining the Same Data Field as Both Alphameric and Numeric

The program assigns a separate core storage location for data associated with each field name. The same source-data field (input-card columns) may therefore be defined more than once and with different data formats, provided each definition of the field is on a separate field-description line and is assigned a different field name.

This technique can be used to advantage when a numeric Control-Level field (or Matching Field) may contain 11- or 12-overpunches that are an essential part of the control-group (or matching-group) identification, and this field is also involved in numeric compare, arithmetic and/or editing operations. For the latter three kinds of operations, the field must be defined as numeric (0-9 in col. 52). However, when a field is defined as numeric, all zones are stripped for Control-Level and Matching-Fields comparisons (see Decimal Positions, col. 52, above). The solution is to define the same input field twice, with two different names: on one line as numeric (0-9 in col. 52), with that field name referenced in calculation and/or editing operations, and no entry in Control Level or Matching Fields; on the other line as alphameric (col. 52 left blank), with Control-Level (cols. 59-60) and/or Matching-Fields (cols. 61-62) entries in this line.

This dual definition is also useful if a field is to be used in arithmetic operations, but it is also desired to test it for blanks (as distinct from zeros) in the input specifications (see Field Indicators) or for high-order-position zones in calculation specifications (see TESTZ).

If a field is defined solely to serve for Control Level or Matching Field, or Field Indicators, and not used in calculation or output specifications, the warning message "unreferenced field names" is printed during generation of the object program. Generation, however, proceeds properly. Actually, the field name is not used at all by the program if the field is defined solely for Field-Indicator, Control-Level or Matching-Fields operation. It must be given, however, to prevent a diagnostic error stop for missing field name.

Using Input Data Fields for Constant Data (Heading Cards)

The term "constant" is applied here to information, or an item of data, that does not change as different data cards are processed; it may be required to remain fixed for the entire job on a given day, remain fixed for part of the data deck, or be permanently fixed whenever a given report is run.

Examples of constant data might be report date, report title, identification for different portions of a report, and report-column headings.

The output-format specifications provide for defining data that remain permanently constant for the report, such as report title or report-column headings. A constant defined in the output specifications is limited to a maximum of 24 positions, although this limit can be circumvented by specifying several constants for successive sets of print or punch positions. (See Output-Format Specifications chapter.)

The input specifications offer a convenient means of entering constants that

1. exceed 24 columns--such as a long report title; and/or

2. may have to be changed each time the report is run--such as a date; and/or

3. differ for successive sections of a report--such as separate report headings for executive, regular salary, and hourly-rated payroll reports, when there are otherwise no differences in the processing of the reports.

The easiest way to enter such constants is to identify the card containing the constant data as a separate input-only card type, and assign a field name that is not repeated for any other field or card type to the columns containing the constant. The card type containing the constant is placed in the data deck wherever desired: if it is a date card or report-heading card, it would normally be placed at the front of the input-data deck. If there are separate constants cards for different sections of the report (such as report-section headings), they can be placed at the beginning of the pertinent sections of the input-data deck; when a new constants card is read, its data will replace the data from the previous one--until that point, the data is preserved because no other input card has the same field name assigned.

When constants cards are interspersed in the data deck (to change constants for different sections of a report), they may have control fields and Control Levels assigned, to assure that they are in the correct group and/or to make it possible to sort or merge them into the deck mechanically. Simple calculation specifications can ensure that a constants card is always at the front of its section.

If the constants card is defined as a separate file, is should be designated the primary file, so that it is read first, and the constant information is available before the first data card.

If multiple input (or combined) files of data cards are processed, the constant card(s) may appear just ahead of any file or file section. If no Matching Fields are specified for the constants card, it will be read ahead of Matching-Fields cards in the other files. (For specifics, see chapter titled Matching of Files.)

Consecutive Numbering (Page Numbering)--Heading Cards

The output-format specifications provide a simple method for printing consecutive numbers on successive pages of output forms, or printing or punching consecutive numbers in cards, beginning with 1 as the first number.

If numbering is to begin with a number other than one (or if it is to begin again with 1 at points in the data deck that cannot be specified with conditioning indicators), provision for loading initial page numbers must be made in the input specifications. It is accomplished as follows:

1. Input Specifications

    a. Define a separate input-only card type--just as for a constants card (see section immediately above). (Alternatively, include PAGE data in a constants heading card.)

    b. Assign the field name PAGE to a four-column card field.

    c. Define the field as numeric without decimal places (0 in col. 52).

2. PAGE (i.e., Consecutive-Number) Card

    a. Punch a value one less than the desired starting number into the pertinent four-column field of a card, together with the appropriate card-type identification punches. (A positive or negative value is permitted, and will be incremented arithmetically.)

    b. Place the card ahead of the data deck.

    For multiple input (or combined) files, or to restart numbering at numbers higher than 1 at several points of the data deck, place consecutive-number cards as explained for constants (heading) cards.

    PAGE cards may also be inserted in the data deck, even though numbering is to begin with 1, if numbering is to restart with 1 at various points in the report that cannot be conveniently identified by conditioning indicators in the output specifications (i.e., if this is required at points in the data deck that cannot be recognized by the program by such occurrences as a control-level break or a certain card type, etc.). The contents of the consecutive-number field should then be left blank or punched with zeros, so that the starting number is 1.

Figure 25 shows field description entries discussed so far (and a few incidental pointers). The example is rather artificial so that each entry chosen can illustrate at least one of the foregoing points.

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS

IBM System/360

Form X24-3350-1
Printed in U. S. A.

Date ________

Program ________

Programmer ________

Punching Instruction	Graphic							
	Punch							

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

Line (3-5)	Form Type (6)	Filename (7-14)	Sequence (15-16)	Number (1-N) (17)	Option (O) (18)	Resulting Indicator (19-20)	Record Identification Codes 1: Position (21-24)	Not (N) (25)	C/Z/D (26)	Character (27)	2: Position (28-31)	Not (N) (32)	C/Z/D (33)	Character (34)	3: Position (35-38)	Not (N) (39)	C/Z/D (40)	Character (41)	Stacker Select (42)	Packed (P) (43)	Field Location From (44-47)	To (48-51)	Decimal Positions (52)	Field Name (53-58)	Control Level (L1-L9) (59-60)	Matching Fields or Chaining Fields (61-62)	Field-Record Relation (63-64)	Field Indicators Plus (65-66)	Minus (67-68)	Zero or Blank (69-70)	Sterling Sign Position (71-74)
01Ø	I	FIELDDES	DA			98	8Ø		C	-									2												
02Ø	I																				1	5	Ø	@DATE							
03Ø	I		PG			99	8Ø		C	&									2												
04Ø	I																				Ø1	Ø4	Ø	PAGE							
05Ø	I		Ø1	1		1Ø	8Ø		C	1																					
06Ø	I																				21	4Ø		NAME							
07Ø	I																				2	6	Ø	EMPL#1							
08Ø	I																				2	6		EMPL#2	L1						
09Ø	I																				41	55	9	MAX15							
10Ø	I																				62	64	3	DEC3MX		M1					
11Ø	I																			P	65	72	8	MAX8PA							
12Ø	I																				65	72		CNTRPA	L2						
13Ø	I		Ø5	1		15	8Ø	N	C	1																					
14Ø	I																				75	79		EMPL#2	L1						
15Ø	I																				65	72		DIFNAM	L2						
16Ø	I																				62	64		DECNO		M1					
17Ø	I																				41	55	9	MX15AL							
	I																														
Ø25	I																				11	5Ø		RPTNAM							
	I																														

Card Electro Number ________

Figure 25. Field Descriptions--Part I

Explanation of Figure 25

Incidentals:

<u>Cols. 3-5 (Line Number)</u>. To illustrate the optional treatment of col. 5, zeros were entered instead of leaving it blank. The last line entered (025) illustrates how to handle an insertion: the field RPTNAM (cols. 11-50) in the first heading card had been forgotten. The specification is assigned any line number between 020 and 030. After the specification cards have been punched, the card with line number 025 is placed behind that for line 020. This method obviates copying and shifting the entries for an entire page.

<u>Lines 010 and 030</u> include a specification to select the Date and PAGE heading cards to stacker 2; other cards enter the normal stacker.

<u>Line 130</u> exemplifies the least number of Record Identification Codes specifications to make all four card types mutually exclusive: "Not Character 1" distinguishes the fourth card type from the third; a specification for absence of (-) and (&) in col. 80 of the fourth card is not needed since the program always tests first for cards of the first and second type, because they have alphabetic Sequence codes (see <u>Nature of the Card-Type Sequence Check</u>).

Field Descriptions

<u>Lines 010, 020, and 025</u> illustrate how to enter constants (e.g., date and report heading) via a card defined as a separate card type. Wherever a card of that type appears in the data deck--at the front or interspersed--new date and report-heading data from that card supersede the previous contents of the core storage areas for @DATE and RPTNAM.

Line 020 also illustrates that alphabetic characters, required in the first column of Field Name, include three special characters (@ is one of these three).

Date contains no decimal places; but it is defined as numeric (0 in col. 52 = numeric, with 0 decimal places), so that the printout can be edited.

Line C25 assigns forty positions (cols. 11-50) to the report heading (RPTNAM). Entering the report heading via an input card overcomes the limitation of twenty-four positions maximum per constant in the output specifications, and allows insertion of new report headings at any desired point in the data deck. RPTNAM is defined as alphameric (col. 52 left blank); therefore, it cannot be edited in the output-format specifications--any edit symbols to be printed, such as a slash or decimal point, must be contained in the data in cols. 11-50.

Lines 030 and 040 show how to provide for loading of initial "page" number, if automatic numbering is to be specified (in the output-format specifications) but is not to start with 1 (or is to restart with 1 at points in the data deck that cannot be identified by conditioning indicators). Wherever a card of the type defined here (12-punch in col. 80) appears in the data-card deck, the number in cols. 1-4 (or in whatever columns are specified in Field Location) becomes the (new) starting number. (The number is incremented before it is printed or punched. Therefore, the number entered should be one less than the desired starting number.) Unless and until a PAGE card is read, consecutive-numbering (if called for in the output-format specifications) begins with 1.

The Field Name PAGE must be used, four columns must be assigned to the field, and the field must be defined as numeric without decimal places (0 in col. 52).

Line 040 also shows that leading zeros for "From" and "To" column numbers (note 01, 04 in cols. 46-47 and 50-51) may be recorded. Other specification lines illustrate that they need not be recorded. All field-description lines show that source-data columns are entered right-justified.

Line 060 points out that the size of alphameric input fields (blank in col. 52) is not limited. 20 columns were assigned. It also illustrates that fields need not be defined in the order in which they appear in the source-data cards: cols. 21-40 are recorded ahead of cols. 2-6.

Lines 070 and 080 show a field (input cols. 2-6) assigned two different names. EMPL#1 is numeric (with zero decimal places) so that it can be used in numeric compare, arithmetic, and/or editing operations--for example, to suppress leading zeros in printout. EMPL#2 is alphameric (col. 52 is blank), so that Control Level (L1 in cols. 59-60) will compare on the full characters in the field, including zone overpunches. If the L1 were placed next to EMPL#1, only the numeric parts of characters in cols. 2-6 would be considered in the Control-Level comparisons.

These field names also illustrate that numeric entries are allowed in Field Name, except in col. 53, and that # is not a special character (it is one of the three symbols defined as alphabetic).

Line 090 illustrates the maximum size of a standard (unpacked) numeric field (15 columns), and the maximum number of decimal places (9) allowed. The entry in col. 52 defines the field as numeric and implies, for numeric compare and arithmetic operations, that the data in cols. 41-46 is to the left of the decimal point, and that in cols. 47-55 to the right.

Line 100 emphasizes that the number of decimal places specified must not be greater than the digit capacity of the field: the field is unpacked (no P in col. 43), three columns long (cols. 62-64); therefore, it cannot have more than three decimal places specified.

Line 110 shows the maximum size (eight columns) for a packed (P in col. 43) input field. The entry 9 in col. 52 is valid--it does not exceed the digit capacity of the field--because an 8-column packed field contains 15 digit positions (2n-1 = 2 x 8-1 = 15). Nine decimal places implies that the contents of cols. 65-67 are to the left of the hypothetical decimal point, and cols. 68-72 to its right (a half-byte in col. 72 represents the sign). The field is defined by its actual card columns (cols. 65-72)--not by the number of digits it contains (15).

When Packed (P in col. 43) is specified for a field name, it cannot be used for Control-Level or Matching-Fields operations.

Line 120 shows assignment of a different name to the same source field (cols. 65-72) --CNTRPA versus MAX8PA--with the first entry (line 110) defined as packed and the second as alphameric (col. 52 blank). This illustrates how control (L2 in cols. 59-60) may be maintained (by the entries in line 120) on the entire EBCDIC characters in a packed field; while the entries in line 110 permit use of the same packed input field in arithmetic and/or editing operations and for easily legible (unpacked) printout.

Line 140 shows the assignment of the same field name to different source fields (cols. 2-6 versus cols. 75-79) in two different card types (see line 080). When the name is the same, the field size and data format must be identical (col. 52 must be coded identically in all input lines where the field name appears). Note that, when either data-card type is read, the data from the new card replaces the previous data stored for EMPL#2--the same core storage area applies to both card types. Only one core storage area is assigned to data for one field name, regardless of how often that field name appears in the specifications.

Line 150 uses a different name in the second card type for the same source field shown in line 120 for the first card type. Data in the storage locations for DIFNAM and CNTRPA is thus conserved until another card of the same type is read.

As shown in lines 080 and 140, and 120 and 150, it is immaterial for the Control-Level operation whether the field name is the same or different for the same Control Level (cols. 59-60). This is also true for Matching Fields (cols. 61-62). However, a Control-Level field or a Matching Field of a given level (here, L2) must always be the same length in all card types--in this example, it is 8 columns long in both card types (line 120 and line 150).

Lines 160 and 100 have a Matching-Fields specification (M1 in cols. 61-62). Different field names are assigned to equivalent source fields in the two card types. This permits difference in format: Line 100 specifies the field as numeric, with 3 decimal places; line 160 defines the field as alphameric.

Line 160 is presented to emphasize that, notwithstanding the different field names in the two card types, certain restrictions exist when Control Level or Matching Fields is specified:

1. The field length must be the same. It is three columns in both card types.

2. Once a Control Level or Matching-Fields level has been defined as numeric in one specification, all Control-Level or Matching-Fields operations, respectively, for that level ignore zone punches.

Therefore, although line 160 is blank in col. 52 (i.e., the field is defined as alphameric), the sequence check (or matching of cards, if the different card types were in different files) is performed on the numeric portion of the field only--since DEC3MX in line 100 is defined as numeric (col. 52 is coded). For other uses of these fields (not Control-Level or Matching-Fields operations), the format conforms to the differing specifications in col. 52--DECNO data is treated as alphameric; DEC3MX as numeric, with 3 decimal places.

Line 170 illustrates that the same source-data field in two card types (MX15AL in line 170 and MAX15 in line 090) may be specified with a different number of decimal places (8 and 9, respectively), provided the fields are assigned different names.

Note: As discussed with columns 59-60, Control-Level fields must be recorded in ascending sequence of significance within card type: L1 must appear in an earlier specification line than L2, etc. See lines 080 and 120, and 140 and 150. Note particularly lines 140 and 150 where the fields had to be specified in a sequence different from that in which they appear in the source-data cards--DIFNAM is in data-card columns ahead of EMPL#2, but had to be defined on a lower line because L2 is higher than L1.

Control Level--Cols. 59-60

Any of the indicators L1-L9 may be entered in these columns. This establishes the field defined in that specification line as a control field (as the term is known in Unit Record parlance--see also Definition of Terms), and designates that L-indicator as a resulting indicator. Nine distinct control and total levels (besides LR for final total) are thus available--L9 is the highest level, L1 the lowest.

Whenever a card with an L-indicator specified in cols. 59-60 is read, the data in the card columns defined in that specification line (in cols. 46-47 and 50-51) is compared with that stored from the last card with the same L-indicator specified. If the data differs, the L-indicator specified in cols. 59-60 turns on; all L-indicators of lower number also turn on. These indicators turn on just before total-time processing for the new card (i.e., after the previous card has been completely processed), and are set off by the program after detail-time processing of the new card (see Figure 6, RPG Program Logic). The L-indicators are thus available to condition calculations and output at total time following the last card of a control group and/or to condition detail-time calculations and output for the first card of a control group. (See also references to Control Levels under Decimal Positions, Col. 52.)

Normally, L-indicators are used to:

1. Condition certain calculations to be performed only at the end of a control group

2. Condition certain punching to be performed only for group totals of particular levels (summary punching)

3. Condition certain printing to take place only at the end of control groups of particular levels (total printing)

4. Conditioning certain calculations and/or output operations to occur only on the first card of a control group of a particular level (e.g., group indication).

See the Calculation Specifications and the Output-Format Specifications for application of the L-indicators as conditioning indicators.

Note: No automatic decimal alignment is performed in Control-Level operations on numeric fields.

Split Control Fields

Control fields may be split; i.e., one Control Level may be assigned to two or more areas of the same input card. The program then combines the data, from the several sets of columns with the same L-indicator assigned, into one continuous control field--in the order in which the portions appear in the input specifications. Thus, the portion (subfield) of a split control field recorded first is stored in the Control-Level data storage area to the left of the portion in the next specification line.

Special rules for split Control-Level fields:

1. a. The length of the portions of split control fields may vary for different card types (if the field names differ), and

   b.* A field may be split for some card types and not for others (if the field names differ), but:

   the aggregate number of columns for one Control Level must be uniform for all card types.

2.* The aggregate number of columns of all portions (subfields) of one split numeric control field may exceed 15 columns--provided:

   a. No individual portion (subfield) exceeds 15 columns, and

   b. The sum of all control fields does not exceed 144 columns (with each L level specified counted once)

3. If one portion of a split Control-Level field is defined as numeric (i.e., col. 52 has an entry), the entire field is treated as numeric (zones stripped) for Control-Level operations in all card types.

4. No other Control-Level entry may intervene in the input specifications between the several specification lines for portions of one control level. (For compatibility with other RPGs, no other field-description lines should intervene, either.)

*Note: Figure 26B illustrates that several numeric data fields--each not longer than 15 columns--may be portions of a single Control-Level field longer than 15 columns. It also shows that the same Control Level may be assigned in another card type to a single non-split field longer than 15 columns, provided it is defined there as alphameric and assigned a different name.

General Rules for Control Fields

1. If several Control Levels are specified (in cols. 59-60) for one card type they must be recorded in the input specifications in ascending sequence of level: the specification line with L1 must precede the line with L2, etc. This may require specifying input fields in a sequence that differs from the order in which the data appears in the input-data cards.

   However, the specification lines for different Control Levels need not be consecutive--lines for other fields, without Control-Level specifications, may intervene.

2. The number of columns (i.e., the field size) that constitute a Control-Level field must be uniform in all card types where that Control Level is specified.

3. The card columns for control fields of different Control Levels in the same card type may overlap; but the aggregate number of columns for all Control Levels must not exceed 144 (with each L level specified counted once).

4. There is no requirement that, if a certain Control Level higher than L1 is assigned, all lower-numbered levels must also be assigned.

Note: Additional rules apply to Control Levels used in conjunction with Field-Record Relation (cols. 63-64), and are discussed in that section.

Control-Level resulting indicator specifications in cols. 59-60 were already illustrated and explained in Figures 5, 11, 13, and 25; additional examples follow discussion of Field-Record Relation. Aspects of L-indicator operations were fully explained in the sections Program Logic Flow (and Figure 6), Indicators, Packed (Input Specifications, col. 43), Decimal Positions (Input Specifications, col. 52), and Field Name (Input Specifications, cols. 53-58).

To refresh the reader's memory, some points are repeated here in condensed form:

1. Control operation for a given Control Level is on a numeric basis (all zones stripped) for all card types if any control field or Split-Control portion for that Control Level is defined as numeric (i.e., if col. 52 has an entry)--even though the field names may differ. (But consider defining the same field twice for the same card type, with different names--as discussed previously.)

2. Field names are ignored by Control-Level operations--contents of specified data-card columns are compared with data stored from a previous card at the location assigned to that Control Level. Therefore, field names for the same Control Level in different card types may be the same or different.

3. A Control-Level specification cannot be assigned to an input field defined as Packed (P in col. 43). (But consider defining the same field twice for the same card type, with different names--as discussed previously.)

4. A Control-Level field defined as numeric is limited to a maximum of 15 columns. (See special case under Split Control Fields, above.)

5. The same or different Control Levels may be assigned to different card types; or none may be assigned to some card types.

   Comparing on control fields occurs only for the card types and fields with Control Level specified. When a card for which a given Control Level is not specified is processed, the data for that Control Level in storage from a previous card remains undisturbed.

6. Control-Level compare operations are performed for cards in the order in which they are processed, regardless of the file from which they come.

Note: While Control-Level indicators may be equated in purpose with control breaks on Unit Record accounting machines, the two operations are quite different. No automatic "control break", with its attendant total-print and group-indicate cycles, occurs on Model 20. Instead, indicators are made available to perform any desired operations at the end of a control group and at the beginning of a new one.

Matching Fields--Cols. 61-62

Any of the codes M1, M2, or M3 may be entered in these columns, with these effects:

1. If the program provides for processing of only a single input (or combined) file, entry of M1, M2, or M3 in cols. 61-62 causes sequence checking of the contents of the field(s) defined in the particular specification line(s).

   However, programming a sequence check by entries in the calculation specifications usually consumes less core storage space than utilizing the Matching-Fields entry for that purpose alone. Sequence checking by calculation specifications also permits detection of duplicates, as well as saving processing time.

2. If the program provides for the processing of two or three input (or combined) files, entry of M1, M2, or M3 causes

   a. sequence checking of the contents of the fields defined in specification lines in which M1, M2, or M3 is entered, and

   b. matching of the contents of these fields

      (i) between successive cards in the same file and

      (ii) between cards in the primary file and cards in the secondary and (if applicable) the tertiary file.

      This determines the order in which cards from the two or three input (or combined) files are processed.

      When a card from the primary file matches a card from the secondary or tertiary file on all Matching Fields specified, the MR indicator is on during the processing of these matched cards. The MR indicator is on for detail-time processing of a matching card through the total time and overflow time that follows the card (see RPG Program Logic, Figure 6). The status of the MR indicator may be used to condition the execution of calculation and/or output specifications. (See the Calculation Specifications and the Output-Format Specifications for applications of the MR indicator as conditioning indicator.)

One, two, or three fields may be matched and/or sequence checked in one operation. If more than one field is specified for matching and/or sequence checking, the M-levels must be assigned to correspond to the significance levels of the fields. For example: if three fields are involved, M3 is assigned to the most significant (highest-order) and M1 to the least significant (lowest-order) field. To put it another way: the contents of the three fields may be regarded as one continuous value, with the M3 value at the left and the M1 value at the right.

If only one Matching Field is used, it must be assigned M1; if two are used, M1 and M2 must be assigned to them. A Matching Field cannot be split within the same card; i.e., one Matching Field (M1, M2, or M3) must represent a single entry of contiguous card columns with the field read from left to right as high-order to low-order.

One matching field (M1, M2, or M3) must take up a contiguous number of card columns and cannot be split up within the card. Up to three matching fields can be specified per card, and, in addition, the card columns of one field may overlap those of another. See Figure 25A.

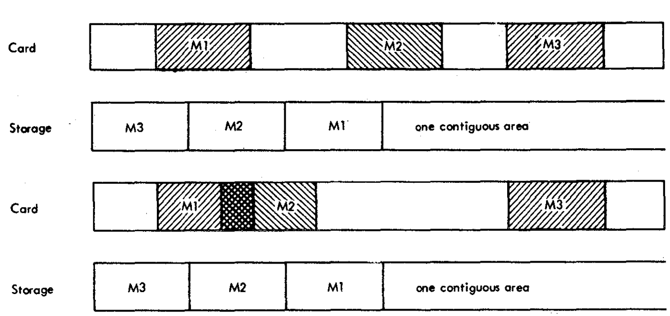


Figure 25A. Specification of Matching Fields

Note: No automatic decimal alignment is performed in Matching-Fields operations on numeric fields.

Matching-Fields specifications were already illustrated and discussed in Figures 13 and 25, and the MR indicator in Figures 11, 12A, and 13. Aspects of Matching Fields and MR-indicator operations are fully explained in the sections Program Logic Flow (and Figure 6), Indicators, File Description Specifications, Packed (Input Specifications, col. 43), Decimal Positions (Input specifications, col. 52), Field Name (Input Specifications, cols. 53-58), and Matching of Files.

To refresh the reader's memory, some points are repeated here in condensed form:

1. With multiple input (or combined) files, at least one card type in each file must have an entry in Matching Fields, and sequence checking is mandatory for card types with Matching Fields specifications. A sequence error stops the program. (It can be restarted.)

2. When Matching Fields are used, card types with Matching Fields specified must be in the same sequence in all files--ascending or descending. (The direction of sequence is designated in the File Description Specifications.)

3. Comparing on Matching Fields occurs only for card types with Matching Fields specified. Processing of card types without Matching Fields specified does not disturb the Matching-Fields data stored from a previous card. (The MR indicator is off during the processing--detail time through next overflow time--of a card type for which Matching Fields is not specified.)

4. Card types for which Matching Fields are specified must all have the same number of Matching Fields specified.

5. The number of columns (i.e., the field size) that constitutes a Matching Field of a given level (M1, M2, or M3) must be uniform for all card types with Matching Fields specified.

6. The card columns for Matching Fields of different levels (M1, M2, M3) in the same card type may overlap; but the aggregate number of columns for all Matching Fields in one card type must not exceed 144.

7. An input field defined as Packed (P in col. 43) cannot be assigned as Matching Field. (But consider defining the same field twice for the same card type,

with different names--as discussed previously.)

8. Matching-Fields operation for a given level (M1, M2, or M3) is on a numeric basis (all zones stripped) for all card types if any Matching Field of that level is defined as numeric (i.e., if col. 52 has an entry)--even though the field names may differ. (But consider defining the same field twice for the same card type, with different names--as discussed previously.)

9. A Matching Field defined as numeric is limited to a maximum of 15 columns.

10. Field names are ignored by Matching-Field operations--contents of specified data-card columns are compared with data stored from other cards at locations assigned by the program to Matching-Fields data. Therefore, field names for the same Matching-Fields level in different card types may be the same or different.

11. The order in which input (or combined) files are entered in the input specifications determines their order of precedence when matching two or three files.

12. Data from cards with higher precedence can be available when processing matching cards of lower precedence, but not vice versa.

13. The order in which specification lines for a card type with Matching Fields are entered need not conform to the level of the Matching-Fields specifications--e.g., the line with M3 in cols. 61-62 could fall between the lines with M1 and M2.

    Also, specification lines without Matching-Fields entry in cols. 61-62 may intervene between lines with Matching-Fields entry.

14. Matching-Fields (M1, M2, M3) operations are independent of Control-Level (L1-L9) operations. (However, they are related to the extent that control-field comparisons are only performed when pertinent cards are processed--and that, in turn, is based on the Matching-Fields operation.)

Note: Additional rules apply to Matching Fields used in conjunction with Field-Record Relation (cols. 63-64), and are discussed in that--the next--section.

Field-Record Relation--Cols. 63-64

These columns are used in conjunction with records in an OR relationship (see Records in an OR Relationship). Entries in cols. 63-64 permit associating fields only with a particular one of several card types in an OR relationship. The distinction is made by entering in cols. 63-64 the code of one of the Resulting Indicators assigned in cols. 19-20 to the several card types in an OR relationship.

Field-Record Relation can be used when two or more card types differ in their Record Identification Codes (cols. 23-41) but a majority of their input fields are in the same columns, have the same format, and identical field names apply--and these fields are described only once. However, some of the input fields differ in field name, location of source columns, and/or format, and/or size--and each different field requires a separate field description line. (The field name may be the same, provided the sole difference for that field between the card types is in location of source columns; otherwise the name must also be different.)

Different card-type Resulting Indicators are assigned in cols. 19-20 to some or all of the several card types in the OR relationship. By entry of the card-type Resulting Indicator for the appropriate card type in Field-Record Relation (cols. 63-64), a field description is associated only with a particular one of the card types in an OR relationship. Field descriptions without an entry in Field-Record Relation apply to all the card types in the OR relationship. This saves entering specification lines and, if the proportion of identical fields preponderates, it also saves core storage space.

Control-Level fields and/or Matching Fields may also be associated with particular card types in an OR relationship by entry of the relevant card-type Resulting Indicator in Field-Record Relation. An entry for the same L- or M-indicator (cols. 59-60 or 61-62, respectively) without a Field-Record Relation entry then applies only when none of the pertinent card-type Resulting Indicators, in Field-Record Relation for the L- or M-indicator, is on.

Note: The core storage saved by the single entry of field description lines that apply to all the card types in an OR relationship must be weighed against the core storage cost to the object program in having to test indicators for the field descriptions that differ for the separate card types. Half the fields common to all card types in the OR relationship may be used as a rule of thumb for the break-even point.

Special Rules for Use of Field-Record Relation (cols. 63-64)

For each set of card types in an OR relationship:

1. Core storage space is conserved by grouping together (i.e., in consecutive lines) all field descriptions with the same indicator in Field-Record Relation (cols. 63-64), and by grouping together all field descriptions without Field-Record Relation indicator.

2. When the same Control Level (L1-L9--cols. 59-60) or Matching-Fields level (M1-M3--cols. 61-62) is assigned both to field description without Field-Record Relation indicator, and to one or more field descriptions with Field-Record Relation indicator(s), the Control-Level and Matching-Fields entries without Field-Record Relation indicator must appear first.

3. In view of (1) and (2) above, all the field-description lines without Field-Record Relation entries should appear before those with Field-Record Relation.

4. The program treats split control fields (see Control Level, cols. 59-60) of one Control Level as a single entity, for purposes of Control-Level operations.

Therefore, it is not possible (for Control-Level operations) to assign different field portions of a single Control Level to different Field-Record Relation indicators, or to assign a portion to a Field-Record Relation indicator and have another portion in a field-description line without Field-Record Relation indicator.

The same result is easily achieved by repeating all portions of the split control field--even that which might apply regardless of OR-relation card type--for all pertinent Field-Record Relation field-description lines. (Figure 26B, lines 06 and 17, illustrates this--see explanation in point 3, under Explanation of Entries in Figure 26B.)

5. When the same Control Level or Matching-Field level is assigned both to field description without Field-Record Relation indicator, and to one or more field descriptions with Field-Record Relation indicator(s), only the specification with the pertinent Field-Record Relation indicator is used--for Control-Level and/or Matching-Fields operations--when that indicator is turned on. If none of the Field-Record Relation indicators for that Control Level or Matching-Fields level is on, the specification without Field-Record Relation indicator assigned is used for that level.

   Control-Level and Matching-Fields specifications to which no Field-Record Relations indicator is assigned for any of the card types in the OR relationship are used with all card types in the OR relationship.

6. The number of Matching Fields specified (one, two, or three) must be uniform for all card types for which Matching Fields are specified.

   It is not allowed, therefore, to match (or sequence check by entry in cols. 61-62) on a different number of fields for different card types in an OR relationship. It is, however, permissible to match (or sequence-check by entry in cols. 61-62) on the appropriate number of fields for some card types--and not at all for others in the OR relationship. The latter implies that all field-description lines with Matching Fields specified contain also a Field-Record Relation indicator entry; otherwise--as explained in 5, above--a Matching-Fields line without Field-Record Relation indicator is applied whenever no such indicator is on for that level. (See also Matching Fields, cols. 61-62, and Matching of Files.)

7. The number of Control Levels (L1-L9) specified for different card types in the OR relationship may differ. It is also permitted to have no Control Level for certain card types, and any number of Control Levels for other card types.

8. While--for Control-Level and Matching-Fields operations--entries with Field-Record Relation indicator assigned take precedence, when the relevant indicator is on, over those without an indicator entry in cols. 63-64, this is not true for other processing of the data in these fields:

   The data from the card field defined in every field-description line which has no Field-Record Relation entry is read from all card types in the OR relationship. This data is read into the core storage area assigned by the program to that field name, which is not the same area where Control-Level or Matching-Fields information--which ignores field name--is stored.

   If it is desired to read into the field-name storage area for a field only from certain card types in the OR relationship--or to read the same field from different card columns for the different card types--then each field-description line for that field must have an appropriate indicator entered in Field-Record Relation.

Figures 26A, B, and C illustrate input-specifications entries for Control Level, Matching Fields, Field-Record Relation, and other OR relationships. (See also Figure 25 and earlier figures.)

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS

IBM System/360

Form X24-3350-1
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	Filename	Sequence	Number (1-N)	Option (O)	Resulting Indicator	Record Identification Codes 1: Position	Not (N)	C/Z/D	Character	2: Position	Not (N)	C/Z/D	Character	3: Position	Not (N)	C/Z/D	Character	Stacker Select	Packed (P)	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field-Record Relation	Field Indicators Plus	Field Indicators Minus	Field Indicators Zero or Blank	Sterling Sign Position
01	1	TYPE1OR2	AB			8∅	1		C	A	74		D																		
02	1	O	R				1		C	B	74		D						2												
03	1	O	R				1		C	C	74		D						3												
04	1	O	R			85	74		D	8									3												
05	1	O	R			88	74		D	9																					
06	1																				5	10	∅	EMPLNO	L1	M1					
07	1																				11	30		NAME							
08	1																				8∅	8∅		DIVISN		M3					
09	1																				31	31		SHIFT							
10	1																				2	5		DEPT	L3	M2					
11	1																				75	78	2	PAYRAT							
12	1																				79	8∅		CORPOR	L4						
13	1	SECNFILE	AC			89	1		C	A	74	N	D	8	74	N	D	9													
14	1	A	ND				74		Z	K																					
15	1																				5	10	∅	EMPLNO	L1	M1					
16	1																				11	14		DEPT		M2					
17	1																				75	78	1	HRSWRK							
18	1																				8∅	8∅		DIVISN		M3					
19	1																				79	8∅		CORPOR	L4						
2∅	1																														

Card Electro Number ____________

Figure 26A. Field Descriptions--Part II

Explanation of Entries in Figure 26A--Control Levels, File Matching, and OR Relationship Types 1 and 2 (see Records in an OR Relationship)

Lines 01, 02, and 03 show three records in an OR relationship of Type 1 (see above): Resulting Indicator 80 applies to all three types (it could have been repeated in each line). No distinction can be made (on the basis of card type) in the calculation and/or output-format specifications between the three card types, because the same indicator is assigned to all three.

Lines 04 and 05 show two more records in an OR relationship to the first three; but they are of Type-2 OR relationship: separate Resulting Indicators are assigned to them, to permit calculation- and output-specifications distinction between the fourth card type, the fifth, and the first three (as a group).

Lines 02-04 also illustrate that card types in any kind of OR relationship--even when no distinguishing Resulting Indicator is assigned--may be directed to non-normal stackers by entries in the input specifications.

Lines 01-05 illustrate only OR relationships of Types 1 and 2: the data fields (lines 06-12) are defined only once, and none are limited to a particular one, or group, of the five card types in the OR relationship--Field-Record Relation (cols. 63-64) is therefore blank.

Lines 13 and 14 are another example of AND relationships between four criteria for the definition of one card type.

Lines 06 and 15 specify that Control Level L1 will be a numeric control only (all zones stripped), because col. 52 has an entry.

Lines 06, 10, and 12, and lines 15 and 19 show Control-Level indicators to be in ascending order of significance--as they must be, even if the data appears in the

IBM

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

Form X24-3350-1
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	Filename	Sequence	Number (1-N)	Option (O)	Resulting Indicator	Record Identification Codes 1: Position	Not (N)	C/Z/D	Character	2: Position	Not (N)	C/Z/D	Character	3: Position	Not (N)	C/Z/D	Character	Stacker Select	Packed (P)	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field-Record Relation	Field Indicators Plus	Minus	Zero or Blank	Sterling Sign Position
01	I	TYPE3OR	BC			91	80		C	1																					
02	I	OR				92	80		C	2																					
03	I	OR				93	80		C	3																					
04	I	OR				94													2												
05	I																				1	5		FLDA1	L1						
06	I																				46	50		FLDA2	L1						
07	I																				76	77		DAY							
08	I																				11	13		FLDB1	L2						
09	I																				51	60	2	FLDC1	L3						
10	I																				31	40	0	FLDC2	L3						
11	I																				71	75		FLDA3	L1	M2	91				
12	I																				26	27		FLDA4	L1		91				
13	I																				48	50		FLDA5	L1	M1	91				
14	I																				51	70		FLDC	L3		91				
15	I																				21	25		FLDD	L4		91				
16	I																				6	10		FLDA6	L1	M2	92				
17	I																				46	50		FLDA2	L1		92				
18	I																				74	75		MONTH			92				
19	I																				61	63		FLDB2	L2	M1	92				
20	I																				21	25		FLDD	L4		94				

Card Electro Number ____________

Figure 26B. Field Descriptions - Part III

cards in a different order (note lines 06 and 10).

Line 10 has Control-Level indicator L3 assigned, although L2 is nowhere assigned in this file. This is permissible. When L3 (or any higher indicator) turns on, L2 and L1 also turn on, even though L2 is not assigned.

SECNFILE does not have L3 assigned to any field, although it is assigned in the other file. No L3 Control-Level comparison is, therefore, made when a card is read from the file SECNFILE. The L3 information from the last preceding card from the file TYPE1OR2 is preserved, and compared against the next card processed from that file.

If L4 turns on when reading a card from SECNFILE, L3, L2, and L1 also turn on, even though L3 and L2 are not assigned in this file.

Lines 08 and 10 illustrate that the Matching-Fields specifications (M1, M2, M3 in cols. 61-62) need not appear in ascending order. Regardless of the order in which they are recorded, M3 identifies the high-order (most-significant) field, and M1 the low-order field. Thus, DIVISN contains the most-significant (leftmost) data for the card match and sequence check, DEPT the next part, and EMPLNC represents the right-hand portion.

The entire example also shows:

a. That the number of Matching Fields must be the same for all card types for which matching fields are specified (three fields in all cards of this example);

b. That Matching Fields need not be the same as Control-Level fields;

c. That other field-description lines may be placed between Control-Level and Matching-Fields lines.

Lines 08, 09, and 18 show how to specify Field Location (in cols. 46-47 and 50-51) for single-column fields.

Lines 08 and 12, and 18 and 19 show that fields for Control Level and Matching Fields may overlap. Lines 06 and 10 show that fields for different Control Levels and for different Matching-Fields levels may overlap.

File relationship: File TYPE1OR2 is the primary file, because it is specified ahead of SECNFILE. When cards from the two files match on the data in the three fields specified as Matching Fields, matched primary cards are processed ahead of matched secondaries.

Therefore, when processing a card from SECNFILE, data from the last preceding card from the file labeled TYPE1OR2 can be utilized. For example, gross pay could be calculated by multiplying HRSWRK in each SECNFILE card by PAYRAT from the last preceding card from file TYPE1OR2. The MR indicator is on only for matched cards. Conditioning the calculation specification to be performed, at detail time, only if the indicators MR and 89 are both on, gross pay would be calculated only on cards from SECNFILE and only if the last card from the file TYPE1OR2 matched the SECNFILE card on DIVISN, DEPT, and EMPLNO. Gross pay could not be calculated during the processing of a card from the file TYPE1OR2, because all matching primary cards have completed processing before data (in this case, HRSWRK) becomes available from a matched secondary card.

To illustrate the point that data for fields with different Field Names is stored at different locations, regardless of source columns, PAYRAT and HRSWRK were assigned the same card columns in different card types.

Explanation of Entries in Figure 26B--Split Control Fields, Selective Sequence Check, and OR Relationship Type 3

Lines 01-04 provide for identifying each of the four different card types in an OR relationship, and assigning a different Resulting Indicator to each. Indicator 94 is assigned to cards that do not satisfy the criteria for indicators 91-93 (i.e., not 1, 2, or 3 in col. 80); these cards are selected to stacker 2. No Record Identification Codes are needed in line 04, because lines in an OR relationship are tested in sequence; therefore, card-type Resulting Indicator 94 turns on only if the card does not meet the criterion in line 01, 02, or 03.

Figure 26C itemizes the card columns from which Control-Level data will be taken for each of the four card types in the OR relationship. The following points are noteworthy in Figures 26B and C with regard to Control Level:

1. When neither indicator 91 nor 92 is on, L1 Control Level is based on the L1 entries with no Field-Record Relation specified (lines 05 and 06). When indicator 91 or 92 is on, L1 Control-Level data is based on the entries in lines 11-13 or 16-17, respectively--lines 05 and 06 are then ignored for Control-Level data.

   Similarly, L2 Control Level is based on the entries in line 19 (data-card cols. 61-63) when indicator 92 is on (i.e., the second type of card was read); otherwise, it is based on the entries in line 08 (cols. 11-13). Likewise, L3 Control Level is based on line 14 (data-card columns 51-70) when indicator 91 is on (first type of card); otherwise it is based on lines 09 and 10 (cols. 51-60 and 31-40).

Card-Type Resulting Indic. ON	Card Columns Read for Control-Level Operations						
	L4	L3		L2	L1		
91	21-25	51-70		11-13	71-75	26-27	48-50
92	No L4 control	51-60	31-40	61-63	6-10		46-50
93	No L4 control	51-60	31-40	11-13	1-5		46-50
94	21-25	51-60	31-40	11-13	1-5		46-50

Figure 26C. Card Columns from which Control Fields will be Taken when One of the Card Types Defined in Figure 26B is Read

2. Control Level L4 is operative only when indicator 91 or 94 is on, because L4 is not specified at all without a Field-Record Relation. Card types to which Resulting Indicator 92 or 93 is assigned therefore do not turn on indicator L4. The L4 Control-Level data is preserved from the last preceding card with indicator 91 or 94, to be compared against the L4 data in the next card of type 91 or 94.

3. The L1 Control Level is split into three fields for cards with indicator 91, and into two fields for the other card types. Note that, in all three cases, the total length for L1 is uniform (ten columns): the aggregate length of fields for a split Control Level must be uniform for all card types.

   In lines 06 and 17 the field entries for the low-order portion of the split L1 Control Level are identical (FLDA2, source cols. 46-50). Nonetheless, the field description had to be repeated for Field-Record Relation indicator 92, because the other portion of L1 differs between card type 92 and others (source cols. 6-10 versus 1-5, and Field Name FLDA6 versus FLDA1): part of a split control field cannot be conditioned by a Field-Record Relation indicator unless all parts are so conditioned, even if this means repetition of an identical entry.

4. In lines 09 and 10, the fields to which Control Level L3 is assigned are defined as numeric (col. 52 has entries). Each of the two portions (subfields) is within the limit of 15 columns for a numeric field, although the aggregate length of the L3 control field exceeds 15 columns--it adds up to 20 columns. This is permissible, so long as no individual numeric subfield exceeds 15 columns.

   In line 14, the same L3 Control Level is not split when card-type Resulting Indicator 91 is on. To be uniform for all card types, it must be 20 columns long--which exceeds the 15-column limit for a numeric field. Note that FLDC in line 14 is defined as alphameric (col. 52 is blank); it may therefore legitimately exceed 15 columns in length.

   It is permissible to designate differently named fields for the same Control Level in different formats (i.e., with different Decimal Positions specification). For processing of the data in the fields, format accords with the specification in col. 52; for Control-Level operations, compare is purely numeric (zones stripped) if one of the fields or split portions for that Control Level is defined as numeric. L3 is, therefore, a numeric control field.

5. Control-Level entries must be in ascending order of significance (i.e., L1 appears in an earlier line than L2, etc.) within Field-Record Relation group, and within the group without Field-Record Relation specifications.

6. The Control-Level entries without Field-Record Relation specifications must appear ahead of those conditioned by Field-Record Relation.

7. Lines without Control-Level specification may appear between those with different Control-Level specifications, but (to be compatible with other RPGs) not between entries for the same split Control Level.

<u>Lines 05-10, 11-15, 16-19, and 20</u> illustrate that field-description lines should be grouped by Field-Record Relation indicator, to minimize core storage requirements.

<u>Lines 11 and 13, and 16 and 19</u> contain Matching Fields specifications (in cols. 61-62). The contents of fields FLDA3 and FLDA5 in card type 91, and FLDA6 and FLDB2 in card type 92 will be sequence checked. Card types 93 and 94 are not sequence checked, because M1 and M2 are not specified with Field-Record Relation 93 or 94; nor are they specified without any Field-Record Relation. (If M1 and M2 were also specified in lines without a Field-Record Relation entry, the fields in these lines would be sequence checked whenever neither indicator 91 nor 92 is on.)

Note that, for card types for which Matching Fields are specified, the same number of fields must be specified for matching, and the field size for each M-level must be the same in all such cards.

<u>Data-field specifications</u> are not affected by Field-Record Relation in the same manner as Control Level or Matching Fields:

As pointed out above, whenever a Control-Level or Matching-Fields specification appears in the same line as a Field-Record Relation indicator, only the Control-Level or Matching-Fields specification in that line applies for that level--even if the same Control-Level or Matching-Fields code is also specified in a line without Field-Record Relation.

However, the data for the field specified in a field-description line without Field-Record Relation entry is read into the core storage area assigned to that field, regardless of which card-type Resulting Indicator is on (for the group in an OR relationship). On the other hand, data for fields in lines with Field-Record Relation indicator are read into the storage area for the field only when that particular indicator is on.

Therefore, the data for fields in lines 11-15 is read only from cards with Resulting Indicator 91; that for lines 16-19 only when indicator 92 is on; and that for line 20 only from card type 94. But the storage areas for the fields defined in lines 5-10 receive new data from each of the four card types.

To illustrate by a few examples:

1. The storage area for the field named FLDA3 receives new data only from the card type to which Resulting Indicator 91 was assigned.

2. The storage area for the field named FLDD receives new data from every card of type 91 or 94.

3. The storage area for MONTH receives new data only from cards of type 92.

4. The storage area for the field named FLDA1 receives new data from every card (in the OR-relation group of card types).

5. The field named FLDA2 appears in line 06 without Field-Record Relation, and in line 17 with Field-Record Relation indicator 92, although the source columns (46-50) are identical. This was necessary because it is part of a split control field, and the other part of the split L1 Control Level is assigned to different source columns (cols. 6-10 versus cols. 1-5).

   When a card of type 92 is read, the data for the field named FLDA2 is stored twice in the same process area. Core storage space is saved by using the same field name. (Of course, if different source columns applied in lines 06 and 17, the data described on line 17 would be available for processing whenever indicator 92 is on--it would replace data in the field described in line 06.)

<u>Field Indicators--Cols. 65-70</u>

Any indicator code, except L0, may be placed in any of these three sets of two columns (cols. 65-66, 67-68, 69-70). The corresponding indicator is treated like a resulting indicator for the contents of the field described in that line: the indicator turns on if the contents of the field satisfy the criterion (Plus, Minus, or Zero/Blank, respectively) to which the particular indicator was assigned--otherwise it turns off. These indicators can then be used as conditioning indicators in calculation and/or output specifications: they can serve to condition the execution of a calculation or output specification to occur only when a particular input field was or was not positive, negative, or zero/blank, or when a particular status combination of several input fields obtains.

Assignment of Field Indicators to a numeric field causes the contents to become signed (hexadecimal C or D--see EBCDIC table, Appendix D, Figure D1) if the input field was unsigned (hexadecimal E or F). A -0 field becomes +0.

<u>NOTE</u>: To test a numeric field for Plus, Minus, or Zero/Blank, each column must contain a valid decimal digit or blank with or without sign; i.e., all entries must be represented in EBCDIC table rows 0-9 (see Appendix D, Figure D1).

Plus (Cols. 65-66)

Enter the code of the indicator that is to be turned on whenever the value of the associated input field is positive.

An input field is treated as positive if the punch combination in the low-order card column is represented in any of the columns of the EBCDIC table (see Appendix D, Figure D1) except D--but excluding EBCDIC 60--and provided all punch combinations in the field do not fall in row 0 of the EBCDIC table.

Expressed in terms of common usage: the field is treated as positive if the low-order position does not have an 11-overpunch, provided the numeric contents of the entire field are not zero or blank. (See special rules for packed input data, under <u>Packed</u>, Column 43.)

Columns 65-66 (Plus) may have an entry only for fields defined as numeric (0-9 in col. 52).

Minus (Cols. 67-68)

Enter the code of the indicator that is to be turned on whenever the value of the associated input field is negative.

An input field is treated as negative if the punch combination in the low-order card column is equivalent to EBCDIC 60 or is represented in column D of the EBCDIC table (Appendix D, Figure D1), provided all punch combinations in the field do not fall in row 0 of the EBCDIC table.

Expressed in terms of common usage: the field is treated as negative if the low-order position has an 11-overpunch, provided the numeric contents of the entire field are not zero or blank. (See special rules for packed input data, under Packed, col. 43.)

Columns 67-68 (Minus) may have an entry only for fields defined as numeric (0-9 in col. 52).

Zero or Blank (Cols. 69-70)--Field Defined as Alphameric (Col. 52 Blank)

Enter the code of the indicator that is to be turned on whenever the associated input field is completely blank. (Zeros, and 11- and 12-punches, are not treated as blanks.)

Zero or Blank (Cols. 69-70)--Field Defined as Numeric (0-9 in Col. 52)

Enter the code of the indicator that is to be turned on whenever the associated input field consists entirely of zeros and/or blank columns and/or zone punches.

Expressed broadly, the indicator assigned here is turned on if all punch combinations in the field are represented in row 0 of the EBCDIC table (Appendix D, Figure D1). (See special rules for packed input data, under Packed, col. 43.)

Note, for example, that a field of zeros with a 12- or 11-overpunch (in the low-order or any other position) turns on the indicator assigned here--not the indicator assigned to Plus or Minus.

Therefore, if the signs are in the high-order column of input fields, and that column could contain zero for its data portion, the signs should be tested by TESTZ in the calculation specifications--not by defining the high-order column as a separate input field and attempting to test for Plus or Minus.

If it is necessary to use the instruction TESTZ in the calculation specifications (to identify a sign in the high-order position of the field), or if it is desired to determine whether a field is blank (as distinct from zero)--yet the field is to be used in arithmetic operations--the field can be defined twice: once as alphameric (to be used for TESTZ or to test for blanks) and once as numeric (for arithmetic operations), with different field names.

Field Indicators are actually turned on or off--based on the status of the associated input field--just before detail-time calculations. (See RPG Program Logic, Figure 6.)

An input field may be assigned different indicators for two, or all three, of the conditions (Plus, Minus, Zero or Blank). When the program turns on the indicator for the condition that applies, it turns off (if they were on) the indicators assigned for that field to the conditions that do not apply. Thus, with the exceptions stated in "Points to Note" below, only one Field Indicator is on at one time for one field.

The same indicator may be assigned to more than one criterion for the same field --for example, to Plus and Zero. It is then turned on if either condition is satisfied.

Points to Note

1. The indicators normally used as Field Indicators are 01-99, H1, H2. Use of any others requires a complete grasp of the sections Program Logic Flow, Indicators, and Indicator Hierarchy, in the chapter Programming for RPG--General Information.

   Assignment of indicator H1 or H2 causes the program to halt after processing of the card for which H1 or H2 is turned on, unless the indicator is turned off by a programmer's instruction in the detail-time calculation specifications for that card. (It can only be turned off at detail time, because Field Indicators are not turned on until just before detail-time calculations, and the halt--if H1 or H2 is on--occurs shortly after detail-time output (see Figure 6, RPG Program Logic).)

2. Field-Description entries are associated with the particular card type defined above them in columns 19-41; the specifications in a field-description line are executed only when a card of that type has been read. Therefore, the status of a Field Indicator can change (apart from exceptions itemized here) only after a card of the pertinent type has been read. It may, therefore, remain on or off while cards of other types are being processed. Consequently, different field indicators assigned to fields in different card types may be on concurrently.

   On the other hand, if the same Field Indicator is assigned to fields in different card types, its status will be based on the contents of the relevant field in the last card of a pertinent type read.

3. If the same indicator is assigned to more than one field in the same card type, the last field entered with which the indicator is associated determines its status.

4. If the same field name is assigned to two different sets of source columns in the same card--once with Field Indicator and once without--the status of the indicator will correctly reflect the contents of the field with which it is associated. Only one core storage area is assigned to the field name; it will contain the contents of the field specified last with that name. (Of course, size and format must be uniform for both fields using the same name.)

   This is a technique for setting an indicator based on the contents of a field--when the field is not otherwise needed for calculations or output--without consuming any core space to store the data of that field.

5. The same indicator used as a Field Indicator in the input specifications may also be assigned as a Resulting Indicator, or specified to be SETON or SETOF, in the calculation specifications. This may change its status during the processing of a card.

6. Indicators assigned to Plus (cols. 65-66) or Minus (cols. 67-68) are off at the beginning of object-program execution. They do not turn on until the criterion is satisfied when a card of the pertinent type has been read.

   On the other hand, indicators 01-99 --but not H1 or H2--assigned to Zero or Blank (cols. 69-70) are on at the beginning (1P time) of program execution--before the first data card is read (see Figure 11, Hierarchy and Summary of Indicators). They remain on until a card of the pertinent type is read, and the field being tested for zero or blank does not satisfy that criterion. Thus, caution is called for in basing calculation or output operations solely on the status of such indicators.

   Exception: If the same indicator is used both as card-type Resulting Indicator and as Zero-or-Blank Field Indicator, it is not on at the beginning of program execution (1P time), because card-type indicators take precedence and are all off at the beginning (see Hierarchy and Summary of Indicators).

7. Change of value in a field during calculations or output does not in itself change the status of a Field Indicator set on the basis of the value in the field at time of input.

   However, if Blank-After (B in col. 39) is designated for a field in the output specifications, the field is set to blanks (if alphameric) or to zeros (if numeric) immediately after the data is moved to the output storage area (the data is then lost to any subsequent output operation). If a Field Indicator is assigned to Zero-or-Blank for that field in the input specifications, the indicator turns on at that point during output, regardless of its prior status (see also Blank-After under Program Logic Flow). It is then on during processing of additional output specification lines and until turned off when the field is tested again in the next input card of the pertinent type, and found not to satisfy the Zero-or-Blank condition.

   Note, however, that any Field Indicator assigned to Plus or Minus in the input specifications does not turn off (if it was on) when the indicator assigned to Zero-or-Blank for the same field is turned on by the Blank-After instruction.

8. If Blank-After (B in col. 39) is designated for a field in the output-format specifications, and more than one indicator has been assigned to that field to represent the condition Zero-or-Blank, only the first-assigned indicator is turned on by Blank-After. For example:

   a. An indicator (say, 25) is assigned to Zero-or-Blank for a field in Field Indicators of the input specifications; and

   b. Another indicator (say, 40) is assigned to Zero-or-Blank as Resulting Indicator, for the same field used as result field in the calculation specifications (arithmetic or TESTZ operation); then

   c. Blank-After turns on indicator 25--the first-assigned indicator--not indicator 40.

9. Assignment of Field Indicators causes an unsigned positive numeric-field value (EBCDIC-table column E or F) to become signed hexadecimal C.

Figure 27 illustrates assignment of Field Indicators.

INTERNATIONAL BUSINESS MACHINES CORPORATION

IBM REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS

IBM System/360

Form X24-3350-1
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	Filename	Sequence	Number (1-N)	Option (O)	Resulting Indicator	Record Identification Codes 1: Position	Not (N)	C/Z/D	Character	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Field Indicators Plus	Field Indicators Minus	Field Indicators Zero or Blank
01	I	PAYROLL	Ø1	N		25	6		C	5								
02	I										1	5	Ø	EMPLNO	L1			H1
03	I										11	30		NAME				1Ø
04	I										31	34	1	HRSWRK			Ø1	Ø2
05	I										7	1Ø	1	HRSWRK				Ø3
06	I										7	1Ø		HRWRKD				H2
07	I		Ø2	1		28	6		C	2								
08	I										6	1Ø	Ø	EMPLNO	L1			H1
09	I										1	5	2	BONDS		31	32	33
10	I										11	14	2	UNDUES		34		4Ø
11	I										15	18	2	BLUCRS		35		35
12	I										61	65	5	PERFRM			Ø3	
13	I																	
14	I																	

Figure 27. Field Indicators

Explanation of Entries in Figure 27

Specification line 02 shows how to turn on indicator H1 if cols. 1-5 of the data card are blank and/or contain zeros and/or zone punches only. Because the field is defined as numeric (col. 52 has an entry), no distinction is made in Field Indicators between blank, zero, and zone punches. (H-indicators assigned to Zero-or-Blank are not on at the beginning--see Figure 11.)

The H1 indicator may be used--like indicators 01-99--to condition calculation and output specifications; e.g., NH1 may be designated as a condition so that a particular specification is executed only when EMPLNO is not zeros (or blank). If EMPLNO is zeros (or blank) in a card, the system halts after that card has been processed, unless H1 is turned off during detail-time calculations by a programmer's specification.

Line 03 specifies that indicator 10 turns on if cols. 11-30 are blank--but not if they contain zeros, since the field is defined as alphameric. (Only Zero-or-Blank may contain an entry for alphameric fields.)

Line 04 causes indicator 01 to turn on if the field in cols. 31-34 is negative, and indicator 02 to turn on if it is zero (or blank). This illustrates assignment of different Field Indicators to two conditions in the same field.

Line 05 causes indicator 03 to turn on if cols. 7-10 contain zeros (or are blank).

Lines 04 and 05 also illustrate how to set indicators based on the status of a field (cols. 31-34) that is not needed for any other purpose, without tying up core storage space for it. The data in cols. 7-10 is to be used subsequently, and is stored at the location for HRSWRK. Note that the format must be uniform for the two fields that were assigned the same name.

Line 06 specifies indicator H2 if cols. 7-10 are blank--as distinct from zero. In line 05, an indicator (03) was specified if the same field is zero or blank. We have arbitrarily--to illustrate a point--made the assumption that hours worked may legitimately be zero; but that a blank field represents an error for which we wish to bypass processing (by using NH2 as conditioning indicator in calculations and output) and after which we want to halt. (Perhaps the card was missed by the keypunch operator.) To recognize the blank field, the field had to be specified as alphameric; to use the data in arithmetic operations, the field must be defined as numeric: hence, the field is described twice, with different names. The alphameric field also allows testing for high-order zone punches in the calculation specifications; these zone punches might be intended to identify special situations.

Lines 02 and 06 used up the available halt indicators. Therefore, although a blank NAME field represents an error, H1 or H2 cannot be used in line 03; another indicator (in this case 10) can be assigned, and used in the detail-time calculation specifications to turn on H1 or H2 if a halt is desired.

If H1 had been assigned to Blank for NAME, as well as to Zero-or-Blank for EMPLNO, then: when the NAME field is not blank, H1 is turned off even if EMPLNO is zero (or blank); i.e., the later test supersedes any earlier test for the same indicator.

Line 08 again assigns indicator H1 to zero (or blank) in the EMPLNO field. However, this line applies to a different card type. The status of this indicator is thus revised for each card; but the status of H1 does not conflict for two fields within the same card.

Line 09 shows assignment of three different indicators for the three possible states of values in one field.

Line 10 identifies positive values in the field by one indicator (34) and zeros (or blank) by another (40).

Line 11 assigns indicator 35 to cards with a positive value in cols. 15-18, or with zeros (or blank) in cols. 15-18. It will, therefore, be on if either condition is satisfied.

Line 12 illustrates use of the same indicator for different purposes in different cards (see line 05). Its status will, therefore, be revised for each card.

Points to be Especially Noted in Figure 27

1. Indicators 01-99 assigned to Zero-or-Blank (cols. 69-70) are on at the beginning of object-program execution. Thus, indicators 02, 03, 10, 33, 35, and 40 are on during heading-and-detail-time output preceding the reading of the first card.

   Indicators H1 and H2 are off at the start of program execution because H1 and H2 take precedence, in the indicator hierarchy (see Figure 11), over Zero-or-Blank indicators.

   The fact that indicators 01-99 assigned to Zero-or-Blank are on initially calls for caution in two respects:

   a. Detail-time output operations conditioned only by the ON status of any of these indicators will be executed before the first card has been read--i.e., during 1P time; and

   b. These indicators remain on until the first pertinent card has been read.

      For example, with the programming in Figure 27: If the first ten cards all happen to be type 25 (i.e., the first type listed, to which card-type Resulting Indicator 25 was assigned), indicators 33, 35, and 40 are on while these cards are processed--since no pertinent card (type 28) has yet been read to test the fields to which these indicators are assigned.

2. Once the status of Field Indicators has been determined for the fields in a card, the status is not revised until the next card of a pertinent type is read. (Exceptions: Blank-After, described below; H1 and H2; and changing the status of a Field Indicator by an entry in the calculation specifications.) For example:

   The status of the Field Indicators in lines 03 and 04 is revised only when a card with a card-type Resulting Indicator 25 has been read; the status of those in lines 09-11 only when a card with Resulting Indicator 28 has been read. This fact must be borne in mind when conditioning calculation or output specifications by these indicators.

   The H1 and H2 indicators are always reset by the program before a new card is processed (after restart by means of the CPU START key), if not set off before then by an instruction in the calculation specifications.

   Indicator 03 appears as Field Indicator in line 05 and line 12. Its status is therefore revised for each card.

3. Field Indicators for Zero-or-Blank (cols. 69-70) are turned on immediately when the corresponding field is set to blank or zero by entry of a B in col. 39 (Blank-After) in the output-format specifications. Any other Field Indicator previously set (for Plus or

Minus) for the field is not turned off thereby. For example:

If B is entered in col. 39 in the output-format specifications next to EMPLNO, the H1 indicator turns on as soon as the EMPLNO data has been transferred to the output area during an output operation involving EMPLNO (if H1 is on at the end of detail-output time, the system halts thereafter). Similarly, indicators assigned to Zero-or-Blank for other fields turn on during an output operation if Blank-After is specified for those fields.

In the case of BONDS, for instance, a Blank-After specification in conjunction with an output operation turns on indicator 33. This indicator remains on until the BONDS field is again tested after a card of type 28 has been read. Note, however, that--if indicator 31 (Plus) or 32 (Minus) was on for the BONDS field, it is not turned off by the Blank-After operation. Indicator 33 is then on concurrently with 31 or 32.

GENERAL INFORMATION

The calculation specifications (see Figure 28) particularize

1. The operations to be performed by the object program upon

   a. the input data, and

   b. data obtained as a result of other calculations

2. Look-up of data contained in tables

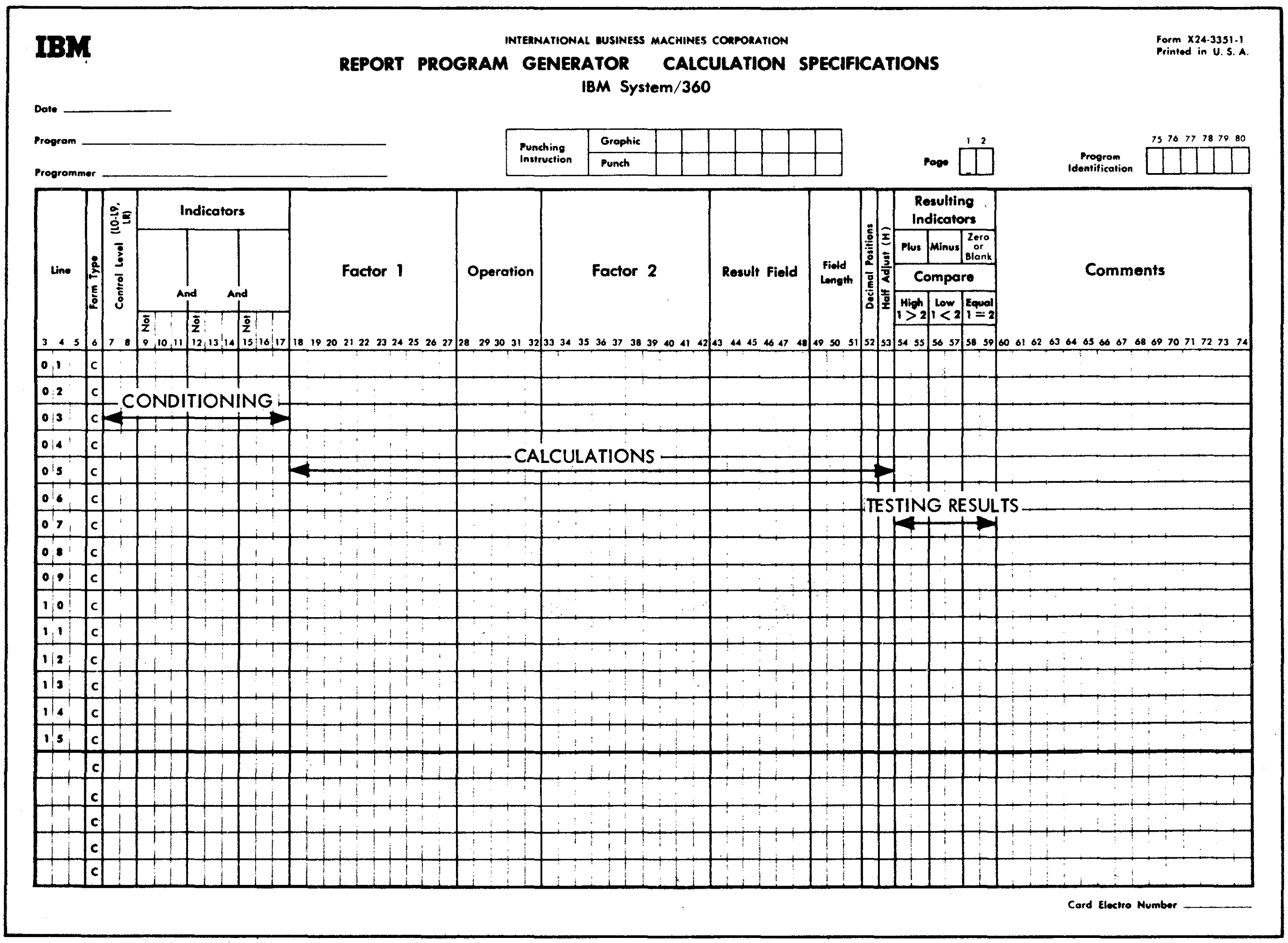
3. The identification of data conditions, to facilitate control of subsequent calculations and of output operations based on calculation results.

Three general rules govern the writing cf calculation specifications:

1. Each operation is specified on a separate single line of the form (and, hence, punched into a separate specification card).

2. Calculation operations to be performed at detail time must all be specified ahead of those to be performed at total time. However, total-time calculations need not be grouped by level of total (i.e., different L-indicator lines may be intermixed).

3. Within the grouping of detail time or total time, calculation operations are performed in the order in which they are specified. (See GCTO, below, for exception.)


IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-3351-1
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	Control Level (L0-L9, LR)	Indicators: Not, And Not, And Not	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Plus / Compare High 1 > 2	Minus / Low 1 < 2	Zero or Blank / Equal 1 = 2	Comments
3 4 5	6	7 8	9 10 11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27	28 29 30 31 32	33 34 35 36 37 38 39 40 41 42	43 44 45 46 47 48	49 50 51	52	53	54 55	56 57	58 59	60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
01	C													
02	C													
03	C													
04	C													
05	C													
06	C													
07	C													
08	C													
09	C													
10	C													
11	C													
12	C													
13	C													
14	C													
15	C													
	C													
	C													
	C													
	C													
	C													

Card Electro Number

Figure 28. The Calculation Specificaticns Form

Note: At the beginning of object-program execution, fields defined as alphameric are blank (hexadecimal 40) and those defined as numeric contain "unsigned" zeros (all zeros, except low-order position is hexadecimal 0F). Therefore, no calculation specification is required solely to clear fields at the start.

The entries for the Calculation Specifications form are divided into three categories, as shown in Figure 28.

1. Conditioning Fields--Columns 7-17

   Indicator codes entered in these fields determine the conditions under which the calculation specification in that line is to be executed.

   Note: Grouping specification lines with identical conditioning indicator entries (in cols. 7-17) saves core storage space and program execution time.

2. Calculation Fields--Columns 18-53

   Entries in these fields define the kind of operation to be performed, the data involved in the operation, and the result field.

3. Result-Testing Fields--Columns 54-59

   Any indicator code may be entered in any of these Resulting Indicators fields. When the operation specified in a line has been executed, the indicator (if any) assigned to the condition that accords with the result is turned on; those (if any) assigned to other result conditions are turned off.

   These Resulting Indicators can be used as conditioning indicators to condition the execution of other calculation specifications and/or output specifications.

CONDITIONING PERFORMANCE OF CALCULATIONS (COLS. 7-17)

If the conditioning fields (cols. 7-17) are blank, the specifications in that line are executed at detail time of every program cycle.

The Control-Level field (cols. 7-8) provides for determining whether a calculation specification is to be performed at detail time in the program cycle (i.e., cols. 7-8 are blank), or at total time of a given level (i.e., there is an L-indicator entry in cols. 7-8). The Indicator fields (cols. 9-17) provide for determining--within the limits established by the contents of cols. 7-8--which other conditions, in terms of status of indicators, must be satisfied to implement the specifications in that line.

The four fields (cols. 7-8, 9-11, 12-14, 15-17) are in an AND relationship; i.e., all stated conditions must be satisfied for the specifications in that line to be executed.

There is an essential difference between applying indicators as conditioning indicators--as exemplified by cols. 7-17 here, and cols. 23-31 in the output-format specifications--and assigning indicators as Resulting or Field Indicators--as exemplified by cols. 19-20, 59-60, and 65-70 in the input specifications, and cols. 54-59 in the calculation specifications:

A Resulting or Field Indicator changes status (on or off) as the condition with which it is associated is tested, and the criterion to which it is assigned is either satisfied or not satisfied.

The same indicator, reflecting a criterion previously tested, may then be applied to condition a calculation or output specification to be executed only if the indicator is on, or if it is not on, as desired. Applying the indicator as a conditioning indicator never changes its status (on, or not on).

Control Level--Cols. 7-8

If this field is blank, the operation specified in that line is to be executed at detail time--and subject to conditioning indicators specified in cols. 9-17.

If a Control-Level indicator code (L0, L1-L9, or LR) is entered in cols. 7-8, the operation specified in that line is to be executed at total time, provided the particular L-indicator specified is on--and subject to other conditioning indicators specified in cols. 9-17.

If a Control-Level indicator L1-L9 is turned on in the normal manner--by a control break--it is on from (and including) the total time following the last card of the previous control group through detail time of the first card of the new control group. The last-record indicator (LR) is on during total time following the last data card of the appropriate file(s). The L0 indicator is on at the start of program execution and is never turned off by the RPG program itself. It can only be turned off by a programmer's specification (SETOF). It is turned on again by the RPG program immediately after detail-time output, in other words, after a new card has been read. Any of the indicators LR, L9-L2--if turned on in the standard fashion---

also turns on all lower-level L-indicators (except L0) for the same period.

The operation of L-indicators, and detail and total times in the program cycle, have been thoroughly covered elsewhere. See RPG Program Logic (Figure 6), Relation of Total Time to Card Movement, and Total-Time Processing on "Run-In"--all under Program Logic Flow; L1-L9, Special Considerations for Indicators L1-L9 on "Run-In", L0, LR, and Indicator Hierarchy (with Figure 11)--all under Indicators; Control Levels--under Matching of Files; Decimal Positions (col. 52), Field Name (cols. 53-58): Defining the Same Data Field as Both Alphameric and Numeric, Control Level (cols. 59-60), and Field-Record Relation (cols. 63-64)--all under Input Specifications.

Indicators--Cols. 9-17

If these fields are blank, the specifications in that line are executed in every program cycle--subject to the status of any L-indicator that might be specified in cols. 7-8 (see Control Level, cols. 7-8, above). If cols. 7-8 are also blank, the specifications are executed at detail time of every program cycle.

An indicator code in cols. 10-11, 13-14, or 16-17 instructs the program to execute the specifications in that line only if that indicator is on. If an N (=Not on) is entered in the column preceding the indicator code (col. 9, 12, or 15, respectively), the program is instructed to execute the specifications in that line only if the associated indicator is not on.
Note: Any EBCDIC character other than N in col. 9, 12, or 15 has the same meaning as a blank.

Up to three different conditioning indicators may be designated by entries in the three Indicators fields for one specification line. Each may be required to be on (first column of relevant Indicators field blank), or off (N in first column of relevant Indicators field), as a condition of execution of the specifications in that line.

The three fields (cols. 9-11, 12-14, and 15-17) are identical in function. If only one or two conditioning indicators are assigned, it does not matter which of the three fields are used. Entries in these fields are in an AND relationship to each other, and to any L-level indicator in cols. 7-8. More than three conditioning indicators cannot be specified in an AND relationship directly; nor can OR relationships be specified directly. Methods for achieving the equivalent results are shown in Programming Tips, Appendix E.

Any indicator may be specified in cols. 9-17. The programmer should remember, however, that the status of an indicator may have a different significance at total time and at detail time--and it is the entry, or absence of entry, in cols. 7-8 that determines whether the specifications in that line are executed at total or detail time. For instance:

1. At total time, the MR indicator reflects the matching status of the previous card--not that of the new card that caused the control break. At detail time, however, MR reflects the matching status of the card being processed.

2. With Control Level (cols. 7-8) blank, an L-indicator (L1-L9) in one of the fields in cols. 9-17 does not pertain to an operation at total time--it conditions the specifications to be executed only at detail time of the first card of a new control group of that, or higher, level.

Note: If execution of a calculation-specification line is conditioned by indicators in columns 7-17, remember that the operation will not be executed during each program cycle unless the status of the conditioning indicator is appropriate. Therefore, resulting indicators (cols. 54-59) could reflect an earlier, and possibly inapplicable, operation.

Assuming only standard methods of assigning and utilizing Resulting and Field Indicators, the Indicator codes entered in these fields (cols. 9-17) condition execution of a calculation specification based on the following factors:

1. If the indicator was assigned as card-type Resulting Indicator in the input specifications (cols. 19-20), execution of the calculation specification is confined to the processing of a particular card type, or--if N is entered (in the first column of the relevant conditioning Indicators field)--to any card type other than that particular one.

2. If the indicator was assigned as a Field Indicator in the input specifications (cols. 65-70), execution of the calculation specification is dependent on the status of the input-data field with which the indicator was associated. The Field Indicator reflects the input-data status after the field was last read as input or, if the indicator was assigned to Zero-or-Blank, the status before the field was read or after it was cleared by a Blank-After instruction in the output specifica-

tions--whichever occurred most recently. The status of a Field Indicator is not altered by changes in the contents of the associated field that might be the result of calculation specifications.

3. If the indicator is assigned as a Resulting Indicator in this or another line of the calculation specifications (cols. 54-59--discussed later), execution of the calculation specifications in this line is controlled by the result of a calculation performed earlier. Note that:

   a. The Resulting Indicator reflects the result last obtained. If the pertinent calculation has not yet been performed, the indicator is off (unless it is assigned to Zero-or-Blank, when it is on initially and is also turned on by Blank-After--see Output-Format Specifications).

      If the pertinent calculation is only performed under certain conditions, the indicator may still reflect an earlier--possibly obsolete--result.

   b. If the conditioning Indicator (cols. 9-17) is also a Resulting Indicator in the same line (cols. 54-59), its status is not changed by the instructions in that line until after the specifications in the line have been executed.

Programmer

Line (3 4 5)	Form Type (6)	Control Level (L0-L9, LR) (7 8)	Indicators: Not (9 10 11)	And: Not (12 13 14)	And: Not (15 16 17)
0 1	C				
0 2	C		16		
0 3	C		N16		
0 4	C		25	N18	NH1
0 5	C		25	L1	
0 6	C		MR	16	
0 7	C	L2	1Ø	NL3	
0 8	C	L1			
0 9	C	LØ	OF	NL1	
1 0	C	LR			
1 1	C				

Figure 29. Calculation Conditioning Indicators

Remember that, at total time, a card-type Resulting Indicator is that of the next card, whereas a Field Indicator is based on a previous card.

The operation of indicators, and detail and total times in the program cycle, have been thoroughly covered elsewhere. See sections Program Logic Flow, Indicators, and Matching of Files--all under Programming for RPG--General Information; Resulting Indicators, Control Level, Matching Fields, and Field Indicators--all under Input Specifications; and Resulting Indicators in Calculation Specifications, discussed in this chapter.

Figure 29 illustrates the use of conditioning indicators with calculation specifications. Only standard applications of indicators have been assumed; non-standard uses are encompassed by the explanations under Indicators in Programming for RPG--General Information.

Explanation of Entries in Figure 29

Line 01 specifications will be executed at detail time (cols. 7-8 blank) of all cards (cols. 9-17 blank).

Line 02 specifications will be executed at detail time for all cards for which indicator 16 is on at that point. If, for example, indicator 16 is a card-type Resulting Indicator, the line is executed at detail time for all cards of type 16.

Line 03 specifications will be executed at detail time for all cards for which indicator 16 is not on at that point.

Line 04 specifications will be executed at detail time for all cards for which indicator 25 is on, and indicators 18 and H1 are not on, at that point.

Line 05 specifications are executed at detail time of the first card of a control group, provided indicator 25 is also on at that point. If, for example, indicator 25 represents a card type, the specifications in the line are executed at detail time of the first card of a control group, if that is a card of type 25.

Line 06 specifications are executed at detail time if the MR indicator and indicator 16 are both on at that point.

For example: Assume indicator 16 represents a detail card with a value that is to be multiplied by a factor stored from a preceding master card. These Indicators entries assure that the multiplication takes place only during processing of a detail card, and only provided the detail card matched the master card on some criterion field(s)--otherwise the multiplication factor could be taken from the wrong master card.

Line 07 specifications are executed at total time. The L2 in cols. 7-8 specifies execution at total time if a Level-2, or higher, control break occurred; i.e., following the processing of the last card of a Level-2 or higher control group. The specifications are, however, further conditioned to be executed only if indicator 10 is also on at that time, and provided indicator L3 is not on. The latter condition (NL3), implies that the specifications are only executed if there was no control break higher than Level 2; the L2 in cols. 7-8 implies that the control break must be at least of Level 2. Thus, execution of the specifications is confined to exactly a Level-2 control break.

Note that, if indicator 10 is a card-type Resulting Indicator, it refers to the first card of the new control group; if it is a Field Indicator, it reflects the status of an input field the last time a pertinent card type was read preceding the control break. The data available at total time is still that from the last card of the control group.

Line 08 is executed at total time following the processing of the last card of every control group: since a control break of any level turns on the L-indicator for that level and for all lower levels, L1 turns on for a control break of any level. The data from the last card of the control group is still available at this time.

Line 09 illustrates an application of the L0 indicator. Assumptions are: it is desired to calculate a value at the end of each page, to be printed at the bottom of each page, except when a control break occurs at the same point.

In order to calculate before forms advance to the new page, yet when it is already known whether a control group has been completed and whether carriage-tape channel 12 was encountered at detail-output time, the calculation must be at total time: total time precedes overflow-time output. L-indicators for the beginning of a new control group are already on, and the overflow indicator is also on if carriage-tape channel 12 (i.e., the point for printing the calculated value at the bottom of the page) was encountered during the preceding detail-time output.

Indicator L0 designates that the specifications in the line are to be executed at total time, provided L0 is on (its normal state). Indicator OF further designates that the overflow indicator must be on (i.e., channel 12 encountered during preceding detail-time output) for the specifications in the line to be performed. Since the calculation is not desired at the end of a control group, NL1 suppresses it when a control break coincides with the end of a page. L0--which is defined as a Control-Level indicator--had to be used to associate the line with total time since, by definitionof the problem, no other Control-Level indicator is on when the specifications in the line are to be executed.

Line 10 specifications are executed at total time following the last data card of the input (or combined) file or, if there are multiple input (or combined) files, the last pertinent file. (See File Description Specifications, col. 17, and Indicators, LR.) When LR is on, all other Control-Level indicators (L9-L1) are also on. The job terminates following total-time output.

Figure 29 also illustrates that all specifications to be executed at detail time must precede those to be executed at total time; within total-time specifications, order need not be maintained by Control Level.

SPECIFYING THE KINDS OF CALCULATIONS (COLS. 18-53)

Entries in this section (cols. 18-53) of the calculation specifications define the actual calculations (or quasi-calculations) to be performed. The following components of calculation operations are designated in these specification fields:

1. The data fields that enter into the operation: Factors 1 and 2.

2. The type of operation to be performed on the data: Operation.

3. The form of the result: Result Field name, length, decimal-point location, half-adjustment.

The fields in this section are described in the sequence that lends itself best to a clear understanding of their relationship, rather than adhering strictly to the order of fields in the specifications.

Note: At the beginning of program execution, Factor and Result Fields are blank (if alphameric) or "unsigned" zero (if numeric).

Factor 1--Cols. 18-27 and
Factor 2--Cols. 33-42

Factor 1 and Factor 2 contain the names of the data fields, or the actual data (literals), that provide the source information for the majority of the operations. Some operations involve both Factors; some only utilize one; and a few operations do not use a Factor field.

Field Name

If the Factor contains a field name, the program obtains the data from the core storage location it has assigned to that name. The field name must either have been defined in the input specifications (cols. 46-58), or as a result field (cols. 43-52) in some line of the calculation specifications (this may be an earlier or later line in the calculation specifications, or the same line), or as a table name in the File Extension Specifications.

If a field whose name is used in the calculation specifications appeared in the input specifications, it must have been fully defined there, and cannot be defined differently (as to format, size, decimal point) in the calculation specifications. A field need be defined only once, although an identical redefinition is permitted.

Field names must be recorded in the Factor fields left-justified (i.e., begin in col. 18 or 33, respectively).

Literals

A literal is the actual data to be used in the calculation, rather than a field name representing the location of the data in core storage (see also Literals, under Definition of Terms). The program is able to distinguish between literals and field names by virtue of a restriction on the initial character (col. 18 or 33, respectively):

The first character of a numeric literal is one of the digits 0-9, a decimal point, a plus sign, or a minus sign. (If European notation is specified in the RPG Control Card, a decimal comma takes the place of the decimal point.)

The first character of an alphameric literal is preceded by an apostrophe (')--card punch-combination 5-8.

The first character of a field name is one of the 29 characters defined as alphabetic (the 26 letters of the English alphabet, plus three specific symbols)--see Definition of Terms.

A literal--so long as it is always identical in all respects (including sign and decimal-point location, if any)--is stored by the program only once, no matter how often it is used in the program. Numeric literals may have a maximum length of ten characters, including the symbols (decimal point and/or sign); alphameric literals in calculation factors may have a maximum length of eight characters, plus two mandatory enclosing apostrophes. Literals must be recorded left-justified (i.e., begin in col. 18 or 33, respectively).

Numeric Literals. A numeric literal may consist of any combination of the digits 0 through 9. One decimal point (decimal comma, if European notation) and/or one sign may also be included, but no other characters or symbols. Its maximum total length is ten characters.

Rules for Forming Numeric Literals

1. Blanks must not appear within a numeric literal.

2. If a sign is part of the literal, it must be the leftmost character.

   A plus sign is represented by the punch-combination 12-6-8; a minus sign, by an 11-punch.

   An unsigned numeric literal is treated as positive in arithmetic operations.

   The positive or negative status of a numeric literal is automatically taken into account in arithmetic operations.

3. One decimal point (card punch-combination 12-3-8) can appear anywhere in the literal, even ahead of the first digit. (If European notation, decimal comma applies instead.)

   When the literal is used in an arithmetic or compare operation, the program performs decimal alignment according to the position of the decimal point. If there is no decimal point in the literal, the program assumes the decimal point to follow immediately to the right of the last digit; i.e., the literal is assumed to be an integer.

Alphameric Literals. An alphameric literal consists of any combination of characters, including blank, from the 256-character EBCDIC set (see Appendix D, Figure D1).

An alphameric literal must be enclosed by apostrophes ('), card punch-combination

5-8. The first apostrophe identifies the entry as an alphameric literal; the terminal apostrophe signals the end of the literal (since blanks are valid in alphameric literals, the program has no other means of recognizing the end). The requirement for initial and terminal apostrophes limits the body of an alphameric literal to a maximum of eight characters.

An apostrophe required as part of the literal itself is represented by two apostrophes; i.e., two consecutive columns, each punched 5-8. If such a literal is used for output, only one of the dual apostrophes is punched or printed, and the portion to the right of the internal apostrophe is shifted left one position so as not to introduce a spurious blank. This limits an alphameric literal with one internal apostrophe to seven meaningful characters.

Alphameric literals may be used for compare, move, test zone, and table look-up operations; but they must not be used in arithmetic operations.

Figure 30 depicts some samples of Factor entries. While Factor 1 is shown, Factor 2 would be equally applicable.

Explanation of Entries in Figure 30

Programmer ______________

Line (3 4 5)	Form Type (6)	Control Level (10-19, LR) (7 8)	Indicators: Not (9 10 11)	And: Not (12 13 14)	And: Not (15 16 17)	Factor 1 (18 19 20 21 22 23 24 25 26 27)	28
0 1	C					NETAMT	
0 2	C					@PAR	
0 3	C					12500.00	
0 4	C					12500	
0 5	C					.001256789	
0 6	C					-.01357942	
0 7	C					+1	
0 8	C					'FEBRUARY'	
0 9	C					'O''CLOCK'	
1 0	C					'''66'	
1 1	C					'5/2% DT.'	
1 2	C						

Figure 30. Factor Entries

Line 01 shows a field name: the first character is alphabetic. The field must either have been defined in the input or file extension specifications, or it must be defined as a Result Field (cols. 43-48) somewhere in the calculation specifications.

When the specifications in line 01 are executed, the Factor-1 data is obtained by the program from the core storage location assigned to NETAMT.

Line 02 also shows a field name: @ is one of the three symbols defined as alphabetic (see Definition of Terms).

Lines 03-07 illustrate numeric literals: the first character is numeric, or a sign, or a decimal point.

The literals will be decimal-aligned by the program, in arithmetic and compare operations, in accordance with their specified decimal point. When the literal includes no decimal point, it is treated as an integer. The literal in line 04 is therefore treated as (12500.), and that in line 07 as (1.). The plus sign in line 07 must be punched as 12-6-8.

Numeric literals without sign are treated as positive in arithmetic and compare operations; therefore, the literals in lines 03, 04, and 05 are positive. Note that a sign, if specified, must be leftmost (lines 06 and 07).

A numeric literal terminates with its rightmost character (digit or decimal point); it cannot contain blanks. Therefore, the literal in line 03 ends with the zero in col. 25; the literal in line 04 ends with the zero in col. 22. Note that (except for a decimal comma in European notation) commas are not permitted in numeric literals: for example, the number in line 04 may not be written as 12,500.

Lines 01, 05, 06, 08, 09, and 11 illustrate the maximum permissible lengths for the three types of Factor entries: six positions for a field name; ten positions, including sign and/or decimal point, for a numeric literal; and eight positions, plus the two delimiting apostrophes, for an alphameric literal.

Lines 08-11 portray alphameric literals: the first and last characters are apostrophes.

Alphameric literals might, for example, be compared against data-field contents, serve as search arguments in a table look-up operation, or be printed out after being moved into a data field. They cannot be used in arithmetic operations.

Lines 09 and 10 illustrate the two apostrophes--independent of the apostrophes defining the alphameric literal--that must be recorded to specify a single apostrophe as part of the literal. Line 09 exemplifies a literal with an internal apostrophe, whereas line 10 shows how to record a literal that begins with an apostrophe. The literal in line 09 is treated as O'CLOCK in any operation involving it; the literal in line 10 is treated as '66.

Line 10 also illustrates that an alphameric literal may contain digits. If desired, a numeric value may be expressed as an alphameric literal by enclosing it in apostrophes. (It cannot, then, be used in arithmetic operations.)

Line 11 shows that an alphameric literal may contain spaces and special characters. Throughout, note that all entries are left-justified.

Operation--Cols. 28-32

Entries in these columns specify the operation to be performed using the entries in Factor 1, and/or Factor 2, or Result Field. Each operation is specified by placing the appropriate operation code in this field, left-justified (i.e., beginning in col. 28). Detailed information on the various operations is given in the section titled Entries in the Operation Field.

All numeric compare and arithmetic operations are performed according to the rules of algebra: signs are taken into account, and decimal alignment is automatic.

Result Field--Cols. 43-48

The field name entered in Result Field (cols. 43-48) is associated by the program with the core location at which the result of the operation is to be stored. (In the case of two particular operations, TESTZ and LOKUP, the entry in this field represents the location of source information for the operation.) The user always references the field by the mnemonic name he assigned to it, and need never concern himself with the actual core storage location.

If the field name appeared in the input or file extension specifications, it must have been fully defined there (as to size, format, position of decimal point). The field name in Result Field then suffices to reference the storage location, and no definition of the field is required in the calculation specifications. If the field name did not appear in the input or file extension specifications, the field must be fully defined once in the calculation specifications, so that the program can assign an appropriate core storage location to it. The definition need not be in the first calculation specification line in which the field name appears.

Defining a field in the calculation specifications consists of:

a. Entering a field name in Result Field (cols. 43-48) in any specification line to which the result field applies.

The name must begin in col. 43 with one of the 29 alphabetic characters, may continue with alphabetic or numeric characters, and may be one to six characters long. (See Definition of Terms, for "alphabetic" and "numeric" characters--neither permits embedded blanks.)

Within these rules, any name may be assigned; except that names starting with PAGE and TAB are reserved for special uses (described later). Names beginning with IN have a special meaning in RLABL specifications; when used in other operations, they must not duplicate the exact characters INxx in the Result Field of an RLABL line.

b. Specifying the length of the field--see Field Length (cols. 49-51).

c. Defining the format--see Decimal Positions (col. 52).

The same field name can be used in any number of calculation specifications; but, once defined in the input, file extension, or calculation specifications, it must never be redefined differently. Once defined, it need never be redefined at all --the field name alone becomes an adequate reference; but, if it is redefined, it must be fully and identically redefined: the contents of Field Length (cols. 49-51) and Decimal Positions (col. 52) must then be identical wherever the field name is defined or redefined and, if the field was defined in the input or file extension specifications, Decimal Positions and field length there must correspond. (But see note, under Field Length, concerning packed input fields.)

Field Length--Cols. 49-51

This field is left blank unless the result field is to be defined (or redefined) in this line--see Result Field, above.

If the Result Field is to be defined (or redefined) in this line, enter the length of the result field for which core storage positions are to be assigned. (Leading zeros may be omitted.) So that the user need not think in terms of internal machine operations, the length is specified in terms of number of characters or digits, regardless of whether the field is defined as alphameric or numeric.

Internally, numeric fields are stored in packed-decimal format (see Appendix D) and normally consume less core positions than the number of digits in the field; nevertheless, Field Length is specified here as though each digit occupied a separate byte (full position) in core. Therefore, in computing the size of results based on the factors used in an operation, an input field that was read in packed format (P in col. 43 of the input specifications) must be assigned, in the calculation specifications, a length equivalent to its digit capacity--not its columns. For example, a packed input field of 5 columns must be treated in the calculation specifications--as a factor, when defining a result based on it, or if redefining it--as being 9 positions long. The general formula is: field size = 2n-1, where n = number of card columns in the packed input field.

Maximum lengths for factors and result fields in calculation specifications are:

15 positions for a numeric field;

256 positions for an alphameric field; except: 40 positions each when comparing two alphameric fields, and 80 positions for table look-up.

(For definition of a field as numeric or alphameric, see Decimal Positions--Col. 52, below and under Input Specifications.)

If the length assigned to the Result Field of an arithmetic operation is insufficient to accommodate the result of the operation, the result is first decimal-aligned--as are all arithmetic results--and then the excess high-order (most significant) positions are truncated (see Figure 31). Resulting Indicators assigned (see cols. 54-59) are then based on the retained digits only.

If Half-Adjust is specified (see col. 53), Field Length applies to the length of the result field after half-adjustment has been executed (i.e., after the extra position required for half-adjustment has been dropped).

Decimal Positions--Col. 52

Column 52 is left blank if:

The operation does not involve a result field; or

It is desired to define the result field as alphameric; or

The result field has been defined elsewhere (in the input specifications, the file extension specifications, or another calculation specification line), and it is not desired to redefine it here. Once defined, it need never be redefined (if redefined, the contents of Decimal Positions, col. 52, must agree with the original definition)--see Result Field, above.

An entry (0-9) in Decimal Positions defines the associated result field as numeric and specifies the number of positions to the right of the decimal point. If no decimal places (i.e., only whole numbers) are to be retained in the numeric result, 0 is recorded in col. 52. If a field that must be defined as numeric is not used in compare or arithmetic operations, any digit 0-9 (within field-size limit) may be specified, regardless of actual number of decimal places.

Fields used in arithmetic operations or numeric compare--or to be edited or zero-suppressed for output--must be defined as numeric. Arithmetic operations comprise addition, subtraction, multiplication, and division (and movement of remainder). In these operations, the program performs automatic decimal-point alignment, in accordance with the decimal positions that have been designated for the fields involved. Move operations to a numeric field also require a Decimal Positions entry where the result field is defined, but decimal alignment is not automatic (see MOVE and MOVEL, below).

The number specified in Decimal Positions (col. 52) must neither exceed 9, nor be greater than the field length specified for the result field. It may, however, be greater or less than the number of decimal digits that result from the operation, provided the assigned field length is large enough. If the Decimal-Positions specification is greater than the number of decimal places that result from the arithmetic operation, an appropriate number of zeros is appended at the right; if it is smaller, the excess right-hand positions are truncated after completion of the arithmetic operation. (If Half-Adjustment is specified by H in col. 53, truncation takes place after half-adjustment.)

Figure 31 itemizes the contents of the result field and the position of the decimal point for a sample multiplication, for different Field-Length and Decimal-Positions specifications. Note that it is not possible to truncate the right-hand position(s) to the left of the decimal point. For example, 121.86984 cannot become 12 by specifying 0 Decimal Positions and a Field Length of 2; instead of the right-hand digit, the most-significant digit is lost, and 21 (rather than 12) is retained.

Half-Adjust--Col. 53

If this column is left blank, the result is stored in the Result-Field core storage location exactly as calculated, retaining the number of decimal places specified in column 52. Column 53 must be left blank for all non-arithmetic operations (and for a Divide operation that is followed by the Move Remainder operation).

If an H (Half-Adjust) is placed in col. 53, the last decimal position to be retained in the result (per specification in col. 52) is rounded, by the equivalent of adding 5 to the next decimal position. The excess decimal positions are then dropped, and the remaining result is stored at the core storage location assigned by the program to the Result-Field name. The program provides for proper rounding of both positive and negative values.

In this RPG program, half-adjustment is actually performed as follows, although the effect is the same as though the leftmost position to be dropped were increased absolutely by 5:

1. The arithmetic operation specified in the line is completed.
2. That portion of the original result that is to be dropped--i.e., the digits in the positions to be dropped--is added algebraically, in the same positions, to the entire original result.
3. The excess right-hand positions are dropped.
4. This final result, conforming to Field-Length and Decimal-Positions specifications, is stored at the location assigned by the program to the Result-Field name.

Note: Since half-adjustment operates upon the digits to the right of the last position to be retained, it is meaningless to perform half-adjustment unless the calculated arithmetic result has at least one more decimal position than is to be retained. Otherwise, the positions to be retained cannot be affected by the half-adjustment, since there cannot be a carry.

Multiplication: 98.76 x 1.234 = +121.86984

DECIMAL POSITIONS (Col. 52) \ RESULT FIELD LENGTH (Cols. 49-51)	10	9	8	7	6	5	4	3	2	1
9	1.869840000	.869840000			NOT PERMITTED: Number of decimal places specified > field length specified.					
8	21.86984000	1.86984000	.86984000							
7	121.8698400	21.8698400	1.8698400	.8698400						
6	0121.869840	121.869840	21.869840	1.869840	.869840					
5	00121.86984	0121.86984	121.86984	21.86984	1.86984	.86984				
4	000121.8698	00121.8698	0121.8698	121.8698	21.8698	1.8698	.8698			
3	0000121.869	000121.869	00121.869	0121.869	121.869	21.869	1.869	.869		
2	00000121.86	0000121.86	000121.86	00121.86	0121.86	121.86	21.86	1.86	.86	
1	000000121.8	00000121.8	0000121.8	000121.8	00121.8	0121.8	121.8	21.8	1.8	.8
0	0000000121	000000121	00000121	0000121	000121	00121	0121	121	21	1

NOTE:
1. Shaded area corresponds to number of Decimal Positions (Col. 52) greater than size of result Field Length (Cols. 49-51) – which is not permitted.
2. Heavy borders outline the combination of Field-Length and Decimal-Positions specifications that provide correct results, for various numbers of decimal places, and various legitimate Result-Field sizes the user may wish to retain – based on the particular factors illustrated.

Figure 31. Result Field Contents, after a Multiplication, for Different Field-Length and Decimal-Positions Specifications

RESULT FIELD LENGTH (cols. 49 - 51)	9			8			6			3
DECIMAL POSITIONS (col. 52)	5	1	0	4	3	2	3	2	1	0
RESULT OF ARITHMETIC OPERATION	+0139.95047	-00000139.95047	+000000139.95047	-0139.95047	+00139.95047	-000139.95047	-139.95047	+0139.95047	+00139.95047	-139.95047
DIGITS TO BE ADDED FOR HALF-ADJUSTMENT	Half-Adjust operation not performed	- 5047	+ 95047	- 7	+ 47	- 047	- 47	+ 047	+ 5047	- .95047
RESULT AFTER ADDING ADJUSTMENT DIGITS	Half-Adjust operation not performed	-00000140.00094	+000000140.90094	-0139.95054	+00139.95094	-000139.95094	-139.95094	+0139.95094	+00140.00094	-140.90094
HALF-ADJUSTED FINAL RESULT TO BE STORED	+0139.95047	-00000140.0	+000000140	-0139.9505	+00139.950	-000139.95	-139.950	+0139.95	+00140.0	-140

LEGEND: ↑ indicates point to the right of which digits are to be dropped before final result is stored in accordance with Decimal Positions specification in column 52.

Figure 32. Examples of Half-Adjustment

Figure 32 illustrates half-adjustment by some examples. The example in the first column (Result Field: 9 digits, Decimal Positions: 5) shows that nothing is accomplished by half-adjusting when the number of decimal positions in the calculated result is no greater than the number of decimal positions retained: The result used has 5 decimal places (say, a multiplier with 2 decimal places and a multiplicand with 3 decimal places were multiplied). We specified (in col. 52) retention of 5 decimal places. Therefore, half-adjustment computation would be based on a value in the 6th position--a dummy position without a significant digit, which could never cause a carry-over to the 5th decimal position.

Figure 33 gives some arbitrary examples of entries in calculation fields--ignoring conditioning Indicators entries (cols. 7-17). The entries in Resulting Indicators (cols. 54-59) in Figure 33 are discussed at the end of the next section.

IBM — INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-3351-1 Printed in U. S. A.

Date ______ Program ______ Programmer ______ Punching Instruction: Graphic / Punch Page __ Program Identification __

Line	Form Type	Control Level (L0-L9, LR)	Indicators	Indicators (And)	Indicators (And)	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators Plus / High 1>2	Minus / Low 1<2	Zero or Blank / Equal 1=2	Comments
01	C						Z-ADD	GRSPAY	FSTNET					H1		
02	C					FSTNET	ADD	BONUS	FSTNET	6	2					
03	C					FSTNET	SUB	ADVANC	FSTNET	6	2					
04	C					GRSPAY	COMP	EXMPTN					25			
05	C						TESTZ		DIVSN				Ø1	Ø2	Ø1	
06	C						MOVE	IDENT	NAME	24						
07	C						MOVE	1ØØØØ	INDEX	5	2					
08	C					FIELDA	MULT	FIELDB	FIELDC	15	2	H				
09	C					DNRALW	ADD	GRSPAY	GRSPAY	6	2					
10	C		Ø2	N25			SETON						1Ø	15	16	
11	C					EMPLNO	LOKUP	TABEMP	TABPAY						Ø2	
12	C															

Figure 33. Examples of Entries in Calculation Fields and in Result-Testing Fields (Resulting Indicators)

Explanation of Calculation-Fields Entries in Figure 33

Specification line 01 shows an operation (Zero and Add) that uses only Factor 2 with the Result Field. It also illustrates defining a Result Field in a subsequent calculation specification (line 02)--cols. 49-52 are blank in line 01.

Line 02 shows the definition of a Result Field as FSTNET, six digits long, including two decimal places to be retained. Although the same Result Field was used in line 01, it is satisfactory to define it in a later line. (BONUS is presumed to be defined in the input specifications or another calculation specification line.)

Line 03 was inserted only to show that a Result Field may be defined more than once, provided that Field-Length and Decimal-Positions entries are identical. (ADVANC is presumed to be defined in the input specifications or another calculation specification line.)

Line 04 portrays an operation (Compare) that uses Factors 1 and 2, but does not involve Result Field. (EXMPTN is presumed to be defined in the input specifications or another calculation specification line. GRSPAY is defined in line 09.)

Line 05 shows an operation (Test Zone) that involves only Result Field which, in this case, contains the name of the source-data field. Decimal Positions must be blank, because TESTZ applies only to alphameric data. The Result Field (DIVSN) is presumed to be defined in the input specifications or another calculation specification line. (If DIVSN did not appear in the input specifications, it could be defined here by an entry in Field Length. It would, however, have to appear also elsewhere in the calculation specifications as Result Field, since TESTZ does not produce any result field.)

Line 06 shows a Move instruction, which uses only Factor 2 and Result Field. The Move is to an alphameric field, because Decimal Positions (col. 52) is blank. Field Length is 24 positions, which requires definition of the field as alphameric: numeric fields are limited to 15 digits. (IDENT is presumed to be defined in the input specifications or another calculation specification line.)

Line 07 shows a Move of a numeric literal to a Result Field named INDEX, defined as 5 digits long including 2 decimal places. After the Move, INDEX represents 100.00: the Move operation itself does not perform decimal alignment.

Line 08 shows a Result Field (FIELDC) defined for the maximum length (15) permitted for numeric data. Two decimal places are to be retained; the second decimal position is to be rounded (H in col. 53) before the excess decimal positions are dropped. (FIELDA and FIELDB are assumed to be defined elsewhere.)

Line 09 defines GRSPAY as six digits long, numeric, with two decimal places. This field is used as Factor 1 in line 04. (DNRALW is assumed to be defined elsewhere.)

Line 10 shows an operation code (SETON) that utilizes neither Factor nor Result-Field entries.

Line 11 specifies that a table of employee numbers (TABEMP) is to be searched for the number matching that stored for the field name EMPLNO. If and when a match is found, the corresponding pay-rate entry in the table TABPAY is to be made available for processing.

TESTING THE RESULTS OF CALCULATIONS (COLS. 54-59)

Entries in the Resulting-Indicators fields of the calculation specifications designate indicators that are to be set on or off, based on results of calculation operations or on direct indicator-setting instructions. The status of these indicators may be used to condition the execution of calculation and/or output specifications. The Resulting-Indicators fields are used in five ways:

1. To reflect the status of the result of an arithmetic operation involving addition, subtraction, multiplication, or division (or Move Remainder).

If the result is ...	the indicator (if any) assigned to ... turns (or remains) on
Positive (excluding $\overset{+}{0}$)	--Plus (cols. 54-55)
Negative	--Minus (cols. 56-57)
Zero (including $\overset{+}{0}$)	--Zero or Blank (cols. 58-59)

The indicators (if any) assigned to the conditions that do not apply, remain (or turn) off. If the same indicator is assigned to more than one of the three alternative Resulting-Indicators criteria, it turns on if the result satisfies one of these criteria.

The setting of the indicators corresponds to the final result--after half-adjustment (if H is specified in col. 53), and after dropping of any excess decimal places (per Decimal-Positions entry in col. 52). A final all-zero result, although signed as plus, causes only the indicator in Zero-or-Blank (cols. 58-59) to turn on.

For example:
If the calculated result is -0.099 and 1 is specified in Decimal Positions (col. 52), without half-adjustment, then the final result is +0.0
This turns on any indicator assigned to Zero-or-Blank--not one assigned to Minus or Plus, although the value was negative before dropping of excess decimal positions, and is signed plus for the final result.

If H is also specified (in col. 53), then the final result is -0.1
This turns on any indicator assigned to Minus.

A Resulting Indicator assigned to Plus (cols. 54-55) or Minus (cols. 56-57) is off at the beginning of program execution. Each time the calculation specifications in the line have been executed, the status of the indicator--on or off--is revised to reflect the result of the calculation.

A Resulting Indicator 01-99 assigned to Zero-or-Blank--in specification lines involving these arithmetic operations--is on at the beginning of program execution. Its status is then revised, to reflect the result of the calculation, each time the calculation specifications in the line have been executed. If the Result Field is also an output field, any indicator assigned to Zero-or-Blank is also turned on (and the field is cleared) immediately when that output field is transferred to the output area for printing, punching, or interpreting if Blank-After (B in col. 39) is specified for the field in the relevant output-format specifications. If Blank-After turns on a Resulting Indicator assigned to Zero-or-Blank, this does not turn off an indicator assigned to Plus or Minus in the same line.

If different indicators are assigned to Zero-or-Blank for the same field in several specification lines--as calculation Resulting Indicator and/or input Field Indicator--only the earliest-appearing Zero-or-Blank indicator for the field is turned on by the Blank-After instruction.

2. To reflect the result of a comparison between two fields (see COMP operation, below).

If	the indicator (if any) assigned to ... turns (or remains) on
Factor 1 > Factor 2	--High (cols. 54-55)
Factor 1 < Factor 2	--Low (cols. 56-57)
Factor 1 = Factor 2	--Equal (cols. 58-59)

The indicators (if any) assigned to conditions that do not apply, remain (or turn) off. If the same indicator is assigned to more than one of the three alternative Resulting-Indicators criteria, it turns on if the result satisfies one of these criteria.

Resulting Indicators assigned to Compare operations are off at the beginning of program execution. Each time the Compare operation has been executed, the status of these indicators--on or off--is revised to reflect the result of the comparison.

3. To identify the zone in the high-order position of an alphameric field. For the specifics, see TESTZ operation, below. However, in simplified (incomplete) terms:

If the high-order position contains ...	the indicator (if any) assigned to ... turns (or remains) on
A 12-punch	Plus (cols. 54-55)
An 11-punch	Minus (cols. 56-57)
Neither a 12- nor 11-punch	Blank (cols. 58-59)

The indicators (if any) assigned to conditions that do not apply, remain (or turn) off. If the same indicator is assigned to more than one of the three alternative Resulting-Indicators criteria, it turns on if the test satisfies one of these criteria.

A Resulting Indicator 01-99 assigned to Zero-or-Blank--in TESTZ specification lines--is on at the beginning of program execution. Its status is then revised, to reflect the result, each time the calculation specifications in the line have been executed. If the Result Field is also an output field, any indicator assigned to Zero-or-Blank is also turned on (and the field is cleared) immediately when that output field is transferred to the output area for printing, punching, or interpreting if Blank-After (B in col. 39) is specified for the field in the relevant output-format specifications. If Blank-After turns on a Resulting Indicator assigned to Zero-or-Blank, this does not turn off an indicator assigned to Plus or Minus in the same line.

If different indicators are assigned to Zero-or-Blank for the same field in several specifications lines--as calculation Resulting Indicator and/or input Field Indicator--only the earliest-appearing Zero-or-Blank indicator for the field is turned on by the Blank-After instruction.

4. In a table look-up operation:

   a. To define whether search is to be for a table argument that matches the search argument, or for the nearest higher (or lower)--but unequal--table argument, or for either;

   b. After the search, to reflect the type of match (if any) between table and search arguments.

      The indicator that reflects the type of match achieved (High, Low, or Equal) turns (or remains) on; any indicator assigned to the other condition turns (or remains) off.

If indicators are assigned both to Equal, and to High or to Low, Equal takes precedence when an exact match between table and search argument exists: the equal value is then selected, and the indicator assigned to Equal turns on. If the same Resulting Indicator is assigned to two conditions (High and Equal, or Low and Equal), the indicator turns on if either assigned criterion is satisfied. An indicator must be assigned to at least one of the three Resulting-Indicators fields (High, Low, or Equal). However, if the table arguments are not in sequence by search argument (ascending or descending), an indicator should only be assigned to Equal (cols. 58-59).

For specifics, see LOKUP operation, below.

<u>Note</u>: A Blank-After instruction in the output-format specifications has no effect on the Result Field or on an

indicator assigned to Equal for a LOKUP operation.

5. To cause designated indicators to turn on or off by the operation code SETON or SETOF, respectively. See SETON and SETOF operations, below.

Points to Note for Calculation-Specifications Resulting Indicators

1. Any indicators may be assigned in cols. 54-59.

   If indicators other than 01-99, H1, or H2 are used, the programmer must be conversant with the contents of the sections Program Logic Flow and Indicators.

   If indicator H1 or H2 is turned on, the program will halt after processing of the card has been completed, unless that indicator has been turned off again before then by another calculation specification. If the program is restarted after an H1 or H2 halt (by pressing the CPU START key twice), the H1 and H2 indicators are turned off by the program.

2. Indicators 01-99 (and H1, H2 within the limits mentioned above) change status (on or off) only when a specification has been executed where the particular indicator is assigned as Resulting Indicator or Field Indicator. (Exception: Zero-or-Blank indicator in conjunction with Blank-After instruction in output specifications--already discussed in this section and under Program Logic Flow.) Therefore:

   a. If the calculation specification is only executed for some cards (i.e., there are conditioning Indicators entries in cols. 7-17), the Resulting Indicators in that line may remain on or off from a previous card.

   b. More than one calculation-specifications Resulting Indicator can be on at the same time.

   c. If the same indicator is assigned as a Resulting Indicator in several calculation specifications, its status will be revised after each such specification has been executed.

   d. If a calculation Resulting Indicator is also assigned as an input-specification Resulting Indicator or Field Indicator, its status is affected: card-type Resulting Indicators turn off when a new card has been read, and the one for the new card turns on before total-time calculations; Field Indicators change status before detail-time calculations. Both take priority over calculation Resulting Indicators (see RPG Program Logic, Indicators and Indicator Hierarchy).

   e. The same indicator may be employed as a calculation Resulting Indicator and as a conditioning indicator (Indicators, cols. 7-17) in the same specification line. Execution of the line is then contingent upon the status of the indicator as set by a prior operation (which could have been the previous time the specifications in the line were performed).

3. Although results of arithmetic operations are always signed, a result value of zero--which carries the equivalent of a plus sign--turns on the Resulting Indicator assigned to Zero (cols. 58-59), not Plus.

   The value in the Result Field of an arithmetic operation in this RPG will never be -0 (minus zero).

Figure 33, previously used to illustrate some calculation-field entries, also depicts the assignment of calculation Resulting Indicators of all of the five types:

Line 01: Indicator H1 turns on if, after GRSPAY has been placed in the Result Field, the Result Field (named FSTNET) is negative. Otherwise, it remains (or turns) off.

Line 04: Indicator 25 turns on, after the contents of GRSPAY have been compared with the contents of EXMPTN, if the former was found to be greater than the latter (Factor 1 > Factor 2). Otherwise it remains (or turns) off.

Line 05: The zone in the high-order position of a field named DIVSN is tested. If it is equivalent to an 11-punch, indicator 02 turns (or remains) on; otherwise, indicator 02 remains (or turns) off, and indicator 01 turns (or remains) on.

Line 10 demonstrates how three indicators (e.g.: 10, 15, 16) can be set on by means of the operation SETON.

TYPE OF OPERATION ↓		Columns 54-55		Columns 56-57		Columns 58-59	
		PLUS	HIGH	MINUS	LOW	ZERO OR BLANK	EQUAL
Arithmetic Operations (except compare)	If the Result Field contains a:	Positive value (except $\overset{+}{0}$)	—	Negative value (there is no $\overset{-}{0}$)	—	Zero value ($\overset{+}{0}$)	—
Compare (COMP)	If the contents of Factor 1 are:	—	Higher in sequence (if alphameric) or algebraically greater in value (if numeric) than contents of Factor 2	—	Lower in sequence (if alphameric) or algebraically smaller in value (if numeric) than contents of Factor 2	—	Equal in sequence (if alphameric) or in value (if numeric) to contents of Factor 2
Table Look-Up (LOKUP)	If the table argument (Factor 2) is:	—	The nearest value higher than the search argument (Factor 1)	—	The nearest value lower than the search argument (Factor 1)	—	Equal to the search argument (Factor 1)

Figure 34. Summary of Conditions that Cause Calculation Resulting Indicators to Turn ON --in Arithmetic, Compare, and Table Look-up Operations

Line 11 specifies a table look-up operation (LOKUP). The entry of an indicator code (02) in "Equal" (Cols. 58-59) instructs the program to search the argument table for a value that exactly matches the contents of the field EMPLNO. If and when such a match is found, indicator 02 turns on.

Figure 34 is a summary of conditions--in arithmetic, compare, and table look-up operations--that cause Resulting Indicators assigned in cols. 54-59 to turn on.

Comments (Cols. 60-74)

The user may enter here any information he would like printed out, next to the other entries in the line, at object program generation time. Apart from this printout, the data is ignored by the program.

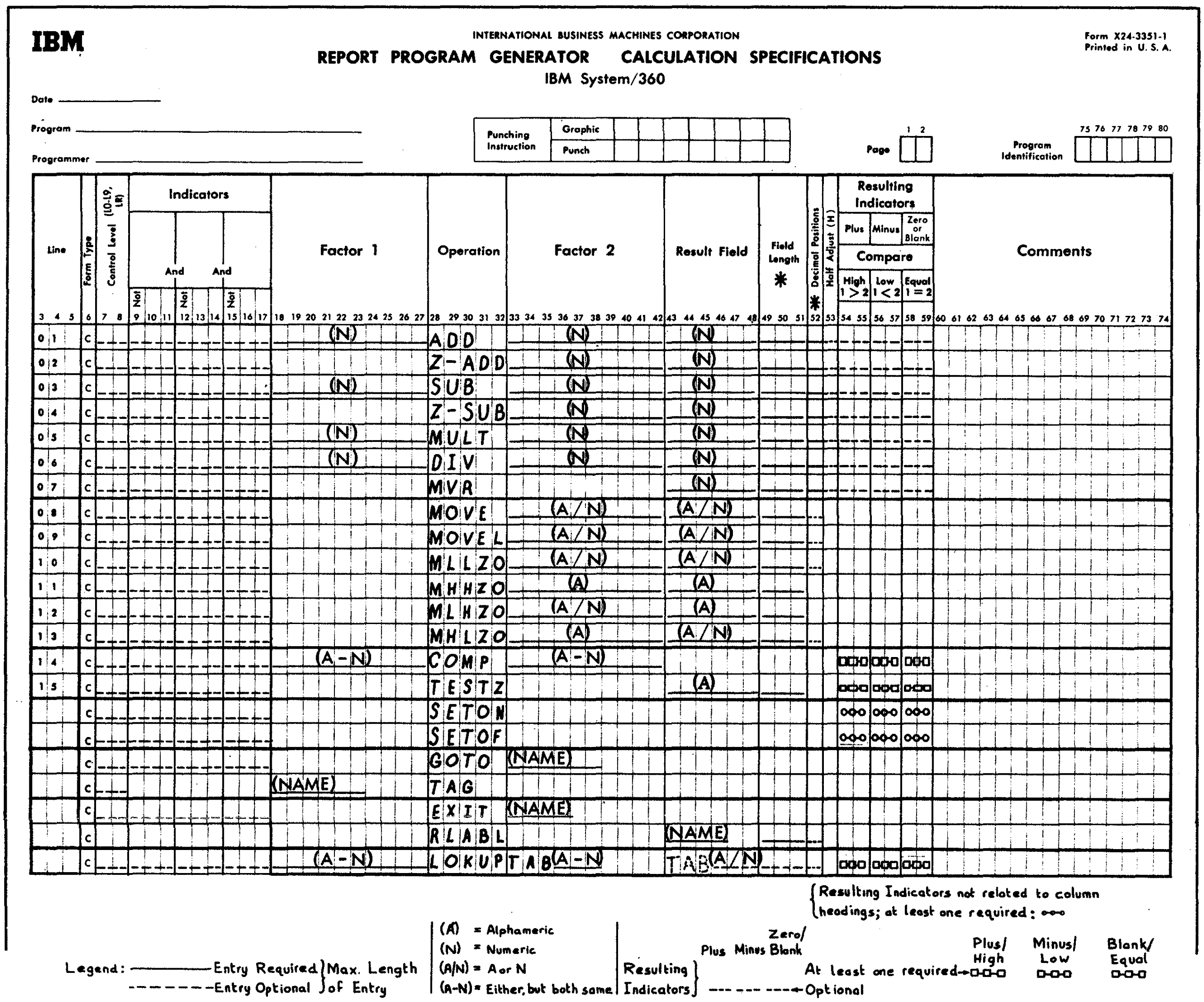
ENTRIES IN THE OPERATION FIELD (COLS. 28-32)

The code for the operation to be performed is entered left-justified (i.e., beginning in col. 28). Figure 35 itemizes, and briefly describes, the operations that can be performed, together with the corresponding mnemonic codes that are to be entered in cols. 28-32. The operations are grouped in Figure 35 by type. They are discussed by group, because some aspects of operations are unique to a type. For a more detailed survey of calculation operations, see Figure G1.

Figure 36 portrays graphically the calculation-specifications fields that apply to each operation code. The figure is repeated in Appendix G, as figure G2, for convenient reference.

Type of Operation	Operation	Operation Code (Cols. 28-32)
Arithmetic Operations	Add Factor 2 to Factor 1	ADD
	Clear Result Field and add Factor 2	Z-ADD
	Subtract Factor 2 from Factor 1	SUB
	Clear Result Field and subtract Factor 2	Z-SUB
	Multiply Factor 1 by Factor 2	MULT
	Divide Factor 1 by Factor 2	DIV
	Move Remainder of preceding division to a Result Field	MVR
Move Operations	Move Factor 2 into Result Field, right-justified	MOVE
	Move Factor 2 into Result Field, left-justified	MOVEL
	Move Zone from low-order position of Factor 2 to low-order position of Result Field	MLLZO
	Move Zone from high-order position of alphameric Factor 2 to high-order position of alphameric Result Field	MHHZO
	Move Zone from low-order position of Factor 2 to high-order position of alphameric Result Field	MLHZO
	Move Zone from high-order position of alphameric Factor 2 to low-order position of Result Field	MHLZO
Compare and Zone-Testing Operations	Compare Factor 1 to Factor 2	COMP
	Identify the zone in the high-order position of an alphameric Result Field	TESTZ
Setting Indicators	Set one, two, or three specific indicators on	SETON
	Set one, two, or three specific indicators off	SETOF
Branching within RPG and to external B.A.L Routines	Branch to another RPG calculation specification line	GOTO
	Identify the name in Factor 1 as a destination label to which GOTO may branch	TAG
	Branch to an external B.A.L. routine	EXIT
	Identify the name in Result Field as a field or Indicator to be referenced in the external B.A.L. routine	RLABL
Table Look-up Operations	Table Look-up	LOKUP

Figure 35. Calculation Operations



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

Form X24-3351-1
Printed in U. S. A.

Line	Form Type	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
01	C	(N)	ADD	(N)	(N)	
02	C		Z-ADD	(N)	(N)	
03	C	(N)	SUB	(N)	(N)	
04	C		Z-SUB	(N)	(N)	
05	C	(N)	MULT	(N)	(N)	
06	C	(N)	DIV	(N)	(N)	
07	C		MVR		(N)	
08	C		MOVE	(A/N)	(A/N)	
09	C		MOVEL	(A/N)	(A/N)	
10	C		MLLZO	(A/N)	(A/N)	
11	C		MHHZO	(A)	(A)	
12	C		MLHZO	(A/N)	(A)	
13	C		MHLZO	(A)	(A/N)	
14	C	(A-N)	COMP	(A-N)		
15	C		TESTZ		(A)	
	C		SETON			
	C		SETOF			
	C		GOTO	(NAME)		
	C	(NAME)	TAG			
	C		EXIT	(NAME)		
	C		RLABL		(NAME)	
	C	(A-N)	LOKUP	TAB(A-N)	TAB(A/N)	

Resulting Indicators not related to column headings; at least one required

Legend: ——— Entry Required, - - - - Entry Optional } Max. Length of Entry

(A) = Alphameric
(N) = Numeric
(A/N) = A or N
(A-N) = Either, but both same

Resulting Indicators: Plus, Minus, Zero/Blank — At least one required; Plus/High, Minus/Low, Blank/Equal; - - - Optional

* NOTE: The entries designated as required in Field Length and Decimal Positions are necessary only if the associated Result Field is not defined elsewhere.

Figure 36. Fields Pertinent to Each Operation Code

Explanation of Symbols Used in Figure 36

A solid straight line indicates that an entry is required in that field. A dotted straight line indicates that an entry in the field is optional.

In the Resulting-Indicators fields:

A straight dotted line signifies optional entries to which the headings Plus, Minus, and Zero/Blank apply.

A line connecting rectangles (□-□-□) signifies that an entry is required in at least one of the three fields, and that the headings High, Low, and Equal--or Plus, Minus and Blank--apply.

A line connecting circles (○-○-○) signifies that an entry is required in at least one of the three fields, but that the column headings are not pertinent.

Absence of any line, dots, or symbols signifies that an entry is not permitted in that field with that operation code.

The length of the line always represents the maximum entry. (It is to be understood that, where the line extends through all ten positions in Factor, this refers to the maximum size of a literal, but field names are limited to six characters.) Entries shown as required in Field Length and Decimal

Positions are necessary only if the associated Result Field is nct defined elsewhere (see Result Field, cols. 43-48, above).

All calculaticn specificaticns to be executed at detail time (ccls. 7-8 blank) must be entered ahead of all those tc be executed at total time (L-indicatcr in cols. 7-8). Within this grcuping, the calculation cperaticns are executed (cr bypassed--depending cn ccnditicning indicators) in the sequence in which they are entered--except when branching (see GOTO operaticn, belcw).

Note:
1. Whenever a field that was defined in the input specifications as packed (P in col. 43) is involved in an cperation, its length must be considered tantamount to a standard (unpacked) numeric input field with the same digit capacity. The general fcrmula is: field length in calculation specifications = 2n-1, where n = length of packed field in input specificaticns.

2. All data is output in true fcrm--ccmplements for negative values are not a consideration.

Arithmetic Operations

General Pcints Applicable to Arithmetic Operaticns

1. All source-data and result fields and all literals involved must be defined as numeric.

2. The program performs autcmatic decimal alignment cf factors and results.

3. Result Fields used in arithmetic operations become signed plus or minus (i.e., hexadecimal C or D) the first time an operaticn is executed with the field as result field, even if the field was read as input without a zone cverpunch in the low-crder position (or with a character in EBCDIC-table column A, B, or E). Zero totals are signed plus (EBCDIC-table column C) ; the result cf an arithmetic operation is never minus zero with this RPG.

   If the field is thereafter punched or printed without editing, zero-suppression, or first removing the zone by a calculation specification (see MHLZO or MLLZO, below), a zcne occurs cver the digit in the lcw-order position, as follows:
      11-punch if field ccntents are negative;
      12-punch if field contents are positive or zero.

   Similarly, if the low-order position of such field is later moved into an alphameric field, the sign may move with it (see MOVE or MOVEL instruction, below). If that position in the new field is then tested fcr zcnes (see TESTZ, below), a positive or zero value will yield Plus (indicator in cols. 54-55).

4. The prcgram perfcrms arithmetic operations, and signs the results, in accordance with the algebraic laws of signs --see Figure 37.

OP. CODE	SIGN IN FACTOR 1	SIGN IN FACTOR 2	SIGN OF RESULT FIELD*	SIGN OF REMAINDER*
ADD	+	+	+	
	-	-	-	
	+	-	Sign of larger absolute value	
	-	+	Sign of larger absolute value	
Z-ADD		+	+	
		-	-	
SUB	+	+	- if Factor 2 > Factor 1; otherwise +	
	-	-	- if |Factor 1| > |Factor 2|; otherwise +	
	+	-	+	
	-	+	-	
Z-SUB		+	-	
		-	+	
MULT	+	+	+	
	-	-	+	
	+	-	-	
	-	+	-	
DIV	+	+	+	+
	-	-	+	-
	+	-	-	+
	-	+	-	-
MVR			+	+
			-	-

*Note: Excluding a Result of zero.
A result of zero is always signed plus.

Legend: | | represents "absolute value of"

Figure 37. Signs in Arithmetic Operations

5. All arithmetic-operation source fields must contain valid digits: Considering an input field before it is packed by the program, the character in each column must be represented in rows 0-9 of the EBCDIC table (Appendix D, Figure D1). For packed input data, the equivalent valid EBCDIC characters were described under <u>Packed</u> (col. 43), in <u>Input Specifications</u>.

   Any characters that do not represent digits (or, digit plus sign in the low-order position) cause an abortive program stop.

6. The Factors and Result Field in an arithmetic operation may each involve the same or different field names. (E.g.: A + B = C, or A + C = C', or C + B = C', or C + C = C', or A + A = C, or B + B = C.)

7. The execution of any arithmetic operation may be made contingent on the status of conditioning indicators specified in Control Level (cols. 7-8) and/or in Indicators (cols. 9-17).

8. Resulting Indicators may be assigned to Plus, Minus, and/or Zero-or-Blank (cols. 54-59) to test the result of any arithmetic operation.

9. With one exception (DIV, when followed by MVR), the result of any arithmetic operation can be half-adjusted (H in col. 53).

10. Fields that provide only source data--as contrasted with receiving result data--for an arithmetic operation are not changed in any way (including sign status) by the operation. Even in the multiplication and division operations, the original values of Factor 1 and Factor 2 are preserved (unless the same field name is used for the Result Field).

11. The Result Field must be defined in the same, or another, calculation specification line, or it must have been defined in the input specifications.

12. When the Result Field is used as a Factor, the user must make sure that the Result Field is large enough to accommodate the new product. If the length of the Result Field is intentionally specified too small (warning message RG117 - Result Field may not be large enough - is printed), and this length is specified as an even number (in the unpacked format), the user must know the digit in the high-order position of the Result Field, since otherwise the digit is lost.

Relationship Between Size of Factors and Results (See also Figures 31 and 32).

Source-data fields and result fields are limited to a maximum length of 15 positions; i.e., 15 digits plus sign. (Internally, in the CPU, this represents 8 bytes.)

However, the immediate (temporary) result of an arithmetic operation (in a work area assigned by the program)--before it is moved by the program to the Result-Field area--may be as large as can be produced by the source-data fields and the operation. This presupposes that the initial result contains enough decimal places to drop--and that the Decimal-Positions entry (col. 52) specifies the appropriately reduced number of decimal places--so that the retained number of digit positions does not exceed 15.

If the result positions to the left of the decimal point, together with the number of decimal places to be retained (per col. 52), exceed the Result-Field Length specified (cols. 49-51)--which must not be greater than 15--a corresponding number of high-order positions is dropped before transfer to the Result-Field location, and the status of any Resulting Indicators assigned reflects the truncated final result. If the Decimal-Positions specification is greater than the actual number of decimal places that result from the operation, an appropriate number of zeros is appended at the right and the number of high-order positions is reduced accordingly, to remain within the Field Length specified.

<u>Note</u>: In the operations ADD, SUB, Z-ADD and Z-SUB, arithmetic overflow may cause a Resulting Indicator assigned to a different result status (+, -, or 0) to be turned on for that specification line, or cause all the indicators to be turned off. The conditions to which this can apply are those listed as requiring only six bytes (or 18, in the case of Z-SUB) of core storage under <u>Processing of Object Program</u>, Calculation Specifications, in Appendix A.

During program execution, no indication of arithmetic overflow is given. (See <u>Programming Tips</u> for a technique to accomplish the equivalent, for result-field-length specifications of less than 15.) During program generation, a warning message ("Result field may not be large enough") is printed if the size of the multiplication or division factors involved could theoretically cause the result, after proper decimal alignment, to exceed the length specified for Result Field. The same message is printed if a Factor field in an addition or subtraction operation exceeds the Result Field size, after decimal alignment. (Note: This message is not provided for the MVR operation.) Often, by familiarity with the particular data involved, the user will know that a result field smaller than the theoretical maximum suffices.

Guarding against exceeding result-field capacity is based on the rules of algebra:

1. Addition and Subtraction

   The maximum number of significant digits that can result from the operation is equal to the number of:

   $$\left.\begin{array}{l}\text{decimal places}\\ \text{+ places to the}\\ \text{left of the}\\ \text{decimal point}\\ \text{+ 1}\end{array}\right\}\begin{array}{l}\text{each from the}\\ \text{factor with the}\\ \text{greater number of}\\ \text{such places}\end{array}$$

For example:

Factor 1: + 9994567898642.06

-(SUB) Factor 2: - 5680975310.25791

= + 10000248873952.31791

Such an operation is legitimate, although the initial result exceeds 15 positions--provided the Result-Field-Length (cols. 49-51) and Decimal-Positions (col. 52) specifications have the proper relationship, and the Result-Field-Length specification is not greater than 15. For instance:

Specified: Result-Field Length	Specified: Decimal Positions	final Result-Field contents	
15	1	+10000248873952.3	OK
15	0	+010000248873952	OK
14	0	+10000248873952	OK
15	2	+0000248873952.31	high-order digits truncated
12	0	+000248873952	
15	6	+248873952.317910	

The preceding example illustrates that any number of decimal places may be dropped (by appropriate Decimal-Positions entry in col. 52) to fit the result within the specified Result-Field Length (cols. 49-51)--which must not exceed 15. If the specified Result-Field Length is then greater than the retained positions, the significant digits are preceded in the Result Field by an appropriate number of leading zeros. If the specified Result-Field Length is too small to accommodate the retained positions--even if no decimal places are retained (0 in col. 52)--a corresponding number of the most-significant positions is lost. The last entry shows that, if the number of Decimal Positions specified exceeds the significant ones that can result from the operation, an appropriate number of zeros is appended at the right and a corresponding number of high-order positions truncated.

2. Multiplication

The maximum number of significant digits (including decimal places) that can result from the operation is equal to the sum of the number of positions in the two factors.

The resulting number of positions always includes a number of decimal places equal to the sum of the number of decimal places in the two factors. For example:

Factor 1: -9876418.34255073

x(MULT)

Factor 2: + 1234.68951027324

= -12194310126.6076055217608614652

i.e., 15 places x 15 places can result in 30 places, and 8 decimal places x 11 decimal places always results in 19 decimal places. The total of 30 places, minus 19 decimal places, equals 11 non-decimal places.

Thus, in this example, any Result-Field-Length specification from 11 to 15, with associated Decimal-Positions specification from 0 to 4, respectively, prevents loss of any high-order position in the result field. For instance:

Specified: Result-Field Length	Specified: Decimal Positions	final Result-Field contents	
15	4	-12194310126.6076	OK
15	2	-0012194310126.60	OK
12	1	-12194310126.6	OK
12	0	-012194310126	OK
11	0	-12194310126	OK
11	1	-2194310126.6	high-order digits truncated
15	6	-194310126.607605	

The following formula can be used to determine whether leftmost positions will be truncated:

$$L_1 - D_1 + L_2 - D_2 + Dr = Lr \text{ where}$$

Lr = Result-Field Length specified (cols. 49-51) $\leq$ 15
L_1 = length of Factor 1
D_1 = number of decimal places in Factor 1
L_2 = length of Factor 2
D_2 = number of decimal places in Factor 2
Dr = number of decimal places specified (col. 52) to be retained in the result field (product)

If Lr turns out to be greater than the specified (cols. 49-51) Result-Field Length:

a. Either Lr must be increased (but it must not exceed 15); and/or

b. Dr must be reduced (but it cannot be negative); and/or

c. It must be known, from the nature of the data, that the product will contain appropriately fewer significant high-order digits than the theoretical maximum.

If none of the above three techniques can be employed to satisfy the equation, the multiplication cannot be performed in its present form.

A possible remaining stratagem is to increase the number of decimal places defined for the Factors (elsewhere in the calculation specifications, or in the input specifications, as the case may be), without increasing the overall length of the Factors. This provides more decimal places that can be dropped to fit the product within the limit Lr, by increasing D_1 and/or D_2. While this reduces the order of accuracy of the result, it prevents truncation of most-significant positions. For example:

```
Factor 1:       135.9
Factor 2:    x   85.2
Product    = + 11578.68
```

If Lr = 4 and Dr = 0 (0 in Decimal Positions, col. 52), the Result Field would contain 1578--a loss of the most-significant digit. If a Result Field greater than 4 positions cannot be used for some reason (say, room in the card) --and, of course, the illustration is similarly applicable where Lr = 15, and therefore cannot be increased--the definition of number of decimal places in the Factors can be changed:

```
Factor 1:        13.59
Factor 2:      x 85.2
             + 1157.868
```

If Lr = 4 now, and Dr = 0, only the least-significant digit to the left of the original decimal point (namely, 8) is lost. (With half-adjustment, the result becomes 1158.) In subsequent operations with this result, the user then bears in mind the misplaced hypothetical decimal point. For instance, if the field is to be printed, a constant 0 can be appended in the output-format specifications, and the value is then printed as 11570 (or 11580, if half-adjusted during the multiplication operation). This provides a value of the proper number of places, and accurate to four significant digits.

3. Division

Decimal Positions. The number of decimal places in the result (quotient) of a division equals the number of decimal places in the dividend less those in the divisor. The RPG program pads either the dividend or the divisor with additional zeros at the right, if this is necessary to yield the number of decimal places specified in col. 52 (Decimal Positions)--which cannot be negative. If half-adjustment is specified (H in col. 53), the program automatically modifies padding to yield one extra decimal position (which is dropped again after half-adjustment).

Two examples:

1. Half-adjustment not specified: If the dividend is 123.643 (3 decimal places), and the divisor is 1.41 (2 decimal places), the quotient contains 1 decimal place (3-2).

   If col. 52 specifies 2 decimal places for the quotient, the program adjusts the dividend to 123.6430 (now, 4 decimal places in the dividend - 2 decimal places in the divisor = 2 decimal places in the quotient). If, on the other hand, 0 is specified in col. 52, the program leaves the dividend unaltered at 123.643, but adjusts the divisor to 1.410 (now, 3 - 3 = 0).

2. Half-adjustment specified (H in col. 53): If the dividend is 579321 (0 decimal places), and the divisor is .46 (2 decimal places), the number of decimal places in the result would be negative (0 decimal places in dividend - 2 in divisor - 1 for half-adjust), which is not possible. A minimum of three 0s must therefore be added to the dividend by the program.

   If col. 52 specifies 0 decimal places for the result, the program adjusts the dividend to 579321.000: 2 decimal places in the dividend + 1 extra dividend place for half-adjustment of quotient - 2 decimal places in the divisor = 1 decimal place in the initial result. After half-adjustment, no decimal place is retained.

   If 3 is specified in col. 52, the program adjusts the dividend to 579321.000000: 5 decimal places in the dividend + 1 extra dividend place for half-adjustment of quotient - 2 decimal places in the divisor = 4 decimal places in the initial result. After half-adjustment, 3 decimal places are retained.

   Expressed by formulas:

   1. Without half-adjustment specified.

      $A + D_1 - D_2 = Dr \ (0 \leq Dr \leq 9)$, where

A = Adjustment factor
D_1 = number of decimal places in Factor 1 (dividend)
D_2 = number of decimal places in Factor 2 (divisor)
Dr = number of decimal places specified (in col. 52) for Result Field (quotient). $0 \leq Dr \leq 9$ states that the number of decimal places in the final quotient must be zero or greater but no greater than nine, because these are the limits for the Decimal-Positions entry in col. 52.

If the equation is satisfied with $A = 0$, the dividend and divisor as stored (under their field names or as literals) fit the result requirements.

If $A > 0$, the program pads the dividend / If $A < 0$, the program pads the divisor — with a number of zeros, in decimal positions at the right, corresponding to the absolute value of A.

2. With half-adjustment (rounding) specified (H in col. 53)

$A + D_1 - D_2 = Dr + 1 \quad (0 \leq Dr \leq 9)$

This equation is identical to the previous one with this exception: The dividend must contain one more decimal place to yield the same number of decimal places in the final result. An extra decimal position is needed in the initial calculated quotient for half-adjustment; thereafter, it is dropped.

Size Restrictions. The rules pertaining to decimal positions in division are defined above. In addition, total Factor and Result-Field sizes are limited to a maximum of 15 positions each, including any zeros appended by the program when padding (see above).

Expressed as equations related to the decimal-places formulas above:

$L_1 + A \text{ (if } A > 0) \leq 15,$

where

L_1 = unpadded (original) length of Factor 1 (dividend)

$L_2 + |A| \text{ (if } A < 0) \leq 15,$

where

L_2 = unpadded (original) length of Factor 2 (divisor)

Alternatively, considered independently of the previous formulas, and before padding by the program, factor sizes and number of decimal positions must satisfy both of the following two equations for the division operation to be executed:

$L_2 + D_1 - D_2 - Dr \leq 15$, and

$L_1 - D_1 + D_2 + Dr + H \leq 15,$

where

L_1 = length of Factor 1 (dividend): ≤ 15
D_1 = number of decimal positions in Factor 1: ≤ 9
L_2 = length of Factor 2 (divisor): ≤ 15
D_2 = number of decimal positions in Factor 2: ≤ 9
Dr = number of decimal places specified for result (quotient): ≤ 9
H = 0, if half-adjustment not specified
= 1, if half-adjustment specified (H in col. 53)

Size of Quotient (Result). Assuming that the divisor field always contains a significant digit in its highest-order position, the quotient contains a number of positions equal to the size of the dividend plus 1, less the size of the divisor, and less 1 if half-adjustment (rounding) is specified. Dividend and divisor sizes refer to padded factors (see above).

If the divisor field always contains a significant digit in the highest-order position, the formula is:

$Lr = 1 + F1p - F2p - H,$

where

Lr = minimum length of Result Field required to accommodate quotient (after half-adjustment, if any)
$F1p$ = length of Factor-1 (dividend) field, after padding (if any)
$F2p$ = length of Factor-2 (divisor) field, after padding (if any)
H = 0, if half-adjustment not specified
= 1, if half-adjustment specified (H in col. 53)

If the position of the highest-order significant digit in the divisor field may vary, the result-field must be larger to accommodate all totals. The result-field length must be increased by a number equal to the maximum number of leading zeros in the divisor field.

Size of Remainder. The remainder (which can be salvaged by an MVR operation--see below) contains a number of positions equal to the length of the divisor, after padding (if any). Its number of decimal places is equal to that in the dividend, after padding (if any).

Effects of Each Operation Code

ADD (Add)

The contents of the field in Factor 2 or the literal entered in Factor 2 is added, algebraically, to the literal or the contents of the field in Factor 1. The result of this addition is placed into the result field specified in cols. 43-48, and replaces any previous data in the result field. Any excess positions in the result field are set to zero.

Factor 1 (say, A), Factor 2 (say, B), and the Result Field (say, C) may all be different fields: A + B = C.

Factor 1 or Factor 2 may be the same field (i.e., have the same field name) as the Result Field. The value of the contents of the result field is then increased, algebraically, by the value represented by Factor 1 or Factor 2, respectively: operation A + C = C' or C + B = C'.

Factor 1 and Factor 2 may be the same (i.e., have the same field name), but may be different from Result Field. Twice the value of either Factor then becomes the result: operation A + A = C = 2A.

Factor 1, Factor 2, and the Result Field may all be the same field (i.e., have the same field name). The absolute value of the contents of the result field is then doubled: operation C + C = C' = 2C.

Z-ADD (Zero and Add)

The result field is set to zero before the contents of the field or the literal in Factor 2 is added algebraically into the cleared result field. Factor 1 must be left blank.

If a literal of 0 is entered in Factor 2, Z-ADD, in effect, causes the result field to be cleared to plus zero. (However, see SUB, below, for a preferred method.)

SUB (Subtract)

The contents of the field in Factor 2 or the literal in Factor 2 is subtracted algebraically from the literal or the contents of the field in Factor 1. The result of this subtraction is placed into the specified result field, and replaces any previous data in the result field. Any excess positions in the result field are set to zero.

Factor 1, Factor 2, and the Result Field may all be different fields: A - B = C.

Factor 1 may be the same field (i.e., have the same field name) as the Result Field. The value in the result field is then reduced, algebraically, by the value in Factor 2: operation C - B = C'.

Factor 2 may be the same field (i.e., have the same field name) as the Result Field. The new result-field value is then the negative of the original result-field value, increased algebraically by the value in Factor 1: operation A - C = C'.

Factor 1 and Factor 2 may be the same field (i.e., have the same field name), but may be different from Result Field. The result is then zero: operation A - A = C = +0. (However, see immediately below for a method of setting the result field to zero that is usually preferable.)

Factor 1, Factor 2, and the Result Field may all be the same field (i.e., have the same field name). This sets the result field to +0 (i.e., all zeros, signed plus): operation C - C = C' = +0.

Note: The operation C - C = C' = +0 is recommended for clearing a numeric field; this method never consumes more core storage space, and often uses less, than other methods (Z-ADD literal 0, or MOVE of 0s or blanks, for instance).

Z-SUB (Zero and Subtract)

The result field is set to zero before the contents of the field or the literal in Factor 2 is subtracted, algebraically, into the cleared result field. This places the negative of the Factor-2 value in the result field. Factor 1 must be left blank.

If a literal of 0 is entered in Factor 2, Z-SUB, in effect, causes the result field to be cleared to plus zero. (However, see SUB, above, for a preferred method.)

Note: Although the result field is cleared before Factor 2 is subtracted into it, the former contents of the result field are available as Factor 2 (i.e., -C = C' is feasible). A Z-SUB operation with the same field name in Factor 2 as in the Result Field is the simplest way to reverse the sign of data.

MULT (Multiply)

The contents of the field or the literal in Factor 1 (multiplicand) is multiplied, algebraically, by the literal or the contents of the field in Factor 2 (multiplier). The product of this multiplication is placed in the result field specified in cols. 43-48, and replaces any previous data in the result field. Any excess positions in the result field are set to zero.

In general, execution time of a multiplication operation is minimized if the multiplicand (Factor 1) has the smaller average sum-of-digits values (crossfoot sum of the digits). Unless knowledge of particular values involved in the factors indicates otherwise, the smaller field may be assumed to contain the smaller average sum-of-digits value, and should therefore be assigned as the multiplicand (Factor 1).

Examples of sum-of-digits values:

Factor = 12348--sum of digits = 18
Factor = 0.92 --sum of digits = 11

Factor 1, Factor 2, and the Result Field may all be different fields: A x B = C.

Factor 1 or Factor 2 may be the same field (i.e., have the same field name) as the Result Field. The new result is then the product of the former result-field contents and a Factor: operation A x C or C x B = C'.

Factor 1 and Factor 2 may be the same field (i.e., have the same field name), but different from Result Field. The result is then the square of either Factor: operation A x A = C = + A^2.

Factor 1, Factor 2, and the Result Field may all be the same field (i.e., have the same field name). The new result is then the square of the former result-field value: operation C x C = C' = + C^2.

Note: When the result field is used as a Factor, the user must make sure that the Result Field is large enough to accommodate the new product. This implies that, if there are significant digits to the left of the defined decimal position in either factor, an equivalent number of high-order positions of zero may have to exist in the original result-field value. At any rate, a diagnostic warning message ("Result field may not be large enough") will be printed at time of object-program generation.

For further details on multiplication, see sections above: General Points Applicable to Arithmetic Operations and Relationship Between Size of Factors and Results; and also Figures 31, 32, and 37.

DIV (Divide)

The contents of the field or the literal in Factor 1 (dividend) is divided, algebraically, by the literal or the contents of the field in Factor 2 (divisor). The result of this division operation (the quotient) is placed in the result field specified in cols. 43-48, and replaces any previous data in the result field. Any excess positions in the result field are set to zero. The remainder is accessible only if a Move Remainder (MVR) operation is next; otherwise it is lost.

A dividend (Factor 1) of zero yields a quotient of zero. A divisor (Factor 2) of all-zero is not permitted; it will cause an error stop.

Half-adjustment (rounding) of the quotient is not permitted if the Move Remainder operation (MVR) follows (see below).

Factor 1, Factor 2, and the Result Field may all be different fields: operation A ÷ B = C.

Factor 1 may be the same field (i.e., have the same field name) as the Result Field: operation C ÷ B = C'.

Factor 2 may be the same field (i.e., have the same field name) as the Result Field: operation A ÷ C = C'.

Factor 1 and Factor 2 may be the same field (i.e., have the same field name) and be either different from the Result Field or the same field. This yields a quotient of 1 (to the left of any decimal point specified for the Result Field, with zeros in all decimal positions): operation A ÷ A = C = 1., or C ÷ C = C' = 1.--an inefficient method of setting a field to 1.

For further details on division, see sections above: General Points Applicable to Arithmetic Operations and Relationship Between Size of Factors and Results; and also Figures 32 and 37.

MVR (Move Remainder)

The remainder from a Divide (DIV) operation is transferred--by a zero-and-add operation supplied by the program--to any result field specified in cols. 43-48 of this specification line. It replaces any previous data in that result field. Any excess positions in the result field will contain zeros. Factor 1 (cols. 18-27) and Factor 2 (cols. 33-42) must be left blank.

MVR is an arithmetic operation. Therefore:

> The result is signed. The sign of the remainder is the same as the sign of the dividend. If the dividend was unsigned, the result of an MVR operation will be signed plus--since results of arithmetic operations are always signed.
>
> Decimal alignment is performed by the program;
>
> Half-Adjustment (rounding) may be specified (H in col. 53);
>
> Resulting Indicators (cols. 54-59) may be assigned;
>
> If the Result Field is not large enough to accommodate all the high-order positions in the remainder (after appropriate decimal alignment), a corresponding number of the most-significant (leftmost) positions is lost. No warning message or error stop occurs, either at generation or object-program execution time, if the Result Field is not large enough.

The length of the remainder of a division is equal to the length of the divisor. "Length of the divisor" refers to the actual divisor used by the program in the DIV operation; this can be longer than the field length specified for Factor 2, if the divisor was padded by the program--see <u>Relationship Between Size of Factors and Results: Division,</u> above.

The value of the remainder (R) can be determined by the following formula

$$R = \text{Dividend} - \text{Divisor} \times \text{Quotient}$$

When the MVR operation is used, it must follow immediately (i.e., in the next specification line) after the pertinent Divide (DIV) operation; and Half-Adjust (H in col. 53) must not be specified for that particular DIV operation. The related DIV and MVR specification lines must have the same conditioning indicators, recorded in the same sequence; i.e., the contents of cols. 7-17 must be identical for the two specification lines.

Most likely applications of MVR are:

1. To test whether the remainder is zero (illustrated in Figure 38), and

2. To perform division expansion (double-precision division).

(See also Figure E11 for another application.)

Figure 38 shows some specifications for arithmetic operations. Specifications for such operations have also already been illustrated in Figures 5, 9, 10, 12, and

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-3351-1
Printed in U. S. A.

Date ____

Program ____

Programmer ____

Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line (3-5)	Form Type (6)	Control Level (L0-L9, LR) (7-8)	Indicators: Not (9-11)	And: Not (12-14)	And: Not (15-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field (43-48)	Field Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Plus / Compare High 1>2 (54-55)	Minus / Low 1<2 (56-57)	Zero or Blank / Equal 1=2 (58-59)	Comments (60-74)
01	C		Ø5				Z-ADD	ALPHA	BETA							
02	C		Ø5			BETA	ADD	-12.32	BETA	5	2			1Ø		
03	C		Ø5			GAMMA	ADD	DELTA	EPSILN	4	1					
04	C		Ø5	1Ø		ALPHA	SUB	ALPHA	ALPHA							
05	C		42				Z-SUB	ZETA	ETA	6	Ø			25		
06	C	L1				EPSILN	MULT	EPSILN	THETA	7	1	H			12	
07	C	L1	N12			864329.Ø15	DIV	THETA	IOTA	1Ø	3					
08	C	L1	N12				MVR		KAPPA	6	3	H			Ø8	
09	C															
10	C															

Figure 38. Examples of Specifications for Arithmetic Operations

33; and Figure 36 identifies the pertinent specification fields for each operation.

Explanation of Entries in Figure 38

The fields named ALPHA, GAMMA, DELTA, and ZETA are assumed to have been defined in the input specifications or elsewhere in the calculation specifications. Note that all detail-time specifications precede all total-time specifications--an absolute requirement. Within these two cycle-time segments, the specifications are executed in the order in which they appear.

Specifications line 01. The field named BETA is set to zero, and the contents of ALPHA are then added algebraically to the value zero in BETA. The field length and decimal places for BETA are defined in line 02; they could equally well be defined in line 01 instead, or in both lines--provided they are defined equally in both lines. The decimal point of ALPHA is aligned to accord with the two decimal places in BETA. Any excess positions in BETA contain zero; and excess high-order positions in ALPHA, beyond the capacity of the BETA field, are lost.

Factor 1 is not used with Z-ADD. The specifications in this line are executed at detail time (cols. 7-8 blank), provided indicator 05 is on.

Line 02. The value -12.32 is added algebraically to (i.e., 12.32 is subtracted from) the contents of BETA. The number of decimal places is equal in both factors. Thus:

```
            BETA  XXX.XX
            ADD  - 12.32
   Result:  BETA ±XXX.XX
```

Indicator 10 turns (or remains) on if the result is negative; otherwise it remains (or turns) off. The specifications in this line are executed at detail time, provided indicator 05 is on.

Line 03. The contents of the field named DELTA are added algebraically to the contents of GAMMA. The result is stored at the location assigned by the program to EPSILN, defined as four positions long, the fourth position being a decimal place. The specifications in this line are executed at detail time, provided indicator 05 is on.

Line 04. If indicator 05 is on, and the value in BETA was last negative (indicator 10 on), then--at detail time--ALPHA is cleared to plus zero. For instance:

```
ALPHA =  1234               -1234
 SUB:    1234   or  SUB:    -1234
      = +0000             = +0000
```

This is as efficient (in terms of core storage consumption) as, or more efficient than, any other technique for setting a numeric field to zero.

Line 05. The field named ETA is set to zero, and the contents of ZETA are then algebraically subtracted from the value zero in ETA. ETA is defined as containing no decimal places, and being six positions long.

If the value in ZETA is positive, indicator 25 will be on after the operation (subtraction of a positive value from zero yields a negative result).

Factor 1 is not used with Z-SUB. The specifications in this line are executed at detail time, provided indicator 42 is on.

Line 06. The value in the field named EPSILN is squared. (The result will always be positive or zero: the product of two positive or two negative values is positive.) Since EPSILN consists of 4 positions including one decimal place (see line 03), the product will contain 2 decimal places within a maximum of 8 positions total length. By specifying only 1 decimal position for THETA, a total length of 7 positions for THETA is certain to accommodate the maximum result. The second decimal position is retained for half-adjustment (H in col. 53), and then dropped before the final result is placed into the location of THETA.

Indicator 12 will be on after the operation if the final (half-adjusted) result, after the second decimal place has been dropped, is zero (actually, plus zero).

The specifications in this line are executed at total time (L-indicator in cols. 7-8), and provided indicator L1 is on.

Line 07. The literal in Factor 1 is divided algebraically by the value in THETA. The operation takes place at total time, if L1 is on, but is suppressed if the value in THETA is zero (indicator 12 on if THETA is zero): division is not possible with a divisor of zero.

THETA contains 1 decimal position (see line 06), and 3 are defined for the Result Field (IOTA). Since the literal in Factor 1 has only 3 decimal places, the program pads the dividend by appending a 0 as fourth decimal place; then, 4 decimal places in the dividend minus 1 in the divisor will provide the 3 decimal places called for in the quotient.

The dividend, after padding, is 10 positions long. Providing 10 positions in the result field (IOTA) allows for any number of leading zeros in the divisor. If it is known that there is always a significant digit in the high-order position of THETA, IOTA need be no larger than 4 positions: 10 dividend positions (after padding) - 7 divisor positions + 1 = 4--see Relationship Between Size of Factors and Results: Division, above.

Half-adjustment (rounding)--which, if specified, would have padded the dividend by an additional zero--is not permitted because an MVR operation follows.

Line 08. The field named KAPPA is set to zero. Then, the remainder from the division operation in line 07 is placed in KAPPA. THETA, the divisor, is 7 positions long, including 1 decimal place. Therefore, the remainder is also 7 positions long. The dividend, after padding, includes 4 decimal places; therefore, the remainder has 4 decimal places. KAPPA is to contain 3 decimal positions (3 in col. 52); therefore, 6 positions will accommodate the entire remainder after the last decimal place has been dropped. Half-adjustment (specified by H in col. 53) occurs before the last decimal position is dropped. The half-adjusted (rounded) remainder is henceforth available at the location named KAPPA; without the MVR operation it would be lost.

Indicator 08 is on after the operation if the remainder, after half-adjustment and dropping of the fourth decimal position, was zero.

The Factor fields are not used with MVR.

Note that the MVR specification must be in the line immediately following the pertinent DIV operation, and that the two specification lines must have identical entries for conditioning indicators (cols. 7-17).

Move Operations

These operations move part or all of the literal in Factor 2, or of the contents of the field named in Factor 2, to the field named in Result Field. The contents of the field or the literal in Factor 2 remains unchanged. The data moved to the result field replaces the former contents of the corresponding positions of the result field. The Result Field must be defined in the same, or another, calculation specification line, or it must have been defined in the input specifications. Move operations differ from arithmetic operations in several significant respects. Points generally applicable to move operations follow:

1. No automatic decimal alignment is performed by the program. Nevertheless, a numeric result field must be defined as numeric somewhere by an entry (0-9) in Decimal Positions (col. 52), which also locates the decimal point for possible compare or arithmetic operations with that field.

2. When data is moved only to a portion of the result field, the contents of the remaining portion are not changed.

3. At the beginning of program execution, data fields are set to:

   Blank (EBCDIC 40) in all positions--if defined as alphameric;

   Zero (EBCDIC 0) in all digit positions, with low-order positions unsigned (lowest-order half-byte EBCDIC F)--if defined as numeric.

   Therefore, if no data has been placed in a result field by a prior operation, any portion of the result field that exceeds the source field in a move operation remains blank or zero (alphameric or numeric field, respectively).

   Note: See Appendix D for code structure.

4. A numeric result field is only signed (other than hexadecimal F) if it was signed before the move, or if a sign is moved into its low-order position. Also, a sign in the low-order position of a numeric field can be removed (i.e., hexadecimal F can be placed in the sign position).

5. A result field can be minus zero: if only zeros and a minus sign are moved (or no sign position is moved but the result field previously contained a minus sign), and no significant digits remain in the result field, it will contain zeros and be signed minus. If the sign position is then tested by TESTZ (assuming the field is then alphameric), a Resulting Indicator assigned to Minus (cols. 56-57) will turn on.

6. Data may--with limitations, defined below--be moved from an alphameric field to a numeric field, and vice versa.

7. The digit portions of numeric fields are not restricted to EBCDIC-table rows 0-9 (see Appendix D, Figure D1); i.e., no test is made that they contain only valid digits that can be used in arithmetic or editing operations.

8. Half-adjustment is not possible; therefore, col. 53 must be left blank.

9. Resulting Indicators cannot be assigned; therefore, cols. 54-59 must be left blank.

10. The execution of move operations may be conditioned by indicator entries in Control Level (cols. 7-8) and Indicators (cols. 9-17).

11. Only one source-data field (Factor 2) is involved in any move operation. Factor 1 must be left blank.

12. Maximum field sizes--
    Numeric fields: 15 positions
    Alphameric fields: 256 positions.

    When an alphameric field is moved to a numeric field, or vice versa, only 15 positions can be transferred.

    Note: The user need not concern himself with the fact that numeric fields are packed (so that, for example, 15 digits are contained in 8 bytes): the program automatically performs the required packing and unpacking, and corrects for any differences in half-bytes utilized. The user treats field lengths as though packing were not a consideration; for example, a field of 15 digits has a field-length specification of 15.

Effects of Each Operation Code

MOVE (Move Right-Aligned)

The contents of the field or the literal in Factor 2 is moved into the specified result field, right-aligned.

If the result field is longer than Factor 2, the excess left-hand positions of the result field remain unchanged. If the result field is shorter than Factor 2, the contents of only the equivalent number of right-hand positions of Factor 2 are placed in the result field.

A source field or literal defined as alphameric can be moved to a result field defined as alphameric or numeric; also, a source field or literal defined as numeric can be moved to a result field defined as alphameric or numeric.

When an alphameric field or literal is moved to a field defined as numeric (i.e., with Decimal-Positions entry in col. 52 wherever the field is defined):

The digit portion of each position to be moved is transferred from the source field (Factor 2) to the equivalent position of the result field; a blank is transferred as zero.

The zone portion of the low-order position of the source field (Factor 2) is also transferred to the low-order position of the result field. Zones in other positions of the source field are not transferred.

Note:
1. The transfer of the low-order-position zone from an alphameric field to a numeric result field adheres to the following rules, so that valid signs for arithmetic operations result. (Refer to EBCDIC table, Appendix D, Figure D1):

    a. If the source position (in Factor 2) contains any of the punch combinations in EBCDIC-table column E or F, the E- or F-zone, respectively, is transferred to the result field, and the result field is considered positive (or zero, if entire result-field is zero).

    b. If the source position contains any of the punch combinations in EBCDIC-table column D, or an EBCDIC 60, D-zone (minus) is transferred to the result field, and the result field is treated as negative (or zero, if entire result field is zero).

    c. All other punch combinations in the source position are transferred to the result field with C-zone (plus), and the field is treated as positive (or zero, if entire result field is zero).

2. If a numeric literal is signed, the sign is recorded in the leftmost position (col. 33) of Factor 2. Nevertheless, the program signs the low-order--not the high-order--position of the literal. Therefore, if the literal is used in a move operation, the sign is in the proper position.

Figure 39 portrays the alignment and zone transfer in MOVE operations. Figure 41 includes specifications for some MOVE operations in Figure 39 (as well as for the MOVEL operations in Figure 40).

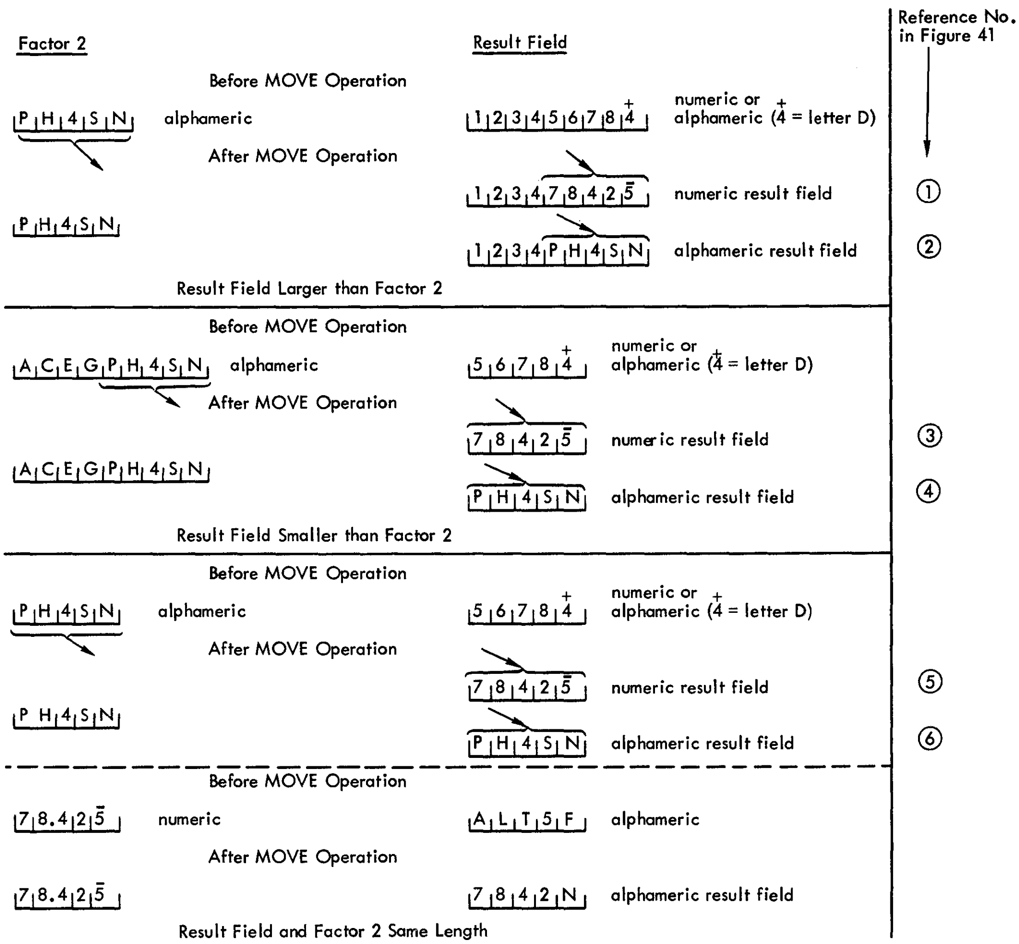


Figure 39. MOVE Operations

MOVEL (Move Left-Aligned)

The contents of the field or the literal in Factor 2 is moved into the specified result field, left-aligned.

If the result field is longer than Factor 2, the excess right-hand positions of the result field--including a sign in a numeric result field--remain unchanged. If the result field is shorter than Factor 2, the contents of only the equivalent number of left-hand positions of Factor 2 are placed in the result field (but see explanation of zone moves, below).

A source field or literal defined as alphameric may be moved to a result field defined as alphameric or numeric; also, a source field or literal defined as numeric may be moved to a result field defined as alphameric or numeric. When moving Factor-2 data to a numeric Result Field, only the low-order position of the Result Field can have a zone transferred to it. If no zone position is transferred, the numeric Result Field retains its former zone; positions other than the low-order positions can contain digits only. Blank in an alphameric field is transferred to a numeric field as zero.

Figure 40 illustrates MOVEL operations, including the behavior of zones (or signs). Figure 41 contains the specifications for the MOVE and MOVEL operations shown in Figure 39 and 40.

Rules for MOVEL zone transfers

1. Factor 2 same length as Result Field

   a. Factor 2 and Result Field numeric: Sign is moved with low-order digit

   b. Factor 2 numeric, Result Field alphameric: Sign is moved with low-order digit; other result-field positions will contain only digits.

	Factor 2	→	Result Field		Reference No. in Figure 41
num.	7, 8.4, 2, $\bar{5}$	Before MOVEL Operation	5, 6, 7.8, $\overset{+}{4}$	numeric	
	7, 8.4, 2, $\bar{5}$	After MOVEL Operation	7, 8, 4.2, $\bar{5}$		⑪
num.	7, 8.4, 2, $\bar{5}$	Before MOVEL	A, K, T, 4, D	alphameric	
	7, 8.4, 2, $\bar{5}$	After MOVEL	7, 8, 4, 2, N		⑫
alph.	P, H, 4, S, N	Before MOVEL	5, 6, 7, 8, $\overset{+}{4}$	numeric	
	P, H, 4, S, N	After MOVEL	7, 8, 4, 2, $\bar{5}$		⑬
alph.	P, H, 4, S, N	Before MOVEL	A, K, T, 4, D	alphameric	
	P, H, 4, S, N	After MOVEL	P, H, 4, S, N		⑭

Factor 2 and Result Field Same Length

	Factor 2		Result Field		Reference No.
num.	Ø, Ø, Ø, Ø, Ø, 8, 4, 2, $\bar{5}$	Before MOVEL Operation	5.6, 7, 8, $\overset{+}{4}$	numeric	
	Ø, Ø, Ø, Ø, Ø, 8, 4, 2, $\bar{5}$	After MOVEL Operation	Ø.Ø, Ø, Ø, $\bar{Ø}$		⑮
num.	9, Ø, 3, 1, 7, 8, 4, 2, $\bar{5}$	Before MOVEL	A, K, T, 4, D	alphameric	
	9, Ø, 3, 1, 7, 8, 4, 2, $\bar{5}$	After MOVEL	9, Ø, 3, 1, 7		⑯
alph.	B, R, W, C, X, H, 4, S, N	Before MOVEL	5, 6, 7, 8.$\overset{+}{4}$	numeric	
	B, R, W, C, X, H, 4, S, N	After MOVEL	2, 9, 6, 3.$\bar{7}$		⑰
alph.	B, R, W, C, X, H, 4, S, N	Before MOVEL	A, K, T, 4, D	alphameric	
	B, R, W, C, X, H, 4, S, N	After MOVEL	B, R, W, C, X		⑱

Factor 2 Longer Than Result Field

	Factor 2		Result Field		Reference No.
num.	7, 8, 4, 2, $\bar{5}$	Before MOVEL	1.3, Ø, 9, 4, 3, 2, 1, $\overset{+}{Ø}$	numeric	
	7, 8, 4, 2, $\bar{5}$	After MOVEL	7.8, 4, 2, 5, 3, 2, 1, $\overset{+}{Ø}$		⑲
num.	7, 8, 4, 2, $\bar{5}$	Before MOVEL	B, R, W, C, X, H, 4, S, A	alphameric	
	7, 8, 4, 2, $\bar{5}$	After MOVEL	7, 8, 4, 2, N, H, 4, S, A		⑳
alph.	C, P, T, 5, N	Before MOVEL	1, 3, Ø, 9, 4, 3, 2, 1, Ø	numeric	
	C, P, T, 5, N	After MOVEL	3, 7, 3, 5, 5, 3, 2, 1, Ø		㉑
alph.	C, P, T, 5, N	Before MOVEL	B, R, W, C, X, H, 4, S, A	alphameric	
	C, P, T, 5, N	After MOVEL	C, P, T, 5, N, H, 4, S, A		㉒

Factor 2 Shorter Than Result Field

Figure 40. MOVEL Operations

c. Factor 2 alphameric, Result Field numeric: Zone and digit portions of low-order character are moved; zones in other positions are not moved.

d. Factor 2 and Result Field alphameric: All characters are moved to the equivalent Result Field positions.

Note: When Factor 2 and the Result Field are the same length, the MOVEL and MOVE operations perform identical functions.

2. Factor 2 longer than Result Field

   a. Factor 2 and Result Field numeric: The sign from the low-order position of Factor 2 is moved over the low-order digit of the Result Field.

   b. Factor 2 numeric, Result Field alphameric: No sign is transferred; result field will contain only digits.

   c. Factor 2 alphameric, Result Field numeric: Zone from the low-order character of Factor 2 is moved over the low-order digit of the result field; other result-field positions will contain only digits.

   d. Factor 2 and Result Field alphameric: The appropriate number of leftmost characters in Factor 2 is moved to the equivalent positions of the result field.

3. Factor 2 shorter than Result Field

   a. Result Field numeric: The digit portion of the Factor-2 data replaces the contents of the equivalent number of leftmost positions in the result field. These result-field positions will contain only unsigned digits. The sign in the low-order position of the result field is not changed.

   b. Result Field alphameric: The entire characters in Factor 2 replace the contents of the equivalent number of leftmost positions in the result field. No change is made in the zone of the low-order position of the result field.

Note:
1. Whenever the operation does not involve transfer of a sign (or zone) portion to the low-order position of the result field, the sign (or zone) previously in the low-order position of the result field is left unchanged.

2. Whenever the operation involves transfer of a zone portion from an alphameric Factor 2 to the low-order position of a numeric result field, the particular zone placed over the low-order digit of the numeric result field follows the rules itemized in the Note under MOVE, above.

MOVE and MOVEL operations are useful in a number of situations. For example, they facilitate splitting a field so that characters (such as a hyphen) can be printed between the portions while retaining high-order zeros in the printout (use of an edit word in the output-format specifications causes suppression of at least one leading zero). This application is described in Programming Tips. Also, a literal consisting of zeros or blanks can be moved into a result field to clear all or part of it.

Item Nos. in Figures 39/40	Factor 2 Defined elsewhere as: Length	Format	Operation (28–32)	Factor 2 (33–42)	Result Field (43–48)	Field Length (49–51)	Decimal Positions (52)	Half Adjust (H) (53)
1	5	alpham.	MOVE	FIELDA	FIELDB	9	Ø	
2	5	alpham.	MOVE	FIELDA	FIELDC	9		
3	9	alpham.	MOVE	FIELDD	FIELDE	5	Ø	
4	9	alpham.	MOVE	FIELDD	FIELDF	5		
5			MOVE	'PH4SN'	FIELD6	5	Ø	
6	5	alpham.	MOVE	FIELDH	FIELDI	5		
11		-	MOVEL	-78.425	FIELDØ	5	2	
12			MOVEL	-78.425	FIELD1	5		
13			MOVEL	'PH4SN'	FIELD2	5	Ø	
14			MOVEL	'PH4SN'	FIELD3	5		
15			MOVEL	-ØØØØØ8425	FIELD4	5	4	
16	9	num.	MOVEL	FIELD5	FIELD6	5		
17	9	alpham.	MOVEL	FIELD7	FIELD8	5	1	
18	9	alpham.	MOVEL	FIELD7	FIELD9	5		
19	5	num.	MOVEL	FIELDZ	FIELDY	9	8	
20			MOVEL	-78425	FIELDX	9		
21	5	alpham.	MOVEL	FIELDW	FIELDV	9	Ø	
22			MOVEL	'CPTSN'	FIELDU	9		

NOTE: For ease of illustration the Result Field is defined in each Move-specification line.

Figure 41. MOVE and MOVEL Specifications for Moves Illustrated in Figures 39 and 40

Points to Note in Figures 40 and 41.

1. Factor 2 may be a field name or a literal (see Figure 41: items 5, 11, 12, 13, 14, 15, 20, 22).

2. Although--if a sign is specified for a numeric literal--it must be recorded

leftmost, the program treats the sign as being located in the low-order position: compare items 11, 12, 15, and 20 in Figures 40 and 41.

3. Item 15 illustrates that a minus zero result can occur after a move operation.

4. The decimal-point location in the Result Field is independent of any decimal point in the source field (Factor 2): see all numeric items--no decimal alignment takes place between the source and result fields in a move operation. Nevertheless, a Decimal-Positions entry (col. 52) is required wherever a numeric result field is defined, and there must be no Decimal-Positions entry for alphameric fields.

5. The zone (or absence of zone) in the low-order position of the result field is not changed when the move operation does not involve transfer of a zone portion to the low-order position of the result field: see items 19-22 in Figure 40.

6. Items 19 and 21 also indicate that the low-order position of a numeric result field can originally contain a sign, but can also be unsigned (hexadecimal F).

7. Factor 1 must be left blank in all Move operations.

Move Zone

This operation has four variations, to provide for moving a zone from the high-order (leftmost) or low-order (rightmost) position of one field to the high- or low-order position of another. The zone (if any) in the specified position of the field or literal in Factor 2 is moved to the specified position of the Result Field, replacing any zone previously in that position.

A zone can be moved from an alphameric field or literal to an alphameric or numeric field, or from a numeric field or literal to a numeric or alphameric field. However, since numeric fields can have a zone only in the low-order position, a zone cannot be moved from or to the high-order position of a field defined as numeric.

MLLZO (Move Low-order zone to Low-order ZOne position). The zone in the low-order position of the field or literal in Factor 2 is moved to the low-order position of the Result Field. Factor 2 and/or the Result Field may be numeric or alphameric.

MHHZO (Move High-order zone to High-order ZOne position). The zone in the high-order position of the alphameric field or literal in Factor 2 is moved to the high-order position of the alphameric Result Field.

MLHZO (Move Low-order zone to High-order ZOne position). The zone in the low-order (rightmost) position of the numeric or alphameric field or literal in Factor 2 is moved to the high-order (leftmost) position of the alphameric Result Field.

MHLZO (Move High-order zone to Low-order ZOne position). The zone in the high-order position of the alphameric field or literal in Factor 2 is moved to the low-order position of the alphameric or numeric Result Field.

A zone is moved to an alphameric Result Field exactly as it appears in the source field (Factor 2). When transferring to numeric fields, certain zones that may appear in alphameric fields are first converted--see the Note under MOVE, above.

Figure 42 illustrates Move-Zone operations. Figure 43 includes some specifications for the operations illustrated in Figure 42.

Factor 2		Result Field		Result Field after Move-Zone Operation			
Format	Contents (field or literal)	Format	Contents before Move-Zone Operation	MLLZO	MHHZO	MLHZO	MHLZO
Alph.	AH5SR	Alph.	S51T	S51L	B51T	K51T	S51C
Alph.	AH5SR	Num.	12.3	$12.\bar{3}$	not appl.	not appl.	$12.\overset{+}{3}$
Num.	$852.\overset{+}{4}$	Num.	$06.28\bar{2}$	$06.28\overset{+}{2}$	not appl.	not appl.	not appl.
Num.	$4.752\bar{4}$	Alph.	$\overset{+}{b}$KD	ƀKM	not appl.	-KD	not appl.
Alph.	6	Alph.	AFJRX	AFJR7	1FJRX	1FJRX	AFJR7
Num.	6	Num.	$580\overset{+}{2}$	5802	not appl.	not appl.	not appl.

Figure 42. Move-Zone Operations

The last line in Figure 42 shows how Move-Zone operations can conveniently be employed to remove a C-zone (+) from a positive numeric field to prevent punching of a 12-overpunch, or printing of a letter or symbol, when the field is used in output--without the necessity of editing and, thus, offering the ability to retain leading zeros. The literal or position in the field from which the zone is to be moved, to eliminate a C-zone (+) or D-zone (-) in the numeric result field, must contain an F-zone (see EBCDIC, Figure D1); i.e., any of the digits 0-9 or any of the remaining six characters in EBCDIC-table column labelled F.

Compare and Zone-Testing Operations

These operations test data conditions without effecting any changes in data fields. Results of the tests are signified by the status of Resulting Indicators assigned in cols. 54-59. Execution of the specifications for these operations may be conditioned by the status of indicators designated in Control Level (cols. 7-8) and Indicators (cols. 9-17).

Effects of Each Operation Code

COMP (Compare)

The contents of the field or the literal in Factor 1 is compared against the contents of the field or the literal in Factor 2. Any Resulting Indicators specified in cols. 54-59 reflect the result of the comparison. (Factor 1 and Factor 2 normally contain different field names or literals; but it is permitted to compare data to itself (Factor 1 and Factor 2 identical), which always yields a comparison result of "Equal.") The Result Field (and related fields--i.e., cols. 43-53) must be left blank.

A Resulting Indicator must be assigned to at least one of the three possible conditions:

High (cols. 54-55): Factor 1 > Factor 2
Low (cols. 56-57): Factor 1 < Factor 2
Equal (cols. 58-59): Factor 1 = Factor 2

After the Compare operation, the Resulting Indicator (if any) assigned to the condition found to exist, turns on; any indicators assigned to the other two possible conditions turn off. However, the same indicator may also be assigned to more than one of the three possible conditions (e.g., High and Low, or High and Equal, or Low and Equal); it then turns on if one of the conditions to which it was assigned applies. Different indicators may be assigned to two, or all three, of the possible conditions. The status of the Resulting Indicators can be used to condition the execution of calculation specifications (by entries in cols. 7-17) and/or output-format specifications (by entries in cols. 23-31).

If execution of the calculation-specification line that contains the Compare operation is itself conditioned by indicators (in cols. 7-17), the user must remember that the comparison will not be executed during each program cycle unless the status of the conditioning indicators is always appropriate. Therefore, Resulting Indicators could reflect an earlier, and possibly inapplicable, comparison.

Factor 1 and Factor 2 must both be alphameric or both numeric. Certain aspects of the Compare operation differ for alphameric and numeric fields or literals, as follows.

Alphameric Fields or Literals:

1. Fields (or literals) of unequal length are left-aligned. The shorter field is assumed to contain an equivalent number of blank right-hand positions to equate the length of both fields.

2. Blanks within a field (or literal) are treated as blanks, not as zeros.

3. Maximum length of the (longer) Factor is 40 positions.

4. The comparison is based upon the internal Model 20 collating sequence, which corresponds to the EBCDIC-table sequence (see Appendix D, Figure D1). Note that, in EBCDIC, the most-commonly used characters follow this sequence: ƀ, &, -, /, A-Z, 0-9. Thus, a digit signed positive (if the field is defined as alphameric) is lower in sequence than one signed negative, or than an unsigned digit.

   The user has the option of substituting any sequence of his choice for the standard EBCDIC sequence, by means of a translation table (see Altered Collating Sequence, Appendix D). The altered collating sequence will then apply both to alphameric Compare operations and to Matching-Fields operations.

Numeric Fields or Literals:

Numeric Compare is tantamount to an arithmetic operation. Therefore:

1. Fields or literals of unequal length are aligned at the (defined or implied) decimal point. Where one field or literal is then shorter than the other (at the high-order or low-order end), it is assumed to contain an equivalent number of zeros.

2. The maximum length of a Factor is 15 positions. (This refers to the literal or field specified in the Factor; any left or right zeros assumed by the program for decimal alignment--see 1, above--are not counted when considering the limitation of 15 positions.)

3. The comparison is algebraic; i.e., a positive value is treated as higher than the same value signed negative. The sequence, from low to high, is: $\bar{9}$ to $\bar{1}$, $\bar{0}$, 1 (or $\overset{+}{1}$) to 9 (or $\overset{+}{9}$). $\bar{0}$ = 0 = $\overset{+}{0}$. A value with an unsigned (hexadecimal F) low-order digit is treated as positive (unless all digits are zeros). A technique for performing absolute numeric Compare (i.e., ignoring signs) is given in <u>Programming Tips</u>, Appendix E.

> If the low-order position of a field does not contain a zone acceptable as a sign (hexadecimal zone C, D or F; or hexadecimal byte 05-06), an error stop occurs.

4. All positions of both Factors must contain valid digits:

   Considering an input field before it is packed by the program, the character in each column must be represented in rows 0-9 of the EBCDIC table (Appendix D, Figure D1). For packed data, the equivalent valid EBCDIC characters were described under <u>Packed</u> (col. 43), in <u>Input Specifications</u>.

   - Any characters that do not represent digits (or, digit plus sign in the low-order position) cause an abortive program stop.

Compare is frequently a more efficient method of sequence-checking than the use of Matching Fields for that purpose alone. It also allows checking for duplicates, which the Matching-Fields specifications cannot accomplish. Figure 68--Part I includes an illustration utilizing Compare for this purpose.

Figure 43 includes some specifications for Compare operations:

<u>Specifications line 07</u>. The contents of the field SLS66 (say, 1966 sales) are compared with the contents of SLS65. If 1966 sales exceeded 1965 sales, Resulting Indicator 21 is turned on; if they were less, Resulting Indicator 26 turns on; if the two years had equal sales, 30 turns on. The two inapplicable indicators will be off after the compare operation.

<u>Line 08</u>. The alphameric literal OCTOBER is compared against the contents of the field named MONTH (which must also be defined as alphameric). If the MONTH field does not consist exactly of the word OCTOBER, indicator 15 is turned on; if it does consist exactly of OCTOBER, indicator 15 will be off after the compare operation.

<u>Line 09</u>. The contents of the field named GRSPAY (which must be defined as numeric) are decimal-aligned with the numeric literal 1250.00, and then compared algebraically against it. If GRSPAY contains a value algebraically equal to or larger than 1250.00, indicator 04 turns on; if its value is algebraically less than 1250.00, indicator 05 turns on. The inapplicable indicator of the two (04 or

05) will be off after the compare operation.

Line 10. The contents of the field named NETPAY (which must be defined as numeric) are decimal-aligned with the numeric literal 0 (for which the decimal point is assumed after the 0; thus: 0.), and then compared algebraically against it. If NETPAY is greater than zero, indicator H1 will be off after the compare operation. If NETPAY is zero or negative, indicator H1 turns on; if it is not turned off by a subsequent calculation operation, the system will halt after detail-time output.

TESTZ (Test Zone)

The zone portion of the high-order (leftmost) position of the alphameric field named in Result Field (cols. 43-48) is tested. Any Resulting Indicators specified in cols. 54-59 reflect the result of the test; i.e., the type of zone present in the high-order position (refer also to EBCDIC table, Appendix D, Figure D1). The Result Field must be defined (in this line or elsewhere) as alphameric (col. 52 blank). Factor 1, Factor 2, Decimal Positions (col. 52), and Half-Adjust (col. 53) must be left blank.

A Resulting Indicator must be assigned to at least one of the three possible conditions--

Plus (cols. 54-55) --Equivalent of 12-punch: &, or any of the characters that have the same zone as the letter A.

In terms of the EBCDIC table: code 50, or any of the 16 characters in the column labelled C.

Minus (cols. 56-57)--Equivalent of 11-punch: -, or any of the characters that have the same zone as the letter J.

In terms of the EBCDIC table: code 60, or any of the 16 characters in the column labelled D.

Blank (cols. 58-59)--Other zones, or equivalent of no zone: Any of the 222 EBCDIC characters not considered Plus or Minus (see immediately above).

After the TESTZ operation, the Resulting Indicator (if any) assigned to the condition found to exist, turns on; any indicators assigned to the other two possible conditions turn off. However, the same indicator may also be assigned to more than one of the three possible conditions (e.g., Plus and Blank, or Minus and Blank, or Plus and Minus); it then turns on if one of the conditions to which it was assigned applies.

Different indicators may be assigned to two, or all three, of the possible conditions. The status of the Resulting Indicators can be used to condition the execution of calculation specifications (by entries in cols. 7-17) and/or output-format specifications (by entries in cols. 23-31).

Note: An indicator 01-99 assigned to Blank (cols. 58-59) is on at the beginning of program execution. Any indicator in Blank of a TESTZ specification is also turned on when the field is transferred to the output storage area by an output specification, if Blank-After (B in col. 39 in the output-format specifications) is specified. (If several different indicators are assigned to Zero-or-Blank of the same field in different specifications lines, the earliest-assigned is turned on by Blank-After.) Where Blank-After turns on the indicator assigned to Zero-or-Blank, this does not turn off an indicator assigned to Plus or Minus, if it was on.

Figure 43 includes some examples of specifications for the TESTZ operation.

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-3351-1
Printed in U. S. A.

Date ____________

Program ____________

Programmer ____________

Punching Instruction	Graphic	+
	Punch	

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

Line (3-5)	Form Type (6)	** Control Level (L0-L9, LR) (7-8)	** Indicators (9-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field (43-48)	* Field Length (49-51)	* Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Plus / Compare High 1>2 (54-55)	Minus / Low 1<2 (56-57)	Zero or Blank / Equal 1=2 (58-59)	Comments (60-74)
01	C				MLLZO	FCTLN1	RESLN1	4						Specifications for some of the examples in Figure 42. Factor 2 may contain field name or literal.
02	C				MHHZO	FCTLN1	RESLN1							LINE 1
03	C				MLHZO	'AH5SR'	RESLN1							
04	C				MHLZO	'AH5SR'	RESLN1							
05	C				MLLZO	+852.4	RESLN3	5	3					LINE 3
06	C				MLLZO	6	RESLN6	4	Ø					LINE 6
07	C			SLS66	COMP	SLS65					21	26	3Ø	
08	C			'OCTOBER'	COMP	MONTH					15	15		
09	C			GRSPAY	COMP	125Ø.ØØ					Ø4	Ø5	Ø4	
10	C			NETPAY	COMP	Ø						H1	H1	
11	C				TESTZ		OMEGA	1			12	13	14	
12	C				TESTZ		PSI	25			16	H2	16	
13	C				TESTZ		CHI	8				99		
14	C				SETON						84	MR		
15	C	L1			SETOF						H1	2Ø	LØ	

*NOTE 1: Field Length and Decimal Positions, if applicable, need be defined here only if not defined elsewhere. Factors are assumed to be defined elsewhere.
**NOTE 2: Conditioning Indicators (cols. 7-17) may be used with all of these operations.

Figure 43. Specifications for Move-Zone, Compare, Test-Zone, and SETON/SETOF Operations

Setting Indicators

One, two, or three indicators, specified in cols. 54-59, can be either set on (SETON) or set off (SETOF) by a single calculation specification, without concurrent performance of any other calculation operation. An indicator must be specified in at least one of the Resulting Indicators fields (cols. 54-55, 56-57, 58-59). The column headings over cols. 54-59 (Plus/High, Minus/Low, Zero-or-Blank/Equal) are irrelevant to these operations.

Factor 1, Factor 2, Result Field, Decimal Positions, and Half-Adjust must be left blank.

Execution of these operations may be conditioned by the status of indicators designated in Control Level (cols. 7-8) and in Indicators (cols. 9-17).

Effects of Each Operation Code

SETON (Set On)

The indicators assigned by entry in cols. 54-55, 56-57, and 58-59 are turned on.

SETOF (Set Off)

The indicators assigned by entry in cols. 54-55, 56-57, and 58-59 are turned off.

Points to Note:

1. Any indicator may be SETON or SETOF.

2. If the LR indicator is SETON during total-time calculations, processing is terminated after total-time output. If it is SETON during detail-time calculations, it is turned off again by the program after detail-time output for that card.

3. If the MR indicator is SETON or SETOF, it assumes--before the next detail-calculation time--the status it would have without the prior SETON or SETOF instruction.

4. If the CF or OV indicator is SETON or SETOF, it assumes--after completion of the nearest following detail-time or total-time output--the status it would have without the prior SETON or SETOF instruction.

5. If indicator L0 is SETOF, it is turned on again by the program after the next detail-time output.

6. If indicator H1 or H2 is SETON, and has not been turned off by a specification before the next detail-time output, the system halts after detail-time output. If the system is restarted (by pressing the CPU START key twice), the program sets H1 and H2 off at that point.

7. SETON or SETOF of any L-indicator (LR, L9-L1, L0) does not automatically set any other L-indicator on or off.

8. Indicators 1P and L1-L9 are always set off by the program after detail-time output.

9. The status of any indicator assigned as card-type Resulting Indicator (cols. 19-20 in the input specifications) is revised--based on card type read--after detail-time output, regardless of any prior SETON or SETOF instruction.

10. The status of any indicator assigned as Field Indicator (cols. 65-70 in the input specifications) is revised--based on the contents of the input field--before detail-time calculations, regardless of any prior SETON or SETOF instruction.

11. An indicator assigned to Zero-or-Blank in Field Indicators (input specifications), or in Resulting Indicators (calculation specifications) of an arithmetic or TESTZ operation, turns on upon execution of a Blank-After instruction (output-format specifications), regardless of any prior SETOF operation.

All of these points were fully discussed earlier under Program Logic Flow, Indicators, and Indicator Hierarchy.

Figure 43 includes SETON and SETOF specifications. SETON is also illustrated in Figures 5D and 33.

Branching

Within the grouping of detail time and total time, the RPG program normally executes specifications in the order in which they appear. Branching implies deviation from this natural sequence.

Two types of branching are possible with this RPG:

1. Branching within RPG: Skipping to an RPG calculation specification other than the next one in the normal sequence.

   This involves the RPG operation code GOTO for the point of origin of a branching operation, and the pseudo-operation code TAG for the point of destination of a branching operation.

   Branching within RPG, by a GOTO instruction, can be useful in several situations. For instance:

   To bypass an entire calculation section that is inapplicable when certain conditions apply (see Figure 44).

   To call in a complete RPG routine that applies only under certain circumstances (e.g., square root).

   To call in an RPG routine that applies to several, but not all, card types. This method may consume less core storage space than repeating the specifications in several places.

   To bypass detail or total output under particular conditions (see Figure 44).

   To iterate a sequence of specifications; i.e., to create a program loop (see Figure 44A, lines 05-13).

   To repeat the same output several times, based on a control number which may vary for different input cards (see Programming Tips, Appendix E).

2. Branching to an external subroutine: Transferring control of the program from RPG to an external routine in machine language, provided by the user or supplied by IBM.

   The routine must be in relocatable form, and must include the various control cards (such as ESD and RLD) normally created when a program is written in Basic Assembler Language and then converted to machine language with the Basic Assembler program. The end of the routine must contain instructions to return control to the RPG program. A thorough understanding of the SRL publications IBM System/360 Model 20 Basic Assembler Language (Card and Tape), Form C26-3602 and IBM System/360 Model 20 Functional Characteristics, Form A26-5847 is a prerequisite.

   This kind of branching involves the operation code EXIT for the point of origin of a branch, and the pseudo-operation code RLABL. The latter identifies specification lines which define fields that are used both in an external routine and in the RPG program, and indicators that are referenced in an external subroutine.

Branching to an external routine may enable the user to incorporate, in a program principally written for RPG, operations not easily accomplished with RPG itself (e.g., selection of the last card of each control group--see Programming Tips, Appendix E).

Any number of external subroutines may be used with an RPG program, within the limits of core storage positions available.

Both the GOTO and the EXIT operations may be conditioned by the status of indicators designated in Control Level (cols. 7-8) and Indicators (cols. 9-17).

Branching Within RPG

GOTO (Branch To)

This operation code defines the point of origin for a skip (branch) to an RPG calculation specification line other than the next in sequence. No other operation--besides initiating a branch--is performed by the specifications in this line. Branching can be to an earlier or subsequent calculation specification line. For branching from detail-time calculations to total-time calculations, or vice versa, see subsection below.

Factor 2 must contain the name (the address) of the point of destination of this branch instruction. The rules for forming this name are identical with those for field names (i.e., one to six characters in consecutive columns, the first of which must be in col. 33 and must be alphabetic, the remainder being alphabetic or numeric). The same point-of-destination name may be used with any number of GOTO points of origin; i.e., several points of origin may all branch to the same RPG routine. But, of course, any one point-of-destination name can only be associated with a single destination point (see TAG, below). The point-of-destination name must not be used for any other purpose; i.e., it must not also be a field name in input specifications or in other calculation operations.

Factor 1, Result Field (and the associated fields for Field Length, Decimal Positions and Half-Adjust), and Resulting Indicators--i.e., cols. 18-27 and 43-59--must all be left blank.

The GOTO operation can be an unconditional branch--cols. 7-17 are then blank--or it can be a conditional branch--i.e., there are entries in Control Level and/or Indicators, cols. 7-17.

If Control Level (cols. 7-8) contains an entry (L0-L9, or LR), the branch is executed at total time, provided the particular L-indicator is on--and subject to the status of indicators that may be designated in Indicators (cols. 9-17). If Control Level (cols. 7-8) is blank, the branch is executed at detail time--subject to the status of indicators that may be designated in Indicators (cols. 9-17).

TAG (Destination of a Branch Operation)

This pseudo-operation code merely designates the specification line as a destination point to which a GOTO operation may branch. Factor 1 contains the point-of-destination name (left-aligned, starting in col. 18) that was assigned in Factor 2 of the related GOTO specification line(s).

A point-of-destination name cannot be associated with more than one TAG specification line--otherwise the program could not know to which of several destination points it should branch--nor can it serve as a field name anywhere else except as a destination name in GOTO specification lines.

Factor 2, Result Field (and its associated fields), and Resulting Indicators--i.e., cols. 33-59--must be left blank. The Indicators fields (cols. 9-17) must also be left blank.

If the TAG line is preceded by total-time specification lines (i.e., lines with L-indicator entries in Control Level, cols. 7-8), the TAG specification line must also have an L-indicator (L0, L1-L9, or LR) in Control Level (cols. 7-8). If the TAG line is followed by detail-time specification lines (i.e., lines without entries in Control Level, cols. 7-8), the TAG line must also be blank in Control Level (cols. 7-8).

The presence or absence of an L-indicator in Control Level (cols. 7-8)--as stipulated in the preceding paragraph--serves two purposes at object-program generation time:

1. The program checks that all specifications for detail time precede all specifications for total time. The TAG line must satisfy this check.

2. Absence of an L-indicator in Control Level (cols. 7-8) of the TAG line signifies that the destination of the branch operation lies within detail-time calculations; presence of an L-indicator in Control Level of the TAG line signifies that the destination of

the branch operation lies within total-time calculations. The significance of this distinction becomes apparent when branching between detail-time and total-time calculations is considered (below).

Note: It is the presence or absence of an (any) L-indicator (in Control Level of the TAG line) at generation time that determines whether the branch destination lies within detail or total time. At object-program execution time, it is immaterial whether that L-indicator is on or off: the TAG line will still serve to identify the branch destination.

When performing a GOTO operation, the program skips to the operation that follows the TAG line which contains (in Factor 1) the same name that the particular GOTO line contains (in Factor 2). That next operation may be another specification line in the same cycle time-segment (detail or total time, respectively); i.e., either the GOTO line and the line following TAG are both blank in Control Level (cols. 7-8), or both contain L-indicators in Control Level. The operation in that line is then executed--subject to conditioning indicators in cols. 7-17. Thence, operations proceed sequentially, line by line, as is normal. If the TAG line is the last calculation specification line in a cycle time-segment (detail time or total time), the pertinent output operations follow (detail-time or total-time output), and the program continues in its standard sequence as shown in Figure 6, RPG Program Logic. (If the GOTO and TAG specifications are not in the same cycle time-segment, see immediately below.)

Branching Between Detail-Time and Total-Time Calculations within RPG

a. Branching from total-time calculations to detail-time calculations.

   If the GOTO line contains an L-Indicator in Control Level (cols. 7-8) but the associated TAG line does not, the program skips from the point of the GOTO line in total-time calculations to the point following the TAG line in the detail-time calculations.

   Total-time and overflow-time outputs are bypassed. The status of all indicators remains unchanged--including L-indicators (L0-L9, LR), overflow indicators (OF, OV), MR indicator, and Field Indicators. Figure 6, RPG Program Logic, shows what normal program actions are bypassed on a skip from total-time to detail-time calculations.

   The data from the previous card remains available. If the program then continues in the normal sequence, the data from the next card to be processed never becomes available.

   If the LR (Last Record) indicator is on before the branch operation, it remains on until completion of detail-time output, at which time it is turned off by the program. Normal end-of-job routines are bypassed; the exact consequences, if an end-of-job situation exists, are difficult to predict in generalized terms, because they depend on a combination of conditions.

b. Branching from detail-time calculations to total-time calculations.

   If the GOTO line is blank in Control Level (cols. 7-8) but the associated TAG line contains an L-indicator in Control Level, the program skips from the point of the GOTO line in detail-time calculations to the point following the TAG line in the total-time calculations. No detail-time output occurs. Data from the next card is not transferred to the input work area; the data from the card being processed remains available. (The same input data is repeatedly transferred to the process area each time the program advances to detail-calculation time.)

   Total-time calculations from the point of the TAG line are executed (again) sequentially--subject to the status of conditioning indicators in cols. 7-17, as is normal. This is followed (unless bypassed by another GOTO instruction) again by total-time output and, if OF or OV is on, by overflow-time output--after which detail-time calculations recur. Figure 6, RPG Program Logic, should be consulted for the implications.

   Note: When branching from detail time to total time, the pertinent total-time calculations and output are performed--even if total time would otherwise be bypassed (see Total-Time Processing on "Run In", under Program Logic Flow). However, total time is still bypassed on subsequent cards--if it would normally be bypassed--unless total time is reached by a GOTO instruction.

   If L-indicators or card-type Resulting Indicators were turned on or off (by SETON or SETOF, or as Resulting Indicators) during detail-time calculations preceding the GOTO operation, they retain that status. If indicators assigned as Field Indicators, or the MR indicator, were turned on or off during detail-time calculations preceding the GOTO operation, they revert--after each repeated total-time output--to the setting they had immediately preceding the original detail-time calculations for the card being processed.

If an overflow-indicator (OF and/or OV) was turned on or off during detail-time calculations preceding the GOTO operation, it reverts--before overflow-time output--to its status at the end of the preceding total-time output. However, if OF (or OV) was off at the conclusion of the preceding total-time output, and a carriage-tape channel-12 punch is encountered during one of the repeated total-time outputs, OF (or OV) turns on. Once turned on by channel 12, it will always be on at overflow-output time, until the next detail-time output has been completed.

Normally, branching from detail time to total time is employed to repeat printed output on paper forms. However, if punching or printing into combined-file cards is performed during repeated total-time output, it takes place in successive cards of the file and these additional cards are not read. (See Multiple-Time Output to Cards during One Program Cycle, under Program Logic Flow.)

Note: When the calculation specifications call for branching (GOTO and TAG) between detail time and total time, a warning message is printed during generation of the object program: "GOTO AND TAG ARE NOT IN THE SAME CALCULATION TIME."

Figures 44A and B illustrate GOTO and TAG specifications. (The particular specifications used in Figure 44--other than GOTO and TAG--were random choices.)

Explanation of Entries in Figure 44A

If the result of the subtraction in line 01 was negative, control is transferred--by the specifications in line 02--to RTN1 (say, Routine 1) in line 09--both points wholly within detail time. Otherwise, the multiplication in line 03 is first executed, and then control is transferred

IBM — INTERNATIONAL BUSINESS MACHINES CORPORATION — REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS — IBM System/360 — Form X24-3351-1 Printed in U.S.A.

Date ______ Program ______ Programmer ______ Punching Instruction: Graphic / Punch — Page — Program Identification

Line	Form Type	Control Level (L0-L9, LR)	Indicators Not (9-11)	And Not (12-14)	And Not (15-17)	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Plus / High 1>2	Minus / Low 1<2	Zero or Blank / Equal 1=2	Comments
01	C					ALPHA	SUB	BETA	GAMMA	5	2			1Ø		
02	C		1Ø				GOTO	RTN1								
03	C					DELTA	MULT	GAMMA	EPSILN	8	2					
04	C						GOTO	RTN1								
05	C					RTN2	TAG									
06	C							Some detail-time								
07	C							calculation specifications								
08	C															
09	C					RTN1	TAG									
10	C							Some detail-time								
11	C							calculation								
12	C							specifications							15	
13	C		N15				GOTO	RTN2								
14	C						TESTZ		ZETA				2Ø		2Ø	
15	C		2Ø				GOTO	ENDDTL								
16	C						MHLZO	ZETA	GAMMA							
17	C						SETOF						1Ø			
18	C					ENDDTL	TAG									
19	C	L1	N99			GAMMA	ADD	TOTGAM	TOTGAM							
2Ø	C															

Card Electro Number ______

Figure 44A. Examples of Branching within RPG - I to line C9. This illustrates a conditional branch (Indicator 10 in line 02) and an unconditional branch (no Indicator in line 04); it also exemplifies forward branching from several points of origin (lines 02 and 04) to the same point of destination (line C9).

If the last operation in Routine 1 does not turn on indicator 15, the program branches at that point to RTN2 (say, Routine 2)--both the point of origin and the point of destination lying within detail time. Routine 2 followed by Routine 1 are then executed repetitively until indicator 15 turns on, whereafter the next sequential specification after Routine 1 is executed. This group of entries illustrates use of a program loop (line 13 to lines 05-12), and branching back to an earlier specification line within the same cycle time-segment.

After indicator 15 has turned on, the next sequential operation (line 14) is carried out. If the zone in the high-order position of ZETA is the equivalent of minus (i.e., indicator 20 is not on), the program continues sequentially; but, if indicator 20 is on, detail-time calculations are terminated. Detail-time output is then the next operation. These entries show how to bypass remaining detail-time calculations: the GOTO branch is to a TAG line that is the last detail-time specification--the ENDDTL TAG-line is blank in Control Level (cols. 7-8), which defines it as a detail-time specification; the next specification line has an L-indicator in Control Level (cols. 7-8), which defines it as a total-time specification. Therefore, the ENDDTL TAG-line is the last detail-time calculation specification.

If, instead of blanks, an L-indicator were entered in Control Level (cols. 7-8) of the ENDDTL TAG-line, the effect would be--

> If indicator 20 is not on: no difference.
>
> If indicator 20 is on: the specifications in line 14 would be the last detail-time operations performed. The next operation would be at total time--line 19, or a subsequent line, depending on the status of conditioning indicators designated in cols. 7-17. No detail-time output would occur until, possibly (depending on execution of GOTO instructions next detail time), following detail-time calculations after the next total-time output.

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-3351-1 Printed in U. S. A.

Date ______

Program ______

Programmer ______

Punching Instruction: Graphic / Punch

Page

Program Identification

Line	Form Type	Control Level (L0-L9, LR)	Indicators Not (9-11)	And Not (12-14)	And Not (15-17)	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators Plus / Compare High 1>2	Minus / Low 1<2	Zero or Blank / Equal 1=2	Comments
01	C					BEGDET	TAG									
02	C							Some								
03	C							detail-time calculation								
04	C							specifications								
05	C		NMR				GOTO	MIDTOT								
06	C	L1				FIELDA	ADD	FIELDB	FIELDB	6	1					
07	C	L2				5Ø.25	MULT	FIELDA	RATE	9	2	H				
08	C	L1				TYPE	COMP	'EXEMPT'					Ø2	Ø2		
09	C	LØ				MIDTOT	TAG									
10	C	L2						Some								
11	C	L1						total-time calculation								
12	C	L2						specifications							12	
13	C	L2	12				GOTO	BEGDET								
14	C	L1				SALES	SUB	RETRNS	NETSLS	8	2			99		
15	C	L1	99				GOTO	BEGDET								

Figure 44B. Examples of Branching within RPG - II

Explanation of Entries in Figure 44B

The specifications in lines 02 to 04 are always executed at detail time. If the MR (Matching Record) indicator is on, detail-time output follows, subsequently followed by total-time calculations in the normal manner. If the MR indicator is off, detail-time output is bypassed, and the program executes total-time specifications, beginning with line 10--or a subsequent line, depending on the status of the L-indicators in Control Level (cols. 7-8); if L1 and L2 are both off, total-time output is the next operation after detail-time calculations. This illustrates not only bypassing of detail-time output, but also the facility of entering into any point of the total-time calculations. It also points out that, if the status of all conditioning indicators happens to prevent execution of any calculation specification in a cycle time-segment, total output for a time segment can occur without any preceding calculation operations.

If indicators L2 and 12 (line 13) or L1 and 99 (line 15) are on, total-time and overflow-time outputs are bypassed, and specification line 02 of detail-time calculations is executed next. In the former case (Indicators L2 and 12 on), the last total-time calculation specification is also bypassed. If neither the indicator pairs L2 and 12 , nor L1 and 99 are on, total-time calculations are completed in the normal manner, followed by total-time output. These specifications illustrate the following: branching from one of two points of origin to one point of destination; bypassing of total-time output, and optional concurrent bypassing of some total-time calculation specifications (line 14).

Note: The TAG lines defined as total-time lines (see, for example, line 09 in Figure 44B) by entry of an L-indicator in Control Level (cols. 7-8) will perform their function regardless of whether the particular indicator is on.

Branching to an External Routine

EXIT (Branch To)

This operation code defines a point in the RPG calculation specifications at which control of the program is transferred to a designated external subroutine previously prepared in System/360 Model 20 machine language. The address for return to the RPG program is stored in register 14.

Factor 2 must contain the name (the address) of the subroutine to which control is to be transferred. (This name is entered at START in the subroutine.) The name may be one to four positions long. The first character must be alphabetic and must be in col. 33; the optional (one, two, or three) additional characters may be alphabetic or numeric (special characters and embedded blanks are not permitted). The name must be unique: it must not also be a field or TAG name within RPG.

Factor 1, Result Field (and the associated fields for Field Length, Decimal Positions, and Half-Adjust), and Resulting Indicators--i.e., cols. 18-27 and 43-59--must all be left blank.

The EXIT operation can be an unconditional branch: --cols. 7-17 are then blank; or it can be a conditional branch--i.e., there are entries in Control Level and/or in Indicators, cols. 7-17.

If Control Level (cols. 7-8) contains an entry (L0-L9, or LR), the branch is executed at total time, provided the particular L-indicator is on--and subject to the status of indicators that may be designated in Indicators (cols. 9-17). If Control Level (cols. 7-8) is blank, the branch is executed at detail time--subject to the status of indicators that may be designated in Indicators (cols. 9-17).

Position of EXIT Specifications. The EXIT operation may be used anywhere in the RPG program. However, the user should be aware of the following considerations regarding four specific positions of the EXIT operation code in the program:

1. If the EXIT operation is the first detail-time calculation specification of the program, control will be transferred to the subroutine upon completion of the input routine; i.e., as soon as the pertinent input data has been placed in core storage, ready for processing by detail-time calculation specifications.

2. If the EXIT operation is the last detail-time calculation specification of the program, control will be transferred to the designated subroutine immediately before detail-time output. Upon return from the subroutine, the RPG output routine is entered.

3. If the EXIT operation is the first total-time calculation specification of the program, the exit to the subroutine takes place just after the record type has been determined and any control fields have been tested.

4. If the EXIT operation is the last total-time calculation specification, control will be transferred immediately before total-time output. Upon return from the subroutine, the RPG output routine is entered.

Requirements and Restrictions related to use of external subroutines with Model 20 card RPG:

1. All subroutines to be incorporated in this RPG program must have been coded in Model 20 Basic Assembler Language (using the IBM System/360 Basic Assembler Short Coding Form, X28-6506), and converted separately (from the RPG program) to machine language with the Basic Assembler program. The resulting object-program deck is loaded at RPG program-generation time. If the object program is punched out, the subroutine becomes an integral part of the punched object deck in TXT-card format, so that it is reloaded with the object deck each time it is loaded to perform the particular job.

   Since the subroutines and the RPG program are to be linked, the subroutines must be relocatable and all Basic Assembler linking conventions must be observed.

2. a. Fields used in both the RPG program and subroutines must be defined and identified in the RPG program--see RLABL, below.

      Note: A field cannot be defined in the subroutine for use in RPG; i.e., the ULABL statement is not available.

   b. Indicators used in a subroutine must be identified in the RPG program--see RLABL, below.

3. A subroutine can have only one entry point, and this must be its first instruction.

4. The subroutines must not consist of more than a single segment each, nor can control be transferred from one subroutine to another.

5. Fields defined in one subroutine cannot be used in another subroutine.

6. Data in fields defined as numeric in the RPG program is transferred to a subroutine in packed format. Data in a field defined as numeric that is transferred from a subroutine to the RPG object program must be in packed format.

7. The facility for branching to external subroutines (i.e., operation code EXIT) is not intended for the performance of input or output operations.

   Input or output operations via instructions in subroutines should not be attempted without a comprehensive grasp of device instructions, device and program time relationships, and the internal logic of the RPG object program. Almost invariably, difficulty will be experienced by the user whose subroutine addresses any of the same I/O devices employed in the associated RPG program.

Use of Registers in External Subroutines calls for observance of the following rules

1. Register 15 must be the base register for all subroutines. (The first instruction must be BASR 15,0 if instructions in the subroutine reference other points within the subroutine.)

2. RPG automatically stores the return address in register 14. The return address is the address of the RPG calculation-specification statement or other operation to which control is to be returned upon completion of the subroutine; i.e., the address of the statement or operation following the EXIT operation.

3. The contents of any other registers used within the subroutine must be preserved before the subroutine is executed.

   Such registers must be restored to their original contents before control is returned to the main (RPG) program.

4. Registers 12 and 13 should not be used if the program is ever to be run on a System/360 model higher than Model 20.

Use of Indicators in Subroutines requires RLABL statements in RPG (see below), and the following information:

1. a. The hexadecimal representation for the indicator-ON condition is F0.

   b. The hexadecimal representation for the indicator-OFF condition is 00.

2. To turn an indicator ON or OFF, set the data located at INxx (see RLABL, below) to F0 or 00, respectively.

3. To test the status of an indicator, examine the data located at INxx (see RLABL, below) for hexadecimal F0 (=ON) or hexadecimal 00 (=OFF).

RLABL (Reference Label)

All fields that are used both in the RPG program and in an external subroutine must be defined in the RPG program. In addition, each field used in both the RPG program and an external subroutine, and each indicator referenced in a subroutine, must be especially identified in the RPG program.

The pseudo-operation code RLABL designates a specification line that identifies either a field used both in the RPG program and an external subroutine, or the code of an indicator utilized in a subroutine. For each such field or indicator, RLABL is recorded in cols. 28-32 (Operation) of a calculation specification line, and the name of the field or indicator in the first four columns (cols. 43-46) of Result Field; Field Length and Decimal Positions are designated if the field is not defined elsewhere. All other fields in the specification line must be left blank.

RLABL lines may appear in any position among the calculation specifications; but core space during object-program generation is conserved by grouping all of them as the last calculation specifications.

A maximum total of 14 indicators and/or RPG field names--i.e., up to 14 field names and indicators identified in RLABL lines--can be used in one external subroutine; but any field name or indicator identified in an RLABL statement may be used in any number of subroutines. (By means of a special subroutine that simulates indexing, it would be possible to exceed this limit of 14.)

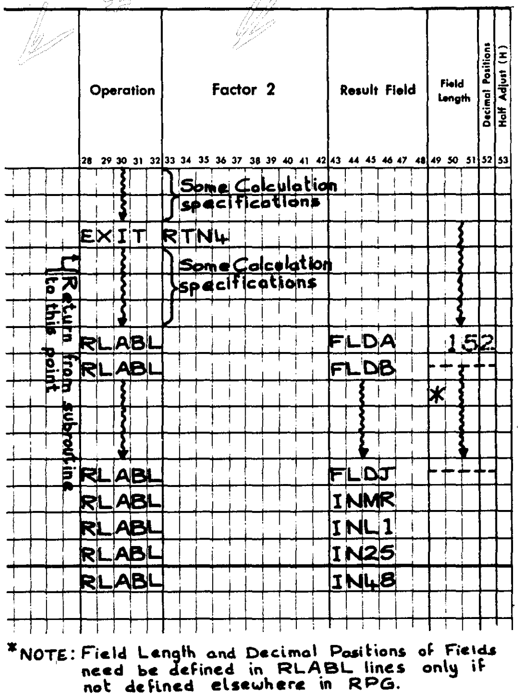
The Field Name in an RLABL line may be from one to four characters long, beginning in col. 43. The first character must be alphabetic; the optional (one, two, or three) additional characters may be alphabetic or numeric (special characters and embedded blanks are not permitted). If the field name is not defined in the input specifications, or elsewhere in the calculation specifications, it must be defined here: field length must then be entered in cols. 49-51 and, if the field is to be defined as numeric, an entry (0-9) is made in Decimal Positions (col. 52)--see sections on Result Field, Field Length, and Decimal Positions, above.

Field names must not be identical with a GOTO destination address (i.e., must not duplicate the full field name identified in a TAG line); nor may they begin with IN in an RLABL line, because INxx is reserved for indicators. A table name may be entered as a field name in the Result Field of an RLABL line; but such a table name must not consist of more than four characters (including TAB)--the data selected from that table in the last LOKUP operation is thus available to external subroutines.

An Indicator that is used in a subroutine is identified in the Result Field of an RLABL line by the letters IN in cols. 43-44, followed by the letters or numbers of the relevant indicator. In the subroutine, that indicator can then be referred to as the data located at INxx. For example, if the MR (Matching Records) indicator is tested or set in an external subroutine, an RLABL line with the characters INMR in cols. 43-46 is required; in the subroutine, that indicator can then be referred to as the data located at INMR.

Figure 45 illustrates both RPG and Basic Assembler Language coding skeletons for an external subroutine operation.

SEARCH ARGUMENT—
TABLE ARGUMENT



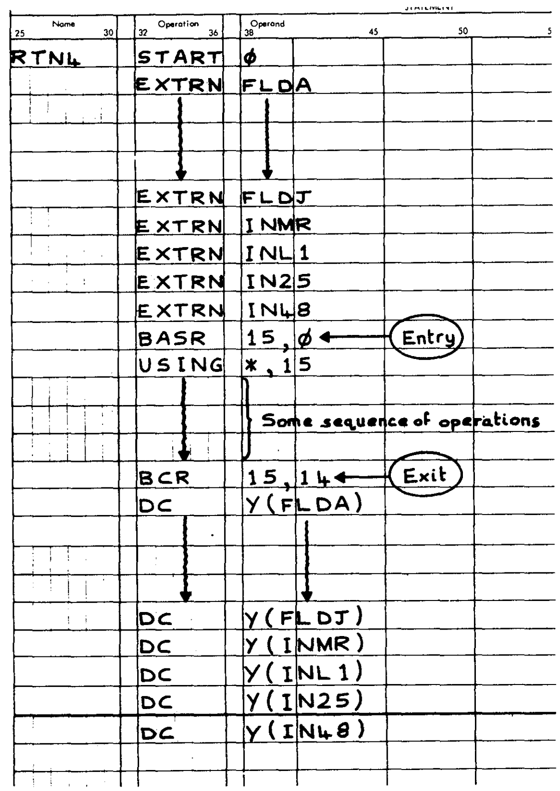


Figure 45. Coding Skeletons for Sample External Subroutine Application

Table Look-Up Operations

General Introduction

Model 20 RPG provides the ability to search through a core-stored table--known as an argument table--for table data--known as the table argument--that bears a predetermined relationship (high, low, or equal) to other designated RPG data (a literal, or the contents of a field)--known as the search argument. Execution of this look-up operation may be conditioned by indicators assigned to Control Level (cols. 7-8) and/or Indicators (cols. 9-17). The status of one or two Resulting Indicators reflects the type of match attained (high, low, equal, or none). These Resulting Indicators may be assigned to condition the execution of calculation and/or output specifications.

If desired, subsequent calculation and/or output specifications can call forth the argument-table data that satisfied the designated relationship, and/or an associated data entry from another table--termed a function table. The table argument and function data each remain available until the same tables are again used in a look-up operation, and an appropriate argument match is attained; if the predetermined search relationship is not satisfied, the former data still remains available to the program, unless an external subroutine instruction has placed other data into the same location.

Tables can be in ascending, descending, or random sequence; but unsequenced tables can only be scanned for an "equal" match. Any number of tables may be used in one program, within the limits of available core storage space. Argument and function tables need not be in the same sequence, nor need they have the same data format (alphameric or numeric). Any table in core storage can be employed as an argument table or as a function table, and this assignment need not be uniform for different look-up instructions in the program.

The table name, sequence (ascending, descending, or random), table-input arrangement (argument and function alternating or separate), and number of entries in a table, as well as the format of the entries themselves (alphameric or numeric, location of decimal point, size of field), are defined in the File Extension Specifications--described in the next chapter.

Tables in core storage cannot be updated or changed in any way by the Model 20 card RPG program.

Note: It is possible, by means of an external subroutine (see EXIT, above), to update tables during execution of an RPG program and, optionally, to punch them out in convenient format at object-program execution time.

However, a table cannot be expanded by additional entries at object-program execution time--unless its size specification was deliberately inflated and the table-input deck was padded with blank cards (see Number of Table Entries per Table, in File Extension Specifications).

Tables are loaded with the RPG source program at program-generation time, and are printed out at generation time exactly as entered. If the object program is punched out, the tables form an integral part of the punched object deck, in TXT-card format, so that they are reloaded with the object deck each time it is loaded to perform the particular job.

LOKUP (Table Look-Up)

This operation code, entered in cols. 28-32, causes a table search to be performed. It is used in conjunction with Resulting Indicators assigned in cols. 54-59 and with designation of a search factor (search argument), the name of a table to be scanned (argument table) and, optionally, the name of a table (called a function table) that is to provide a function associated with the argument.

The pertinent associated entries, in the same calculation-specification line, are:

1. Control Level (cols. 7-8)--optional.

    a. If blank, the look-up is performed at detail time, subject to Indicators (cols. 9-17).

    b. If L-indicator (L0, L1-L9, LR) is entered, the look-up is performed at total time, provided the particular L-indicator is then on, and subject to Indicators (cols. 9-17).

2. Indicators (cols. 9-17)--optional.

    On or off status of indicators may be designated to condition execution of the look-up operation.

3. Factor 1 (cols. 18-27): search argument--required.

    Enter the search argument, left-aligned (i.e., beginning in col. 18).

    This may be either:

    a. An alphameric or numeric literal, or

    b. A field name defined in the input specifications or elsewhere in the calculation specifications, or

    c. The name of a table. A table entry selected in a previous LOKUP operation may thus become a search argument.

4. Factor 2 (cols. 33-38): argument table--required.

    Enter (left-aligned) the name of the table to be searched for data that bears a predetermined relationship (see Resulting Indicators, item 8, below) to the search argument.

    All table names begin with the letters TAB, followed by one, two, or three alphabetic and/or numeric characters--see File Extension Specifications, section Table Name--Cols. 27-32.

    Note: Data in the argument table must have the same total field length and format (alphameric or numeric) as the search argument; but the decimal-point location (for numeric fields) need not be identical. No decimal alignment is performed by the program in a LOKUP operation.

5. Result Field (cols. 43-48): function table--optional

    Enter (left-aligned) the name of the table from which an associated function is to be retrieved (after an appropriate table argument--Factor 2--has been located to satisfy the search argument --Factor 1).

    Note: The manner in which the program determines the function-table entry that corresponds to an argument-table entry is described below (Performance and Results of a Table Look-Up Operation, part 2).

    The Result Field is left blank if a corresponding function from another table is not needed. It may only be desired to ascertain whether a table argument of the appropriate relationship to the search argument is present --as indicated by the status of one or two Resulting Indicators (see item 8, below); or, it may be desired to use only the table argument that satisfied a criterion of High or Low, and therefore is not identical to the search argument.

6. Field Length (cols. 49-51) and Decimal Positions (col. 52)--optional; may always be left blank. These fields must be left blank if no function table (Result Field) is specified.

   If a function table is specified (in cols. 43-48), Field Length and Decimal Positions (if numeric) may be defined here. This is, however, superfluous, since all tables must in any case be defined in the File Extension Specifications. If the Result Field is redefined here, the Field-Length and Decimal-Positions entries must accord with those in the File Extension Specifications.

7. Half-Adjust (col. 53): leave blank

8. Resulting Indicators (cols. 54-59)--at least one entry required (but never more than two--see below).

   The argument table (Factor 2) is searched for an entry that bears to the search argument (Factor 1) the relationship designated by the assignment of one or two Resulting Indicators.

   Note: Tables are stored in the order in which the table data is loaded--no check is made to assure that the table sequence adheres to any sequence (ascending or descending) that may be specified in the File Extension Specifications. The search always commences with the table entry loaded first, and progresses entry-by-entry until the designated search condition is satisfied or the search is terminated. Thus, if an ostensibly sequential table is out of order, inappropriate table data may be selected and incorrect Resulting-Indicator setting may be caused (see samples, below).

Resulting Indicators assignments in a LOKUP operation have the effects itemized below:

A Resulting Indicator assigned to Equal (cols. 58-59) instructs the program to locate an argument-table (Factor 2) entry equal to the search argument (Factor 1). The indicator turns on if such an entry is found; otherwise, it turns off.

Note: The status of a Resulting Indicator assigned to Equal of a LOKUP operation is not affected by a Blank-After instruction in the output-format specifications, nor is it set on at the beginning of program execution.

An indicator assigned to Low (cols. 56-57) instructs the program to locate that argument-table entry that is nearest to, but lower in sequence than, the search argument. The indicator turns on if such an entry is found; otherwise, it turns off.

An indicator assigned to High (cols. 54-55) instructs the program to locate that argument-table entry that is nearest to, but higher in sequence than, the search argument.

At least one Resulting Indicator must be assigned. If an indicator is assigned to Equal and to High, or to Equal and to Low, the program searches for an argument-table entry that satisfies either one of the two designated conditions, with Equal given precedence. The indicator for the condition that was satisfied turns on; the other indicator turns off--unless the same indicator is assigned to both conditions, in which case it will be on. If neither condition is satisfied, the indicators turn off.

When several successive identical entries exist in the argument table, the first one encountered that meets the appropriate search criteria is selected: for an Equal condition, this is the first equal value; for a High or Low condition, it is the high or low entry physically closest to equal, provided the table is in proper order. The significance of this (i.e., why it matters which of several equal entries is treated as the "hit") becomes apparent when function tables are discussed (below).

If the argument table is not specified to be in ascending or descending sequence, a search can only be made for an Equal condition. (If an indicator is assigned to High or Low, but sequence is not designated for the argument table in the file extension specifications, a warning message is printed at program-generation time.)

Note: The column headings of High and Low for Resulting Indicators must be considered reversed for the LOKUP operation--for LOKUP

   High stipulates: Factor 2 > Factor 1
   Low stipulates: Factor 2 < Factor 1

The expanded explanations below cover further particulars--including the contingency of an argument table, defined as being in sequence, being out of order:

1. If no sequence is designated in the file extension specifications (col. 45 or 57)--regardless of whether the table is actually in sequence: A (any) Resulting Indicator must be assigned to Equal (cols. 58-59).

   Note: If no sequence is specified in the file extension specifications, a Resulting Indicator assigned to High (cols. 54-55) or Low (cols. 56-57) will be ignored--i.e., retains its former setting--if an indicator is also assigned to Equal; if none is assigned to Equal, the indicator assigned to High or to Low is treated as though assigned to Equal. If no Resulting Indicator is assigned to Equal, but different indicators are assigned to both High and Low, the indicator assigned to High is treated as assigned to Equal; the other one is ignored. (A warning message is printed during program generation.)

The program searches through the argument table (Factor 2) until it either finds the first data that matches the search argument (Factor 1) exactly, or the entire table has been scanned--whichever occurs first.

If a match is found, the Resulting Indicator assigned to Equal turns on; if no match is found, it turns off.

Example: Search argument (Factor 1) = 5.
Argument table (Factor 2) = 1, 3, 5, 5, 8, 9.
The first (underlined) 5 is selected, and the Resulting Indicator assigned to Equal turns on.

2. If Ascending argument-table sequence is designated in the file extension specifications: The table is assumed to be loaded in ascending sequence.

A (any) Resulting Indicator must be assigned to at least one of the three fields (cols. 54-55, 56-57, 58-59); but the same or different indicators may also be effectively assigned to two fields: High and Equal, or Low and Equal. The indicator for the condition that is satisfied turns on; the indicator assigned to a second condition turns off, unless it is the same indicator.

If indicators are assigned to High and Equal, or to Low and Equal, Equal takes precedence if the Equal condition can be satisfied (and provided the table is in proper sequence).

a. If a Resulting Indicator is assigned only to Equal (cols. 58-59): The look-up operation is identical to that described under 1, above. Nothing is gained by designating a sequence in the file extension specifications.

Example--which illustrates the search for Equal through the entire table, even though the table is out of order:

Search argument = 5.
Argument table = 1, 2, 9, 10, 4, 5, 6.
5 is selected.

b. If a Resulting Indicator is assigned only to High (cols. 54-55): The first argument-table entry encountered which is higher in sequence than the search argument is selected (i.e., becomes the data accessed whenever, thereafter, the table name is used as a field name, before another LOKUP operation on the same table).

Examples:

(i) Search argument (Factor 1) = 5.
Argument table (Factor 2) = 1, 3, (5), 8, 9, ...
8 is selected as satisfying the search conditions, regardless of whether the entry 5 is present in the table.
The Resulting Indicator assigned to High turns on.

(ii) Search argument = 5.
Argument table = 1, 1, 2, 3, 3, 4, 5, 5, 5.
No table entry satisfies the search condition.
The Resulting Indicator assigned to High turns off.

But note:

(iii) Search argument = 5.
Argument table = 1, 8, 8, 6, 5, 6, ... (table out of sequence)
The first (underlined) 8 is selected, although 6 is nearer to 5 in value and position; but the first 8 is the first value greater than 5 that is encountered.
The Resulting Indicator assigned to High turns on.

c. If a Resulting Indicator is assigned only to Low (cols. 56-57): The argument-table entry that is nearest to, but lower in sequence than, the search argument is selected--provided the table is in proper sequence.

Note: To generalize for any sequence in which the table might actually be (when ascending sequence is specified): The last argument-table entry is selected which precedes the first entry that is either equal to, or higher than the search argument.

Examples:

(i) Search argument = 5.
Argument table = 1, 3, (5), 8, 9 ...
3 is selected as satisfying the search condition, regardless of whether the entry 5 is present in the table.
The Resulting Indicator assigned to Low turns on.

But note:

(ii) Search argument = 5.
Argument table = 1, 2, <u>2</u>, 10, 3, 4, 5, 6 (table out of sequence).
The second (underlined) 2 is selected, although 4 is closer to 5 in value and position. The second 2 is the entry that immediately precedes the first-encountered entry (10) that is equal to or higher than the search argument (5).
The Resulting Indicator assigned to Low turns on.

d. If Resulting Indicators are assigned both to High and Low (irrespective of whether an indicator is also assigned to Equal): The indicator assigned to Low is ignored, and retains its former status. (The effect of the indicator assigned to High is explained above and below.)

e. If Resulting Indicators are assigned to High (cols. 54-55) and Equal (cols. 58-59): The first argument-table entry encountered which is either equal to, or higher in sequence than, the search argument is selected.

Examples:

(i) Search argument (numeric) = +5.
Argument table = 1, 3, <u>5</u>, 5, 7, 9.
The first (underlined) 5 encountered is selected as satisfying the search conditions: an entry is always tested first for Equal, if an indicator is assigned to Equal.
The Resulting Indicator assigned to Equal turns on; the indicator assigned to High turns off, if it is a different indicator. If the same indicator is assigned to both conditions, it turns on if either condition is satisfied; it turns off only if neither an Equal nor a High condition was satisfied.

(ii) Search argument (numeric) = 5.
Argument table = -8, -5, -4, 0, +1, 3, +7, 9.
+7 is selected, being the first-encountered value algebraically equal to or higher than 5.
The Resulting Indicator assigned to High turns on; the indicator assigned to Equal turns off, unless it is the same indicator.

(iii) Search argument = 5.
Argument table = 1, 1, 2, 3, 3, 4, 4.
No table entry satisfied the search conditions.
The Resulting Indicators assigned to High and to Equal turn off.

f. If Resulting Indicators are assigned to Low (cols. 56-57) and Equal (cols. 58-59): The first argument-table entry that satisfies either condition--as explained in (a) and (c) above--is selected. However, each table entry is tested first to see whether it is equal to the search argument: if higher, the program then selects the last preceding lower entry.

Examples:

(i) Search argument (alphameric) = N.
Argument table = A, +5, E, J, -5, N, P, S, 5.
-5 (=N) is selected (i.e., first equal encountered): an entry is always tested first for Equal, if an indicator is assigned to Equal.
The Resulting Indicator assigned to Equal turns on; the indicator assigned to Low turns off, unless it is the same indicator.

(ii) Search argument (alphameric) = N.
Argument table = A, +5, E, J, <u>J</u>, P, S, 5.
The second (underlined) J is selected: P is higher than N (in the EBCDIC sequence); the program therefore selects the last preceding

lower entry.
The Resulting Indicator assigned to Low turns on; the indicator assigned to Equal turns off, unless it is the same indicator.

But note:

(iii) Search argument (alphameric) = N.
Argument table = A, E, P, E, N, S (table out of sequence).
E is selected: when P (higher than search argument) is reached, the program selects the last preceding lower entry, although an exact match (N) exists in the table.
The Resulting Indicator assigned to Low turns on; the indicator assigned to Equal turns off, unless it is the same indicator.

3. If Descending argument-table sequence is designated in the file extension specifications: The table is assumed to be loaded in descending sequence.

The effect of LOKUP operations is identical as for ascending tables (see 2, above). The only differences occur when supposedly sequential tables are out of order.

Examples

a. If Resulting Indicators are assigned to Low (cols. 56-57) and Equal (cols. 58-59).

(i) Search argument (Factor 1) = 5.
Argument table (Factor 2) = 9, 8, 6, <u>5</u>, 5, 3, 1.
The first (underlined) 5 encountered is selected.
The Resulting Indicator assigned to Equal turns on; the indicator assigned to Low turns off, unless it is the same indicator.

(ii) Search argument = 5.
Argument table = 9, 8, 6, 6, <u>3</u>, 3, 1, 0.
The first (underlined) 3 encountered is selected.
The Resulting Indicator assigned to Low turns on; the indicator assigned to Equal turns off, unless it is the same indicator.

(iii) Search argument = 5.
Argument table = 9, 9, 8, 7, 7, 6.
No table entry satisfies search conditions.
The Resulting Indicators assigned to Low and to Equal turn off.

But note:

(iv) Search argument = 5.
Argument table = 9, 1, 4, 5, 4, 3 (table out of sequence).
1 is selected, although there is an Equal value in the table: 1 is the first value encountered that is lower than 5, and it is encountered before 5.
The Resulting Indicator assigned to Low turns on; the indicator assigned to Equal turns off, unless it is the same indicator.

b. If Resulting Indicators are assigned to High (cols. 54-55) and Equal (cols. 58-59):

(i) Search argument = S.
Argument table = Z, V, T, R, B.
T is selected.
The Resulting Indicator assigned to High turns on; the indicator assigned to Equal turns off, unless it is the same indicator.

But note:

(ii) Search argument = S.
Argument table = Z, W, R, T, S, K (table out of sequence).
W is selected, although there is an entry S in the table: W is the nearest entry preceding an entry lower in sequence than the equal value (S), and the lower value (R) is encountered before the equal value (S).
The Resulting Indicator assigned to High turns on; the indicator assigned to Equal turns off, unless it is the same indicator.

Performance and Results of a Table Look-Up Operation

1. Using a single table

This provides indicator settings that reflect the success (if any) and nature of the match achieved (High, Low, or Equal), and--if a match was achieved--access to the appropriate argument-table data selected.

As explained in detail above: The operation code LOKUP is specified in Operation;

The search argument (literal or field name) is entered in Factor 1;

The name of the argument table is entered in Factor 2;

One or two Resulting Indicators are assigned--to determine the type of match desired between the argument-table and the search-argument entries, and to reflect the result;

An L-indicator is entered in Control Level if the LOKUP is to take place during total-time calculations; Control Level is left blank if the operation is to be performed at detail time;

If desired, execution of the operation is made contingent on the status of conditioning indicators entered in Control Level (cols. 7-8) and in Indicators (cols. 9-17).

When the LOKUP operation is terminated:

The status of the Resulting Indicators reflects the result of the table search.

A core storage "hold" area, previously assigned to each table by the program, contains the argument-table data that was selected because it satisfied a search criterion (Equal, High, or Low). (This is a separate core location--not part of the location where the table is stored. The integrity of the storage of the table itself is not disturbed.)

If a search criterion was not satisfied (i.e., no match was found of the type specified by the indicators assigned in Resulting Indicators), no move to the hold area is performed. It then retains its former contents: the data selected as a result of a previous LOKUP operation on the same table; or, data placed there by using that table name as a result field in an external subroutine; or, if nothing was ever placed into the field, blanks (if alphameric) or zeros (if numeric).

Subsequent availability of data selected from a table in a LOKUP operation:

Whenever a table name is used as a field name in Factor 1 or Factor 2--in an operation other than LOKUP--this actually references the hold area for LOKUP-selected table data, not the table-storage area itself. The last data placed in that hold area is thus accessed by use of the field name in any other operation.

By entering the argument-table name as a Factor in other calculation operations, the data last selected from the argument table can be used as source data for other calculation specifications; by using the argument-table name as a Field Name in the output-format specifications, the data last selected from the argument table can be printed and/or punched.

By using the argument-table name as Result Field in an external subroutine, the contents of that hold area can be changed--but this does not alter the table data, which is stored elsewhere. (The selected argument-table data in the hold area can be made available to external subroutines by identifying the field by the field name in the Result Field of an RLABL line. Note, however, that the table name must consist of exactly four characters--TABx.) If the argument-table hold area is not referenced as a field immediately after the LOKUP operation, the user must be careful not to alter its contents (by use in an external subroutine) if he intends subsequently to utilize the selected argument-table data.

The argument-table name may also be used as Factor 1 (search argument) in a LOKUP operation. The search argument is then the data selected from the argument table in a previous LOKUP operation, or data placed in that hold area by an external subroutine.

A table name must not be used in Result Field except in a LOKUP or RLABL operation.

<u>Note</u>: For the format of a table name, refer to the File Extension Specifications, section <u>Table Name--Cols. 27-32</u>.

2. Use of two tables in a LOKUP operation.

   All statements made for use of a single table (see 1, above) apply. In addition, data in a corresponding position of a second table--called a function table--becomes available when a match is achieved between the search argument and data in the argument table.

   The name of the function table is specified in Result Field (cols. 43-48). (If desired, Field Length and Decimal Positions (if numeric) may also be redefined; but this is redundant, since they must be defined in the file extension specifications.)

<u>Note</u>: For the format of a table name, refer to the File Extension Specifications, section <u>Table Name--Cols. 27-32</u>.

Data in a function table need not conform to the data format--field length, alphameric or numeric data, or decimal positions--in the argument table with which it is to be associated; nor need the function table adhere to the same data sequence as the argument table.

The function table may contain the same number of entries as, or more entries than, any argument table with which it is to be associated.

If a function table contains less entries than an argument table with which it becomes associated in a LOKUP operation, a warning message is printed at program-generation time. At object-program time:

Proper function data is selected (i.e., made available, by the table name as a field name, to subsequent operations) if the selected argument-table entry is the nth entry (counted from the front of the argument table), and n is equal to or less than the number of entries in the function table. If n is greater than the number of entries in the function table, the function data selected is unpredictable: it is obtained from a core storage location outside the area occupied by the function table. (Blanks or zeros can be assured in such case by increasing the specified size of the function table to match that of the argument table, and appending an appropriate number of blank cards to the table.)

When the LOKUP operation is successful in satisfying a designated relationship (Equal, High, or Low--as specified in Resulting Indicators) between an argument-table entry and the search argument, the corresponding entry from the specified function table is also placed in a "hold" area. Thereafter, its data is available--in the same manner as explained above for the argument table--for subsequent operations, by using the table name as a field name. This includes the possibillity of specifying that table name as Factor 1 in a subsequent LOKUP operation. The function selected in a previous LOKUP operation can thus serve as search argument for a subsequent one.

The funtion-table data selected for the hold area is determined by the program as follows:

The program establishes the relative position of the selected argument-table entry (Factor 2), by counting from the front of the argument table. It then selects, from the specified function table (Result Field), the same relative entry, counted from the front of the function table.

For example:

Search argument = 5.
Argument table = 1, 3, 4, <u>5</u>, 5, 6, 8.
Function table = B, R, T, K, A, W, F, G, L.
Resulting Indicator assigned to Equal.
The indicator assigned to Equal turns on.

The first (underscored) entry with 5 is selected from the argument table for its hold area. Subsequent operations referencing the argument-table name access that hold area, and the data supplied to the operation is the value 5.

The first 5 is the fourth entry in the argument table. Therefore, the fourth entry--which contains the character K--is moved from the function table to its hold area. References to the function-table name in subsequent operations access that hold area, and the data supplied to the operation is the character K.

This also illustrates that the program's consistent selection of a particular one of several equal argument-table entries (see the two 5s above) affects function-table entry selection. (If the second 5 had been selected--although irrelevant to subsequent use of the argument-table name as a field name--a different function-table entry would have been selected--namely, A.)

Note the effect on function-table-entry selection when the argument table is not sequenced or, although supposedly sequenced, is out of order:

Search argument = 5.
Argument table = 1, 3, 4, 6, 8, 5, 5.
Function table = B, R, T, K, A, W, F, G, L.
Resulting Indicator assigned to Equal.
The indicator assigned to Equal turns on.

The first (underscored) entry with 5 is selected from the argument

table. This is now, however, the sixth (rather than the fourth) entry (see previous example). Therefore, W (instead of K) is selected from the function table.

There is no inherent connection between an argument table and a function table. Each table is stored separately--even if loaded from cards with alternating entries for two different tables (see below). Any table that has been properly defined and loaded may be utilized as an argument table--by entering its name in Factor 2 of a LOKUP operation--or as a function table--by entering its name in Result Field of a LOKUP operation. Furthermore, the same table may serve the two different purposes in different LOKUP operations.

Examples of table look-up operations, and related calculation specifications, follow discussion of the File Extension Specifications--next chapter.

Points to Note for LOKUP operations

1. Comparison between data in the argument table and the search argument is:

   a. Algebraic--if the field is defined as numeric; i.e., negative values are lower in sequence than positive or unsigned values.

   b. Logical--if the field is defined as alphameric; i.e., the comparison is based on the EBCDIC sequence (see Figure D1, Appendix D), and this sequence cannot (for LOKUP) be altered by a translation table.

2. The search argument and the argument-table data (but not necessarily the function-table data) must have the same data format: both defined as alphameric or as numeric, and both must have the same total field length (including any decimal positions); but the number of decimal places (if numeric) may differ between search argument and argument-table data.

3. No decimal alignment is performed between search argument and argument-table data: the two fields are compared in their entirety.

4. Table fields may have a maximum size of

   a. 15 positions, if defined as numeric

   b. 80 positions, if defined as alphameric

5. More than one function can be selected for one associated argument.

   The several functions must occupy contiguous groups of columns which are jointly defined as one field in the function table; i.e., the length of a field is defined as the aggregate of the number of columns for the several function fields.

   After a LOKUP operation, the several functions are separated into different fields by MOVE and MOVEL operations. (See Figure 48C and <u>Programming Tips</u>, Appendix E.)

6. Since a table is searched sequentially from the beginning, look-up for an Equal match can be significantly expedited by creating the table with entries in decreasing order of frequency of occurrence of the search argument: the argument that occurs most often should be the first table entry, etc. (Of course, any function table to be associated with such an argument table must be organized to correspond.)

Creating Table-Input Cards

<u>Table-Input Format</u>. A set of table-input cards may be devoted to

1. Entries for a single table, or

2. Alternating entries for two tables.

Each set of cards representing a single table, or alternating entries for two tables, must be loaded together (and in the proper sequence, if sequence is relevant) at program-generation time. The sets of cards for different tables must be grouped, for loading, in the same order in which the tables are described in the File Extension Specifications.

While it is common, when entries for two tables alternate in each card, that they represent the related arguments and functions for two associated tables, this need not be the case. When the tables are loaded, the program stores all entries for one table in one contiguous area, and those for another table in another area--irrespective of whether the two tables are read as alternating entries in one set of cards or from two different sets of cards. The association of two tables by alternating entries in one set of cards has no bearing on their serving subsequently as argument or function tables: any table that has been loaded can be stipulated to serve as argument table (by entry of its name in Factor 2) or as function table (by entry of its name in Result Field) in any LOKUP operation.

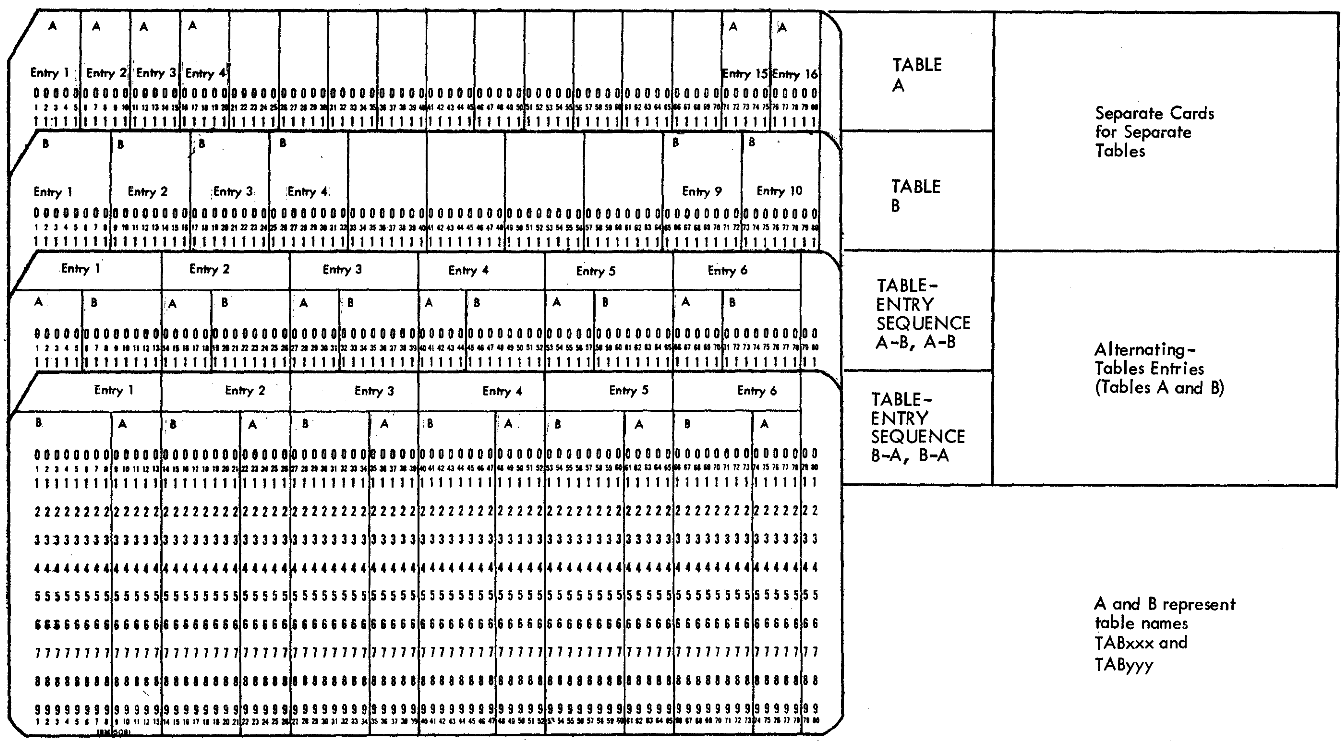


Figure 46. Two Methods of Creating Table-Entry Cards

Benefits of alternating entries for two tables in one set of table-input cards may lie in:

1. Keypunching or reproducing convenience, if the data for the two tables was grouped in the source documents; or

2. Assurance that, if the associated entries are to serve as argument and function, respectively, the related data cannot get out of phase with each other if the cards should get out of order.

Figure 46 shows examples of different techniques for entering two tables.

Rules for Creating Table-Input Cards

1. A table may have entries defined as alphameric or as numeric; but all entries for one table are defined (in the file extension specifications) as one or the other. (Of course, a field defined as alphameric may contain numeric data.)

   Fields defined as numeric are, as always, limited to 15 columns each.

   Fields defined as alphameric are limited to 80 columns each (see item 6, below).

2. Data-entries must begin in column 1 of each table-input card (note Figure 46).

3. All cards (except the last) of a table-input deck must contain the same number of table entries. (The last card may contain less entries.) It is, however, not required--although usually done--that these represent the maximum number of complete entries that can fit in a card.

4. All entry fields in a table-input card must be contiguous: intervening blank spaces (that are not part of the defined field length) are not permitted (note Figure 46).

5. All entries for one table must be the same length: i.e., the field length for one table must not vary. (Of course, with the alternating-table format, the fields for the two tables may be of different lengths.) Note Figure 46.

6. Entries must not be split between two cards. Sufficient columns must be left blank at the right (high-column-number) end of each card--if necessary--to complete an entry in a single card (note Figure 46, Alternating-Table Entries). (This precludes alphameric table entries in excess of 80-column length.)

In alternating-table format, the entries for the two tables also must not be split between two cards.

7. In alternating-table format, each card must begin with an entry for the same table as every other card in the set.

8. Each table may be ascending, descending, or in no particular sequence. In alternating-table format, the two tables need not be in the same sequence.

9. Any of the 256 EBCDIC characters are allowed in alphameric fields. Thus, blanks are permitted as contents of alphameric table-entry fields.

   Blanks within numeric table-entry fields are converted to zeros (as the program packs the field); however, a blank in the low-order position will yield an invalid sign (hexadecimal zone 4; i.e., EBCDIC-table row labelled 4)--this will cause an abortive program stop if that field is used in LOKUP or arithmetic operations (including numeric compare).

10. Packed format cannot be specified for table-input data.

11. The table-input cards for each table must contain the exact number of entries specified in the file extension specifications.

    Note: Blank cards (or fields) may be appended (or interspersed) to satisfy a larger number of entries specified in the file extension specifications (but note item 9, above). The user must then understand the possible effect of blank or zero-value entries on a LOKUP operation with Resulting Indicators assigned to High or Low.

12. Use of the alternating-table format requires that both tables in the single table-input card deck have the same number of entries. (If one table contains more entries than the other, the equivalent of the necessary number of additional entries for the shorter table can be achieved by leaving the corresponding columns blank, so that the length of entries remains uniform.)

FILE EXTENSION SPECIFICATIONS (OPTIONAL)

GENERAL INFORMATION

Each table to be used with the RPG program (see Table Look-Up Operations) must be defined in file extension specifications (see Figure 47).

Each table that occupies a separate set of input cards is described in the left portion (cols. 27-45) of a single file extension specifications line.

When two tables are recorded in alternating format in one set of cards, the table whose entry appears first (leftmost) in the table-input cards is defined at the left (cols. 27-45), and the table whose entry appears second in the table-input cards is defined at the right (cols. 46-57). This has no bearing on which, if either, of the two alternating-entry tables is to serve as an argument or a function table.

Columns 7-26, 43, and 55 are not applicable to this program.

If several tables in separate card decks are involved, the decks must be loaded in the same order in which their names appear in the file extension specifications: the cards for the table defined in the first line must be loaded ahead of those for the table defined in the second line, etc. This is the only means the program has of associating a table with a specific table name.

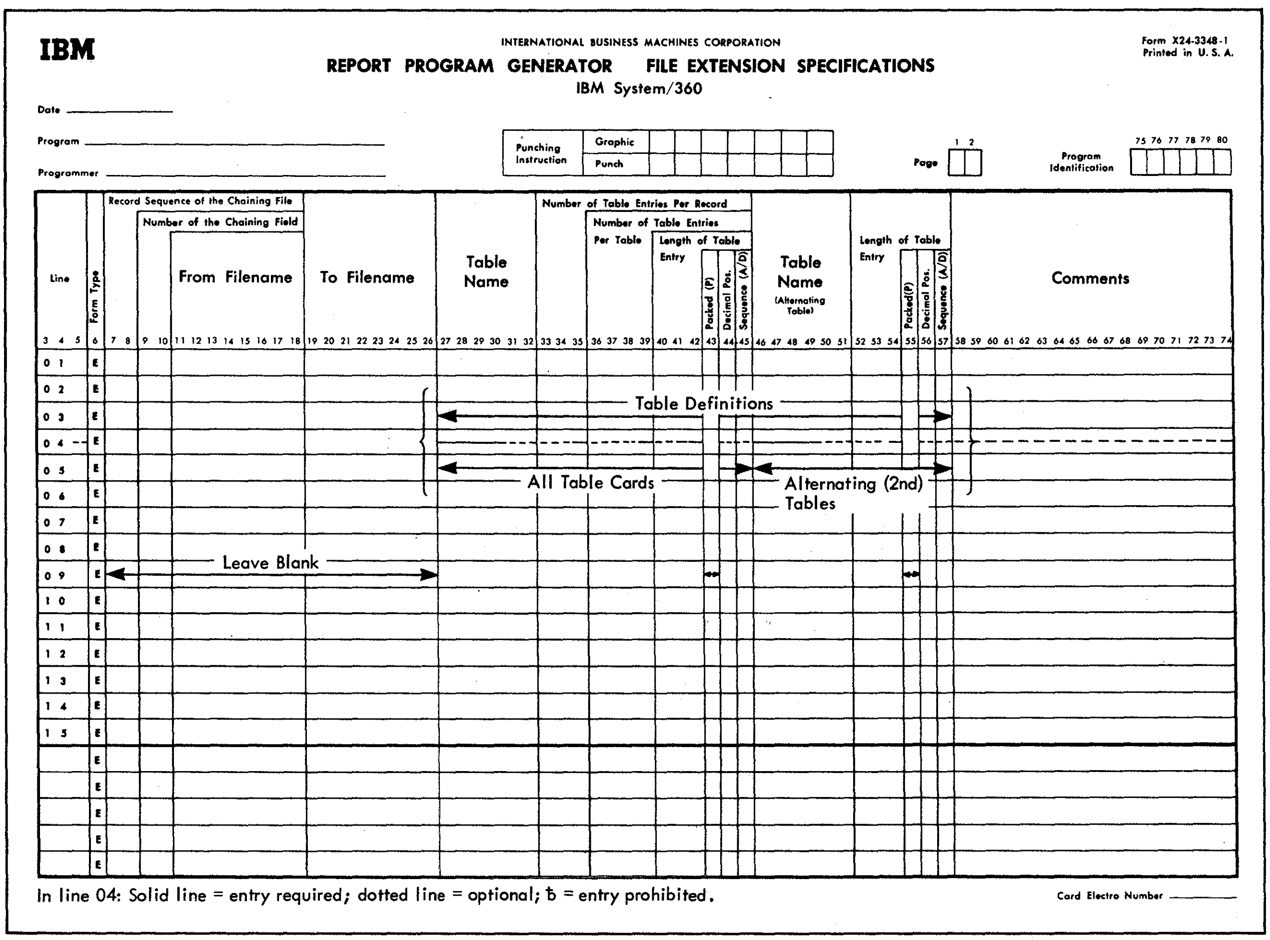


Figure 47. The File Extension Specifications Form

SPECIFICATIONS FOR SINGLE-TABLE DECKS, AND FOR FIRST TABLES OF ALTERNATING-TABLES DECKS (COLS. 27-45)

Table Name--Cols. 27-32

The table name must be four, five, or six characters long, starting in col. 27. The first three characters must be the letters TAB; the additional one, two, or three characters may be alphabetic or numeric (but not special characters or embedded blanks).

If a table is to be identified in an RLABL line, for use in an external subroutine, the table name must be exactly four characters long, the first three being TAB. This is because such a subroutine is written --for card RPG-- in Basic Assembler Language, in which names must not be longer than four characters.

If data for two tables is alternated in a single set of table-input cards, the entry here (cols. 27-32) names the table whose data occupies the first (leftmost) field in the table cards.

Number of Table Entries per Record--Cols. 33-35

The number of table entries per table-input card is recorded in cols. 33-35, right-justified. (The last card may contain fewer entries.) Recording of leading zeros is optional.

If data for two tables is alternated in a single set of table-input cards, the two contiguous entries--each for one of the two tables--is counted here (cols. 33-35) as a single entry. For instance, the specification here (in cols. 33-35) for the alternating-tables (A-B or B-A) example in Figure 46 would be 6 (not 12).

Number of Table Entries per Table--Cols. 36-39

The total number of entries in the table is recorded in cols. 36-39, right-justified. Recording of leading zeros is optional.

This value must correspond exactly to the number of entries in the table to be loaded.

If it is desired to allow for ultimate expansion of (or insertions in) the table, the value here can be inflated--provided an appropriate number of blank cards, or fields, representing the proper number of entry fields, is appended to (or interspersed in) the table-input deck (but note Rules for Creating Table-Input Cards, item 9, under Table Look-up in the Calculation Specifications, above). If data is subsequently to be substituted for the blank entries in the table-input deck, the program must be regenerated. Alternatively, the excess area (containing blanks or zeros) reserved for the table by the program--by virtue of the blank table-input cards or fields and the inflated table size specified--may have table data placed in it by an appropriate external subroutine.

Note: Since the number of entries for the two tables in alternating-table format must be equal, the specification in cols. 36-39 is the same, regardless of which of the two tables is considered--the number represents the count of entries for one table.

Length of Table Entry--Cols. 40-42

The number of columns for one entry for one table is recorded in cols. 40-42, right-justified. Recording of leading zeros is optional.

Tables defined as numeric (see col. 44) are limited to a maximum of 15 columns per entry. Alphameric table entries may be up to 80 columns long.

If data for two tables is alternated in a single set of table-input cards, the specification here (cols. 40-42) applies to the table whose entry appears first (leftmost) in the table-input cards. For instance: for the table exemplified by the third sample card in Figure 46, the specification in col. 42 would be 5; for the fourth sample card, it would be 8.

Each entry in tables used as argument tables must have the same total length (including any decimal places) as the search argument (see Look-Up Operations, above).

Packed--Col. 43

Leave blank. This program does not permit the designating of table-input format as packed.

Decimal Positions--Col. 44

Format definition for data in the first (or only) table of table-input deck:

- ƀ = alphameric
- 0 or N = numeric, with no positions to the right of the decimal point
- 1-9 = numeric; 1-9 positions, respectively, to the right of the decimal point.

Entries in tables used as argument tables must be defined with the same data format (alphameric or numeric) as the search argument; but the defined position for the decimal point may differ. No decimal alignment is performed during the LOKUP operation--the entire search-argument field is compared with an entire argument-table field. Therefore, if the table fields are not used in compare (COMP) or arithmetic operations, any of the codes N, 0-9 (within field-size limit) may be assigned to a numeric field, regardless of the actual number of decimal places.

Comparing between argument table and search argument is algebraic for tables defined as numeric (i.e., negative values are smaller than zero or than positive values); comparing is "logical" (according to the EBCDIC sequence) for alphameric tables.

If data for two tables is alternated in a single set of table-input cards, the specification here (col. 44) applies to the table whose entry appears first (leftmost) in the table-input cards.

Sequence--Col. 45

If the table will be used as an argument table, and entries are to be matched as "high" or "low" against the search argument (Resulting Indicators assigned to High or Low in the calculation specifications), the table must be defined as being in either ascending of descending sequence:

A = ascending sequence
D = descending sequence

No check is made by the program that the table conforms to the specified sequence.

If the table either will not be used as an argument table, or its entries will only be matched against the search argument for an "equal" condition (no Resulting Indicator assigned to High or Low in the calculation specifications), no entry is required in Sequence (col. 45)--an entry will be ignored.

If data for two tables is alternated in a single set of table-input cards, the specification here (col. 45) applies to the table whose entry appears first (leftmost) in the table-input cards.

SPECIFICATIONS FOR SECOND TABLES OF ALTERNATING-TABLES DECKS (COLS. 46-57)

All fields in this right-hand portion of the file extension specifications have the identical significance as the like-titled fields in the left-hand portion (cols. 27-45)--but they apply only to the second table (i.e., the table whose entry appears second) in table-input cards with alternating-tables entries.

This section is left blank for a table-input deck that contains entries for only a single table.

Table Name--Cols. 46-51

The name assigned to the second table in one table-input deck.

Length of Table Entry--Cols. 52-54

The number of columns in an entry for the second table in one table-input deck.

Packed--Col. 55

Leave blank.

Decimal Positions--Col. 56

Format definition for data in the second table of one table-input deck.

ƀ = alphameric
0 or N = numeric, with no positions to the right of the decimal point.
1-9 = numeric; 1-9 positions, respectively, to the right of the decimal point.

Sequence--Col. 57

Sequence of the second table in an alternating-table input deck.

Note: There are no specifications, for the second table in a single table-input deck, for "Number of Table Entries per Record" or "Number of Table Entries per Table": specifications in cols. 33-35 and 36-39 cover both tables in one table-input deck.

COMMENTS--COLS. 58-74

The user may enter here any data he wishes to have printed out, next to the specifications in the line, at program-generation time. Apart from this, the entries are ignored by the program.

Figures 48A, B, and C illustrate file extension, table look-up, and related calculation specifications. In order to demonstrate a variety of possibilities, some of the examples are rather artificial from an applications viewpoint.

ILLUSTRATION AND EXPLANATION OF USE OF TABLES--FIGURES 48A, B, AND C

Calculation Specifications--Figure 48C

All operations shown are performed at detail time: Control Level (cols. 7-8) is blank.

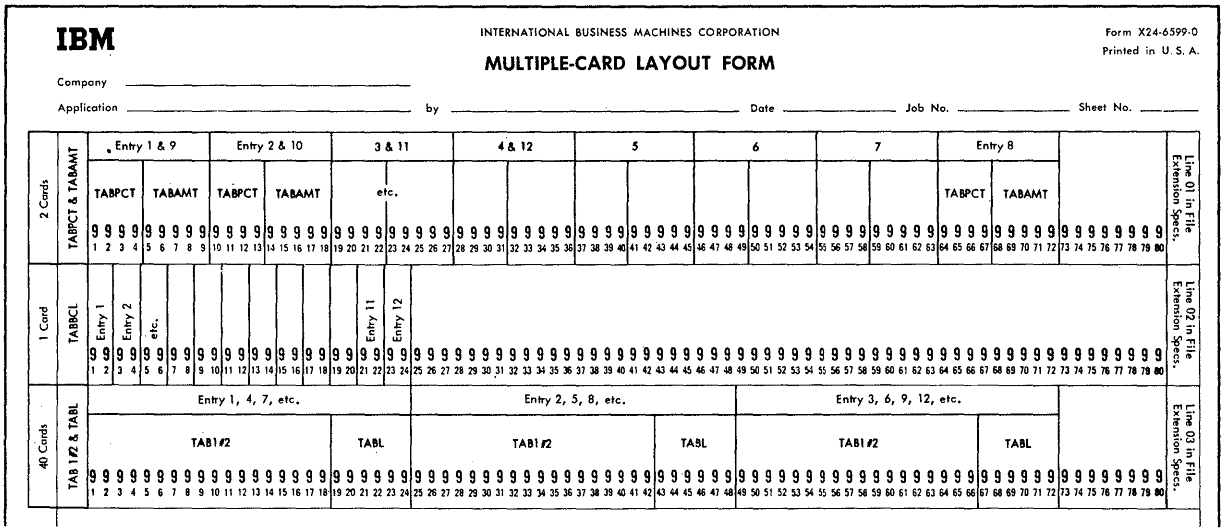


Figure 48A. Table Look-Up--Table-Input Card Format

Line 01. Entries in the table named TABPCT (say, bonus percent)--the argument table by virtue of its entry in Factor 2--are compared with the contents of a field named PRFPCT (say, performance percentage)--the search argument, by virtue of its specification in Factor 1. The field PRFPCT must be defined (elsewhere) as numeric and four columns long, because TABPCT is so defined (in the file extension specifications): search argument and argument-table entries must have the same format and total length, except for location of the decimal point. The comparison is algebraic, because the fields are numeric (unsigned--EBCDIC zone F--values are therefore treated as equal to values signed plus).

The table is defined (in the file extension specifications) as in ascending sequence. The same Resulting Indicator (20) is assigned to Equal and High; it turns on if either condition is satisfied. The program first attempts to locate an equal entry in the table; if there is none in the proper sequential position, it chooses the closest table entry that is higher in value than the search argument. We have assumed that bonus percent figures go in steps in the table; if none matches the exact performance percentage, the employee is to be credited with the nearest higher value.

If an argument-table entry (say, the nth entry in this table) satisfies the search criteria specified (in Resulting Indicators), that TABPCT value is stored in the hold area for this table. The entry in the corresponding nth position of the table TABBCL (say, bonus class)--the function table, by virtue of its specification in Result Field--is also selected, and placed in the hold area for that table. If the search did not result in a "hit", neither hold area is disturbed.

TABBCL is defined (in the file extension specifications) as alphameric, each entry 2 columns long, and no sequence is specified for the table. (The program does not verify the sequence of the table even if Sequence is specified in the File Extension Specifications.)

Line 02. The applicable bonus class code selected is to be used in output-format specifications; it must, therefore, be preserved before the TABBCL table is again employed in a LOKUP operation.

The MOVE of TABBCL to BCNCLS transfers the bonus class code selected in line 01 from the TABBCL hold area to a new field. TABBCL is defined (in the file extension specifications) as 2 columns long per entry, and alphameric. BCNCLS is therefore similarly defined (it could have been defined differently if there were a reason therefor).

The specification is executed only if the preceding operation yielded a "hit" (indicator 20 on).

Line 03. The specification of TABBCL in Factor 1 causes the contents of the TABBCL hold area--the previously-selected (line 01) entry from the function table TABBCL--to become the data for Factor 1. Since this is a LOKUP operation, Factor 1 contains the search argument; i.e., the last-selected TABBCL function-table entry now becomes a search argument.

IBM

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 (3-5)	Form Type (6)	Record Sequence of the Chaining File (7-8)	Number of the Chaining Field (9-10)	From Filename (11-18)	To Filename (19-26)	Table Name (27-32)	Number of Table Entries Per Record (33-35)	Number of Table Entries Per Table (36-39)	Length of Table Entry (40-42)	Packed (P) (43)	Decimal Pos. (44)	Sequence (A/D) (45)	Table Name (Alternating Table) (46-51)	Length of Table Entry (52-54)	Packed (P) (55)	Decimal Pos. (56)	Sequence (A/D) (57)	Comments (58-74)
01	E					TABPCT	8	12	4		1	A	TABAMT	5		2		
02	E					TABBCL	12	12	2									
03	E					TAB1*2	3	12Ø	18			D	TABL	6		N	D	
04	E																	
05	E																	

Figure 48B. Table Look-Up--Table Definitions

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-3351-1
Printed in U. S. A.

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

Line (3-5)	Form Type (6)	Control Level (L0-L9, LR) (7-8)	Indicators (9-11)	And (12-14)	And (15-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field (43-48)	Field Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators Plus / Compare High 1>2 (54-55)	Minus / Low 1<2 (56-57)	Zero or Blank / Equal 1=2 (58-59)	Comments (60-74)
01	C					PRFPCT	LOKUP	TABPCT	TABBCL				2Ø		2Ø	
02	C		2Ø				MOVE	TABBCL	BONCLS	2						
03	C		2Ø			TABBCL	LOKUP	TABBCL	TABAMT						21	
04	C		2Ø			TABPCT	SUB	PERCNT	DIFFR	3	Ø	H				
05	C		2Ø			GRSPAY	ADD	TABAMT	FINGRS	6	2					
06	C		N2Ø			-1256.5Ø	LOKUP	TABL	TAB1*2					25		
07	C		N2Ø	25			MOVE	TAB1*2	PART2	1Ø						
08	C		N2Ø	25			MOVEL	TAB1*2	PART1	8						
09	C					CRITRN	LOKUP	TAB1*2					H1		H2	
10	C						RLABL		TABL							
11	C															

Figure 48C. Table Look-Up--Calculation Specifications

TABBCL is also specified in Factor 2, and therefore is the argument table in this operation. It is needed to ascertain the count of entries (n) to the point of match between the search argument and the pertinent argument-table entry. This permits the program to select the nth entry from the function table (TABAMT--say, amount of performance bonus), and to place it in the hold area for TABAMT.

The comparison is logical (rather than algebraic), because TABBCL is defined as alphameric.

An exact match is required between search argument and argument table; therefore, a Resulting Indicator (21) is assigned only to Equal. (An exact match is known to exist, because the search argument was originally derived from the argument table.) TABBCL is not defined (in the file extension specifications) as being in sequence, because this makes no difference in a search confined to an "equal" match: the program will search through the entire argument table until it finds an equal value (if it exists). Since alphameric comparison is logical, the identical EBCDIC character must be located (e.g., 5 ≠ +5, 0 ≠ 0^+ ≠ $\bar{0}$).

TABAMT is defined (in the file extension specifications) as numeric, each entry 5 columns long, including 2 decimal places.

The specifications in this line are executed only if the first LOKUP in this program cycle yielded a "hit"--otherwise a bonus class could still be in the TABBCL hold area from an earlier program cycle.

Line 04. The difference is calculated between the bonus percent selected in line 01 from the table TABPCT (which steps in increment groups) and placed in its hold area, and the precise performance percentage. The difference (≥0) is stored as field DIFFR, 3 columns long, rounded to no decimal places--the original values contained one decimal place. The operation is performed only if the original LOKUP selected pertinent data (indicator 20 on).

This illustrates that there may be occasions when the entry selected from the argument table in a LOKUP operation is of interest, because it differs from the search argument when a Resulting Indicator is specified for High or Low only, and because it may differ when High and Equal or Low and Equal are designated.

Line 05. The amount of bonus pay selected from table TABAMT in line 03, and placed in its hold area, is added to basic gross pay (GRSPAY) to provide final gross pay (FINGRS). TABAMT is 5 columns long, including 2 decimal places and, therefore, fits in FINGRS (6 columns long, including 2 decimal places).

The operation is only performed if the first LOKUP operation yielded a "hit" (indicator 20 on). Indicator 21 is not needed, because the second LOKUP must yield a "hit".

Line 06 illustrates a numeric literal as search argument. Table TABL--the argument table--must be defined with the same entry length (6) as the search argument and the same format (numeric)--see line 03 in file extension specifications. Table TABL is defined (in the file extension specifications) as descending.

Comparison is algebraic: the first-encountered value in TABL that is algebraically smaller than -125650, i.e., negative and larger in absolute value, satisfies the criterion. (Decimal point is ignored in LOKUP, and its position is not counted as a column in field length.) If the criterion is satisfied, indicator 25 turns on; the TABL entry that satisfied the condition is stored in the TABL hold area, and the corresponding (nth) entry from table TAB1#2 is stored in its hold area. If no argument-table entry meets the specified condition (Low), indicator 25 turns off, and the two hold areas are not disturbed.

TAB1#2 is defined as in descending sequence because of another operation (line 09)--this is irrelevant here.

The operations in this line are performed only if no successful match was achieved in the first LOKUP operation (line 01); i.e., if indicator 20 is off.

The name TAB1#2 illustrates that the characters after TAB may be numeric (1, 2) and/or alphabetic (# is an alphabetic character--see Definition of Terms). The other table names chosen all happen to be completely alphabetic.

A numeric literal is used here as a search argument. Alphameric literals may also be used; the argument table must then be defined as alphameric, and comparison is logical rather than algebraic.

Lines 07 and 08. These specifications were included to point out that a function table could include more than one function.

Assume that each entry in table TAB1#2 really contains adjacent data for two functions--the left-hand function-1 field being 8 columns long and the right-hand function-2 field being 10 columns long, together forming the 18-column entries defined (in the file extension specifications) for TAB1#2. The single LOKUP operation in line 06 then supplies both functions associated with the selected argument-table entry.

By MOVE and MOVEL operations to fields of appropriate sizes the dual-function data in the TAB1#2 hold area can be split, and made available separately.

The operations are conditioned on the LOKUP in line 06 having been performed (indicator 20 off) and data having been selected from table TAB1#2 (indicator 25 on).

Line 09 demonstrates the possibility of performing a LOKUP operation solely to ascertain the relationship of data in an argument table to the search argument, without use of the selected data and without selection of data from a function table.

The system is to halt (after completion of detail-time output), and/or calculation or output operations are to be modified, if TAB1#2 (the argument table) contains an entry logically equal to or higher in EBCDIC than the contents of the field CRITRN. Different Resulting Indicators (H1 and H2) were assigned to the two error conditions being checked, to show that different indicators may be assigned to High and Equal or Low and Equal. (That the same indicator may be assigned was shown in line 01.)

The same table (TAB1#2) is used as an argument table here and as a function table in line 06.

TAB1#2 is defined (in the file extension specifications) as alphameric, and each entry as 18 positions long. CRITRN must therefore be defined identically somewhere (in input specifications or elsewhere in the calculation specifications). TAB1#2 is also assigned a sequence (descending) so that a comparison can be made for High or Low.

Line 10 entries make data selected (in a LOKUP operation) from table TABL (as placed in its hold area) available to any external subroutines, by reference to the field name TABL. Note that the table name cannot be longer than 4 columns for this purpose. (For the format of a table name refer to File Extension Specifications, section Table Name--Cols. 27-32.)

Lines 01, 03, and 06 also show that argument and function tables need not have the same field lengths, formats, numbers of decimal places, or sequence.

File Extension Specifications (Figure 48B) and Card Formats (Figure 48A)

The tables TABPCT and TABAMT are contained in one set of cards, in alternating format. They must therefore be defined in one line of the file extension specifications. TABPCT appears first in each table-input card; therefore, it must be described in the left portion of the file-description line. The same applies to TAB1#2 and TABL.

The table TABBCL is alone in its set of table-input cards; therefore, no entry is made in the right portion of line 02 in the file extension specifications.

Note, in the calculation specifications, that the loading of two tables from one set of table-input cards has no bearing on whether a table is used as argument or function table, or which tables are related to each other as argument and function tables.

Note also that, when two tables are alternated in one card, one entry for each table is jointly considered a single entry for purposes of "Number of Table Entries per Record" (cols. 33-35). A comparison of the entries in cols. 33-35 (of the file extension specifications) with the card layout form will clarify this.

In line 03, the entry N in col. 56--to define TABL as numeric with no decimal places--could equally well have been 0.

GENERAL INFORMATION

The output-format specifications (see Figure 49) serve to specify the kinds of output files to be produced and the location of the specific data fields in output cards and reports, and to define the conditions under which the particular output is to take place.

The specifications for output can be divided into two categories:

1. File identification and control--cols. 7-31:
   - Identification of output files (output media)--File name;
   - Stacker selection of cards;
   - Forms control on the printer--Space, Skip;
   - Segment of the program cycle during which the output is to occur (detail time, total time, or overflow time);
   - Conditions under which the output file is to be created--Output Indicators.
2. Field description and control--cols. 23-70:
   - Identification of fields whose contents are to be output--Field Name;
   - Location of data fields on the report and/or in output cards--End Position in Output Record;
   - Format in which each output field is to be printed or punched--including Zero Suppress, Edit Word, Packed Field;
   - Definition of constants;
   - Clearing of data fields for subsequent operations--Blank-After;
   - Conditions under which a particular field is to be output--Output Indicators;
   - For output to cards: whether the data described in a particular specification line is to be punched or document-printed--End Position in Output Record.

The Sterling Sign Position (cols. 71-74) applies to Sterling currency fields (British monetary system) only. It is not covered in this manual. See IBM System/360 Model 20, Sterling Currency Processing Routines, Form C26-3605.

A File Identification occupies a separate specification line (or--if there are AND or OR lines--a group of lines) followed by all Field-Description lines for that file output--one line per output field or constant.

Note: When stacker selecting input-file cards, based on file matching and/or calculation results, the file must be defined as combined. Therefore, a File Identification entry is made in the Output-Format specifications, but it is not followed by a Field-Description entry.

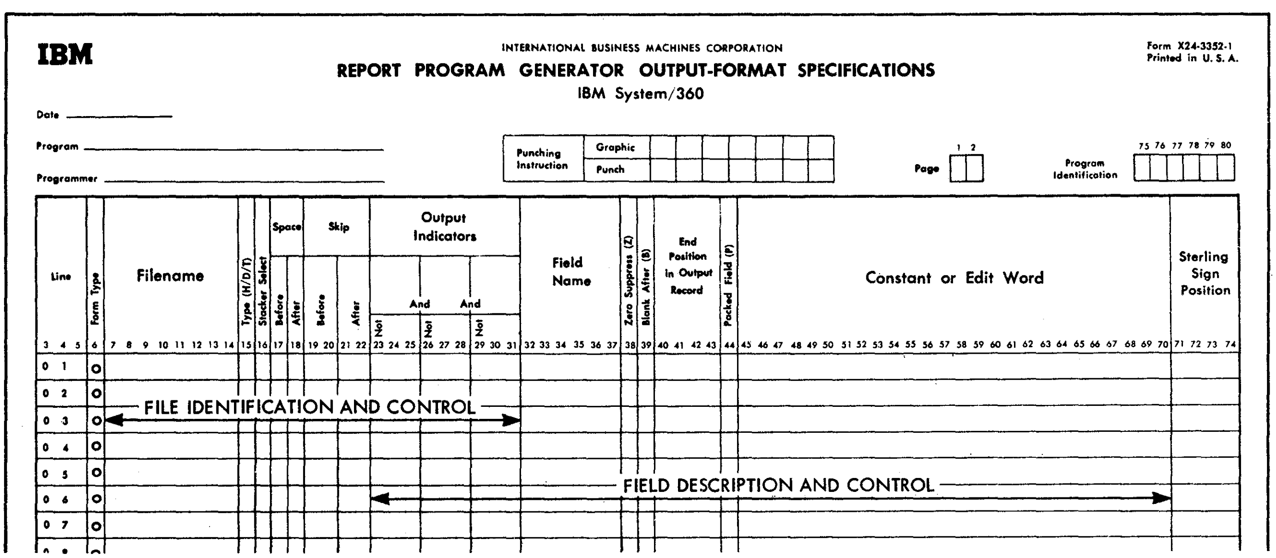

IBM
INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS
IBM System/360
Form X24-3352-1
Printed in U. S. A.
Date
Program
Programmer
Punching Instruction: Graphic, Punch
Page 1 2
Program Identification 75 76 77 78 79 80
Line | Form Type | Filename | Type (H/D/T) | Stacker Select | Space: Before, After | Skip: Before, After | Output Indicators: Not, And Not, And Not | Field Name | Zero Suppress (Z) | Blank After (B) | End Position in Output Record | Packed Field (P) | Constant or Edit Word | Sterling Sign Position
FILE IDENTIFICATION AND CONTROL
FIELD DESCRIPTION AND CONTROL

Figure 49. The Output-Format Specifications Form

-215

Output Indicators in a File-Identification line condition the occurrence of output for the entire file (i.e., the report or the card); in a Field-Description line, they only condition output of the particular field, and are relevant only when the file output is executed.

Only output and combined files are entered in the Output-Format Specifications. Files defined in the File Description Specifications as input files (I in col. 15) must not be entered in the Output-Format Specifications. The Printer is considered an output file--or, two output files, when utilizing the upper and lower feeds of the Dual-Feed Carriage special feature.

No check is made automatically by the program or the hardware that an output card is blank in the columns into which data is to be punched. Such a test can be programmed by designating the file a combined file, and assigning Field Indicators to the relevant fields, defined as alphameric, in the input specifications; if all fields to be punched are contiguous, they could be defined in the input specifications as one long single field, for purposes of testing them for "blank".

<u>Sequence of Specifications</u>

Specifications for all detail-time output must precede specifications for all total-time output.

Users accustomed to Unit Record applications should realize that card cycles and total cycles as such do not exist in RPG, nor does higher-level-total output necessarily occur later than lower-level: the program steps through total time and detail time in each complete program cycle, thus offering greater flexibility. Any RPG operation may be performed in either cycle segment--including the printing or punching of totals during detail-time output.

However, detail time and total time occur at different points in the cycle. The conditions reflected by various indicators may differ at these different points in the cycle, and the data available for output may represent different cards and/or different stages of calculation (depending on the user's program).

Detailed information is presented in the earlier section titled <u>Program Logic Flow</u>, and in Figure 6: <u>RPG Program Logic</u>.

Within the grouping of detail time and total time, the sequence of output operations corresponds to the sequence of the output-format specifications lines. There are three exceptions:

1. Overflow output occurs at overflow time, following total-time output.

2. The lower and upper feeds of the Dual-Feed Carriage special feature are considered two output files; yet, under certain conditions, output to both files is concurrent.

3. Data for card printing is transferred to the output data-storage area after the transfer of data for punching of the same card. This need concern the user only when Blank-After is specified for such a field.

These divergences will be further clarified later.

Each File-Identification line (or group of lines, when there are AND or OR lines), together with its subordinated Field-Description line(s), if any, represents one file output operation. (Punching and document-printing in the same card are parts of a single file-output operation.)

If there are several separate File-Identification (and groups of Field-Description) lines for the same output file at different points in the output-format specifications--and the status of any Output Indicators assigned calls for performance of several of these output-file specifications in one program cycle--the same output file is acted upon several times in the same program cycle. This has the following effect:

1. If the output file is the printer--printing occurs several times. If spacing and skipping between lines is suppressed, the successive printing is on the same line of the form.

   If spacing or skipping between lines is specified, the printing is on separate lines of the form. This is the normal method to accomplish, for example:

   a. Printing of group totals below detail item lines. (Usually, the detail items are printed at detail time and the totals at total time, but this need not be so.)

   b. Printing totals of different levels on different lines (for instance, the L2-level total under the L1 total).

   <u>Note</u>: A print line conditioned by L2 as Output Indicator is not printed later than (or under) a print line conditioned by L1, unless the File-Identification line conditioned by L2 appears later in the output-format specifications than the L1 File-Identification

line: within one cycle segment (detail time or total time), the sequence of output operations adheres to the specifications-line sequence.

There is no such event--as with Unit Record accounting machines--as major-total output automatically preceded by intermediate, preceded by minor. By proper assignment of Control-Level indicators (cols. 59-60, Input Specifications), indicator L3 can be made to represent the equivalent of a major control break, L2 an intermediate one, and L1 a minor one. However, the program does not recognize the difference between L-indicators of different levels, or between L-indicators and other indicators, as related to a particular class of total: any indicators may be assigned in Output Indicators (cols. 23-31) to condition the execution of an output-specifications line. Therefore, if L2 represents the equivalent of a total class higher than L1, and the L2 totals are to be printed underneath the L1 totals, the File-Identification and Field-Description Specifications conditioned by the L2 indicator must appear later than those conditioned by L1 (if both apply to the same cycle segment).

c. Printing of several lines from one input card; for instance, name and address on three lines from a single input card.

d. Printing forms-overflow identification. This occurs at overflow time. If the same output file (the printer) is also specified (say, for group indication) by another File-Identification line--which is the normal situation--care must be taken not to unintentionally print some data twice. (This situation is discussed further, below.)

2. If the output file consists of cards (output or combined file)--successive cards are punched, document-printed and/or stacker selected. (See Multiple-Time Output to Cards during One Program Cycle, under Program Logic Flow.)

Therefore, all output operations--punching, card-printing, stacker-selection--pertinent to one card must be included in a single File-Identification line (and its related Field-Description line(s), if any).

If multiple output is required during the same cycle segment to each of several files, faster throughput may be obtained, through maximization of overlap, by alternating the output specifications for the files. Assume for instance:

On a Level-2 (L2) control break, Level-1 totals are to be printed on one line on the printer, followed by Level-2 totals. It is also desired to summary-punch an output-file card with the Level-1 totals, and another with the Level-2 totals. These operations are all to be performed in the same cycle segment (total-time output, or detail-time output, as desired).

The File-Identification and Field-Description specifications lines for the L-1 print line must be written ahead of those for the L-2 print line, because output sequence in one cycle segment is determined by the specifications sequence. By interposing the specifications for one of the output operations to the card file between the two print-line outputs, throughput is usually enhanced.

E.g.: Specifications for output of L1 totals to printer; then
Specifications for output of L1 totals to card file; then
Specifications for output of L2 totals to printer; then
Specifications for output of L2 totals to card file.

<u>Note</u>: Even if no card punching is required at the L2-level, it is still advantageous to interpose the L1-level card-punch specifications between the L1- and L2-level printer specifications.

Figure 4 shows which operations can be carried out concurrently.

Specifying Output Units

Each file name is associated with a particular input, output, or input/output unit (or device) by the entries in the File Description Specifications. Designation of a file name in the output specifications therefore suffices to determine the file to be operated upon.

Organizing for Output-Format Specifications

Writing the output-format specifications becomes a simple task if the user has first analyzed his report and output-card requirements and laid out

1. The printed report (if relevant) on a Printer Spacing Chart (IBM Form X24-3115--see Figure 1), and

2. The format of any output-file cards on one of the many card layout forms available (e.g., see Figure 48 A).

FILE IDENTIFICATION AND CONTROL--COLS. 7-31

One File-Identification line (or group of lines, when there are AND or OR lines) is to be specified per output operation per output file. Each such File-Identification line (or group of lines) is followed by all Field-Description lines pertinent to that output operation.

Note: When stacker-selecting input-file cards, based on file matching and/or calculation results, the file must be defined as combined. Therefore, a File-Identification entry is made in the Output-Format specifications, but it is not followed by a Field Description entry.

File Name--Cols. 7-14

Each output file is given a separate name by the programmer--and the same name must be used for that file in the File Description Specifications, to associate the file name with a particular I/O device. The same name must not be assigned to more than one file. However, when a card file is used both for input and output, it is termed a "combined" file, and the same file name is entered both in the Input and the Output-Format Specifications. A file name must begin in col. 7 with one of the 29 alphabetic characters, and may continue with alphabetic or numeric characters (but not special characters or embedded blanks); it may be one to eight characters long. (Further details on files and file names appear under Input and Output Files, in Introduction--RPG Functions and Characteristics; Definition of Terms; and File Description Specifications.)

The file name must be recorded in this field (cols. 7-14) in the File-Identification line for an output (or combined) file the first time that file appears in the Output-Format Specifications. The same file may be specified several times--for repeated output to the same file in the same program cycle (see Sequence of Specifications, above). The file name need then not be repeated, unless specifications for another file intervene: if the file-name is blank in a File-Identification line, the program applies the nearest preceding file name--this is true even if the entries apply to different segments of the program cycle (Type D versus Type T in col. 15). No file name may appear in an AND or OR line.

Type--Col. 15

The mnemonic code letter entered here designates the type of output record being specified.
The following three entries can be made.

H = Heading-line output
D = Detail-time output
T = Total-time output

Note: Code H and code D are distinguished only for the user's convenience. The RPG program treats both specification types in the same way. No "Type" entry is made in an OR line; the same type code is assumed to apply.

All detail-time output (Type D) must be specified ahead of all total-time output (Type T).

Reference to RPG Program Logic (Figure 6) makes it apparent that detail-time output will most-commonly deal with data from and/or to individual detail cards--such as listing from detail cards, printing the results of detail calculations, or punching into detail cards; whereas total-time output lends itself best to printing and/or punching of totals at the end of control groups, when data from the next card is not yet available. However, the use of detail-time and total-time outputs is by no means thus restricted--comprehension of the RPG program cycle (with its attendant data flow, indicator relationships, and card movement) permits other arrangements.

Note: Although all File-Identification specifications must be designated Type D (or H) or Type T, output conditioned (in Output Indicators, cols. 23-31), in the File-Identification entries, by indicator code OF (or OV) occurs at overflow-output time--not at detail-output or total-output time. (This is discussed more fully under Output Indicators--OF and OV, below, and has already been described under Program Logic Flow, and in the explanation for Figure 5E.)

Stacker Select--Col. 16

Stacker Select applies only to card files. (If a Stacker-Select entry is made in the File-Identification specifications for a Printer output file, it is ignored.)

If no Stacker-Select assignment is made for a card file in either the input or the output-format specifications, the cards enter the normal stacker for the particular card punch or read-punch device. If the device contains more than a single stacker, cards may be program-directed to a non-normal stacker by designation of the desired stacker in the input or output--

format specifications--subject to certain rules listed below. Figure 24 (Input Specifications) itemizes the normal and additional stackers for card input/output units with multiple stackers, and the associated stacker-select codes. For single-stacker I/O devices, Stacker Select should be left blank (however, any entry is simply ignored by the program).

Note 1: When stacker 5 is designated, but the I/O device referred to is the 2560 MFCM Model A2, the card is directed to stacker 4.

Note 2: In the case of the IBM 2520 Card Punch or Read-Punch, cards with punch errors are automatically directed to stacker 2--the non-normal stacker--by the system.

Rules for Stacker Selection

Input-File cards can only be stacker-selected by an entry in the input specifications.

Stacker selection of input-file cards, based on file matching and/or calculation results, is possible. In this case, however, the file must be defined as combined and the file name entered in the Output-Format specifications.

Note: It is also possible to perform stacker selection on input-file cards, based on file matching and/or calculation results, by means of the EXIT operation code and BAL subroutines (see _Programming Tips_, Appendix E).

Output-file cards can only be stacker-selected by an entry in the output-format specifications. This is accomplished by entering the number of the desired stacker in col. 16 of the relevant File-Identification line. If cards are to enter the normal stacker for the I/O device that contains the file, col. 16 may either be left blank or coded with the number of the normal stacker (which is always 1, except for the secondary feed of the MFCM).

Combined-file cards can be stacker-selected by an entry in the input specifications, but only when selection can be based solely on card type. They can be stacker-selected by an entry in the output-format specifications to reflect any desired condition: card type, Matching-Record status, results of calculations, etc. If for a card file, or for certain card types in a file, stacker selection is the only operation desired in the output-format specifications, only the pertinent File-Identification specifications, including Stacker Select, are required. It is permissible to select some card types within a file via the input specifications, and others in the same file via the output specifications. The following criteria must be observed:

1. The same card type must not have stacker-select instructions in both the input and output-format specifications.

2. If any output operation (punching and/or card-printing) is to be performed on cards of a type, any stacker selection to be designated for that type must be in the output specifications.

   Therefore, card types for which stacker selection is designated in the input specifications must not have any output operations specified. When output File-Identification specifications are written for a file, any cardtype(s) within that file that had an input stacker-select specification must be eliminated from output operations by appropriate designation of Output Indicators (cols. 23-31)--otherwise, output is to the _next_ card and that next card is never read.

   For example, assume:

   a. File DETAIL contains three card types to which card-type Resulting Indicators 10, 11, and 12 were assigned in the input specifications; and

   b. Type 11 has a stacker-select instruction in the input specifications; then:
   Only types 10 and 12 may have output operations specified; the entry N11 in cols. 23-25 is the simplest way to accomplish this.

3. If stacker selection is to be based on the status of any indicator except card type (such as MR, or one reflecting the results of calculations), it must be designated in the output specifications.

   Stacker selection based on matching of files (matching records) requires that the file be defined as a combined file. (But see _Programming Tips_, Appendix E, for BAL subroutines to accomplish this with Input Files.)

4. If no entry at all appears in the output-format specifications for a combined-file card, and no stacker selection is specified for that card in the input specifications either, the card enters the pertinent normal stacker.

   Note: See also further Stacker-Select information for combined files under _Input Specifications_.

Stacker selection at total-output time (Type T in col. 15) causes selection of the "next" card. At this time:

a. Data from that new card has not yet been available for calculation, and

b. The Matching-Record indicator still reflects the status of the previous card, and

c. Field Indicators still represent the previous card; but

d. The card-type Resulting Indicator for the new card is on--that for the old card is off, and

e. If the old card was the last of a control group, the pertinent L-indicators are already on.

(See RPG Program Logic, Figure 6.)

A card's position as the last of a control group can not be recognized by RPG as a criterion in time for stacker selection of that card. (The PLACE card in the Punched-Card Utility Collate Program provides for this; or, the RPG program may branch, by operation code EXIT, to a B.A.L. subroutine to accomplish this selection--see Programming Tips, Appendix E.)

Stacker Selection of matched or unmatched cards in a file-matching application, based on the status of the MR indicator (see Output Indicators, below), should be specified for detail-time output (Type D in col. 15). The MR indicator then correctly reflects the match status of the card that would be selected. At total time for a new card, the MR indicator reflects the match status of the preceding card.

Stacker selection for OR lines (see Output Indicators, below) is independent of that for the basic File-Identification specifications line. It behaves like stacker selection for any other line: if col. 16 is blank, the cards defined in the OR line enter the normal stacker; if a stacker number is specified, the cards enter that stacker. (But see item 3 under Stacker Select--OR Lines, in Input Specifications.)

Space, Skip--Cols. 17 and 18; Cols. 19-20 and 21-22

These fields are left blank in File-Identification specifications for card files.

The Space and Skip fields provide for printer forms-movement control. They apply each time the particular printer-output File-Identification specifications are executed, even when:

1. The particular printer output specifications are repeated during the same program cycle--by a GOTO operation (see Calculation Specifications); or

2. No data is actually printed, because the status of the particular Output Indicators assigned in the individual Field-Description specifications lines (see below) prevent printing of all associated fields or constants.

If the printer is the IBM 2203, and the Dual-Feed Carriage special feature is installed, forms control applies only to the forms carriage with which the particular File Name is associated (through the Device Code in the file description specifications).

Separate specifications may be given for OR lines. However, if an OR line is blank in all of these forms-control fields, the space and/or skip specifications from the nearest preceding File-Identification line (within the same file output entry) with such specifications are applied by the program also to that OR line. If there are also no specifications in these fields in any of the preceding File-Identification lines (main or OR lines) of that file-output entry, the field is considered to be blank in the OR line too.

Note that a zero (in contrast to blank) in any of the columns 17-22 in an OR line prevents application to the OR line of Space or Skip specifications from a preceding file-identification line. If at least one of the cols. 17-22 in an OR line contains a zero, and the remainder are zero or blank, no spacing or skipping takes place, before or after, when output is based on the OR line.

There must be no Space or Skip entries in an AND line.

Note: Relationships between Space and Skip specifications are discussed at the end of this section.

Space, Before--Col. 17

The paper form in the printer is advanced 0, 1, 2, or 3 lines before printing by entering 0, 1, 2, or 3, respectively, in col. 17 of the pertinent File-Identification specifications.

A blank in col. 17 has the same effect as entering a 0; i.e., no space before printing.

Space, After--Col. 18

Equivalent to Col. 17, but controls line spacing after printing.

Skip, Before--Cols. 19-20

Any number from 01-12 may be entered to cause the paper form in the printer to be advanced, before the line is printed, until a punch is sensed by the tape-reading brushes in the corresponding channel of the synchronized forms-carriage control tape. If a tape punch in that channel is already lined up with the brushes, the form will nevertheless advance, until a punch in that carriage-tape channel reaches the tape-reading brushes again.

A leading 0 need not be recorded with this RPG; i.e., ƀ1 = 01.
00 is treated as equal to ƀƀ

Skip, After--Cols. 21-22

Equivalent to cols. 19-20, but controls forms skipping after printing.

Points to Note

1. If the user's application offers a choice

   a. Between Space/Before and Space/After, Space/After should be employed; or

   b. Between Skip/Before and Skip/After, Skip/After should be employed.

   Forms movement after printing of a line usually gives better throughput: it permits overlap of subsequent processing with the forms movement; whereas, if forms movement takes place ahead of printing of a line, execution of the print instruction has to await completion of forms movement.

2. Line spacing may be stipulated for both before and after printing of a line. Thus, a maximum of 6 line spaces (i.e., 5 intervening blank lines) may be achieved between successive print lines without skipping.

3. Forms skipping may be stipulated for both before and after printing of a line.

4. If Space/Before and Skip/Before are both stipulated for the same print line: the Skip is executed first, followed by the Space operation.

5. If Space/After and Skip/After are both stipulated for the same print line: only the Space operation is executed.

6. A forms advance to the next carriage-tape channel-1 punch is automatic:

   a. At the conclusion of program generation--unless the RPG Control Card (card H) contains a B in Col. 11, which suppresses program listing during generation (see the <u>Operating Procedures</u> manual).

   b. After total-output time if, at any time in that program cycle (i.e., during detail-time output or during total-time output), a line was printed at or below the point at which a carriage-tape channel-12 punch was sensed by the forms-carriage brushes--and provided OF (or OV) is not assigned in Output Indicators of any File-Identification specifications for that file.

   This implies that, if (by virtue of the user's RPG specifications) more than one line may be printed in a single program cycle without a specified skip to a new page, the distance of the channel-1 carriage-tape punch from the channel-12 punch must be long enough to allow all lines in a program cycle to be printed without exceeding the maximum desired print lines on a page.

   <u>Note</u>: If OF appears in Output Indicators of <u>any</u> <u>File</u>-Identification line for the standard (or lower) printer-carriage (i.e., file PRINTER or PRINTLF), no automatic overflow forms skip to the channel-1 punch ever occurs for that file: overflow forms skipping must then be specified in an overflow-time File-Identification line--see below. The equivalent applies to OV with the upper carriage--file PRINTUF. (These statements do not apply if only NOF or NOV appears in Output Indicators for the respective file, nor if OF or OV appears only in <u>Field</u>-Description Specification lines.)

7. Successive printer outputs can be printed on the same line of the printer form (i.e., without intervening forms movement) by appropriate space and skip specifications of blank or zero, to effect "space suppression": if cols. 17-22 are blank or zeros, no spacing or skipping occurs before or after the line is printed (but see distinction between blank and zero for OR lines, above).

   Note however that, if the multiple outputs--intended for a single line on the printer form--occur during different program cycles, they may become separated to different pages: the automatic forms advance operates as described in item 6 (b) above, even though all space and skip specification fields for these outputs may be zero or blank.

If the multiple outputs to one printer line are in the same program cycle, no automatic forms advance can separate the lines (since the automatic advance to the channel-1 punch takes place only after total-output time).

8. A Skip (by a specification in cols. 19-20 or 21-22) past a carriage-tape channel-12 punch--i.e., from a point on the page higher than the channel-12 punch--has the following effects:

   a. If the skip is to or past a channel-1 punch: The overflow indicator (OF or OV) is not turned on.

   b. If the skip is to a carriage-tape punch in any channel other than channel 1, and a channel-1 punch is not passed or reached during this skip: The overflow indicator (OF or OV) is turned on, after the line at or past channel 12 has been printed.

9. Once a line has been printed at or below the point at which a carriage-tape channel-12 punch was sensed, an internal switch is set which will cause the overflow indicator (OF or OV) to turn on _at the end of_ (not during) that cycle-segment output time. It cannot be turned off by a skip-to-channel-1 specification (contrary to the situation described in 8 (a) above). Therefore:

   a. If OF (or OV--see Dual-Feed Carriage) is not specified in Output Indicators of any File-Identification specifications line for that file, an automatic skip to channel 1 (as stated in 6 (b), above) will then occur after total-output time of that program cycle--even if a skip to channel 1 was specified (and executed) in a File-Identification specifications line whose output followed the detection of the channel-12 punch, in the same program cycle.

   b. If OF (or OV) is specified in Output Indicators of any File Identification specifications line for the respective file, the program will execute overflow File-Identification specifications for that file at overflow-output time following total-output time--even if a skip to channel 1 was specified (and executed) in a File-Identification specifications line whose output followed the detection of the channel-12 punch, in the same program cycle.

10. If it is desired to cause the overflow indicator (OF or OV) to turn on (at the end of the program cycle-segment), so that overflow output can be performed after total-time output--but it is necessary to skip to channel 1 from a point on the page higher than the channel-12 punch--two Skip specifications are required: first skip to channel 12, then to channel 1. As explained in item 9, above, the skip to channel 1 will not turn off the overflow indicator, once it has been signaled to turn on by sensing of the channel-12 punch.

Note 1: With the IBM 1403 Model 2 or N1 Printer, successive punches in the same channel of the carriage-control tape must be at least 8 lines apart, because of the high-speed skip capability of the dual-speed carriage on these models (see the publication _IBM 1403 Printer_, Form A24-3073). By making use of both Space/Before and Space/After, up to six spaces (five blank lines) can be obtained between two successive print operations.

If it is nevertheless desired to utilize Skip for distances of less than eight lines between consecutive tape-channel punches, punches may be placed in the same positions in two different tape channels. The Skip instruction must then be alternated between the two channels. Note, however, that a skip to a channel-1 punch can have implications that differ from skipping to punches in other channels (see _Output Indicators--CF and OV_, below). Channel 1 should therefore be avoided, in some situations, as one of the alternating channels.

Note 2: For compatability with other RPGs, there should be an entry in at least one of the Space or Skip fields of the first File-Identification specifications line for each printer output (i.e., it is not necessary in an OR line). If the user requires no entry (no Space or Skip desired), he should enter a zero in Space/After (col. 18).

Output Indicators--Cols. 23-31
(File-Identification Specifications)

Indicator codes entered in these fields determine the conditions under which the output operations defined in this File-Identification specifications line, and in its subsidiary Field-Description specifications lines, are to be executed.

Note that Output Indicators in the File-Identification specifications line control the occurrence of that entire output--not of a particular field. (Output Indicators may also be assigned to individual fields. This is discussed under Field Description and Control, below.)

Absence of an entry calls for execution of that output at detail time or total time--D (or H) or T, respectively, in col. 15 (Type)--each program cycle. Note that detail time includes detail-output time before the first card has been read (when the 1P indicator is on).

Any indicator--except OF or OV (discussed separately below)--may be entered in cols. 24-25, 27-28, or 30-31 to instruct the program to execute the particular output specifications only if that indicator is on at that time (detail time or total time, as determined by the entry in col. 15). If an N (=Not on) is entered in the column preceding the indicator code (col. 23, 26, or 29, respectively), the output is performed only if that indicator is not on.

Note: Any EBCDIC character other than N in col. 23, 26, or 29 has the same meaning as a blank.

The three fields (cols. 23-25, 26-28, 29-31) are identical in function. If less than three conditioning indicators are assigned, it does not matter which of the three fields are used. Up to three different conditioning indicators may be designated in one File-Identification specifications line. All indicators assigned to one line are in an AND relationship to each other; i.e., the conditions for all indicators in the line must be satisfied for the output to be performed. Each of the several indicators may individually be required to be on or not on (N) as a condition of performance of the output operation.

If more than three Output Indicators in an AND relationship are needed to condition an output operation, additional lines may be used, each able to accommodate up to three more indicator entries. Such lines must be immediately below the initial File-Identification specifications line for the particular output. They must be blank except for the word AND required in cols. 14-16 and the desired entries in Output Indicators (cols. 23-31).

Different Output Indicators may be placed in an OR relationship to each other; i.e., the output operation is to be performed if any one of several indicator criteria is satisfied. A separate File-Identification specifications line is used for each OR line, and placed immediately beneath the initial File-Identification line (or any AND or OR lines) for the particular output. The word OR is entered in cols. 14-15, the desired indicators in Output Indicators, and--optionally--Stacker Select and forms control instructions in cols. 16-22.

Both AND or OR relationships may be specified in conjunction with each other--i.e., the output operation is to be performed if any one of several combinations of indicator conditions is satisfied.

When there are AND or OR lines for an output operation, then every AND line, every OR line, and the initial File-Identification specifications line for that output operation must each have at least one entry in Output Indicators. If the indicators for successive lines in an OR relationship are not completely mutually exclusive, the program executes the specifications (Stacker Select and forms control, if any) of the first line whose indicator criteria are satisfied (except if one of the Output Indicators specifications is OF or OV, for the lower or upper feed respectively--see below).

Entries in Output Indicators utilize indicators only to condition the execution of an operation--they do not set them as Resulting or Field Indicators. Therefore, the use of an indicator in Output Indicators never changes its status (on or not on). An Indicator applied in Output Indicators reflects the status (on or not on) it previously assumed:

1. As card-type Resulting Indicator, or
2. As Field Indicator, or
3. As calculation Resulting Indicator, or
4. As Control Level indicator, or
5. As Matching Fields indicator (MR), or
6. Through a SETON or SETOF instruction, or
7. As its initial status at the beginning of program execution--if never changed, or
8. As a result of a Blank-After output instruction, or
9. As a consequence of forms-control carriage-brush sensing of a carriage-tape channel-12 punch--if OF or OV indicator.

The status of an indicator may have a different significance at detail time and at total time--for example:

(a) It may reflect different cards. For instance:
At total time, the MR indicator and Field Indicators reflect the previous input card whereas L-indicators and card-type Resulting Indicators already reflect the new input card; or

(b) Its use may have a different effect. For instance, with L-indicators employed in the normal manner, with Control Levels:
An L-indicator in Output Indicators of a total-time output operation (T in col. 15), makes printing or punching at total time contingent on occurrence of a control break of that or higher level--the standard method for programming output of group totals or of specifying a group-printed report; but

An L-indicator in Output Indicators of a detail-time output operation (D or H in col. 15), makes printing or punching at detail time contingent on a preceding control break of that or higher level--a method for programming group-indication (printing identifying data only from the first card of a control group).

For details on indicators, see also Program Logic Flow, RPG Program Logic, (Figure 6), Indicators, Indicator Hierarchy, and Matching of Files, all under Programming for RPG--General Information; Resulting Indicators, Field Indicators, Control Level and Matching Fields, all under Input Specifications; and Indicators and Result-Testing Fields, under Calculation Specifications.

Points to Note

1. The output operation called for by the File-Identification specifications occurs in a program cycle--at detail time if D or H is specified in col. 15 (Type); at total time if T is specified in col. 15--if either of the following situations in Output Indicators (cols. 23-31) applies:
   (a) The Output-Indicators fields are blank; or
   (b) Any indicators specified, and not preceded by N, are then on; and any indicators specified, and preceded by N, are then off.

   These criteria are valid also at the detail-output time that precedes the reading of the first input card (the uppermost I/O block in Figure 6, RPG Program Logic).
   If the output is to be suppressed at detail-output time preceding the reading of the first input card--and it should normally be suppressed at that time, except for the printing of constant data as report or column headings--an indicator must be assigned in Output Indicators. This may either be the code of an indicator known to be off before the first card has been read (such as a card-type Resulting Indicator); or it can be an indicator known to be on at that time, but the entry is preceded by N, so that the output is performed only when that indicator is not on (for instance, N1P).
   All indicators are off before the first input card has been read, with the exception of the following which are on at the start of object-program execution (see also Indicator Hierarchy and Figure 11):
   1P and L0; and any indicator assigned to "Zero or Blank" in input Field Indicators or in calculation Resulting Indicators (arithmetic operations or TESTZ).
   Permitting any output operation--apart from the printing of constant report-heading data (see Constants, below)--before the first input card has been read, may produce spurious effects: such as a line of zeros printed, a card punched or printed with zeros, or a combined-file card never read, etc. (See also Output Before First Card is Read, under Program Logic Flow.)

2. Total time is always bypassed in the first program cycle--and in the first n program cycles under certain circumstances. (For a full explanation, see Total-Time Processing on "Run-In", under Program Logic Flow.)
   Bypassing of total time does not, however, prevent the proper setting of L-indicators to reflect group-control breaks. Thus, even though total-time output is bypassed, L-indicators specified in Output Indicators to control group-indication (i.e., printing of identifying data from the first card of a control group) at detail time will operate properly also for the first data card of the deck (but see 3, below).

3. The L-level control fields in the first data card (of a type for which control fields are specified) are compared against zeros (EBCDIC F0) in core storage. Zeros (and, for numeric fields, also blanks) in control fields of such a card result in an "equal" comparison and therefore do not turn on the relevant L-indicators. This may present a problem in group-indication

for the first control group when the control fields of the first group contain no significant data. (See Special Considerations for Indicators L1-L9 on "Run-In", under Indicators, in the section Programming for RPG--General Information.)

A technique for circumventing any such problem is presented in Programming Tips, Appendix E.

Figures 50 A and B illustrate specifications for File Identification and Control, temporarily excluding the OF and OV indicators (discussed next). The reader is asked to assume, for purposes of this illustration, that each File-Identification specifications line (or group of lines, when there are AND or OR lines) is followed by at least one Field-Description specifications line (discussed later).

Programmer ______________

Line (3 4 5)	Form Type (6)	Filename (7–14)	Type (H/D/T) (15)	Stacker Select (16)	Space Before (17)	Space After (18)	Skip Before (19 20)	Skip After (21 22)	Output Indicators Not (23 24 25)	And Not (26 27 28)	And Not (29 30 31)
0 1	O	REPORT*1	H			3	Ø1		1P		
0 2	O										
0 3	O		D			1			N1P	NH1	
0 4	O										
0 5	O		T		1	2			L1		
0 6	O										
0 7	O	SUMCARD	T						L1		
0 8	O										
0 9	O	REPORT*1	T					Ø1	L2		
1 0	O										
1 1	O		T				1Ø	Ø1	LR		
1 2	O										

Figure 50A. Simple Examples of Entries for File Identification and Control (Excluding OF and OV Indicators)

Explanation of Entries in Figure 50A

Assumptions: A straight listing is desired, with two classes of total and grand total. Headings of constant data are to be printed on one line across the top of the report on the first page. At each Level-1 control break, a summary card is to be punched. The printer has been assigned (in the file description specifications) the file name REPORT#1, and the card punch device has the file name SUMCARD.

Specification line 01 causes printing only at detail time and only before the first data card has been read:

REPORT#1 is associated with the printer; H in col. 15 (Type) specifies detail time (a D, instead of H, would have been synonymous);

The 1P indicator is on at the beginning of program execution, and is turned off by the RPG program itself immediately after the first card has been read.

Thus, the output is limited to printing at detail time, before the first card has been read; i.e., the first detail-output time only. The Field-Description specifications following this File-Identification line are assumed to contain constants, to be printed as headings across the first print line of the first page.

If 1P were not specified, but Output Indicators left blank, the heading constants would be printed at detail-output time of every program cycle.

Skip/Before to the next carriage-tape channel-01 punch is specified, to make sure to start at the top of a fresh page. After the heading, the form is advanced 3 lines, so that two blank lines intervene before the first detail-data line.

Line 03 calls for printing the data in the fields presumed to be designated in Field-Description specifications beneath this line. The file name (REPORT#1, cols. 7-14) need not be repeated, because no other file name intervened--but it may be repeated.

This output is at detail time (D in col. 15--H could have been used instead);

The output is suppressed if the H1 and/or the 1P indicator is on. NH1 was arbitrarily chosen to point out that H1 might be assigned to an error condition, to halt the system after detail-output time, and the same indicator can conveniently be utilized to suppress output. If N1P were not assigned, the output would also occur before the first card has been read.

The 1 in col. 18 causes single spacing after each detail-output line.

Line 05 causes printing at total time (T in col. 15), provided the L1 indicator is on. This is the normal method for printing totals at the end of a control group of Level 1. The file name need not be repeated, even though this specification is for total-time output and the previous one was for detail time.

To offset the total line from a group of preceding detail lines, a Space/Before is specified. This creates one blank line between the last detail line (where 1 Space/After was specified) and the L1-total line. After the L1-total line, the form is advanced 2 lines, to leave a blank line before the next detail line.

Line 07 directs the program to punch a card (SUMCARD is associated with a card output or combined file) at total time, if the L1 indicator is on--a standard method of punching group totals into summary cards.

The SUMCARD output specifications could have followed the L2- or LR-level printer output; but they were deliberately interposed between the L1- and L2-level printer output specifications: alternating output media for the same program cycle tends to speed throughput. Therefore, the higher the proportion of L2 control breaks in relation to L1-only breaks, the more is gained by interposing the card-punch output between the two printer operations (see Sequence of Specifications, above).

Line 09 causes printing at total time, provided indicator L2 is on. This is the normal method for printing totals at the end of a control group of Level 2.

Because this output File-Identification specifications line is below the L1 line, the L2 total will be printed below the L1 total. If the specifications in line 09 were to precede those in line 05, then--at every control break of Level 2--the L2 totals would be printed ahead of the L1 totals: within the same program cycle segment (detail or total), the operations for the same output device are executed in the order in which the specifications appear in the output-format specifications (see Sequence of Specifications, above).

After every L2-level printer output, the forms-control-carriage tape advances to the next channel-1 punch, i.e., the top of a new page.

The file name REPORT#1 must be recorded because output to another file (SUMCARD) intervened.

Line 11 is equivalent to lines 05 and 09, but is operative only when the Last Record indicator is on. Final totals are printed on a predetermined line (Skip/Before to carriage-tape channel 10), and a Skip/After to the top of a new page takes place after printing. The file name is the same as for the preceding output and therefore need not be repeated.

Throughout, note that (1) all detail-output specifications must precede all total-output specifications, and (2) Space and Skip are forms-control specifications and can, of course, only be entered in File-Identification lines for printer output.

Programmer

Line (3-5)	Form Type (6)	Filename (7-14)	Type (H/D/T) (15)	Stacker Select (16)	Space Before (17)	Space After (18)	Skip Before (19-20)	Skip After (21-22)	Output Indicators Not (23-25)	And Not (26-28)	And Not (29-31)
01	O	INVNTRY	D				Ø1	Ø2	1P		
02	O										
03	O	OLDBALCE	D						MR	Ø5	
04	O	O	R	2					NMR	Ø5	
05	O										
06	O	TRSACTNS	D	3					MR	21	4Ø
07	O	A	N	D					62		
08	O	O	R	3					MR	21	N4Ø
09	O	A	N	D					N62	14	
10	O										
11	O		D						NMR	21	
12	O										
13	O	TRSACTNS	D	2					Ø1		
14	O										
15	O	INVNTRY	D			2			Ø1		
16	O										
17	O		T				Ø1	Ø1	LR		
	O										
	O										

Figure 50B. Further Examples of Entries for File Identification and Control (Excluding OF and OV Indicators)

Explanation of Entries in Figure 50B

Assumptions; The MFCM and a printer are used to produce an inventory status report; to update the inventory item-master card file; and to punch unit and extended prices into item-order detail cards, on the basis of quantity in the detail cards and unit price in the master inventory cards. Behind the item-order-card group for each stock number represented is a blank trailer card, which is to become the updated inventory-item master card. Inventory-item masters are in hopper 1, transaction (item-order and blank) cards in hopper 2.

The old inventory master-card file is named OLDBALCE; the file with the item-order detail and blank cards is named TRSACTNS. Both files are defined (in the File Description specifications) as combined files: the TRSACTNS file because its cards are to be read, punched, and stacker-selected at output time; the OLDBALCE file only because, besides being read, some of its cards are to be stacker-selected on the basis of the MR indicator--which must be in the output specifications. The Files are matched (Matching Fields--M1) on stock number.

The file INVNTRY is associated with the printer. The File Description specifications call for turning the LR indicator on when both card files are exhausted.

Specification line 01 causes printing only at detail time (D in col. 15--H would have had the identical effect), and only before the first data card has been read (1P in Output Indicators). (Heading data, in the form of constants, is presumed to be contained in the Field-Description specifications.) Before the line is printed, the form skips to the top of a new page (01 in cols. 19-20); and after printing, the form advances to the next carriage-tape channel-2 punch (02 in cols. 21-22).

Lines 03 and 04 cause old inventory master cards to enter the normal stacker (stacker 1) for the primary hopper of the MFCM (blank in col. 16--a 1 could equally well have been specified) whenever there is at least one matching detail (item-order) card--MR indicator on--but to enter stacker 2 (2 in col. 16) when there is no matching detail--NMR. In line 13, new inventory masters are also selected to stacker 2.

Lines 03, 04 and 13 jointly have the effect of selecting out (to stacker 1) old inventory masters (line 03) that are being replaced by updated new ones (line 13); but directing to stacker 2 those old inventory masters for which no new ones are being created (line 04). At the conclusion of the job, stacker 2 contains the updated complete inventory master-card file: newly-punched updated cards to reflect transactions, plus old masters for items on which there were no transactions.

Besides MR and NMR, respectively, the card-type Resulting Indicator (05) assumed to have been assigned to the OLDBALCE cards in the input specifications is also specified here--otherwise, in every program cycle in which a TRSACTNS card is processed, the next OLDBALCE card would also be fed through, but never read; and NMR alone would allow an OLDBALCE card to be fed through at the beginning, without being read.

Lines 06-09 illustrate AND and OR specifications. The operations are performed if either of these combinations of conditions exists:

Indicators MR and 21 and 40 and 62 are all on; or
Indicators MR and 21 and 14 are all on, and indicators 40 and 62 are both off.

These are presumed to be two types of item-order detail cards, to be processed alike. Both types are selected to stacker 3 (by entry of different stacker numbers in lines 06 and 08, the two types could be directed to different stackers).

Line 11. All item-order detail cards (say, card-type Resulting Indicator 21) should have a matching inventory master card (OLDBALCE file). If there is no master (indicator condition NMR), either a master card is missing or the detail card is punched with a wrong stock number. The detail card is directed to the normal stacker (col. 16 blank) for the MFCM secondary hopper, to be investigated.

A second indicator specification (besides NMR) is required (card type 21 was used) to prevent performance of this output before the first data card has been read and each time an OLDBALCE card is processed, and to distinguish this card from the blank card (see line 13) at the end of each stock-number detail-card group.

The file name (cols. 7-14) need not be repeated, because no other file name intervened.

Line 13 specifies the output for the blank card at the end of each stock-number detail-card group. This card will be punched with the updated inventory information, and becomes the new inventory master for the particular stock item.

Resulting Indicator 01 was assigned to this card type in the input specifications. The cards are selected to stacker 2 to form--in conjunction with old master cards for which there were no transactions (see line 04)--an updated complete inventory master deck. The file name (TRSACTNS) was repeated just to show that this is permissible--it is not necessary.

Output to this card could be performed at total time (T in col. 15). However, although totals for a preceding group of cards are to be punched, detail time (D in col. 15) was chosen, to illustrate that there is no fundamental difference between the operations that can be performed in these two segments of the program cycle--provided the appropriate data and indicator settings are available: this card type (the blank card), although part of a combined file, serves only for output; no data is read from it; the data is ready for "summary" punching when the preceding card has been processed, and the status of the MR indicator is not relevant. Therefore, this card can be punched at total or detail time. If it is desired to perform output to the blank trailer card only if the detail cards matched the OLDBALCE cards, MR should be specified (in addition to 01) in Output Indicators; the output should be performed at total time (T in col. 15), when the MR indicator still reflects the matching status of the previous card. (Because all total-time specifications must follow all detail time specifications, the specifications now in line 13 would have to be moved beyond line 15.)

Line 15 provides for printing the updated inventory information after the last transaction card of each stock-number group. This is the program cycle during which the blank card (indicator 01) at the end of each group is being processed; therefore, indicator 01 is specified in Output Indicators. Again, the printed output is performed at detail time (D in col. 15); but it could equally well be performed at the preceding or following total time. Either way, it illustrates a group-printed report, since the individual transaction cards are not printed.

The form is spaced 2 lines after each printing.

Line 17 provides for the printing of grand totals; the output is performed only when the LR indicator is on. This operation must be performed at total time, because--when the LR indicator is on--the job is terminated after total output.

The grand totals are printed at the top of a new page (01 in cols. 19-20), and the form is again advanced to the top of a new page after printing (01 in cols. 21-22).

Note that:

1. All detail-output specifications must precede all total-output specifications.

2. Card output operations contingent upon the status of the MR indicator (applied in the normal manner, to the matching of files) at detail time reflect the matching status of the card being processed; at total time, MR still reflects the matching status of the preceding card: this can be utilized for output to a card based on the matching status of the preceding card.

3. In this example, Control Level was not utilized: a blank (trailer) card was assumed to have been merged previously behind each stock-number group of transaction (item-order) cards. The program cycle for the trailer card is used to perform the group-end operations.
   This re-emphasizes that Control Level (L-indicators) and matching Fields (M1, M2, M3, and the associated MR indicator) have no inherent connection with each other--applications involving matching-records groups do not necessarily require Control Levels.

In both Figures 50A and 50B, forms advance to the next channel-1 punch is automatic after total-output time whenever a line has previously been printed at or below the channel-12 punch--because OF (or OV) is not designated in Output Indicators of a File Identification line.

Overflow Indicators--OF, OV

The overflow indicators are related to printer forms movement. Overflow indicator OF is associated with the standard forms-control carriage, and with the lower feed of the Dual-Feed Carriage special feature (see below) available for the IBM 2203 Printer. OV is the overflow indicator associated with the upper feed of the dual-feed carriage.

The principal functions of the overflow indicators are (for their respective carriages):

1. To provide for the control of output operations--among them such as forms advance to a new page, and page and column headings after the bottom of a page has been reached.

2. To condition the execution of calculation specifications on the basis of whether the bottom of a page was reached (by entry of OF, OV, NOF, or NOV in Indicators, cols. 9-17, of the calculation specifications).

The relevant overflow indicator turns on at the conclusion of a program-cycle segment--i.e., after completion of all detail-time output, or all total-time output--if, during that program-cycle segment, either of the following situations cccured in that file.

1. A line was printed at or below the point of a carriage-tape channel-12 punch during detail- or total-output time--i.e., after a punch in channel 12 was encountered (sensed) by the carriage-tape stop brushes; or
2. A line was printed after a programmed forms-skip was executed (during detail- or total-output time) past a carriage-tape channel-12 punch, to a punch in a channel other than channel 1 and without passing a channel-1 punch. (A forms skip past channel 12 to or past channel 1, before the overflow indicator was turned on as explained in 1 and 2 above, does not turn it on.)

When either of these two conditions occurs, it stores a signal to turn on the overflow indicator at the end of that output time and, if this is detail-output time, then also after the next total-output time.
(No new overflow signal is created if channel 12 is passed during overflow-output time.)

Once the signal to turn on an overflow indicator is stored (as a result of 1 or 2 above), the signal and the overflow indicator are not turned off again by the program until completion of the next detail-output time (unless they then remain on because an overflow condition occurred again during that detail-output time). Once the condition for turning the overflow indicator on has been met, even a Skip to channel 1 will not turn it off: thus, overflow-output-time information can be printed before and/or after a skip to any channel.

Four points inherent in the above statements should be emphasized:

1. Regardless of whether the overflow condition occurred during detail-time or total-time output of a program cycle, the indicator does not turn off again until after the next detail-time output.
   Therefore, if the status of the overflow indicator is to be used to control the performance of calculations--and, by the nature of the application, the overflow point could be reached during either detail or total output--the calculations conditioned by the overflow indicator should be specified for detail time to obtain consistent results: if the overflow point was reached during either detail or total time, the overflow indicator will be on during the next detail-time calculations; however, during total-time calculations, it is on only if the overflow condition occurred during the preceding detail-time output.
2. The overflow indicator does not turn on during output time as soon as a channel-12 punch is sensed: it turns on after all output operations for one cycle time-segment (detail-time output or total-time output) have been completed, if an overflow condition occurred at any time during that output time (i.e., at least one line was printed at or beyond channel 12).
   Therefore, printed output cannot be conditioned based on occurrence of an overflow condition during printing of a previous line in the <u>same program-cyle segment</u>.
3. Although the overflow indicator does not turn on until completion of output for the program-cycle segment during which overflow was signalled, once either condition (1 or 2 in previous paragraph) that determines overflow has occurred, the indicator will turn on--even if a Skip instruction to channel 1 follows the overflow signal within the same cycle segment.
4. Skipping past a channel-12 punch to a punch in channels 2-11, without passing a channel-1 punch, creates an overflow condition; but skipping past channel 12 to or past channel 1 does not. This has certain implications:
   (a) If it is desired to skip to channel 1 from a point above a channel-12 punch without turning on the overflow indicator, no problem exists.
   (b) If it desired to skip from a point above a channel-12 punch, past a channel-12 punch, to a punch in any of the channels 2-11--without passing a channel-1 punch--and the overflow indicator is to turn on, no problem exists.
   (c) If it is desired to skip past a channel-12 punch to a channel-1 punch, but an overflow condition is to be created, skipping must be specified twice: first to channel 12, and then to channel 1.

If an overflow indicator is on, because of an overflow condition during detail- or total-time output, then--after conclusion of all total-time output in that program cycle--one of two events occurs:

1. If ȣOF (or ȣOV, respectively) does not appear in Output Indicators (cols. 23-25, 26-28, or 29-31) of <u>any File-</u> Identification specifications line of that file (see also <u>Dual-Feed Carriage</u>, below): The form (for that file) is automatically advanced until the next

channel-1 punch is sensed by the carriage-control stop brushes. (If a channel-1 punch is already at the carriage-control brushes, the form is advanced to the next channel-1 punch.) Nevertheless, the overflow indicator remains on until conclusion of the next detail-time output.

2. If bOF (or bOV) appears in Output Indicators of any File-Identification specifications line of that file, and that indicator is on:
   The program next performs overflow-time output (see below: Output Indicators--OF, OV).
   No automatic forms advance takes place in the pertinent file (standard or lower-feed file for OF, upper-feed file for OV); but automatic forms advance is retained for the other file (if dual-feed carriage used). The overflow indicator remains on until conclusion of the next detail-time output.

   NOTE:
   (a) An entry of NOF (or NOV) in Output Indicators does not cause overflow-time output to take place, nor does it prevent automatic overflow forms advance to channel 1--unless there is an OF (or OV, respectively) specification elsewhere in File-Identification Output Indicators for that file.
   (b) Entries of OF or OV in Output Indicators of Field-Description specifications lines have no effect on automatic forms advance, and do not cause the program to perform overflow-time output.

Note that the automatic overflow forms advance or the alternative performance of overflow-time output occurs after total-output time. Thus, with detail and total printing programmed in the conventional manner--at detail time and total time, respectively--all detail lines and all total lines of one program cycle are completed before forms advance or overflow output takes place. Therefore, the channel-12 punch must be placed high enough to permit completion of the maximum detail-time and total-time output lines of one program cycle beneath the location of the channel-12 punch. (It is possible to program forms overflow to take place prior to total-time output--see Programming Tips, Appendix E.)

If an overflow indicator is turned off or on by a SETOF or SETON instruction, or as a Field or Resulting Indicator (in input or calculation specifications), it reverts --at the conclusion of the output time that follows its programmed setting--to the status it would have had otherwise.

Further details on the behavior of overflow indicators appear in Figure 6 (RPG Program Logic) and under Program Logic Flow (Overflow-Time Output), Indicators (OF, OV), and Indicator Hierarchy--all in Programming for RPG--General Information; under Space, Skip (Points to Note: 6,8,9, 10), above; Output Indicators--OF, OV, immediately below; under Dual-Feed Carriage, below; and under Output Indicators, in Field-Description Specifications, below.

Output Indicators--OF, OV
(File-Identification Specifications)

Entries of indicators other than OF and OV are discussed above (under Output Indicators--Cols. 23-31). Entries of NOF or NOV operate like entries of any other indicators in these fields (cols. 23-31), except that the output is then conditioned to be performed only if that overflow indicator (OF or OV, respectively) is not on at the particular output time (detail or total time--D or T in col. 15).

The conditions under which overflow indicators (OF, OV) are on are explained above (Overflow Indicators--OF, OV).

If OF (or OV) is specified in Output Indicators of a File-Identification line, that output is always executed following total-time output, and only provided the OF (or OV) indicator is then on. Execution is also subject to the status, at overflow-output time, of any other indicators specified in an AND relationship to the OF (or OV) indicator.

Expressed another way: specification of OF (or OV)--but not NOF (or NOV)--in Output Indicators of a File-Identification line assigns that output to a special program-cycle segment known as overflow output (see Figure 6, RPG Program Logic), timed to take place after total-time output. Performance of the output at overflow-output time remains subject to the status of all Output Indicators for that output. If the overflow indicator is off--or any additional indicator in an AND relationship is not in the specified status--at overflow-output time, the output is not performed.

During overflow-output time, all outputs conditioned by the OF (or OV) indicator are performed--subject to appropriate status of Output Indicators assigned--in the order in which the File-Identification lines appear in the output-format specifications, except: all "total" overflow output (T in col. 15) precedes all "detail" overflow output (D or H in col. 15). Although all overflow-time output occurs during a separate program cycle segment, the File-Identification lines for overflow-time output must nevertheless be grouped with the other detail (D in col. 15) or total (T in col. 15) output lines.

WARNINGS

1. During overflow-output time, the card-type Resulting Indicator for the next card is on. If output is suppressed on that card type--or conditioned to occur only on some other particular card type--no forms advance to the new page occurs. (There is no automatic overflow forms-skip to a channel-1 punch of a carriage when OF (or OV, respectively) is specified in Output Indicators of any File-Identification line of that file.) Conditioning overflow-time output by card-type Resulting Indicator or Field Indicator--when one cannot be sure at what point of the card deck the overflow will occur--can create the impression that the overflow operation failed: in reality, it may have been suppressed--by indicators in an AND relationship--during the one program cycle during which the overflow indicator was on. It does not remain on beyond the next detail-time output, merely because no forms-skip took place.

   Similar comments apply to calculation specifications whose performance is made contingent on the status of an overflow indicator and a card-type Resulting or Field Indicator.

2. Other, seemingly peculiar, results can occur when not all types of input (or combined-file) cards print detail output. For example--

   <u>Assume</u>: Input cards of type A and type B. Only type A is listed (at detail time); but both types are included in group control (Control Level). Group totals are printed at total-output time. OF is specified in Output Indicators, for forms advance and printing of overflow-page headings.
   <u>Effect</u>: As previously explained, all group totals are printed on the old page, before overflow-time output, when a control change occurs in the same program cycle in which a channel-12 punch is passed (because overflow-time output follows total-time output). This remains true and is manifestly true when a type-A card is the last card of a control group.
   However, if overflow is signalled during the printing of a type-A card, and the next card is of type B and is the last card of a control group, the group totals are printed on the next page, giving the (false) impression that overflow forms-advance took place after detail-time output before total-time output.

What actually happens is: the overflow signal is triggered during detail-time output printing of a type-A card. This is followed by total-time output (of the same program cycle), during which nothing is printed (no control break). This, in turn, is followed by overflow-time output during which the form is advanced to the next page and overflow-page headings are printed. The next card is of type B, for which nothing is printed during total- or detail-time output. This type-B card concludes the control group. Totals are therefore printed before processing of the next card. Since forms advance took place during processing of the preceding type-A card, and nothing has been printed from the type-B card, the group totals are the first non-overflow data on the new page. It now looks as though overflow occurred before total-time output; but the overflow operations and the group-total printing actually occurred in two different program cycles.

File-Identification Specifications in AND and OR relationships are explained above (under <u>Output Indicators</u>).

File-Identification lines with OF (or OV) in Output Indicators may be in AND relationship with preceding and-or following lines (when more than three indicators are required in an AND relationship). The user must remember that the status of the other indicators at <u>overflow</u>-output time is then relevant--not their status at detail time, even if Type (col. 15) is designated D.

File-Identification lines with OF (or OV) in Output Indicators may be placed in an OR relationship with preceding and/or following lines. The user must then be careful that the output does not occur twice--once at overflow-output time and once at total- or detail-output time--when the Output Indicators in two lines in an OR relationship satisfy the criteria. (Figure 5E, lines 09 and 14, partially illustrates the point.) Execution of the overflow specifications should then be suppressed when the OR condition also exists (e.g.: OFNL1). (See also Figure 51A.)

An example:

If a group identification is to be printed at detail-output time of the first card of each control group, and also at the top of an overflow page, one common programming technique involves two File-Identification lines in an OR relationship. One line has OF in Output Indicators, the other L1. (D is specified in col. 15.) However, if printing at the overflow point (at or below the channel-12 punch) coincides with the last card of a control group, group identification is printed twice: that of the old group at the top of the new page (during overflow-time output), and the identification of the new group at detail-output time of the first card of the new group. If forms advance is specified, it also occurs twice.

A simple way to prevent such undesired duplication is to specify OF and NL1 in an AND relationship in Output Indicators of the overflow File-Identification line. The L-indicator for a control break is already on before overflow-time output; thus, the overflow output is prevented when group-control output provides the necessary data. (This method assumes that forms advance to the next page is desired after every control break of this level, as well as when the overflow point has been reached.)

Sometimes it is desired to print the same column headings at the top of the first page and of overflow pages. Two convenient approaches are shown in Figure 51B.

Note: Passing channel 12 during overflow-output time does not cause the overflow indicator to turn on again for the next cycle. Therefore, it is possible to skip to more than one new page during overflow-output time, without this itself causing overflow after total output of the next cycle.

Explanation of Entries in Figure 51A-Part I

The File Name PRINT is assumed to have been associated with the printer, in the File Description specifications.

It is desired to print the column headings (the words ACCOUNT, N A M E, BALANCE) across the top of each page--on the first page, on each overflow page, and on the new page to be started after each L2 Control-Level break. The example illustrates a simple method for printing the same constant information under each of these three conditions.

Printing must be at detail-output time (D or H in col. 15) in order (1) to print constants before other data from the first card of a control group and (2) to skip to a new page on a control break after--not before--the group totals have been printed.

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System/360

Form X24-3352-1
Printed in U. S. A.

Date

Program

Programmer

Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line (3-5)	Form Type (6)	Filename (7-14)	Type (H/D/T) (15)	Stacker Select (16)	Space Before (17)	Space After (18)	Skip Before (19-20)	Skip After (21-22)	Output Indicators Not (23-25)	And Not (26-28)	And Not (29-31)	Field Name (32-37)	Zero Suppress (Z) (38)	Blank After (B) (39)	End Position in Output Record (40-43)	Packed Field (P) (44)	Constant or Edit Word (45-70)	Sterling Sign Position (71-74)
0 1	O	PRINT	D			3	Ø1		L2	NOF		Specify one						
0 2	O	O	R						OF	NL2		or the other						
0 3	O														8		'ACCOUNT'	PART I
0 4	O														24		'N A M E'	
0 5	O														42		'BALANCE'	
0 6	O																	
0 7	O	PRINT	D			3	Ø1		OF	NL2								
0 8	O	O	R						L2									PART II
0 9	O											ACCT			8			
1 0	O																	
1 1	O																	

Figure 51A. Forms Advance and Printing of Constants or Identification on Overflow and After Control Break

Specifications line 01 causes the output to occur at the beginning of each L2 control group. The "constant" data (explained under Field Description, below) is therefore printed on the first page, as well as on every other new page started when a new L2 control group begins.

Line 02 provides for the same output--the column headings of constant data--at the top of each overflow page.

Because overflow output and detail output take place in separate distinct time segments of the program cycle, either the operation in line 02 or that in line 01 must be suppressed when an L2-level control break occurs in the same cycle as an overflow signal. If neither NL2 in line 02 nor NOF in line 01 were specified in Output Indicators, and an overflow signal and L2 control break coincided in one program cycle, the events would be:

1. Skip to channel 1 at start of overflow output;
2. Printing of constant data;

(OF specifications)

3. Skip to channel 1 at start of detail-time output
4. Printing of constant data

(L2 specifications)

In this example, it is immaterial whether overflow output is suppressed when L2 is on (line 02: OF NL2; line 01: L2), or L2 output is suppressed when OF is on (line 02: OF; line 01: L2 NOF), because only constants are printed.

However, the time of execution in the program cycle differs: if the OF-specification output is performed, this takes place at overflow-output time; if the L2-specification output is performed, this occurs at regular detail-output time. Therefore, if data from cards is to be printed, output at overflow time can only be from a preceding card, whereas output at detail time can be from the new card. Normally, when control-level break and the overflow point coincide, the data from the new card is to be group-indicated. Thus, the OF line rather than the L2 line must be suppressed. This is illustrated in the second portion of Figure 51A.

At detail time following an L2 control break the carriage skips to the next channel-1 punch, before the headings are printed (01 in cols. 19-20). Thereafter, the form is advanced 3 spaces (3 in col. 18). Since cols. 17-22 are blank in line 02, the forms-control instructions are taken from the last preceding line (of the same group) that contains significant entries, i.e., line 01.

Note: Output should not also be specified, in this application, before the first card has been read (at 1P time). This would cause printing of the constant data, followed by forms advance and another line of the same constant data at detail-output time of the first card (which is normally also the first card of an L2 Control-level break).

Explanation of Entries in Figure 51A-- Part II

This example is intended to be contrasted with Part I. Again, the form is to be advanced to a new page when either a Level-2 control break has occurred or overflow was signalled.

However, instead of constant heading data, the account number (contents of the field ACCT) of the pertinent card group is to be printed at the top of each page. As specified in lines 07 and 08, this will operate correctly:

Line 07: If the overflow indicator is on at overflow-output time, and no L2 control break has occurred (NL2 in Output Indicators), the form is advanced (at overflow-output time) to the top of a new page. The account number from the previous card is then printed. Since no L2 control break has occurred, there must be at least one more card of the same control group; therefore, the account number from the previous card is appropriate to identify the data that will follow on that page.

Line 08: If an L2 Control-Level break has occurred, the form is advanced (at detail-output time) to the top of a new page. At detail-output time, the ACCT field contains the account number from the first card of the new control group. This is the proper indentification for the data that will follow.

Now note what would happen when overflow and L2 control break coincide, if OF were the only specification in Output Indicators in line 07, but L2 NOF were spcified in line 08:

Duplication of page heading is properly prevented; but--when L2 and OF are both on--the specifications in line 07 (not line 08) are executed. These are performed at overflow-output time, when the data from the first card of the new control group is not yet in the process area. The account number at the top of the new page will be that of the last card of the previous control group; but

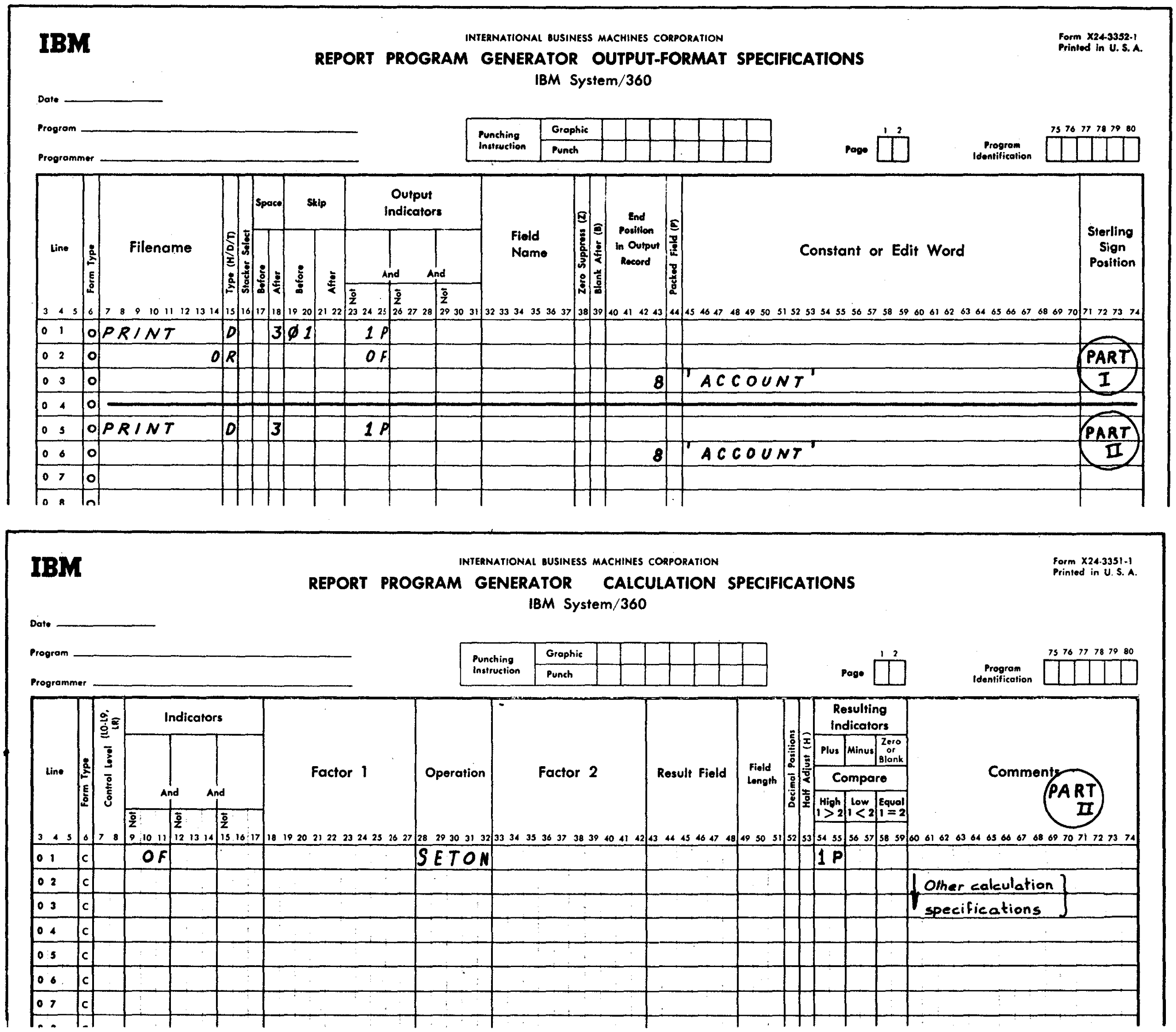


Figure 51B. Forms Advance and Printing of Constants on First and Overflow Pages

the card data that will follow will be from the new group. The group will thus be incorrectly identified.

Explanation of Entries in Figure 51B-Part I

This is a straightforward set of output specifications for printing the same constant data (e.g., the word ACCOUNT) at the top of the first page (before the first data card has been read--1P is on) and at the top of each overflow page (at overflow-output time).

Two File-Identification specifications lines are needed with this method. Part II presents an alternate approach.

Explanation of Entries in Figure 51B-- Part II

This illustrates use of the OF indicator in calculation specifications to accomplish the same as Part I, but without an OR line in the output-format specifications.

The File-Indentification specifications (line 05, output-format specifications) cause the output to be performed at detail time, if indicator 1P is on. It is always on before the first card has been read; therefore, the heading word ACCOUNT is printed in print positions 2-8 on the first page.

The first detail-time calculation specification (line 01, calculation specifications) causes indicator 1P to turn on if the OF indicator is on. The OF indicator is on if a line was printed at or below the channel-12 punch during the preceding detail-time or total-time output of the same program cycle (see Figure 6, RPG Program Logic). The output specifications in line 05 are then the first detail-time output operations performed in the next program cycle.

Indicator 1P is turned off again by the RPG program after a new card has been read.

Note that overflow-time output is not utilized at all with this programming approach. Because OF (or OV, respectively) is nowhere specified in Output Indicators of a File-Indentification line, forms advance to channel 1 is automatic after total-output time, if the overflow indicator is on. Therefore, Skip/Before (cols. 19-20) must not contain 01; otherwise, the form is advanced to a second new page at the beginning of detail-output time following overflow.

Dual-Feed Carriage (DFC)

This is a special feature available for the IBM 2203 Printer equipped with a 39-, 52-, or 63-character typebar. The DFC permits control of two different forms in one job run. Each form has its own forms-control carriage and the forms tractors of the two carriages are controlled independently, each having its own carriage-control tape.

The overflow indicator OV is associated with the extra carriage, the so-called upper carriage. The overflow indicator OF remains associated with the standard, or lower, carriage.

Pairs of forms that are to contain information from a common source--any, all, or none of which may apply to both forms--can be printed in a single run with entirely different spacing and format requirements. The two forms can be completely segregated side-by-side, or they can be partially or entirely overlapped.

For example: payroll checks can be printed alongside a payroll register; or the checks can be above or beneath the register, with different spacing and forms-skipping. Similarly, invoices and an invoice register, or invoices and shipping labels, can be handled side-by-side or partially or fully superimposed. (For further details on the DFC feature see the publication IBM System/360 Model 20, 2203 Printer, Form A26-5926.)

The forms controlled by the two carriages are assigned separate output-file names in separate entries in the File Description Specifications, each name being associated through the Device code with a particular one of the two carriages. Separate File-Identification and Field-Description entries for these two files are required in the output-format specifications, when both files are to be used. The Field-Description specifications may be different, or partially or wholly identical, when desired.

The two files are basically two separate files. However, printing takes place concurrently for the two files (upper and lower carriage)--i.e., output is treated as though to a single file, which can speed output considerably--if all four of the following conditions are satisfied:

1. Output is specified for the same program-cycle segment (both D or H, or both T, in col. 15).

2. The specifications for the two files follow each other in the output-format specifications, without intervening entries for any other output. (If output to the same two files is specified several times, the entries for the two files must be paired wherever they are to be treated as a single file for concurrent output.)

3. The entries in Output Indicators of File-Identification lines (though not necessarily of Field-Description lines) are identical for the two files. (Same overflow indicator for both files also satisfies this criterion.)

   Not only must the same indicators be specified alike (each preceded by ƀ or N, respectively, for both files), but they must also be entered in the same order. If there are AND and/or OR lines, the number of such lines, and their sequence and Output Indicators must correspond for the two files.

   (These requirements preclude simultaneity of output for the two files if different overflow indicators (OF and OV) are specified in Output Indicators of the File-Identification lines of the two files, or if one--but not the other--has one of these overflow indicators specified.)

4. No Output Indicator in a File-Identification line is required to be

off (Nxx) as a condition of output for these two files, if that indicator may turn on as output fields for the first file are transferred to the output area (see "Blank-After"). For example:

N20 appears in Output Indicators of the File-Identification lines for both files;
Field B is an output field in a Field-Description line of the first of the two printer files;
Blank-After is specified for Field B in that Field-Description line (col. 39);
Indicator 20 is assigned to "Zero or Blank" either (1) in Field Indicators (input specifications) for Field B, or (2) in Resulting Indicators (calculation specifications) for Field B as Result Field in an arithmetic or TESTZ operation.

This could cause the indicator to turn on between output for the first and second files. Therefore, if an indicator has been assigned in such manner, the two dual-feed-carriage files are designated--at program-generation time--as two separate files whose output will be performed consecutively, but not concurrently.

Concurrent printing can occur even though there are different Space and/or Skip specifications (cols. 17-22) for the two files, provided that the other conditions (above) are met for treating the two output files as one.

Forms overflow for each file conforms to the normal overflow operation of any printer output file:

1. If ƀOF is specified in Output Indicators of any File-Identification line for the lower-feed file, no automatic overflow forms-advance occurs in that carriage. The particular output is performed at overflow-output time, and forms skipping to a new page must be expressly specified. The same is true for the upper-feed file, if OV appears in any of its File-Identification lines.

2. If ƀOF (or ƀOV, respectively) does not appear as Output Indicator in any File-Identification line for the respective printer file, forms advance to channel 1 is automatic for that file after total-output time, when the pertinent overflow indicator is on.

If ƀOF appears in File-Identification Output Indicators for the lower-feed printer file, but OV does not appear for the upper-feed file, then overflow forms-skipping to channel 1 is automatic for the upper carriage but not the lower; and vice versa. If OF or OV is specified in File-Identification Output Indicators for the other file (i.e., OF with the upper-feed file and/or OV with the lower-feed file), automatic overflow forms advance remains operative in that other file; however, the output called for by the File-Identification and Field-Description specifications occurs at overflow-output time (not at detail- or total-output time)--even though the overflow indicator is that of the other file.

Figure 52 illustrates specifications for both files of a dual-feed carriage. For general information on overflow indicators and overflow forms advance see <u>Overflow Indicators--OF, OV</u> and <u>Output Indicators--OF, OV</u>, above.

Line (3-5)	Form Type (6)	Filename (7-14)	Type (H/D/T) (15)	Stacker Select (16)	Space Before (17)	Space After (18)	Skip Before (19-20)	Skip After (21-22)	Output Indicators Not (23-25)	And Not (26-28)	And Not (29-31)	Field Name (32-37)	Zero Suppress (Z) (38)	Blank After (B) (39)	End Position in Output Record (40-43)
01	O	INVOICE	D			2			Ø5	MR	N14				
02	O	AND							84						
03	O	OR							23	NMR					
04	O											FIELDA			Ø8
05	O											FIELDB	Z		31
06	O								NOV			FIELDC	Z	B	4Ø
07	O	REGISTER	D			1			Ø5	MR	N14				
08	O	AND							84						
09	O	OR			2	3			23	NMR					
10	O											FIELDB	Z	B	65
11	O								OF			FIELDD		B	8Ø
12	O											FIELDE			94
13	O														

Figure 52. Examples of Entries for Dual-Feed Carriage Output

Explanation of Entries in Figure 52

It is assumed that one of the two file names (INVOICE or REGISTER) has been associated, in the File Description Specifications, with the upper-feed carriage (PRINTUF), and the other with the lower-feed carriage (PRINTLF or PRINTER). Field names and print positions have been included for completeness; their use is explained more fully later.

The specifications meet the criteria for concurrent printing to both files (i.e., the program consolidates the two files into one, at generation time):

1. Both file outputs are in the same program-cycle segment (D in Col. 15);

2. The specifications for the two files are contiguous;

3. Identical File-Identification Output Indicators are specified in equivalent positions, and AND- and OR-line entries correspond.

4. It is assumed that neither indicator 14 nor MR is assigned as Zero-or-Blank indicator to FIELDC (the only field in the first file with a Blank/After instruction).

Note that concurrent printing is still accomplished even though forms control (cols. 17-22) may differ for the two files.

Neither OF nor OV is specified in File-Identification lines for the lower or upper-feed carriage, respectively. Therefore, printing is at detail time (D in col. 15)--not at overflow-output time--and overflow forms advance to channel 1 is automatic for both files.

The contents of FIELDD (line 11) are only printed if the OF indicator is also on at detail-output time. Note that indicator OF is in Output Indicators of a Field-Description line, which does not affect execution or timing of the output for the file (i.e., it does not cause output for the file to be at overflow-output time, or to be subject to the status of the OF indicator). Similarly, the contents of FIELDC (line 06) are only printed if the OV indicator is not on at detail-output time. (see also Output Indicators, under Field Description and Control, below.)

Note that the print positions (End Position in Output Record) are continuous for the two files: only a single printer serves for output even though it is assigned two files; if fields for both files were designated to print in the same location on the print line, this would be tantamount to attempting to print different information in the same position at the same time. The program then overlays, in the output core-storage area, the data from the later line over that of the earlier line, and only one character is printed in any one position.

Of course, the entries for output positions in the two files need not be in sequence, so long as none of the output fields in one overlap those in the other. Even this restriction does not apply when it is known that the two files are never output at the same time--either (1) because Field-Description Output Indicators are mutually exclusive, or (2) because output to the two files is not simultaneous. And, of course, the two forms may be overlapped, so that an output field may appear on both forms.

FIELD DESCRIPTION AND CONTROL--COLS. 23-70

One Field-Description specifications line is needed per data field. Field-Description lines follow immediately beneath the File-Identification line(s) for the particular output operation. At least one Field-Description line is required per File-Identification line (or group of lines, when there are AND or OR lines).

Each Field-Description line contains the information necessary to determine the output format of an individual field, its location in the output record, and any conditions restricting output of the field beyond the general restriction on output of the entire file.

Output Indicators--Cols. 23-31
(Field-Description Specifications)

Entries in Output Indicators of a Field-Description line follow the rules for Output Indicators in File-Identification lines, with these differences:

1. The indicator entries apply only to the field or constant described in the particular Field-Description line--not to output of the entire file. They have no significance unless output to the file--as determined by the File-Identification specifications--takes place; then they represent additional restrictions on output of a field--subsidiary to the restriction in the File-Identification specifications on output to the file itself.

   Even if all field output for a file is suppressed in a program cycle (by virtue of the status of indicators in the Field-Description lines), the file output still takes place if the Output Indicators in the File-Identification line have the appropriate setting. Therefore, for instance, if file output to a card file takes place, but all field output is suppressed, the card is transported past the punch and print stations without being punched or printed; if the output file is the printer, a blank line is "printed", but forms movement is implemented as though data had been printed.

2. Entry of an overflow indicator (OF, OV) has no effect on forms control, nor does it cause the output to be shifted (from detail or total time) to overflow-output time. Indicators OF and OV in Field-Description lines are treated like any other conditioning indicators--for example: if OF is specified, output of the field occurs only if the OF indicator is on at the time

output to the file takes place; if NOF is specified, output of the field is contingent on indicator OF being off.

3. Output Indicators in an AND relationship in Field-Description Specifications are limited to the three that can be accommodated in one line; AND lines are not permitted. (See Programming Tips, Figure E6I, for setting of a single indicator to represent three AND conditions.)

4. OR lines as such are not permitted. However, the same output field may be repeated on successive lines, each time conditioned by a different indicator or combination of indicators. The output for the field is then performed if the indicator conditions in at least one of the lines are satisfied. (Also see Programming Tips, Figure E6(b), for setting of a single indicator to represent several OR conditions.)

	Line (3-5)	Form Type (6)	Filename (7-14)	Type (H/D/T) (15)	Stacker Select (16)	Space Before (17)	Space After (18)	Skip Before (19-20)	Skip After (21-22)	Output Indicators Not (23-25)	And Not (26-28)	And Not (29-31)	Field Name (32-37)	Zero Suppress (Z) (38)	Blank After (B) (39)	End Position in Output Record (40-43)
	01	O	PRINT	D			2			44						
	02	O								L2			SALSMN	Z		4
	03	O								OF	Nφ5		SALSMN	Z		4
1	04	O								L1			CUSTMR			12
	05	O											INV#	Z		2φ
	06	O											AMOUNT			28
	07	O								25	N16	φ2	COMSN			4φ
	09	O	PRINT	D				φ1	φ2	φ4						
	10	O	OR							OF	Nφ4					
	11	O											CUSTMR			13
2	12	O								φ4			ORDER			7
	13	O								φ4			SALSMN	Z		22
	14	O								N85			INVOIC	Z		29
	15	O								85			CRMEM	Z		29
	16	O								φ4			DATE			12φ
		O														
		O														

Figure 53. Examples of Output Indicators for Field Description Specifications

Field Selection

Point 4, above, explains output of the same field under OR conditions. The same technique can be applied to select one, from among several different fields, to be used for output to one location.

If several different fields of appropriate size and format are each conditioned by mutually exclusive Output Indicators, and all have the same entry in End Position in Output Record (see below), at most one of these fields will be transferred to that output area.

Figure 53 shows two sets of entries portraying applications of Output Indicators for Field-Description lines. File-Identification, field-name entries, and End Position in Output Record have been included for the sake of clarity. (Field names and output-record positions are more fully covered shortly.) The numbers to the left of the figure correspond to the explanatory sections that follow.

Explanation of Entries in Figure 53

Example

1. Assume that the file PRINT has been associated with the printer in the File Description Specifications.

   Printing takes places at detail-output time, provided indicator 44 is on. Five fields are printed:
   The contents of INV# and AMOUNT are printed each time the file output takes place.
   Besides output being subject to indicator 44:
   The contents of the field SALSMN are printed only either
   (a) when the L2 indicator is on at detail-output time--the standard method for group-indicating a field on the first card following a Level-2 control break; or
   (b) when the overflow indicator is on at detail-output time--the standard method for group-indicating a field on the first printed line of an overflow page, when overflow-output time is not utilized (OF is not assigned in Output Indicators of the File-Identification specifications).
   SALSMN is further conditioned to print on the first line of an overflow page only if indicator 05 is then not on.

   Besides group-indication (i.e., suppression of listed data except at the beginning of a new group), the specifications for the SALSMN field also illustrate:
   (a) How to handle OR relationships for fields: lines 02 and 03 are in an OR relationship; if either L2 is on at detail-output time, or OF is on and 05 is off at that time, the field is printed (if indicator 44 is also on).
   It is also shown (by OF and N05 in one line) that each OR condition may

consist of a combination of indicators.
(b) That it is not necessary to make the OR conditions mutually exclusive in this situation: The OF indicator in a Field-Description line operates like any other indicator; it does not cause the output to be switched from detail-output time to overflow-output time (as it does when entered in a <u>File</u>-Identification line). Therefore, the output described in lines 02 and 03 takes place in the same program-cycle segment (detail-output time)--thus, the field cannot print twice even if overflow and the end of a Level-2 control group coincide. (Note the difference when OF is assigned in a File-Identification line: see <u>Output Indicators--OF, OV,</u> under <u>File Identification and Control</u>; Figures 5E and 51A; and Figure 53, Part 2.)

If overflow and an L2 control break occur in the same program cycle, both lines 02 and 03 are executed; but the data for line 03 is transferred to the output area after that for line 02. Since the data is the same (the contents of the SALSMN field), and is moved to the same output location (ending in print position 4), no harm is done.

The contents of the field CUSTMR are printed (subject to indicator 44) only when the L1 indicator is on at detail-output time--the standard method for group-indicating on the first card of a control group.

The contents of the field COMSN are printed (subject to indicator 44) only if indicators 25 and 02 are on, and indicator 16 is off.

This is an example of the maximum of three indicators in an AND relationship.

The overflow indicator (OF, or OV) is not specified in Output Indicators of a File-Identification line. Therefore, overflow forms advance to channel 1 is automatic.

<u>Example</u>

2. Assume that (1) the file PRINT has been associated with the printer in the File Description Specifications, (2) indicator 04 represents a heading card followed by a group of listed detail cards, (3) printing of some fields is to be suppressed when printing headings on overflow pages, and (4) Invoice No. (INVOIC) is to be replaced by Credit Memo No. (CRMEM) when indicator 85 is on ("field selection").

Printing takes place at detail-output time if indicator 04 is on, and at overflow-output time if indicator OF is on and indicator 04 if off. If the overflow point and the reading of a new heading card can happen in the same program cycle, line 10 must have N04 in Output Indicators; otherwise, when the overflow signal coincides with a new heading card, the headings would be printed twice: first the headings for the old (completed) group during overflow-output time, then the new data from the heading card at detail-output time. (If the nature of the application is such that overflow and a type-04 card cannot coincide, then N04 in line 10 is not needed.)

The contents of all five fields are printed at detail-output time when a type 04 card is being processed; but only CUSTMR and INVOIC or CRMEM are printed at overflow-output time.

<u>Field selection</u> is performed between the fields INVOIC and CRMEM: one or the other is transferred to the same output area, depending on the status of indicator 85. (If the field CRMEM is no shorter than INVOIC, N85 in line 14 is not needed: the data from CRMEM would be overlaid over the INVOIC data if indicator 85 is on.)

Before a new heading card or an overflow-page heading is printed, the form is advanced to the top of a new page (01 in cols. 19-20); after printing of the heading, it is skipped to the nearest channel-2 punch.

WARNING: An important point should be made for the applications in which the overflow signal (channel 12) and the reading of a new heading card can occur in the same cycle, and the output requirements parallel those exemplified here (i.e., the two file output specifications are in an OR relationship, and the printing of certain fields is suppressed for overflow output):

The programmer has the choice of suppressing the printing of some fields (see lines 12, 13, and 16) during overflow-output time by specifying either (1) the indicator of the condition to which the printing of the field is to be restricted (04 in this example), or (2) the negative of the indicator that is on when the field is not to be printed (NCF in this example).

As illustrated--with indicator 04 in the pertinent Field-Description lines--

the application will work, even when overflow and a type-04 card coincide.

However, if NOF (in place of 04) were specified in lines 12, 13, and 16, the application would work correctly during each overflow heading, and for the printing from each type-04 heading card--but the latter only if overflow was not signalled during the same program cycle in which the new heading card was read. The reason is that--although overflow output is not executed (because N04 is entered in line 10) if overflow was signalled in the same cycle in which a new heading card was read --the overflow indicator is nevertheless still on at detail-output time, when the new heading-card data is printed. In that situation, the output from the fields ORDER, SALSMN, and DATE would be suppressed--by NOF--when the new heading card is printed.

Field Name--Cols. 32-37

Entry of a field name here designates the contents of that field for output--forms printing, card punching, or document-printing--subject to Output Indicators in this Field-Description line and in the preceding File Identification. The output device (printer or card punch) is determined by the file name (see File Name, above).The location of the data in the output record is determined by the entry in cols. 41-43.

The field name is entered left-aligned (to begin in col. 32). With one possible exception (PAGE--see Consecutive Numbering, below), the name must have been previously defined in the input specifications (Field Name), file-extension specifications (Table Name), or the calculation specifications (Result Field).

All previously defined field names are permitted, except the following:
ALTSEQ, a name that begins with CONTD, or
PAGE followed by one or two characters.
The field name PAGE itself has a special significance (see Consecutive Numbering, below).

If the field name corresponds to the name of a table defined in the file extension specifications, the output consists of the contents of the "hold" area for that table--i.e., normally the data selected from the table in the last LOKUP operation. (For details, see Table Look-Up Operations, under Calculation Specifications.)

Output fields need not be recorded in the sequence in which their data is to appear in the output record; that sequence is determined by entries in cols. 41-43 (End Position in Output Record).

The sequence in which the fields are specified can nevertheless be important under some circumstances--principally when Blank-After is specified (col. 39): with one exception (below), fields are transferred for output to the designated (cols. 41-43) location of the output core-storage area in the order in which they are recorded (under the File-Identification specifications). Therefore:

If successively specified output fields are assigned partially or completely overlapping positions in the output record (i.e., based on entries in cols. 41-43), the data from the field specified later (lower down) replaces any data in the same output-area positions of a field recorded higher up. (The same applies regardless of whether the field name is the same or different--it is possible for the contents of the same field to change during output, by a Blank-After instruction.)

Of course, if several Field-Description entries specify transfer to the same output record area, but only one of the transfers is executed because of either (1) mutually exclusive Output Indicators, or (2) association with different program-cycle segments, no overlay problem exists.

Exception:

When card document-printing (interpreting) and punching are both specified for the same card (under one File-Identification specification), then transfer of the data of all appropriate fields to the output punch-storage area precedes transfer to the card-print output storage area. This calls for caution in the use of Blank-After instructions with such fields.

If, instead of the contents of a field, a constant is to be transferred to the core-storage area for the output-record location specified in cols. 41-43, Field Name (cols. 32-37) is left blank. (See Constant, below.)

Figures 5E and F, 51A and B, 52 and 53 already illustrated the use of output field names and constants. Several further examples appear in Figures 54A and B.

Consecutive-Numbering (Page Numbering)

RPG provides automatic page numbering or consecutive card-numbering simply by using

PAGE (in cols. 32-35) as the name of an output field for the pertinent file.

Only one page- or consecutive-numbering field can be set up in this manner. If serial numbers are needed for several output files--such as both files of a dual-feed carriage, or a printer file and a card file, etc.--numbering of the additional files must be handled with another field name and use of calculation specifications.

The field named PAGE is basically treated like any other output field:

1. Output of the field is contingent on output to the file; i.e., the conditions set up by entries in Type (col. 15) and in Output Indicators of File-Identification lines must be satisfied;
2. The contents of the PAGE field are tranferred to the proper output core-storage area to appear in the output record in the location specified in cols. 41-43; and
3. The contents of the PAGE field may be printed on report forms, document-printed on cards, or punched into cards.

However, the PAGE field differs in other significant respects:

1. The contents of the field are always incremented by +1 (by the RPG program itself) immediately before output from the field. (At the beginning of object-program execution, the field contains zeros.)
   Therefore, if PAGE appears only in the output-format specifications, output from the field the first time is 0001; the second time, it is 0002; etc.
   If a value was entered into the PAGE field from an input card (see Consecutive Numbering--Header Cards, under Input Specifications) or by calculation specifications, whatever value stands in the field at time of output is incremented by +1 before output. Thus, if (for example) the most recent entry in the field PAGE from an input card was 1000, and 25 was subtracted from PAGE by a calculation instruction, output will be 0976. The next output (assuming no new input or calculation specifications changes to the field) will be 0977; etc.
   If the PAGE field is used as output several times in one program cycle, the number is incremented before each output.
2. The field is always numeric, and 4 digits long.
3. The low-order position is always signed (normally plus, although minus is possible if a negative number was entered from a card or in calculation specifications).
   Zero Suppress or an edit word may be specified (see Zero Suppress and Edit Word, below). If this is not done, leading zeros will be printed or punched for numbers of less than four significant digits, and the low-order position will be signed: when punching, the low-order position will then contain a 12- or 11-overpunch; when printing, the character will be as shown in the EBCDIC-table column labelled C or D, respectively. Depending on the typebar, chain, train, or MFCM print-mechanism set of graphics, a signed zero may only print a plus or minus sign, or the position may remain blank. Normally, when printing page numbers, Zero Suppress (Z in col. 38) is specified to eliminate the leading zeros, and avoid zoning by the plus sign.
4. Output Indicators cannot be assigned in a PAGE Field-Description line to make output of the field subject to the status of indicators. Field Output takes place whenever output to the file is performed.
5. Output Indicators may be designated in a PAGE Field-Description line, and--when output to the file is performed (subject to File-Identification Output Indicators)--have the following effect:
   (a) If not all assigned indicators are in the designated states (on, or not on, as specified): no effect.
   (b) If all assigned indicators have the status stipulated (on, or not on, as specified):
      The PAGE field is set to zero before being incremented by 1, both prior to output. Output is then 0001.

   Note: Blank-After (col. 39) is also a permitted method for resetting the PAGE field to zero; but it is usually awkward in practice to set up the control to execute this reset at the desired time only.

Figure 25 and its accompanying text explain how to employ input cards to initiate a series of numbers for the PAGE field with any desired 4-digit value at any point of an input (or combined-file) deck. Figure 54A includes specifications to print output from the PAGE field.

Zero Suppress (Z)--Col. 38

This column must be left blank if:

1. The field is defined as alphameric (in the input, calculation, or file extension specifications); or
2. The Field-Description line applies to a

constant, in cols. 45-70 (see Constant, below); i.e., when cols. 32-37 are blank because output does not refer to the contents of a named field; or
3. Editing is specified by an edit word, in cols. 45-70 (see Edit Word, below); or
4. Packed Field is specified for output (P in col. 44); or
5. The field does not consist only of valid digits (0-9),except for a sign permitted in the low-order position.

If none of the above applies, the letter Z may be entered in col.38 of a Field-Description line for a field that has been defined as numeric. Specifying Z has two effects on the format of the output:

1. Blanks are substituted for leading (non-significant) zeros; and
2. Zone bits are removed from the low-order (rightmost) position of the field (i.e., the character is assigned the corresponding position, in the same row, in EBCDIC-table column labelled F).

   A numeric field is zoned in its low-order position if (a) it was zoned in that position when read in, or (b) a Field Indicator was assigned to it in the input specifications, or (c) it was a Result Field in an arithmetic operation, or (d) a zone was moved to that position in calculation specifications, or (e) the input field was blank, or (f) the field was cleared by a Blank-After specification.

The Z specification affects only the output determined in the particular Field-Description line; it does not modify the form in which the data is stored at the field location. Therefore, Z may be designated in an output line for a field without affecting the format of output for the same field in a subsequent Field-Description line (this confined effect is in contrast to the consequences of specifying Blank-After for a field--see below).

The Zero-Suppress specification is a simple method of editing a field for printing, when a sign is known to have no significance. For example: if a quantity field can only be positive, but a plus (EBCDIC C-zone) appears over the low-order position, specifying Z assures printing of an unsigned character (0-9) --rather than a letter, symbol or blank space--in the low-order position; at the same time, left zeros are eliminated.

Data to be punched is not usually edited by the specification Z in col. 38, because high-order zeros are normally to be punched. Without any editing, a negative value is punched in the low-order position with an 11-overpunch and a positive value may have a 12-overpunch (if the field was read in with a 12-overpunch, if a Field Indicator was assigned and the field contents were positive or zero, if it was a positive or zero Result Field in an arithmetic operation, or if a C-zone was transferred to it in some other calculation operation, if the input field was blank, or the field was cleared by a Blank-After specification). If the 12-overpunch is not desired, it can be removed prior to punching (see next paragraph, and Programming Tips).

To provide complete flexibility of editing for printing and punching, two other methods are available:

1. Editing by Edit Word (see below); and
2. Limited editing by calculation specifications (see Calculations Specifications, above):
   (a) A zone can be removed from the low-order position of a numeric field by moving any character shown in EBCDIC-table column F (e.g., any of the digits 0-9) to that position by a MHLZO or MLLZO operation. The last line in Figure 42, in conjunction with line 06 in Figure 43, illustrates this. (See also Programming Tips.)
   (b) Leading zeros can be changed to blanks by moving the numeric field to an alphameric field, and then moving in blanks by a MOVEL operation. The appropriate number of blank positions can be determined by move and other calculation operations.

The Z specification has been illustrated in Figures 52 and 53. Further examples appear in Figures 54A and B.

Blank After (B)--Col. 39

A "B" entered in this column causes the field to be cleared immediately after its contents have been transferred to the output core-storage area (i.e., as the specifications in the output line are executed). This is a convenient method, for example, of clearing a group-total field as the group total is transferred for printing--so that the field is ready for data accumulation of the next group.

An alphameric field is set to blanks; a numeric field is set to zeros, signed plus.

Once a field has been reset by Blank-After, it is blank or zero until data has again been placed in it by an input or calculation operation. Therefore, it is blank or zero at least for the remainder of the same program-cycle segment (total-time, overflow-time, or detail-time output). Thus, if Blank-After is specified for a field, the field is already cleared before the transfer to the output core-storage area specified in the next Field-Description line--even if that transfer involves the same field--and it is blank or zero at the time any subsequent File-Identification specifications in the same program-cycle sigment are executed.

Therefore, if output from the same field is required several times (e.g., to several media--say, for forms-printing, card punching, and/or card document-printing), care must be applied to designate the B in col. 39 only in the last of the pertinent File-Identification and Field-Description lines: as explained above, under Field Name, fields are transferred to the output core-storage area in the sequence in which they appear in the output-format specifications. However, if card document-printing (interpreting) and card punching are both specified for the same fields (under one File-Identification specification), all transfers for punching are performed first. In this situation, any Blank-After specification should be in the last Field-Description line that specifies card document printing for that field.

Note: If the output line pertains to a constant in cols. 45-70 (i.e., Field Name, cols. 32-37, is blank), the core-storage area that contains the constant is set to blanks by the Blank-After instruction. It remains blank thereafter until the object program deck is reloaded, or the program is regenerated.

Relationship of Indicators to Blank After

A Field Indicator or Resulting Indicator assigned to "Zero or Blank" turns on immediately when that same field is cleared by a Blank-After instruction during output. This applies to indicators assigned to "Zero or Blank":

1. In Field Indicators in the input specifications--cols. 69-70; and
2. In Resulting Indicators in the calculation specifications--cols. 58-59--for arithmetic or test-zone operations (specifically: ADD, Z-ADD, SUB, Z-SUB, MULT, DIV, MVR, TESTZ).

However, the fact that the Blank-After instruction may turn on an indicator for a field does not cause any other indicator for that field to turn off. For example, assume:

Indicator 25 assigned to Minus (cols. 67-68) in Field Indicators for FLDA, in input specifications;

Indicator 20 assigned to Zero or Blank (cols. 69-70) in Field Indicators for same field (FLDA);

The field is negative at input time. Therefore, indicator 25 turned on when the input data was transferred to the process area before detail-calculation time.

At detail-output time for FLDA, Blank-After is specified for FLDA.

Then: immediately after transfer of FLDA at detail-output time, indicator 20 turns on; but indicator 25 also remains on (unless previously turned off by a calculation specification).

Once an indicator assigned to "Zero or Blank" has been turned on by a Blank-After instruction, it cannot be turned off again during the same output program-cycle segment (the earliest possibility is calculation time during the next program-cycle segment, setting of Field Indicators before detail-calculation time, or automatic reset for certain indicators after the next card has been read following detail-output time). Therefore, the programmer must realize that the indicator is on for subsequent Field-Description and File-Identification specifications which may be conditioned by its status (it can, however, no longer change execution of the same File-Identification specifications, or the transfer to the output area of data in the same Field-Description line).

Note: If more than one indicator is assigned to Zero-or-Blank for the same field in different specification lines, Blank-After causes only the earliest-assigned indicator to turn on. (However, all of the different indicators assigned to Zero-or-Blank for the field, in Field Indicators or calculation Resulting Indicators of arithmetic or TESTZ operations, are on at the beginning of program execution.)

Blank After has been illustrated in Figures 5E, 5F, and 52. Further examples are included in Figures 54A and B.

The subject of indicators, as related to Blank-After, is also discussed in Program Logic Flow, under Input Specifications, and under Calculation Specifications.

End Position in Output Record--Cols. 40-43

This entry (right-justified in cols. 41-43) designates the location of the field or constant in the output record. Only the location of the rightmost character of the "field" or constant is specified.

"Field", in this case, includes any extension due to an edit word (see Edit Word, below): if an edit word (cols. 46-69) extends to the right of the data in the field, End Position in Output Record refers to the rightmost position of the edit word. For example:

Assume a seven-digit field, with its low-order position to be printed in print position 10;

Assume an edit word that (1) inserts a decimal point between the dollars and cents position, (2) inserts a comma between hundreds and thousands of dollars, (3) allows a blank to the right of the low-order digit, followed by a CR symbol for negative amounts, and (4) provides an asterisk to the right of the CR position.

The maximum printout then looks like this: $\underbrace{\texttt{xx,xxx.xxƀCR*}}_{\text{print position 10}}$

Therefore, End Position in Output Record is 14.

Note that, due to symbols (e.g., decimal point and commas) and characters that may be inserted by edit words, the field may expand to the left (as well as to the right). In the above example, a field which, unedited, would occupy print positions 4-10, occupies print positions 2-14 when that particular edit word is specified. Therefore, whenever fields are edited, care must be applied--when assigning End Position in Output Record--that successive fields are not unintentionally overlapped. (Data from the field transferred to the output core-storage area later replaces any data in the same area from an earlier transfer--see <u>Field Name</u> and <u>Blank After</u>, above, for sequence of transfers.)

Zeros in cols. 41-43 may be entered or omitted. (Col. 40 does not apply to card RPG, and may be left blank or coded zero.)

Card Document-Printing (Interpreting) Specifications (Cols. 41-43)--Special feature, available only on the 2560 MFCM Model A1, attached to an IBM System/360 Model 20, Submodel 1 or 2.

The file name (cols. 7-14) identifies the output device. Thus, End Position in Output Record (cols. 40-43) refers to the printer if the file name was associated, in the file-description specifications, with the printer; it refers to a specific card processing device, if that was the association formed in the file-description specifications.

However, the file-description specifications cannot make a distinction between punching into, or printing on, a card in the MFCM. (If the two functions were to be distinguished as separate files, it would not be possible to punch and interpret the same cards in a single pass.) Therefore, if the file name is associated with the MFCM, output to be punched is distinguished from output to be card-document-printed (interpreted) by the entry in End Position in Output Record.

Since End Position in Output Record cannot be greater than 80 for punch cards, the hundreds position (col. 41) is used to distinguish between punching and interpreting:

Col. 41 = 0 or ƀ: output is punched.
Col. 41 = 1-6: output is document-printed on the card by print head 1-6, respectively.

<u>Note</u>: If card document-printing is specified, the appropriate instructions are generated. If the object program is then executed on a MFCM that is not equipped with the card document-print special feature, the program performs all other operations in the normal manner but, of course, no document-printing takes place.

If card document-printing is specified for more print heads than are installed on the MFCM on which the object program is run, document-printing is performed as specified for the available print heads.

The entry in cols. 42-43 represents the rightmost location occupied by the low-order position of the "field" (including any extension through an edit word) in the card, in either case (punching or interpreting). The maximum value (i.e., rightmost location) possible is:

1. 80 for card punching
2. 64 for card document-printing (interpreting)

Punching, and interpreting of one to six lines (depending on the features installed), may be performed in the same card during a single pass of the deck through the system. However, all punching and interpreting for one card must be specified under a single File-Identification line (or group of AND or OR lines). The Field-Description lines for punching and interpreting may appear in any order under the File-Identification line; but, for one File Identification, all data for punching is always transferred (by the program) to the output core-storage area ahead of the data for interpreting. Therefore, if Blank-After (see above) is specified for a field that is to be punched and interpreted, the B in col. 39 must be entered in the (last) line that specifies interpreting for that field; otherwise, the field is already blank or zero before transfer to the output area for interpreting. If the same field is to be interpreted more than once, Blank-After should be specified in the last line that specifies interpreting for that field: the fields for interpreting are transferred to the output area in the sequence in which they are specified in Field-Description lines, regardless of print-head number.

Note: Other things being equal, output time is conserved if:

1. On serial punches, punching and interpreting is concentrated at the left end of the card--output speed is inversely correlated with the number of the last column punched or last position printed.
   Often, it is possible to confine interpreting to the left end of the card by card-printing data on several lines concurrently, by utilizing several print heads.
2. Printing on the printer can be confined to the first (leftmost) 100 print positions.
   This is of value only if it can be adhered to for the entire job, so that the RPG Control Card (card H) can be left blank, or punched 100, in cols. 23-25 (see the publications IBM System/360 Report Program Generator for Punch-Card Equipment, Operating Procedures, Form C26-3800).

Entries in End Position in Output Record have already been illustrated in Figures 5E, 5F, 51A, 51B, 52, and 53. Further examples, including specifications for interpreting, appear in Figures 54A and B.

Note: The output storage area is cleared by the program after each pertinent output operation.

Packed Field (P)--Col. 44

A field defined as numeric is stored, at its field-name location, in packed format (see Data Formats, Appendix D). If col. 44 is left blank, numeric data moved to the output core-storage area is unpacked during this transfer. This causes the data to appear in cards and on printed reports in customary form--one digit per column or print position (with the low-order position possibly signed).

If P is entered in col. 44, the (numeric) field is transferred to the output storage area in its packed format--two digits (one per half-byte) per column (or print position), represented by the EBCDIC characters for the particular combinations of two digits. The low-order position contains the EBCDIC character for the low-order digit and sign (which may also be hexadecimal F, for "no sign").

Packed output has a field length slightly larger than half that of an unpacked field. (It is greater than half the unpacked field size because the sign position requires a half-byte, and only complete bytes are permissible.) The formula is:

$$Lp = \frac{n+1+E}{2}, \text{ where}$$

Lp = number of positions in the packed output field

n = field length defined in:

a. input specifications of unpacked input field (Field Location), or

b. calculation specifications (Field Length), or

c. file-extension specifications (Length of Table Entry)

E = 0, if n is odd; or

= 1, if n is even.

When specifying End Position in Output Record, the reduced length of a packed numeric output field should be taken into account.

Packed output is intended as an optional format for punching into cards:

1. On a serial punch it may expedite throughput, if punching can be terminated at a lower column number because the data fits into fewer columns.

2. It may significantly reduce punch time if, as a result of packed output, the data fits into fewer cards.

3. It may reduce subsequent card handling if, as a result of packed output, fewer cards are required.

4. It may speed subsequent input, if the number of cards has been reduced as a result of previous packed output.

(See also Packed--Col. 43, under Input Specifications.)

It should be pointed out, however, that numeric data punched in packed format is difficult to decipher (without a conversion table) by visual inspection of a card, and still awkward to read even with reference to the EBCDIC table (see Appendix D). Sorting on packed-decimal fields also presents special problems.

While it is permitted to specify P for printed numeric output (to the printer or for card document-printing), this is not practical because:

1. Many of the EBCDIC characters that represent the combination of two digits (one per half-byte) have no corresponding graphics in the chain, train, type-bar, or MFCM print mechanism; and

2. It is awkward to relate any graphics that are printed to a particular two digits.

Packed Field (P in col.44) must not be specified:

1. For a constant
2. For an alphameric field
3. If Zero Suppress or an edit word is specified.

Figure 54B includes illustration of the Packed-Field specification.

Constant or Edit Word --Cols. 45-70

An entry in cols. 45-70 respresents a constant if Field Name (cols. 32-37) is blank; it represents an edit word if a Field Name is specified.

Although both items are specified in the same field, their uses are quite distinct. They are therefore treated separately, below.

Constant--Cols. 45-70

If Field Name (cols. 32-37) is left blank, the actual data in the "Constant or Edit Word" field--instead of the contents of a named field--is moved to the output core-storage area for the output-record location specified in cols. 41-43. This is a convenient method of placing into the object program any data that does not change troughout the job, nor from one processing of the program to another. The most common use of constants is for report and report-column headings, and for punching a fixed indentification into output cards.

Any of the 256 EBCDIC characters (including blank) may be specified in this field. A constant is always considered an alphameric literal and must, therefore, be enclosed in apostrophes (card punch-combination 5-8). Col. 45 must contain an apostrophe. The constant itself always begins in col. 46 (even if that column is blank) and ends in the column preceding the next single apostrophe. An apostrophe desired within the constant itself is represented by two successive apostrophes. (For further details on alphameric literals, see Alphameric Literals, under Definition of Terms and under Calculation Specifications.) Of course, numeric data also may be used as a constant, by treating it as an alphameric literal enclosed in apostrophes.

Because only 26 columns are available for a constant in one Field-Description line, and two delimiting apostrophes are required, the maximum length of a constant is 24 positions. Under the Input Specifications, a method is described for reading longer constants in from a card (see Using Input Data Fields for Constant Data--Heading Cards). A longer constant for output can also be simulated by continuing the constant in another Field-Description line (see Figure 54A).

A constant is stored only once by the program (i.e., consumes core-storage space only once), irrespective of the number of Field-Description lines in which it is specified. (Consequently, a constant is blanked for the remainder of the job once it has been transferred to the output storage area by specifications in any Field-Description line in which Blank-After (B in col. 39) is specified.)

Note:

1. Zero Suppress (Z in col. 38) must not be specified in the same line as a constant.

2. Field Name (cols. 32-37) must be blank if a constant is entered in cols. 45-70; otherwise, the constant is instead assumed to be an edit word (see below).

3. Packed Field (P in col. 44) must not be specified for output of a constant.

Figures 54A and B illustrate constants, as well as Packed-Field specifications, End Position in Output Record, Blank-After, and Zero Suppress. The examples were chosen to maximize clarification, and are not necessarily a natural sequence of specifications. (Figures 54A and B should be considered as independent of each other.)

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System/360

Form X24-3352-1 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	Filename	Type (H/D/T)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators	And	And	Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
01	O	PRINT	D			2	Ø1		99									
02	O	OR							OF									
03	O														54		'SALES ANALYSIS BY CUSTOM'	
04	O														ØØ78		'ER AND SALESMAN FOR MONT'	
05	O														Ø82		'H OF'	
06	O											DATE			91			
07	O											PAGE	Z		12Ø			
08	O		D					Ø2	99									
09	O	OR							OF									
10	O														28		'AMOUNT'	
11	O														19		' SLSMN ACCOUNT'	
12	O														46		'12.5'	
13	O														6Ø		'15.Ø'	
14	O		T			1			L1									
15	O											ACCNT	Z		19			
16	O											SALSMN			1Ø			
17	O											AMOUNT		B	3Ø		' , Ø. -'	
18	O											COM12	Z	B	48			
19	O											COM15	Z	B	62			
2Ø	O								NØ5			TABBON	Z		7Ø			

Card Electro Number ____

Figure 54A. Some Examples of Entries for Field Name, PAGE Number, Zero Suppress, Blank After, End Position in Output Record, and Constant

Explanation of Entries in Figure 54A

The file named PRINT is assumed to have been associated with the printer, in the file-description specifications.

Specifications lines 01 and 02 cause the output for lines 03-07 to be performed at detail-output time, or at overflow-output time if indicator OF is on. The form is advanced to the next carriage-control tape punch in channel 1 before output, and is spaced 2 lines afterwards.

Lines 03, 04, and 05 show how a report heading that is 52 positions long can be printed, by specification of constants, as one continuous phrase--even though an individual constant is limited to a maximum of 24 positions. (An alternative, involving an input card, is presented under Input Specifications: Using Input Data Fields for Constant Data--Heading Cards.)

Line 06 indicates how data that changes each time the report is run, and may change at points during the run, can be printed as parts of a constant report heading, on the same print line.

It is assumed that a card, defined as a separate card type, has been read in at the start of the object-program run (and at any additional points of the card deck where the information is to be changed). A field in this card was defined, in the input specifications, as DATE. The DATE field in that card type contains the date--in this example, in the form month-name and year (e.g., SEPT.66). (Again, see Using Input Data Fields for Constant Data--Heading Cards.)

Line 07 causes four-digit consecutive page numbers to be printed at the top of each page, in print positions 117-120. Leading zeros are suppressed, and the zone over the units position is eliminated from the print-out. The number is printed on the first page as bbb1, unless a special starting number was entered (see Input Specifications: Consecutive Numbering) and/or the number was modified or created by calculation specifications.

Lines 08-09 provide for printing a second heading line, two lines below the first heading line (2 in col. 18 of specification line 01), in the same program cycle. Whereas the entries in lines 03-07 provided for report heading and page number, lines 10-13 contain specifications for column headings. After this output, the form is advanced to the next channel-2 punch.

Line 11 illustrates that one constant may contain more than one column heading (SLSMAN and ACCOUNT)--it is merely a matter of convenience and appropriate spacing.

Also shown is the fact that a constant may start with blanks, to align it appropriately.

Lines 10 and 11 show also that fields or constants need not be recorded in the sequence in which the data is to appear in the output record (End Position in Output Record: 28 is in an earlier specification line than 19).

In lines 10, 12, and 13 we chose to use separate lines for each column heading, rather than combining some as in line 11.

Lines 12 and 13 also illustrate that constants, which are alphameric literals, may contain numeric values.

Note, throughout, that Field Name (Cols. 32-37) is blank where a constant is assigned.

Line 14 calls for a group-printed report (printing only when L1 is on), printed at total-output time. Printing begins on each page at channel 2, below the two heading lines.

Lines 15 and 16 again illustrate that fields need not be recorded in the order in which the data is to appear in the output record.

Lines 15, 18, 19, and 20 include the Zero-Suppress specification. Data from each of these four fields is printed without any leading zeros, and any zone in the low-order position is not transferred to the output core-storage area.

The fields for which Z is specified in col. 38 must have been defined as numeric.

Line 17 illustrates formatting of an output field by an edit word (discussed in the next section). It is included here to emphasize that Zero Suppress (Z) must not be designated when an edit word is assigned, and to show that an edit word can expand the size of the field (End Position 30 has been specified to center the field around the heading AMOUNT, which ends in position 28).

The field AMOUNT must have been defined as numeric to permit use of an edit word.

The entry in cols. 45-70 is an edit word--not a constant--because Field Name (cols. 32-37) is not blank.

Lines 17, 18, and 19 show resetting (to blank or zero) of fields, by the Blank-After (B in col. 39) instruction. The field is cleared immediately after the data has been transferred to the output core-storage area. At that time, any indicators assigned to "Zero or Blank" for these fields turn on (this applies to "Zero or Blank" indicators for these fields in Field Indicators, and in Resulting Indicators of arithmetic and TESTZ operations).

If such indicator is used in Output Indicators of a subsequent Field-Description or File-Identification line, it is on--until new input data or calculation results turn it off again.

Note: We have treated Figures 54A and B as independent examples. If the output specifications to the file SUMCARD in Figure 54B were considered a continuation of the output specifications in Figure 54A, Blank-After must not be specified in lines 17, 18, and 19 of Figure 54A: otherwise, only zeros will be punched in SUMCARD from the fields AMOUNT, COM12, and COM15--these fields will have been cleared by Blank-After specifications in Figure 54A, before output to the file SUMCARD.

Line 20 illustrates output from the "hold" area of a table (see Table Look-up Operations, under Calculation Specifications). The data last selected (or as subsequently modified) from the table named TABBON is printed, ending in type position 70.

This item is printed only if indicator 05 is not on.

Note: Throughout, note that the entries in cols. 40-43 are right-aligned, and that leading zeros may be recorded (lines 04 and 05) or omitted.

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System/360

Form X24-3352-1
Printed in U.S.A.

Date ____________

Program ____________

Programmer ____________

Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line (3-5)	Form Type (6)	Filename (7-14)	Type (H/D/T) (15)	Stacker Select (16)	Space Before (17)	Space After (18)	Skip Before (19-20)	Skip After (21-22)	Output Indicators Not (23)	(24-25)	And Not (26)	(27-28)	And Not (29)	(30-31)	Field Name (32-37)	Zero Suppress (Z) (38)	Blank After (B) (39)	End Position in Output Record (40-43)	Packed Field (P) (44)	Constant or Edit Word (45-70)	Sterling Sign Position (71-74)
01	O	COMBFILE	D							45	N	MR									
02	O	O	R	2						45		MR									
03	O																				
04	O	SUMCARD	T							L1											
05	O														ACCNT			7			
06	O														ACCNT			1Ø8			
07	O														AMOUNT			212		' , Ø. CR'	
08	O														AMOUNT	Z	B	164			
09	O														AMOUNT			ØØ17	P		
10	O														SALSMN			Ø13			
11	O														SALSMN	Z		117			
12	O														COM12			21	P		
13	O														COM15			25	P		
14	O														COM12	Z	B	22Ø			
15	O														COM15	Z	B	227			
16	O									N	1Ø					LASTYR			29	P		
17	O									N	1Ø								13Ø		'SALS SUMRY'	
18	O														PAGE			79			
19	O																	8Ø		'9'	
	O																				

Card Electro Number ________

Figure 54B. Continuation of Examples of Entries for Field Name, Zero Suppress, Blank After, End Position in Output Record, Packed Field, and Constant

Explanation of Entries in Figure 54B

Lines 01-02 portray stacker selection, based on the status of indicators, for cards on which no output operation is required. The file is assumed to be defined as a combined file in the file-description specifications.

If the Matching-Record (MR) indicator and indicator 45 are on at detail-output time, the card is to be selected to stacker 2; if MR is off, it is to enter the normal stacker of the relevant I/O device.

Line 04 specifies output at total time, provided the L1 indicator is on, to a file named SUMCARD. Assume that SUMCARD is an output file of blank cards. This, then, represents the standard method of summary-punching, into a blank deck, at the end of each control group, and/or of interpreting the summary cards.

Line 05 causes the contents of the field ACCNT to be punched, with its last (low-order) position in col. 7. Punching--not card document-printing--is called for, because col. 41 is blank (or, it could be 0).

Line 06 causes the same data to be printed on the card, by print head 1 (1 in col. 41), in positions 2-8, in unedited format: leading zeros are printed, and any zone in the low-order position remains.

Lines 07-11 show that the fields need not be recorded in the order in which they are to appear in the output record.

Line 07 causes the contents of the 7-digit AMOUNT field to be printed by print head 2 (2 in col. 41) in positions 2-12.

The data is formatted by an edit word (see next section). The edit-word example is included to emphasize that Zero Suppress (Z) must not be specified when an edit word is assigned, and to show that an edit word can expand the size of the field: the 7-digit AMOUNT field requires 11 positions in the output area, with this particular edit word. Care must be used not to overlap fields unintentionally in the output area when edit words may expand them.

The field AMOUNT must have been defined as numeric to permit use of an edit word.

The entry in cols. 45-70 is an edit word--not a constant--because Field Name (cols. 32-37) is not blank.

Line 08 causes the same field also to be printed by print head 1 in positions 58-64. Leading zeros and any zone in the low-order position are eliminated from the output by the Z in col. 38.

The B in col. 39 causes the field to be cleared to zeros after transfer of the data to the output area. Note that the Blank-After instruction is in the last document-printing specifications line for the field. Although data transfer for punching is in a subsequent line (line 09), the program transfers all data for punching to the output area before the data for card document-printing (within the same File-Identification specifications). If the B were in line 09, the AMOUNT field would contain only zeros at the time of the transfers called for in lines 07 and 08.

Line 09 provides for punching of the AMOUNT-field data into cols. 14-17. The output will be in packed format (P in col. 44).

The field AMOUNT is 7 digit-positions long (as implied by the edit word in line 07). In packed format, it consumes 4 positions (see formula above, under Packed Field). The field must have been defined as numeric for packed output.

Neither an edit word nor Zero Suppress is permitted with Packed Field. In any case, it is not usual to use Zero Suppress for punched data, because leading zeros are normally desired in the card and because the sign over the low-order position must ordinarily be punched for future use (at least, if it is a minus sign--11-overpunch).

Line 10 specifies punching of salesman code (field SALSMN), with low-order position in col. 13 (if we assume a 6-digit field, then into cols. 8-13).

Line 11 causes the contents of the SALSMN field to be printed by print head 1, ending in position 17 (if a 6-digit field, then in positions 12-17). Leading zeros and any zone in the low-order position are eliminated (Z in col. 38) from the print-out. The field must have been defined as numeric because Zero Suppress is used.

Note: ACCNT and SALSMN are not punched in packed format, because it is assumed that subsequent reports may require group control (Control Level) on these fields, which is not possible on packed fields. (see Packed, under Input Specifications, for the possibility of defining the same field a second time, as alphameric, in order to control on a packed field. However, printing the group-identifying data then presents a problem.)

Lines 12 and 13 provide for punching the contents of the fields COM12 and COM15 in packed format, in cols. 18-21 and 22-25, respectively. We have assumed that they are 6-digit fields (representing commissions at 12.5 and 15.0 percent, respectively); they each, therefore, require 4 positions in packed format. The fields must have been defined as numeric for packed output.

Lines 14 and 15 specify printing of the same 6-digit fields, by print head 2, in positions 15-20 and 22-27, respectively--in unpacked format.

Leading zeros and any zone in the low-order position of each field are eliminated (Z in col. 38) from the printout. The fields must have been defined as numeric.

The fields are reset to zeros (B in col. 39) after transfer to the output area for document-printing. This is the correct place for the Blank-After instruction, because transfer to the output area for punching (see lines 12 and 13) always takes place first.

Also illustrated here is the grouping--rather than alternating--of punching and document-printing instructions (note lines 12 and 13 for punching of two fields, before the document-printing instruction for either field). Grouping and alternating are equally acceptable.

Line 16 specifies the punching of data from a field (LASTYR; say, comparative sales figure from previous year), in packed format (7-digit amount field packed into 4 positions), without any document-printing--just to make it clear that a field may be punched but not printed, or vice versa, or both.

The output is not performed if indicator 10 is on at that time. Even if indicator 10 is assigned to "Zero or Blank" in Field Indicators or calculation Resulting Indicators (arithmetic or TESTZ operation) of any of the fields AMOUNT, COM12, or COM15, this will not turn on the indicator as a result of the Blank-After instruction for these fields in time to interfere with output of LASTYR: all data for punching is transferred to the output area before data for document-printing. Therefore, indicator 10 does not turn on until after LASTYR has been transferred.

Blank-After is not specified: we have assumed that the total in this field does not represent an accumulation of data from detail cards, but is read in from a single summary card in each control group. Clearing the field is then unnecessary.

Line 17 shows how a constant can be document-printed: the phrase SALS SUMRY is to be printed by print head 1 in positions 21-30.

The output is not performed if indicator 10 is on at that time. If indicator 10 is assigned to "Zero or Blank" in Field Indicators or calculation Resulting Indicators (arithmetic or TESTZ operation) of any of the fields AMOUNT, COM12, or COM15, the output of this constant will always be suppressed as a result of the Blank-After instruction for those fields: the indicator will always be on before transfer of the constant in line 17 to the output area.

Of course, Blank-After is not specified: it would clear the constant to blank, and it would be lost after output to the first summary card.

Note that Field Name (cols. 32-37) is bank when a constant applies.

Line 18 shows how the page-numbering feature can be employed equally well for consecutive-numbering of cards. The cards are punched with consecutive numbers in cols. 76-79--starting with 0001 (12-overpunch in col. 79), unless another starting number was provided through an input card or a calculation specification.

Z may be specified in col. 38 to eliminate the 12-overpunch, but leading zeros are then also replaced by blanks. (If an edit word is used for formatting, at least one leading zero is lost.)

If Figure 54B is considered a continuation of Figure 54A, PAGE cannot be used here; otherwise, the contents of the field PAGE are incremented each time they are printed on the printer (the file named PRINT) and each time they are punched into the file SUMCARD.

Line 19 illustrates the punching of a constant: the summary cards are identified by 9 in col. 80.

It also indicates that a constant (an alphameric literal) may consist of numeric data.

Note that Field Name is blank when a constant applies.

Note: Throughout, note that the entries in cols. 40-43 are right-aligned, and that leading zeros may be recorded (lines 09 and 10) or omitted.

Edit Word--Cols. 45-70

Purpose of Edit Word. An edit word permits formatting of the output from numeric fields.

Edit words provide for:

1. Suppression of leading (non-significant) zeros to a predetermined position of the field;
2. Punctuation with decimal point and commas;
3. Fixed or floating dollar sign;
4. Asterisk protection;
5. Identification of the field by any EBCDIC characters;
6. Elimination of any zone from the output representation of the low-order position;
7. Identification of negative totals by CR symbol or minus sign;
8. Insertion of any constant data within or following the field;
9. Insertion of spaces in the field.

If no edit word is specified in a Field-Description line, the output from the field corresponds to the contents of the field: the digits, including leading zeros, are printed or punched in adjacent positions without spaces, and the character in the low-order position is zoned if the field was zoned (see also point 4 under Rules for Forming Edit Words, below).

The punch combination that corresponds to each zoned EBCDIC character can be ascertained from the EBCDIC table (Appendix D). For standard digits and zones:

A 12-position hole is punched over the low-order digit if the field is signed plus (C zone);
An 11-position hole is punched over the low-order digit if the field is signed minus (D zone);
Only the digit is punched in the low-order column if the field is unsigned (F zone).

The same EBCDIC characters apply to printed output. However:

1. Not all EBCDIC characters are represented on the print medium (chain, train, typebar, or MFCM print mechanism);

2. The number of different graphics available varies with the model and the type of chain, train, or typebar.

3. The user may have had non-standard graphics installed; and

4. In some cases, an EBCDIC character for which the printing device is not designed may cause printing of another character.

Note particularly that a signed zero (a frequent normal condition for the low-order position of a signed numeric field) may be printed as only +(for 0) or -(for 0), or the position may be left blank entirely--depending on the particular printing device.

Numeric printed output, unless it is known to be unsigned, is therefore usually edited either by an edit word or by the Zero-Suppress instruction (see above). (Removal of the Zone is also possible by a calculation specification--see Calculation Specifications: Move Operation.)

General Guidelines Pertaining to Edit Words

If a numeric output field is to be edited (by use of an edit word), the edit word is placed in the "Constant or Edit Word" field in the same Field-Description line.

An edit word is an alphameric literal (see Alphameric Literals, under Definition of Terms). As such, it is enclosed in single apostrophes (card punch-combination 5-8). Col. 45 must contain the initial apostrophe, with the edit word itself starting in col. 46. An apostrophe follows the end of the edit word (see Alphameric Literals, under Definition of Terms and under Calculation Specifications for apostrophes within a literal). An edit word is, therefore, limited to a maximum of 24 positions. Any of the 256 EBCDIC characters (including blank) are valid within an edit word; but some characters, in certain positions, have a unique significance in edit words.

The fact that Field Name (cols. 32-37) is not blank (i.e., contains the name of a field) distinguishes an edit word from a constant (see Constant, above).

No matter how often a particular edit word is used (i.e., in how many Field-Description lines it appears), it is stored by the program only once (i.e., it consumes only a single core storage area).

A Blank-After instruction (B in col. 39) does not destroy the edit word (in contrast to a constant, which is then blanked out).

An edit word may be used to format numeric fields for:

1. Printing on the printer;

2. Document-printing (interpreting) on punch cards;

3. Punching into cards.

   The latter use is not common, because it is not usual to insert punctuation, symbols, constants, spaces, or separate sign positions--or to eliminate leading zeros--in punched numeric fields.

The edit word applies to whatever output is specified in that Field-Description line. Editing output from a field in one Field-Description line has no effect on subsequent output from the same field (in contrast to a Blank-After instruction).

An edit word must not be assigned if:

1. The field is alphameric (i.e., if it has not been defined as numeric);

2. Packed Field (P in col. 44) is specified in the same Field-Description line;

3. Zero Suppress (Z in col. 38) is specified in the same line;

4. The field does not consist only of valid digits (0-9), except for a zone permitted in the low-order position (i.e., the digit portions of the field must not contain hexadecimal A-F).

Where an edit word is assigned, suppression of leading zeros in at least one position--the leftmost position--of the field cannot be prevented. (See Programming Tips for circumvention.)

When an edit word is assigned, any zone over the low-order position of the data is removed from that position in the output. However, a negative sign (hexadecimal D--see EBCDIC table, Appendix D) can be represented as CR or minus (-) to the right of the digits from the field; a plus sign (or any zone other than hexadecimal D) is eliminated completely from the output.

The same edit word cannot be assigned to fields of varying lengths; i.e., all the fields with which a given edit word is used must have the same size.

The number of positions allowed in an edit word for digits from the data field must not be less than the number of positions in the data field--it should be exactly the same number.

Note: While it is permitted to make the edit word larger than necessary to accommodate all positions of the field, this gains the user nothing:

1. The data field is left-aligned within the edit word; therefore, there is still no way to prevent elimination of at least one leading zero.

2. No core-storage space can be saved by making an edit word large enough to accommodate the largest of several fields, since all fields with which one edit word is used must be of equal size.

Edit Word Segments

An edit word is composed of one, two, or three constituent segments:

1. The body--required

2. The status portion--optional

3. The expansion--optional

Figure 55 depicts the segments of edit words.

The body of an edit word governs (1) the transfer of the digits in the data field to the output record, (2) the termination of zero suppression or asterisk protection, (3) punctuation, and (4) the insertion of other constants. The body portion begins at the leftmost position of the edit word (i.e., col. 46), and contains the same number of blank positions as there are digit positions in the data field, plus positions for any desired constants and dollar sign. One zero or one asterisk, if it appears in the body portion, is counted as the equivalent of a blank position. These blank positions (including a maximum of one zero or asterisk) are replaced by digits from the corresponding positions of the data field, or--where there are no significant digits to the left of the zero-suppression limit--the output appears as blank (or asterisks).

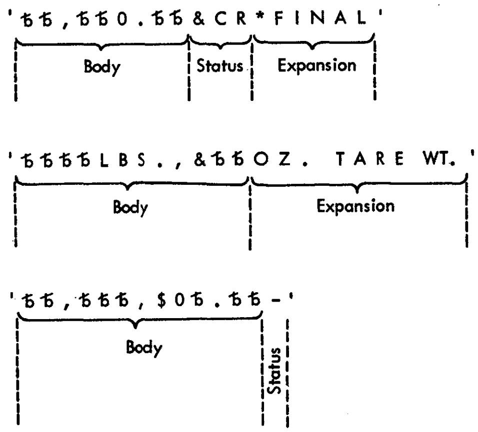


Figure 55. Symbolic Portrayal of the Segments of an Edit Word

The program determines the end of the body segment by counting the blank positions (including the leftmost 0 or *) from left to right until the point is reached where the count is equal to the digit capacity of the data field. The point where that count is satisfied terminates the body of the edit word.

The status portion extends to the right from the end of the body segment, through the first appearance of the two letters CR (credit) or the symbol - (minus). The function of the status portion of an edit word is the identification, by display (or punching) of the CR or -, of a negative quantity. Any EBCDIC characters (including blank) may precede the sign symbol (CR or -) within the status portion. When the field is negative (D zone), the entire status portion (except an ampersand) appears in the output as specified in the edit word; when the field is not negative, the entire status portion appears as blank. (An ampersand in the status portion always appears as blank in the output record.)

Edit words that contain no CR or - (minus) symbol to the right of the body segment have no status portion. The terminal apostrophe is then placed to the right of the immediate right of the body, unless an expansion segment follows.

The *expansion*, if any, consists of any positions to the right of the status portion (or, to the right of the body, if there is no status portion) through the end of the edit word (i.e., to the terminal apostrophe). If the terminal apostrophe follows the body or the status portion, there is no expansion section. Any EBCDIC characters (including blank) may be placed in the expansion section; they will appear unchanged in the rightmost positions of the output area assigned to the field (see cols. 41-43), regardless of whether anything from the body and status portions appears in the output.

Rules for Forming Edit Words

Note: Although the examples in Figure 56 are explained in the next subsection, reference to Figure 56 while reading this subsection will help.

1. Delimiting the edit word.

   The complete edit word must be enclosed in single apostrophes (card punch-combination 5-8).

2. Determining number of positions required in edit word for data-field digits.

   The number of blank positions (including, if present, one position with zero or asterisk) in the *body* portion of the edit word must be no less than (and is normally equal to) the number of digit positions in the data field. Call this value = B. This is the only mandatory portion of an edit word.

   These blanks (and, if present, the first zero or asterisk) in the body are replaced by the significant digits from the corresponding positions of the data field specified in Field Name. Non-significant zeros may be represented in the output record by zeros, blanks, asterisks, or dollar symbol (see Zero Suppression, Asterisk Protection, and Floating Dollar Sign: 5, 6, 8, below).

3. Determining total length of edit word.

   The body of an edit word may contain constants (any EBCDIC characters except blanks) besides the blanks and single zero or asterisk counted in the value B (item 2, above), and a fixed or floating dollar sign. Call the number of these constants = C.

   The status portion may contain any EBCDIC characters (including blanks) preceding the sign symbol. Call the total number of positions in the status portion = S.

   Any EBCDIC characters (including blanks) may make up an expansion section to the right of the status portion (or the right of the body, if there is no status portion)--to the terminal apostrophe. Call the number of positions in the expansion = E.

   The total number of positions in the edit word must then be = B+C+S+E ≤ 24.

   Note: Because the edit word can be considerably longer than the data field, the user must remember:

   a. That End Position in Output Record (cols. 41-43) refers to the rightmost position of the edit word; and

   b. To allow enough room for successive output fields so that the left part of one field is not overlaid over the right part of a previous output field.

4. Zone elimination.

   The assigment of an edit word in itself removes any zone from the low-order digit in the output record. (See Status segment, item 10 below.)

   A numeric field is zoned in its low-order position if:

   a. It was zoned in that position when read in; or

   b. A Field Indicator was assigned to it in the input specifications. (This also converts minus zero to plus zero.); or

   c. It was a Result Field in an arithmetic operation; or

   d. A zone was moved to that position in calculation specifications; or

   e. It was cleared by a Blank-After specification; or

   f. The input field was blank.

5. Zero suppression.

   If no zero (or asterisk--see below) appears in the body of the edit word, all leading (non-significant) zeros are suppressed, in all positions in the body of the edit word, up to the point of the first significant digit. These positions--including those for constants and punctuation in the body to the left of the first significant digit--are blank in the output record. If the data field contains only zeros (apart from a possible sign in the low-order position), and there is no zero or asterisk in the body of the edit word, the entire area assigned to the body in the output record will be blank. (Exception: fixed dollar sign --see 7, below.)

The leftmost zero (or asterisk--see below) entered in the body of an edit word stops suppression of leading zeros _beyond_ that position (the position in the body that contains the suppression-limiting zero is included in zero suppression). _Through_ the point of the first zero in the body of the edit word, or _to_ the point of the first significant data digit--whichever occurs first (further left)--all positions in the body of the edit word, including those containing constants, appear as blank in the output record. (Exception: dollar sign--see 7 and 8, below.)

From the point of the first significant digit in the data field, or the position to the right of the suppression-limiting zero--whichever occurs first--data digits replace the blanks in the corresponding positions of the edit-word body and any constants (including punctuation) are retained for output. The suppression-limiting zero itself is replaced by a significant data digit in the corresponding position of the field; if only leading (non-significant) zeros exist through that position in the data field, the suppression-limiting zero, too, is blanked.

Any zero (or asterisk--see below), in the body of the edit word, to the right of the first zero (or asterisk) is considered a constant and always appears in the output (i.e., the leftmost zero or asterisk terminates zero elimination)--see item 9, below.

Since zero suppression is operative through the position containing the suppression-limiting zero:

(a) It is not possible to retain all leading zeros when an edit word is assigned: even if zero is entered in the leftmost position of the edit word, the leftmost leading zero is still replaced by blank. (But see _Programming Tips_ for a way around this.)

(b) It is not possible, by entries in an edit word, to include a constant (with the exception of a dollar sign--see below) to the left of the beginning of the field--even if the edit word were made larger than the field. For example:

A two-digit field containing 65 cannot be output as .65 by writing the body of the edit word as '.ƀƀ'. Regardless of whether the edit word is written as '.ƀƀ' or 'ƀƀ', the output will appear as 65.

Similarly, .05 for a two-digit field will appear in the output as ƀ5.

The problem is solved by (1) specifying the decimal point as a constant for the preceding position in the output record and (2) not editing the field at all (see _Programming Tips_). If desired, calculation specifications can test for 00 in the field, to allow suppression of the decimal point and the field output in that situation.

(c) With two exceptions, it makes no difference--if leading zeros for the entire field are to be suppressed--whether a zero is entered in the rightmost position of the body, or not at all.

The exceptions are (1) floating dollar sign, and (2) printing of the status portion when the data field consists of all zeros, with a minus sign. Both are explained below.

6. Asterisk protection.

The leftmost asterisk in the body of an edit word--provided there is no zero in the body to its left--has the same effect on zero suppression as a leftmost zero (see 5, above).

However, instead of being blank, all positions in the body (including those containing constants) from the extreme left through the first asterisk position, or to the point of the first significant data digit--whichever occurs first--are filled with asterisks for output.

A significant digit (including a non-leading zero) in the corresponding position of the data field replaces the asterisk in the edit word. From the point of the first significant digit or the asterisk in the edit-word body, any constants in the body are retained for output.

Any asterisk (or zero), in the body of the edit word, to the right of the first asterisk (or zero), is considered a constant and always appears in the output (see item 9, below).

One leading zero is suppressed, even if the asterisk is in the extreme left position of the body.

Note. Neither a fixed dollar sign nor a floating dollar sign (see 7 and 8, below) can be specified in an edit word with asterisk protection.

7. Fixed dollar sign.

A dollar sign ($) placed in the leftmost position of an edit word appears in that position of the output, regardless of suppression of leading zeros. The body of the edit word must be enlarged to provide the extra position for the dollar sign.

If only one leading zero is to be suppressed (and no & symbols precede the suppression-limiting zero)--i.e., the suppression-limiting 0 is in the leftmost digit position adjacent to the dollar sign--the $ becomes a floating (not fixed) dollar sign (see 8, below).

A fixed (or floating) dollar sign cannot be used in an edit word with asterisk protection (see 6, above). Specifying the dollar sign instead as a preceding contiguous constant field is a simple solution.

8. Floating dollar sign.

A dollar sign ($) placed to the immediate left of (i.e., next to) the zero that terminates zero suppression appears in the output record either (1) in the position occupied by that zero in the body of the edit word, or (2) the position immediately preceding the first (high-order) significant digit--whichever is farther left. The body of the edit word must be enlarged to provide the extra position for the dollar sign.

Regardless of where the floating dollar sign is placed in the body of the edit word, the location of constants in the body (such as punctuating commas, for instance) should not be shifted to compensate for the dollar-sign position: the extra position provided because of the floating dollar sign remains at the extreme left of the body (see Figure 56).

A dollar sign in the body of the edit word elsewhere than in the leftmost position or to the immediate left of the first zero is neither a fixed nor floating dollar sign: it is a constant (see 9, below).

A floating (or fixed) dollar sign cannot be used in an edit word with asterisk protection (see 6, above).

Note:
(a) If the dollar sign is placed in the leftmost position of the edit-word body, it is normally a fixed dollar sign (see 7, above). However, if the zero that terminates zero suppression occupies the next position--i.e., only the minimum of one leading zero is to be suppressed (and no & symbols intervene)-- the $ becomes a floating dollar sign: it then appears in the output record either (1) in the position that corresponds to the leftmost leading zero, when the data begins with zero(s) or (2) to the immmediate left of the high-order position of the data field, when the data begins with a significant digit--i.e., it floats between two positions.
(b) If the zero to end zero suppression is placed in the low-order position of the body of the edit word (i.e., all leading zeros are suppressed--as though no zero appeared in the body), the floating dollar symbol (if one is specified) appears in the output record in the low-order position of the body when the entire data field is zero. This programming approach is meaningful only when a floating dollar sign is needed, but all leading zeros are to be suppressed. (See also item 10 below, for status portion with all-zero data field.)
(c) Because a floating dollar sign must be specified in the edit word contiguous to (and to the left of) the zero-suppression-limiting zero, a floating dollar sign cannot be placed ahead of punctuation (e.g., ahead of a comma), to appear in the output to the immediate right of the punctuation if there are no higher-order significant digits. For example, in the edit word 'ƀƀ$,0ƀƀ',the dollar sign is merely a constant (see 9, below) --not a floating dollar sign--because it is not contiguous to the zero. It is not possible to assign a floating dollar sign to appear in the output in place of the zero in the edit word in this situation (where the zero follows a comma or other constant).

9. Constants within the body of an edit word.

With the exceptions enumerated below, any EBCDIC characters within the body of an edit word are treated as constants: they appear in the output exactly as specified in the edit word, and in the corresponding positions.

However, they appear in the output only when they are to the right of the first significant data-digit, or to the right of the leftmost zero or asterisk in the body of the edit word--whichever occurs first. When neither a significant data-digit nor a suppression-limiting zero or asterisk occurs to the left of the constants, they are replaced in the output either (1) by blanks, if asterisk protection is not specified, or (2) by asterisks, if asterisk protection is specified.

The constants most commonly employed in the body of an edit word are a decimal point and commas in amount or quantity fields (or, a decimal comma and periods, in European notation). However, the program does not perform decimal alignment between edit words and data fields: the programmer must place the decimal point in the edit word into the appropriate relative position where he wishes it to appear in the output.

Exceptions:
(a) A dollar sign in the appropriate position for a fixed dollar sign (see 7, above) or floating dollar sign (see 8, above) is treated as described above. However, in any other location, it is treated as a constant.
(b) The leftmost zero or asterisk (one or the other only) is treated as described above (items 5 and 6). However, in any other position, a zero or asterisk is treated as a constant.
(c) The use of blank spaces in the body of an edit word is confined to:
    (i) Output positions that correspond to digit positions in the data field; and
    (ii) A leftmost blank to compensate for the space taken up by a floating dollar sign.

Therefore, spaces between data digits or constants cannot be created by leaving positions blank in the body of the edit word. However, an ampersand (&) in the body of the edit word provides a corresponding blank space in the output representation. An ampersand itself cannot be reproduced in the output representation of the body of the edit word--it is not treated as a constant.

10. Status segment.

The status portion of an edit word is intended for identification (by CR or minus symbol) of negative values.

As described under <u>Edit Word Segments</u> (above), the body of the edit word terminates with the last blank (or replaceable zero or asterisk) position (counted from left to right) required to accommodate all digits of the output field. If the consecutive letters CR, or a minus sign (-), appear in the edit word to the right of the end of the body, the positions from the end of the body through the (first) CR or minus symbol form the status segment of the edit word.

Any of the 256 EBCDIC characters (including blank) may precede the CR or minus symbol in the status segment. If the contents of the data field are negative, the entire status portion appears in the output record exactly as it appears in the edit word--except: an ampersand (&) in the status portion is replaced by a blank space in the output record. (Thus, either a blank or an ampersand in the status portion appears as blank in the output record.) When the contents of the data field are not negative, the entire status portion of the output record is blank.

When the data-field contents are minus zero (possibly if read in as such, or through a move operation), and all leading zeros are to be suppressed, the programmer has two choices:

(a) If no 0 or * is placed into the low-order position of the body of the edit word, the status portion is blank in the output record.

(b) If 0 or * is placed into the low-order position of the body of the edit word, the status portion appears in the output record as specified in the edit word.

If there is no CR or minus symbol in the edit word to the right of the body, there is no status portion. Anything to the right of the body is then part of the expansion (see 11, below). However, negative amounts always appear in the output record as true figures (not complements): if there is no status segment, they merely appear without a sign symbol.

11. Expansion segment.

Any positions to the right of the status segment--or, if there is no status segment, any positions to the right of the body--up to the closing apostrophe, make up the expansion segment of the edit word.

Any of the 256 EBCDIC characters (including blank) entered in the expansion segment appear identically in the corresponding positions of the output record. (An ampersand in the expansion segment also appears as such in the output record.) Be careful not to use the characters CR or -(minus) in an intended expansion segment which is not preceded by a status portion--otherwise the expansion portion through the position occupied by these characters becomes a status portion.

If the terminal apostrophe follows the body or status segment of the edit word, there is no expansion segment.

Figure 56 illustrates edit words. All examples assume that col. 38 (Zero Suppress) is blank (as it must be whenever an edit word is assigned). The symbol ƀ has been used extensively to represent a blank space in the output record, to avoid any possible confusion concerning the number of blank positions. (Zeros have not been slashed where no confusion with the letter O is likely, in order not to clutter up the presentation.)

A vertical dotted line has been inserted in the output representation (it does not, of course, appear in the output record) to clarify for the reader the end of the body and status segments.

Examples labelled A-J are sample edit words for some of the most frequently desired output formats. The numbered examples that follow this first group represent an attempt to illustrate every editing situation which might raise a question in the programmer's mind. Points to be especially noted in each example are itemized below; but the user is assumed to have read the foregoing section on edit words.

Points to Note in Figure 56

(Reference letters and numbers refer to "Example No." in the figure. The lettered items are explained in somewhat greater detail than the numbered ones. For the latter, only the non-routine points are emphasized. All comments assume prior reading of the entire section <u>Edit Word</u>.)

A. Normal method of editing a ten-digit dollars-and-cents field. Decimal point between dollars and cents; commas offset each three positions in dollar area. A space follows the body (in the status portion, either a space or ampersand appears in the output as a space). The status portion (& CR) appears in the output as specified (except that ƀ replaces &) when the data is negative; otherwise the positions are all blank. The expansion, here consisting of an asterisk, always appears in the output record as specified.

Since zero elimination is not terminated until the unit-dollars position (0 in edit word just left of decimal point), leading zeros and constants (e.g., commas) are replaced by blanks until a significant digit is encountered, or through the zero position in the edit word. (The edit-word 0 itself is replaced by any significant digit in the corresponding data-field position.) The decimal point, and data to its right, therefore always appears in the output record. Notice that, since zero elimination proceeds through the position of the 0 in the edit word, that 0, too, is replaced by blank.

B. Illustrates punctuating an eight-digit quantity field with commas. Leading zeros and constants (e.g., commas) are replaced by blanks through the edit-word 0 position (the next-to-last position in the body). Therefore, if the entire data field is zero, a zero appears in the output record only in the low-order position.

The status portion, consisting here only of a minus sign, appears in the output record if the data is negative; otherwise, it is replaced by blank. The expansion, which is any data that follows the status portion, or--if there is no status portion--follows the body, always appears in the output record as it is specified in the edit word (regardless of whether the status portion appears as specified or as blanks).

C. Again, normal punctuating of a ten-digit amount field. By placing the 0 in the body in the ten-dollar position, leading zeros and constants are retained starting with the unit-dollars position.

By placing a dollar sign to the immediate left of the leftmost zero in the edit-word body, it becomes a floating dollar sign: it always appears in the output to the immediate left of the first digit or retained constant--i.e., ahead of the leftmost significant digit, the leftmost retained leading zero, or the leftmost retained constant. Note that an extra position must be allowed for the floating dollar sign at the extreme left of the edit word--not at the location where it is placed: where it is placed, it is replaced by a data digit or a blank (or remains, if the first output-field digit happens to fall to its immediate right).

The status portion (minus sign) appears thus if the data is negative; otherwise, it is replaced by blank. The expansion (asterisk) always appears identically in the output record.

D. Similar to C, except zero elimination is specified up to (but not including) the decimal point, CR is used as the symbol for a negative value, and the expansion segment consists of two asterisks.

In the example, the dollar symbol has "floated" to the left, to precede the first significant digit--which occurred before the zero-suppression termination point. If the data were all zero, the output would appear as $.00ƀƀ**. Note, again, the extra position at the left of the edit word to compensate for the dollar sign. Notice also that the zero-suppression-termination zero in the edit word is replaced in the output by the actual data digit (8) in that position.

E. Similar to D, but there is no status or expansion segment. Also, because the dollar sign is placed in the extreme left position of the edit word, and is not followed immediately by a suppression-terminating zero, it is a fixed dollar sign. It then always appears in the output in that position.

F. This illustrates that a space can be left in the output record between a fixed dollar sign and the first digit, even when the entire field contains significant digits. An ampersand in the body is not replaced by a data digit, and becomes a blank in the output record (except when the ampersand is in an asterisk-protection area).

The status portion consists of ƀ-. The minus appears in the output because the data is negative. In the status segment, either ampersand or blank may be used to represent a blank space in the output record (in example A, an ampersand was used). The program determines the end of the body of the edit word by counting, from the left, the number of positions provided (blanks plus a possible 0 or *) for data digits. Further blanks then belong to the status segment, or to expansion if there is no status segment (no CR or - to the right of the body). Thus, the blank here preceding the minus sign is part of the status segment.

The expansion segment consists of ƀGROSS, because it always begins immediately after the minus sign or CR of the status segment--or after the body segment, if there is no status portion. The contents of the expansion segment always appear identically in the output record, irrespective of the sign of the data (i.e., regardless of whether the status appears in the output as specified or as blanks).

G. Zero elimination is not suppressed at all; data in the output record therefore begins with the first significant digit. If the whole data field were zero, the entire edit word--including the sign, even for negative zero--would be replaced by blanks in the output record (see I, below).

H. Zero elimination is not suppressed at all (it always proceeds through the position of the suppression-terminating zero or asterisk in the edit word). However, because a suppression-terminating zero is entered, the status portion will appear in the output as specified (ƀCR) when the data field contains minus zero. This is the essential difference between H and I (below).

I. The status portion (CR) appears as blank in the output record when the data field consists of minus zero, because no suppression-terminating zero (or asterisk) appears in the body of the edit word (compare with H, above).

The expansion segment (*) always appears in the output record as given in the edit word.

Edit Word	Example No.	Source Data	Appears in Output Record as:
`'  ,   ,  0.  &CR*'`	A	0000000005 -	ƀƀƀƀƀƀƀƀƀƀ.05¦ƀCR¦*
`'  ,   , 0 -*ON HAND'`	B	00000000	ƀƀƀƀƀƀƀƀƀ0¦ƀ¦*ONƀHAND
`'   ,   ,$0 .  -*'`	C	0000000005 +	ƀƀƀƀƀƀƀƀƀ$0.05¦ƀ*
`'   ,   , $0.  CR**'`	D	0034567890 -	ƀƀƀ$345,678.90¦CR¦**
`'$  ,   ,  0.  '`	E	0000000000	$ƀƀƀƀƀƀƀƀƀƀ.00
`'$&  ,   ,  0 .   - GROSS'`	F	1234567890 -	$ƀ12,345,678.90¦ƀ-¦ƀGROSS
`'   ,   ,   .  -'`	G	0000000123 -	ƀƀƀƀƀƀƀƀƀƀ1.23¦-
`'   ,   ,   . 0 CR*'`	H	0000000000 -	ƀƀƀƀƀƀƀƀƀƀƀƀƀƀ¦ƀCR¦*
`'  ,   ,   .  CR*'`	I	0000000000 -	ƀƀƀƀƀƀƀƀƀƀƀƀƀ¦ƀƀ¦*
`'  ,   , * .   -'`	J	0000135792	*****1,357.92¦ƀ
No Edit Word	1	0000135678	0000135678
	2	0000135678 +	000013567H
	3	0000135678 -	000013567Q
`'          '`	4	0000000000	ƀƀƀƀƀƀƀƀƀƀ
`'          '`	5	0000135678 +	ƀƀƀ135678
`'          '`	6	0000135678 -	ƀƀƀ135678
`'         0'`	7	0000135678 -	ƀƀƀ135678
`'0         '`	8	0000135678 +	ƀ000135678
`'           CR*NET'`	9	0000135678 +	ƀƀƀ135678¦ƀƀƀ¦*NET
`'           CR*NET'`	10	0000135678 -	ƀƀƀ135678¦ƀCR¦*NET
`'          &- *NET'`	11	0000135678 -	ƀƀƀ135678¦ƀ-¦ƀ*NET
`'          *NET&CR&*'`	12	0000135678	ƀƀƀ135678¦ƀƀƀƀƀƀƀ¦&*
`'          *NET&CR&*'`	13	0000135678 -	ƀƀƀ135678¦*NETƀCR¦&*
`'          * PROFIT'`	14	0000135678	ƀƀƀ135678¦*ƀPROFIT
`'$          &- NET'`	15	0000135678 +	$ƀƀƀ135678¦ƀƀ¦ƀNET
`'$          &- NET'`	16	0000135678 -	$ƀƀƀ135678¦ƀ-¦ƀNET
`'$0         - NET'`	17	0000135678	ƀ$000135678¦ƀ¦ƀ¦NET
`'       $0  &CR*'`	18	0000135678 -	ƀƀƀ$135678¦~~ƀCR~~¦*
`'       $0  &CR*'`	19	1234567809 -	$1234567809¦ƀCR¦*
`'          &- TOTAL'`	20	0000000000 -	ƀƀƀƀƀƀƀƀƀƀ¦ƀƀ¦ƀTOTAL
`'         0&- TOTAL'`	21	0000000000 -	ƀƀƀƀƀƀƀƀƀƀ¦ƀ-¦ƀTOTAL
`'$          &CR*'`	22	0000000000 -	$ƀƀƀƀƀƀƀƀƀƀ¦ƀƀƀ¦*
`'$         0&CR*'`	23	0000000000 -	$ƀƀƀƀƀƀƀƀƀƀ¦ƀCR¦*
`'$       0   CR GROSS'`	24	0000000000 +	$ƀƀƀƀƀƀƀƀ00¦ƀƀƀ¦ƀGROSS
`'       $0   CR GROSS'`	25	0000000000 -	ƀƀƀƀƀƀƀƀ$00¦ƀCR¦ƀGROSS
`'         $0-'`	26	0000000000 -	ƀƀƀƀƀƀƀƀƀƀ$¦-
`'         *&CR'`	27	0000000000 -	**********¦ƀCR
`'       *  &CR'`	28	0000000000 -	********00¦ƀCR
`'*         '`	29	0000135678 -	*000135678
`'*         '`	30	1234567890 +	1234567890
`'         *'`	31	0000135678 -	****135678
`'  ,   ,   .  &CR* NET'`	32	0000135678 -	ƀƀƀƀƀ1,356.78¦ƀCR¦*ƀNET
`'  ,   ,   .  &CR*-NET'`	33	0000135678	ƀƀƀƀƀ1,356.78¦ƀƀƀ¦*-NET
`'$&0 ,   ,   .   NET'`	34	0000135678 +	$ƀƀ0,001,356.78¦ƀNET
`'   ,   ,$0 .   NET'`	35	0000000005	ƀƀƀƀƀƀƀƀƀ$0.05¦ƀNET
`'   ,   , $0.  '`	36	0000000005	ƀƀƀƀƀƀƀƀƀƀ$.05
`'   ,   , $0.  -'`	37	1234567890 -	$12,345,678.90¦-
`'   ,   , $0.  CR'`	38	0001356789 -	ƀƀƀƀ$13,567.89¦CR
`'  ,   ,  *.  *CR**'`	39	0000135678 +	*****1,356.78¦ƀƀƀ¦**
`'   ,  0.  &CR*'`	40	00000000 -	ƀƀƀƀƀƀƀ.00¦ƀCR¦*

Figure 56. Samples of Edit Words, Part 1 of 2

Edit Word	Example No.	Source Data	Appears in Output Record as:
`'  ,   ,  *.  -'`	41	0000000000 -	**********.00-
`'   ,-$0,   .  -SALES'`	42	0000001234	b̸b̸b̸b̸b̸b̸$,012.34b̸SALES
`'$&  ,   ,  0.  CR*'`	43	1234567890 -	$b̸12,345,678.90CR*
`'  ,   ,   ,   -OLD BALNCE'`	44	1234567890 -	1,234,567,890-OLDb̸BALNCE
`'  ,   ,   , 0 -OLD BALNCE'`	45	0000000000 +	b̸b̸b̸b̸b̸b̸b̸b̸b̸b̸b̸b̸0b̸OLDb̸BALNCE
`'  ,   ,   DOLLARS  CENTS'`	46	0000135678	b̸b̸b̸b̸b̸1,356DOLLARS78CENTS
`'  ,   DOLLARS  CENTS'`	47	000000 +	b̸b̸b̸b̸b̸b̸b̸b̸b̸b̸b̸b̸b̸b̸CENTS
`'  , 0 DOLLARS  CENTS CR'`	48	000000	b̸b̸b̸b̸0DOLLARS00b̸b̸b̸b̸b̸b̸b̸b̸
`'   0 LBS.&  OZ.TARE -'`	49	000002 +	b̸b̸b̸0LBS.b̸02b̸b̸b̸b̸b̸b̸b̸b̸b̸
`'    0LBS.  OZ.TARE'`	50	000002 -	b̸b̸b̸b̸LBS.02OZ.TAREb̸-
`'0  -  -    '`	51	095140036	b̸95-14-0036
`'0 HRS.  MINS. O''CLOCK'`	52	0042	b̸0HRS.42MINS.b̸O'CLOCK
`'  , 0.  '`	53	000000	b̸b̸b̸b̸b̸.00
`'  ,   .0 '`	54	000000	b̸b̸b̸b̸b̸b̸b̸0
`'  ,  $,0  .  '`	55	00123456	b̸b̸b̸1$,234,56
`'  ,   , 0 *  '`	56	0000000000	b̸b̸b̸b̸b̸b̸b̸b̸b̸0*00
`'0,   ,0  '`	57	001234	b̸,012,034
`'  ,  *,   *  '`	58	0000001234	*******,012*34
`'& ,* 0,   '`	59	013579	***130,579
`'  -  -  &LATER'`	60	093069	b̸9-30-69&LATER
`'  &  &  &LATER'`	61	093069	b̸9b̸30b̸69&LATER
`'  /  /  '`	62	100169	10/01/69
`'  ,   ,   .  -'`	63	000000015 -	b̸b̸b̸b̸b̸b̸b̸b̸b̸b̸b̸15-
`'  ,   , 0 .  -'`	64	000000005	b̸b̸b̸b̸b̸b̸b̸b̸0.05b̸

Figure 56. Samples of Edit Words, Part 2 of 2

J. Illustrates asterisk protection. Asterisks replace all positions in the body (including those with constants --like commas, for instance) to the left of the first significant digit, through the position that contains the left-most asterisk in the edit-word body. (The asterisk in the edit word itself is replaced by any significant digit that may be in the corresponding position of the data field.)

If the asterisk were in the right-most position of the edit-word body, asterisks would appear in the output record for the entire body of the edit word when the data is all zero (if it were minus zero, the minus sign would appear).

1, 2, and 3. No edit word. The data appears in the output record as contained in the data field. Note that the low-order position is zoned in the output record if zoned in the data field.

4, 5, and 6. A blank edit word is assigned. All leading zeros are blanked and a zone in the low-order position is eliminated in the output record. Negative values are not identified.

7. The effect is the same as in 4, 5, and 6. If the edit word contained a status portion, the 0 in the body would cause the status portion to appear in the output record also for a value of minus zero. In examples 4, 5, and 6, a status segment would appear as blank for a minus-zero value.

8. Although the suppression-limiting 0 is at the extreme left, suppression of the first leading zero cannot be avoided.

9 and 10. The status portion appears in the output for negative values; it is blank for positive values. Because the first ten blank positions accommodate the data field, the eleventh position--also blank--is part of the status segment. Any positions to the right of the status-segment end (CR or minus symbol) represent expansion, which always appears identically in the output record--even if the status segment is blanked.

11. An ampersand in the status segment also appears as blank in the output. A minus sign, instead of CR, is illustrated. The blank following the CR or minus is part of expansion.

12 and 13. The *NET& to the left of CR or minus (and to the right of the body) is part of the status segment. Therefore, it appears in the output record as blank when the status does not apply (value not negative). However, the &* to the right of CR or minus represent the expansion segment, and therefore always appear in the output.

14. There is no minus or CR to the right of the body. Therefore *bPROFIT is expansion, and appears in the output regardless of the sign of the data field.

15 and 16. Similar to 11, but a fixed dollar sign is shown. Note that an extra position was added to the body to accommodate the dollar sign.

17. When the dollar sign appears to the immediate left of the suppression-limiting 0, it becomes a floating dollar sign--even when it occupies the leftmost position in the edit word (see No. 34 for contrast).

18 and 19. Floating dollar sign is illustrated for different numbers of leading zeros. Note extra position in edit word to compensate for the dollar sign.

20 and 21. When the contents of the data field are minus zero, the status portion is blanked unless a 0 (or asterisk) appears in the body of the edit word. Regardless, however: the expansion segment is reproduced in the output record.

22 and 23. This illustrates the same as items 20 and 21, but points out that--even with a dollar sign--the status portion is blanked when the value is minus zero, unless 0 (or *) appears in the body of the edit word.

24. An example of some zeros appearing in the output record when the entire field is zero. Zero-elimination extends through the 0 in the edit word, leaving two data positions whose zeros appear in the output (the third blank after the edit-word 0 belongs to the status segment, because all ten data positions have been allowed for before that position).

25. As 24, but with floating dollar sign replacing the last suppressed zero.

26. Presence of a 0 (or *) in the body causes the status segment to appear in the output even for a minus zero value (see also 21 and 23). Because the dollar sign is adjacent to the 0 in

the low-order position, it is a floating $ and appears in the output record in the low-order position of an all-zero data field. This gives full protection with a floating dollar sign, even for all-zero data, when all leading zeros are to be eliminated.

Incidentally illustrated is a status segment consisting only of a minus sign, and no space.

27. Asterisk protection with complete elimination of all leading zeros. The status segment appears in the output for a minus zero field when there is an asterisk (or zero) in the body of the edit word.

28. Asterisk protection to a certain position; thereafter, any further leading zeros appear in the output.

29 and 30. Asterisk protection and zero elimination for a single position. Note that the asterisk is replaced by a significant digit in that position.

Because there is no status segment, a negative value is not identified.

31. Asterisk protection and zero suppression for entire field. Significant digits take precedence, and replace the asterisk.

32 and 33. A method for punctuating a ten-digit dollars-and-cents field. Punctuation and zeros to the left of the first significant digit are blanked. The decimal point is also lost when there are less than three significant digits (see item 63). The status segment is blanked for an all-zero field. The expansion portion always appears.

The minus sign to the left of NET in No. 33 was inserted to point out that it is part of expansion--not status--because a sign symbol (CR) already appeared to its left.

34. The ampersand--which appears in the output as a space--makes it possible to keep the dollar sign fixed, while limiting zero suppression to the minimum of one position (contrast with item 17). All punctuation is retained--regardless of leading zeros --because the 0 in the edit word is placed to the left of the first comma.

35-38. Standard methods for placing the floating dollar sign to retain at least the decimal point, regardless of number of leading zeros.

Note that the extra position to compensate for the floating dollar sign is at the extreme left of the edit word--not where the dollar sign is recorded; i.e., the location of the comma is not changed.

39. Asterisk protection and zero elimination to the decimal point, retaining the decimal point regardless of number of leading zeros. Note that asterisks replace also the punctuation (or any constants) where leading zeros are suppressed. (See also No. 41.)

The second asterisk is part of the status segment, and appears in the output only for a negative value. The third and fourth asterisks form the expansion, and always appear in the output.

40. A standard programming technique for retaining the decimal point while eliminating all leading zeros to the left.

Similar to A, but shows status segment for minus-zero value.

41. Similar to 39, but the status portion consists only of a minus sign and there is no expansion segment. The effect on a minus-zero field is shown.

42. Shows that the constant (in this case, a comma) follows the dollar sign in the output record, if the floating dollar sign and the suppression-limiting zero immediately precede a constant and there is a relevant number of leading zeros.

In the case of a comma, this has an awkward-looking effect; in case of a decimal point, it is a normal approach (see item 36).

43. How to maintain a space between a fixed dollar sign and the first data digit, when all digits in the field are significant (no leading zeros): an & in the body appears as a space in the output record.

44. Normally punctuated quantity field. However, all leading zeros (including units position) will be suppressed (compare to item 45).

45. Normal method for showing a single zero in the output record when the data-field contains only zeros.

46-50. Other constants within the body of the edit word behave like punctuation (which is the same as constants): constants to the right of the first significant digit or the suppression-limiting zero appear in the output.

Note that a constant to the right of the last data-digit position is part either of the status or expansion segment. If CR or minus symbol follows, it is part of the status, and appears in the output only for negative values (note items 48-50); if no sign symbol follows, it belongs to expansion, and always appears in the output record (see item 47).

Items 48-50 also show the effect on constants of different placement of a suppression-limiting 0. In item 49, an ampersand inserted after the first constant provides a space following that constant in the output record.

51. A hyphen (a minus symbol) is used within the body of the edit word. A social security number is shown.

If the initial zero must appear in the output, the data must be broken up into three separate fields, no edit words can be used, and the hyphens must also be specified as separate output constants. (See <u>Programming Tips</u>.)

52. Again, an edit word containing constants in the body, followed by expansion. Included is an illustration of an apostrophe within an edit word (i.e., within an alphameric literal).

53 and 54. Illustrate effect, on decimal point (or any other constant) and following zeros, of different placements of the suppression-limiting zero--when the field contents are zero.

55. Included to emphasize that a dollar sign that is separated from the suppression-limiting zero--even if only by a comma--is not a floating dollar sign, but a constant.

It is not possible (in this RPG) to place a floating dollar sign so that it replaces a constant (e.g., a comma) in the output record when the first significant digit follows the constant (for instance, when it follows a comma).

56-59. A zero or asterisk after the leftmost zero or asterisk is a constant--not a suppression-limiting or asterisk-protection symbol.

Items 58 and 59 again also show that asterisk protection supplants not only blanks but also other constants to the left, including an ampersand.

60-62. Three examples of editing a date field. Note that one leading zero is suppressed even if 0 were placed in the leftmost position of the edit word: therefore, since month numbers have at most one leading zero, there was no point in specifying a suppression limiting zero.

Item 61 shows the use of ampersand in the edit word to retain a blank space in the output record. The characters &LATER, however, form an expansion section. An ampersand in the expansion appears as such in the output.

63. Shows what happens to the decimal point when there are less than three significant digits in a longer field, and no suppression-limiting zero is specified.

64. Shows one method of preventing loss of the decimal point when there are less than three significant digits in a longer field.

Moving the suppression-limiting zero one position further right still preserves the decimal point, but eliminates the zero to its left in the output.

216 – 243

One application example (Figure 5) appears early in the manual, to serve as an introduction to the RPG approach. The numerous other fragmentary program examples up to this point were developed to illustrate various individual functions that can be performed with the Report Program Generator (RPG).

The following three program examples illustrate the complete program specifications for:

1. A sales commission calculation and report;

2. An order-entry pre-billing application: inventory control and order-item price extension, involving also matching of files; and

3. A simple invoicing operation with summary punching of accounts receivable cards. The detail cards processed in example 2 are utilized; but side-by-side printing of ship-to and sold-to cards is also demonstrated, as well as table look-up.

The first sample program can be compiled and executed on a system equipped with 4K bytes of core storage in the CPU, any one of the Model 20 card-reading devices, and a printer. A card deck comprising the specifications and data for this example is supplied by IBM's Program Information Department together with the RPG compiler. In that deck, the IBM 2501 is assigned as the card reader. However, the application was deliberately designed not to be dependent on a particular read device: the user need only change the Device code in the first File Description Specifications card to correspond to his particular Model 20 card reader--the program will run with any Model 20 reader.

The second, more complex, application example requires 8K bytes of core storage, and has been designed to take advantage of all the features of the IBM 2560 MFCM: reading cards from both hoppers, punching into cards that have also been read (combined file), interpreting (card document-printing), and stacker selection.

The third example can also be run within 4K bytes of core storage. It has been written for the MFCM, to take advantage of interpreting (if installed); but otherwise it can be run on other Model 20 I/O units, if the File Description specifications are amended to correspond.

Note: The interpreting feature is available only on the 2560 MFCM, Model A1.

SALES COMMISSION CALCULATION AND REPORT

A commission report is to be prepared using invoice summary cards (Figure 57) for the source data. The cards are in sequence by salesman number. The invoice summary cards are coded with a 5 in column 1; other cards that are maintained as part of the invoice file--and are not to be processed--do not contain a 5 in col. 1.

The commission amount is calculated on the net invoice amount. The percentage of commission depends upon the net invoice amount:

For net invoice amounts up to (and including) $10,000--10% commission
For net invoice amounts above $ 10,000--12% commission

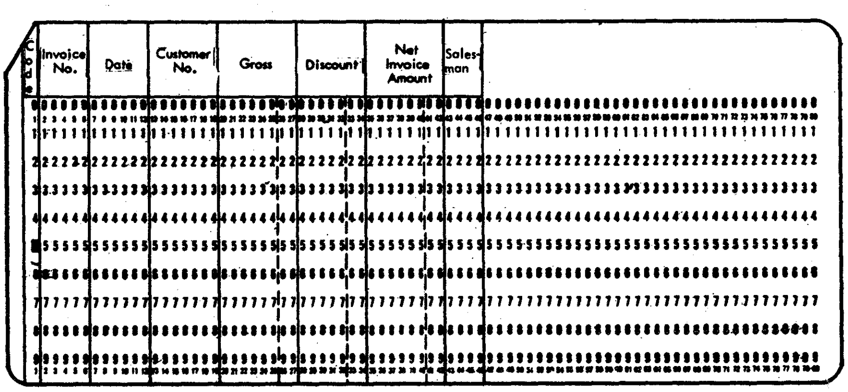


Figure 57. Sales Commission Calculation, Format of Invoice Summary Card

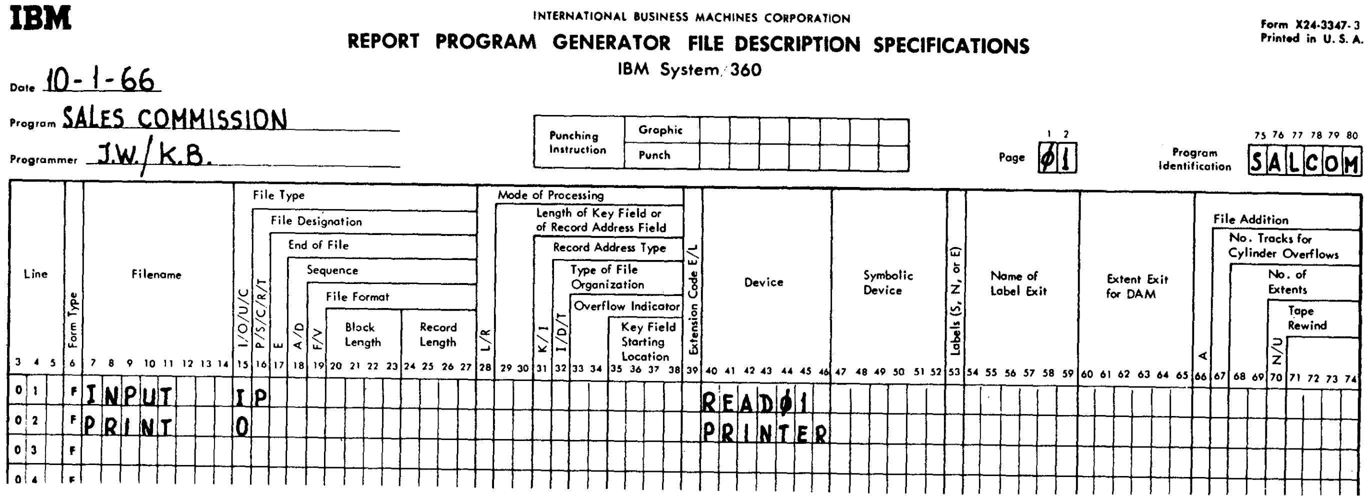


Figure 58. Sales Commission Calculation, File Description Specifications

The commission is calculated on each invoice summary card, and the pertinent input and calculated data is detail printed. A total commission amount for each salesman and a final sum of commission amounts are accumulated and printed.

File Description Specifications (Figure 58)

Only two specifications lines are necessary for this application. In the first line, the input file name INPUT is assigned to the card deck that contains the invoice summary cards, and that file is associated with the IBM 2501 Card Reader by the Device code READ01. In the second line, the output file named PRINT is associated with the printer (IBM 2203 or 1403).

Input Specifications (Figure 59)

Specifications line 01 provides for identification of the invoice summary cards: character 5 in col. 1. Resulting Indicator 11 is assigned to this card type.

The Sequence entry (cols. 15-16) is alphabetic, because the presence of the two

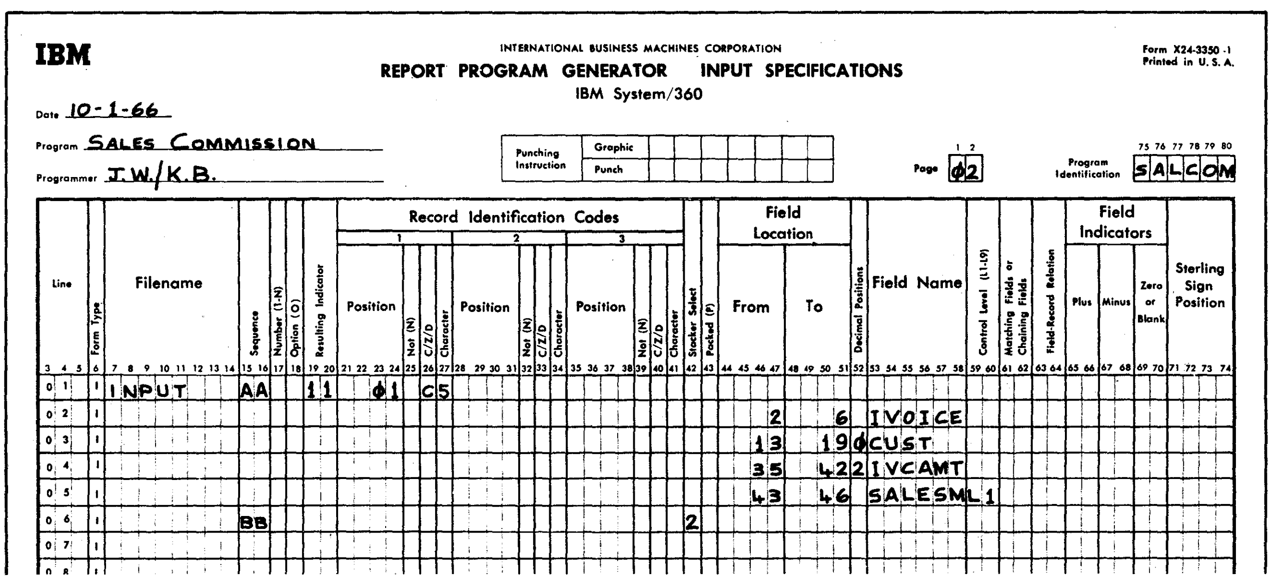


Figure 59. Sales Commission Calculation, Input Specifications

card types is optional and, when both are present, either may be first or they may be intermixed.

Lines 02-05 describe the fields to be read from the invoice summary cards. Note that fields not used in this job are not described: it would waste core storage space to do so, and serves no purpose.

Although invoice number (IVOICE) and Salesman number (SALESM) are numeric, they need not be defined as numeric. On the other hand, customer number (CUST) must be defined as numeric because Zero Suppress is used for that field in the Output-Format Specifications. (Since CUST is not used in an arithmetic or numeric compare operation, any number of decimal positions within field size--i.e., 0-7--can be specified in col. 52.)

Invoice amount (IVCAMT) must have 2 specified in Decimal Positions, because it is to be used in arithmetic operations where proper decimal alignment matters and where all fields must be numeric.

In line 05, salesman number is set up as an alphameric control field of Level 1. Only cards for which indicator 11 is on are considered in the control-field comparison (see also comment for line 01 of page 04).

Line 06 represents other cards in the same input deck. These cards are part of the same deck as the invoice summary cards, but are to be passed through the I/O device without any processing.

Because Sequence (cols. 15-16) in line 01 contains an alphabetic entry, the program always first attempts to match each card against the Record Identification Codes in line 01 (see Nature of the Card-Type Sequence Check). Only if it is not an invoice summary card (i.e., not 5 in col. 1) is the card matched to the Record Identification Codes in line 06. Since cols. 21-41 are blank in line 06, all cards without character 5 in col. 1 satisfy the criteria for line 06. This is a good technique for providing for "other card types." There must be an entry in cols. 15-16 for every card type; if there is no entry in cols. 15-16 for a card type that can occur in the data deck, the system would stop because of an unidentified card type.

A card-type Resulting Indicator (cols. 19-20) is not needed in line 06 in this particular example. All calculation and output operations that are pertinent only to the invoice summary cards are conditioned by indicator 11.

The 2 in col. 42 (Stacker Select) is ignored when the IBM 2501 Card Reader serves as the input device (the form in which this program is written, and supplied by IBM). If a different card reader is employed (after changing the first card in the File Description Specifications to conform), the invoice summary cards will enter the normal stacker of the device and the other cards in the deck will be selected to stacker 2.

IBM — INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS
IBM System/360
Form X24-3351-1
Printed in U. S. A.

Date 10-1-66
Program SALES COMMISSION
Programmer J.W./K.B.
Punching Instruction: Graphic / Punch
Page Ø3
Program Identification SALCOM

Line	Form Type	Control Level	Indicators	And	And	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Plus / High 1>2	Minus / Low 1<2	Zero or Blank / Equal 1=2	Comments
01	C		11			IVCAMT	COMP	1ØØØØ					22			
02	C		22	11		IVCAMT	MULT	.12	COMM	8	2	H				
03	C		N22	11		IVCAMT	MULT	.1Ø	COMM			H				
04	C		11			COMM	ADD	SUM	SUM	8	2					
05	C	L1				SUM	ADD	FINTOT	FINTOT	9	2					
06	C															
07	C															

Figure 6C. Sales Commission Calculation, Calculation Specifications

Calculation Specifications (Figure 60)

Specifications line 01 causes the net invoice amount to be compared with the numeric literal 10000. Resulting Indicator 22 is turned on if the invoice amount exceeds $10,000; otherwise indicator 22 remains (or is turned) off. Note that decimal alignment is performed by the program itself in numeric compare operations.

Line 02 provides for multiplying the net invoice amount by 12 percent, if indicator 22 is on (invoice amount greater than $10,000); whereas line 03 causes multiplication of the invoice amount by 10 percent, if indicator 22 is off (invoice amount less than, or equal to, $10,000). Thus, for any one invoice summary card, the specifications in either line 02 or line 03--but not both--are executed. (Decimal alignment is automatic in arithmetic operations.)

The result Field (COMM) was not an input field, and must therefore be defined in the calculation specifications. This is done in line 02 (it could equally well have been in line 03). Half Adjust is specified in lines 02 and 03. Since each factor (IVCAMT and .12 or .10) contains 2 decimal places (yielding 4 decimal places in the result), and only 2 are specified for the result, the 2 rightmost decimal positions are dropped after half-adjustment.

A Field Length of 8 positions is specified for the result field: an 8-position field is multiplied by a 2-position field, which results in a maximum of 10 positions; two decimal positions are dropped, leaving a maximum of 8 positions.

Line 04 causes the commission to be added into a field named SUM, which is established (at program-generation time) by the specifications in cols. 43-52 in this line. The individual commission amounts calculated on each invoice summary card are accumulated in SUM for a commission total by salesman.

The Blank-After instruction in the Output-Format Specifications clears the SUM field to zero at the end of each salesman's group of cards, so that it is ready for accumulation of commissions for the next salesman.

This line also illustrates using the same field as addend and result field (as does line 05): A+B = B'. (The field names in Factor 1 and Factor 2 could equally well be reversed.)

Note:

1. All of the above operations are performed at detail-calculation time (cols. 7-8 are blank), which is the normal method for handling this type of application.

2. All of the above calculations are conditioned by indicator 11. Therefore, they are performed only when an invoice summary card is being processed.

Line 05 causes the total commission amount for each salesman to be added to the field FINTOT, to provide a grand-total of commissions at the end of the report. FINTOT is one position larger than SUM, to assure capacity for the larger total.

The operation is performed at total-calculation time, provided indicator L1 is on; i.e., at the end of every salesman-number control group--after the commission calculated on the last card of the group was added to SUM, but before the SUM field is cleared at total-output time.

Note that all detail-time calculation specifications precede those for total time.

Output-Format Specifications (Figure 61)

Page 04

Specification lines 01-08 provide for printing columnar headings at the top of each page.

Lines 01 and 02 (in an OR relationship) specify output to the printer, because the file name PRINT was associated with the printer in the File Description Specifications. Line 01 calls for the printing to take place at overflow-output time; line 02--in conjunction with the H in col. 15 (which could equally well be a D) of line 01--provides for the same printing at detail-output time for the first card of each salesman-number control group (Control Level L1).

The NL1 in line 01 prevents duplication of the operation when a control break occurs in the same program cycle in which overflow (carriage-tape channel 12) is signalled. (In this particular application, NOF could have been specified in line 02 in place of NL1 in line 01. But this is true only because all data here involves constants, not information from the first card of a control group.)

IBM | INTERNATIONAL BUSINESS MACHINES CORPORATION | Form X24-3352-1 Printed in U. S. A.

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System/360

Date 10-1-66

Program SALES COMMISSION

Programmer J.W./K.B.

Punching Instruction: Graphic / Punch

Page Ø4 — Program Identification SALCOM

Line (3-5)	Form Type (6)	Filename (7-14)	Type (H/D/T) (15)	Stacker Select (16)	Space Before (17)	Space After (18)	Skip Before (19-20)	Skip After (21-22)	Output Indicators Not (23-25)	And Not (26-28)	And Not (29-31)	Field Name (32-37)	Zero Suppress (Z) (38)	Blank After (B) (39)	End Position in Output Record (40-43)	Packed Field (P) (44)	Constant or Edit Word (45-70)	Sterling Sign Position (71-74)
01	O	PRINT	H			3	Ø1		OF	NL1								
02	O	OR							L1									
03	O														71		'COMMISSION'	
04	O														59		'PERC'	
05	O														51		'AMOUNT'	
06	O														42		'INVOICE'	
07	O														34		'CUST'	
08	O														28		'SALESMAN'	
09	O		D			2			11									
10	O											COMM			71		' , Ø. '	
11	O								N22						59		'1Ø %'	
12	O								22						59		'12 %'	
13	O											IVCAMT			53		' , Ø. '	
14	O											IVOICE			41			
15	O											CUST	Z		34			
16	O								L1			SALESM			26			
17	O		T		2				L1									
18	O														56		'TOTAL'	
19	O											SUM		B	71		' , Ø. '	
	O																	

Card Electro Number

IBM | INTERNATIONAL BUSINESS MACHINES CORPORATION | Form X24-3352-1 Printed in U. S. A.

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System/360

Date 10-1-66

Program SALES COMMISSION

Programmer J.W./K.B.

Punching Instruction: Graphic / Punch

Page Ø5 — Program Identification SALCOM

Line (3-5)	Form Type (6)	Filename (7-14)	Type (H/D/T) (15)	Stacker Select (16)	Space Before (17)	Space After (18)	Skip Before (19-20)	Skip After (21-22)	Output Indicators Not (23-25)	And Not (26-28)	And Not (29-31)	Field Name (32-37)	Zero Suppress (Z) (38)	Blank After (B) (39)	End Position in Output Record (40-43)	Packed Field (P) (44)	Constant or Edit Word (45-70)	Sterling Sign Position (71-74)
01	O		T		3			Ø1	LR									
02	O														56		'FINAL TOTAL'	
03	O											FINTOT			71		' , , Ø. '	
04	O																	
05	O																	

Figure 61. Sales Commission Calculation, Output-Format Specifications

Because no forms-control specifications (cols. 17-22) are entered in line 02, the instructions in line 01 apply to both lines: the form is advanced to the next channel-1 punch before printing at overflow-output time or at detail time for the first card of a control group. After printing of the heading line, the form is advanced 3 spaces, to leave two blank lines before the detail data.

Because OF is specified in Output Indicators of a File-Identification line, no automatic overflow forms advance takes place--it must be specified (as it is in line 01).

Note: Total-time output always precedes overflow-time output in the same program cycle; therefore, totals are normally printed on the old page before forms advance to a new page during overflow time. However, in the particular application described here it may appear as though the total were printed on the next page:

If the last invoice summary card of a control-level group triggers the overflow operation (i.e., is printed at or below the channel-12 punch), but other cards (other than invoice summary cards) follow before the first invoice summary card of the next salesman number, then forms advance to the next page takes place after the last invoice summary card of the group and--apart from constant headings at the top--the next line printed on the new page contains the control-group total. (See Warnings: 2, under Output Indicators--OF, OV--File-Identification Specifications.)

Lines 03-08 define the constants that are to be printed on the first line of each page as columnar headings.

Lines 09-16 specify the detail printing from invoice summary cards.

Line 09 specifies that the output is to take place at detail time (D in col. 15), but only for invoice summary cards (indicator 11). The form is advanced two spaces after each detail line (2 in col. 18). The file name (PRINT) need not be repeated, since it is the same as that for the last preceding File-Identification line.

Lines 10, and 13-16 describe the data fields to be printed.

The output for the fields COMM and IVCAMT is formatted by edit words--leading zeros to the decimal point are suppressed, and appropriate commas are inserted between each three positions where significant digits appear (the fields were defined as numeric, and therefore can be edited). The assignment of an edit word eliminates the plus (C) zone in the low-order position of COMM from the output; if IVCAMT was zoned in the input card, that zone is also eliminated from output by virtue of the edit word.

Any leading zeros, and a possible low-order-position zone, are eliminated from the output for CUST by Zero Suppress (Z in col. 38). The field must have been--and was--defined as numeric.

Field selection is performed in lines 11 and 12: one of the two constants (10 % or 12 %) is printed, based on the result of the COMP operation in the calculation specifications (indicator 22 off or on).

The L1 in Output Indicators of line 16 conditions the contents of the field SALESM to be printed only when indicator L1 is on at detail-output time--i.e., on the first card of a control group (group-indication).

Lines 17-19 provide for the printing of the commission total (SUM) at the end of each control group (L1 in Output Indicators of File-Identification line), at total-output time (T in col. 15). The form is spaced 2 lines before printing, which--in conjunction with the 2 spaces after each detail line--creates 3 blank lines before the total line. The word TOTAL is printed to the left of the amount.

The Blank-After (B in col. 39) instruction resets the field SUM to zero as soon as the contents have been transferred to the output core-storage area. The field is then ready for acculumation of the total for the next control group.

The file name (PRINT) need not be repeated in the File-Identification line.

Page 05

Lines 01-03 contain the specifications for printing the grand total commission amount for the report. (The file name need not be repeated.) Output is at total time (T in col. 15), after processing of the last data card (LR indicator on)--it must be at total-output time, because the operation terminates after total-output time when LR is on.

The words FINAL TOTAL are printed to the left of the amount.

The form is advanced 3 lines before printing (3 in col. 17) and to the next page after the final total (01 in cols. 21-22).

Note: The channel-12 punch must be high enough to allow for 8 additional lines on the same page. Assuming worst cases:

(a) Detail line printed just above carriage-tape 12-punch line, followed by detail line one line below 12-punch (double space after each detail line) = 1 line below 12-punch. Overflow indicator turns on.
(b) Double space after that detail line = 3 lines below 12-punch.
(c) Control-Level 1 total follows--double space before total line = 5 lines below 12-punch.
(d) End of report--triple space before total line = 8 lines below 12-punch.

SALESMAN	CUST	INVOICE	AMOUNT	PERC	COMMISSION
0313	9450	00015	4,730.50	10 %	473.05
	11269	00344	10,665.00	12 %	1,279.80
	20011	10046	30.75	10 %	3.08
	34567	10095	580.35	10 %	58.04
			TOTAL		1,813.97

			FINAL TOTAL		129,454.13

Figure 62. Sales Commission Calculation, Printed Report

Sample Report

Figure 62 shows a sample report based on the program defined above. It portrays the first control group and final total as actually produced if the sample deck supplied by the Program Information Department is run.

PRE-BILLING CALCULATION WITH INVENTORY CONTROL

This example illustrates one of numerous approaches to an order-processing/inventory control job. The application has been arbitrarily slanted to a distribution business--perhaps a mail-order house--with customer orders to be filled from warehouse stock. An attempt has been made to be reasonably realistic in the application, including the complexities of such a multi-purpose operation.

Note: Figures 63 and 64 should be consulted in relation to the discussion that follows.

Basic Assumptions

1. A card has been keypunched:
   (a) For each item line on a customer order--Card 9, no X in col. 11;
   (b) For each item line on a customer return--Card 9, X in col. 11;
   (c) For each item line on a stock receipt (or purchase-order cards are used as stock receipt cards)--Card 5;
   (d) For each stock adjustment--Card 6: No X in col. 11 to reduce on hand, X in col. 11 to increase on hand;
   (e) For each item on a stock purchase order--Card 7: no X in col. 11 when ordered, X in col. 11 if order is cancelled or reduced.
   (f) For a new stock item or a change in price, description, warehouse location, etc. (Obsolete master cards are removed manually or, at least, separately from this operation.)

   Note: The sixth digit of Stock No. could be the check digit for a self-checking number. (See Self-Checking Number Device for keypunches.)

2. An Inventory Master Card file exists, with one card per item carried in stock. Changes to the file are made manually, or in some other data processing operation (i.e., addition and deletion of items, changes in price, warehouse location, etc.).

3. It is desired to process customer order-item cards against inventory records before attempting to fill the orders in the warehouse. At the same time, the inventory records will be updated and an up-to-date inventory report prepared.

   The customer-order cards are thereafter ready for invoicing. (The cards could be sorted by warehouse location prior to invoicing.) A copy of the invoice, or the cards themselves, serve as order-picking medium--i.e., either sequential or bulk picking is employed.

   If orders are processed once daily on this basis, the inventory records are always up to date.

   Note: The third example utilizes these customer-order cards.

4. If the quantity on-hand is insufficient to satisfy the quantity in the customer-order card, no partial quantity will be applied for that item. The item order:
   (a) Will be marked for back order if not previously back-ordered, and

provided stock is on order; or
(b) Will be marked "cancelled" if no stock is on order; or
(c) Will be marked "cancelled" if previously back-ordered. (Say, back orders are reprocessed one week after they became back orders.)

5. Where previously back-ordered item cards are reentered, they are to receive priority for available inventory.

6. Some items have a lower unit selling price when at least the specified criterion quantity is ordered by the customer.

7. Stock adjustments are made without attempt at modifying the unit cost of the item.

8. (a) Besides price extension, gross profit is to be included in the item detail cards for a subsequent report by merchandise class and division, and by Stock No. (The first digit of Stock No. represents merchandising division, the second the classification within division.)
   (b) Value of inventory on hand (average cost basis) is to be continually available.
   (c) Available quantity (on-hand plus on-order) less than an established minimum is to be signalled.

Procedural Details

1. Safeguards.
   (a) Certain control totals will be carried, partially as audit trails. Control totals are presumed to have been established for the various kinds of transaction cards, so that new on-hand and on-order totals can be proved out.
   (b) Customer-order detail cards that are being cancelled will be identified. If such a card is reentered, it is selected out, and calculation for it is bypassed.
   (c) Matched old master cards--for which new ones are created--will be identified, and selected to a separate stacker. If such a card is accidentally reentered, the entire stock-number group is selected to a separate stacker, and calculation is bypassed.
   (d) The entire stock-number group (exept the first card) is selected to a separate stacker, processing is bypassed, and the system halts after the second card, whenever there is more than one master card for a group.
   (e) The entire stock-number group is selected, and calculations are bypassed, when a master card with a negative on-hand quantity has been read.
   When a negative on-hand quantity is created as a result of calculation, the cards from the point of error are selected, and calculations are bypassed.
   (f) Whenever the blank trailer card is missing or mispositioned within the group, all cards in the group from the point of the error detection are selected, the system halts, and further calculation is bypassed for the group.
   (g) Unmatched transaction cards, including the trailing blank card, are selected, and calculations are bypassed.
   (h) If the on-order quantity turns negative, the system halts. The inventory report also indicates this condition.
   (i) For known error conditions that affect the new inventory values, the data is omitted from the report (the cards have been selected to a separate stacker).

2. Any merchandise receipts, stock adjustments, and customer returns precede order-item details, so that the customer orders are correctly applied to the latest on-hand status.

   Stock purchase-order cards are also placed ahead of customer order details, because it was decided not to back-order items for which no stock is on order.

   Former back-order cards precede other order-item cards to get first chance at on-hand goods.

3. The cards are assumed to be in ascending sequence by Stock No.

   Inventory master cards are to be in the primary feed of the MFCM--preceded by a single card to read in today's date. All other cards will be placed in the secondary feed.

   A previous operation has placed a blank card at the end of each Stock No. group of secondary-file cards. These blank cards will become the new (updated) inventory master cards for stock numbers for which there are transactions. (These blank cards were merged in on the MFCM of the Model 20, using the PLACE specification card of the Punched-Card Utility Collate program or an RPG program; or they could have been merged on a collator.)

4. Stacker Selection.
   (a) The Date header card is directed to stacker 1; any other stacker would do equally well.
   (b) All old inventory master cards with stock numbers for which there are transaction cards in the secondary file are directed to stacker 1 (the normal stacker--chosen to contain obsoleted cards), because a new inventory master card will be punched--and placed in stacker 2.

   Each unmatched old inventory master is selected to stacker 2, because no new master is punched in such case.

   Stacker 2 ultimately contains the complete up-to-date inventory master file (except for known error-condition cards)--consisting of new cards where transactions occurred and old masters where no transactions applied.

   (c) Stacker 3 receives the customer order-item cards, ready for warehouse picking (if cancelled and BO (back-orders) are sorted out), or to be sorted on order and account numbers for invoicing.
   (d) Stacker 4 has been assigned to unmatched transaction cards (secondary file), and to all other detected error-condition cards.
   (e) Stacker 5 has been assigned to stock orders, receipts, adjustments, and merchandise returns. These may also be left together with the other transaction cards by directing them to stacker 3 instead; they could easily be segregated later by sorting on cols. 1 and 11.

<u>Note</u>: When stacker 5 is designated, but the I/O device referred to is the 2560 MFCM Model A2, the card is directed to stacker 4.

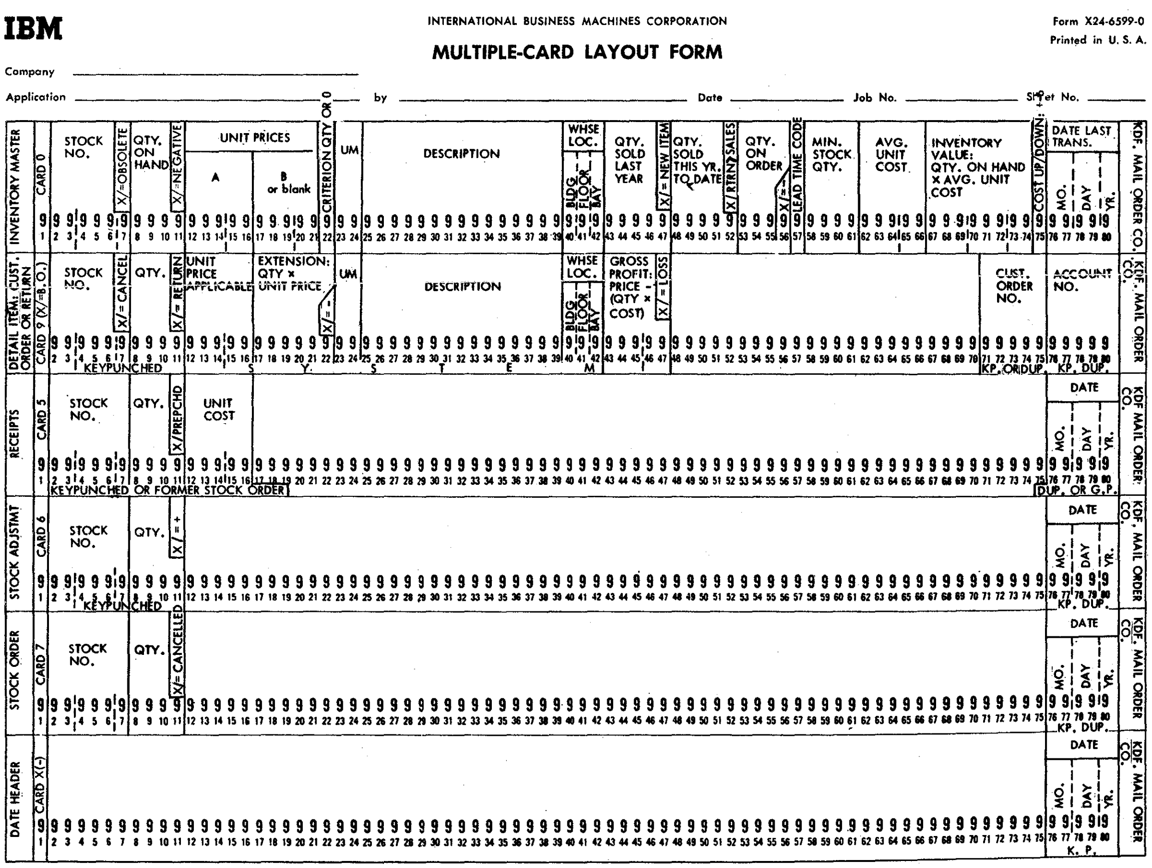


Figure 63. Pre-Billing with Inventory Control, Card Layouts

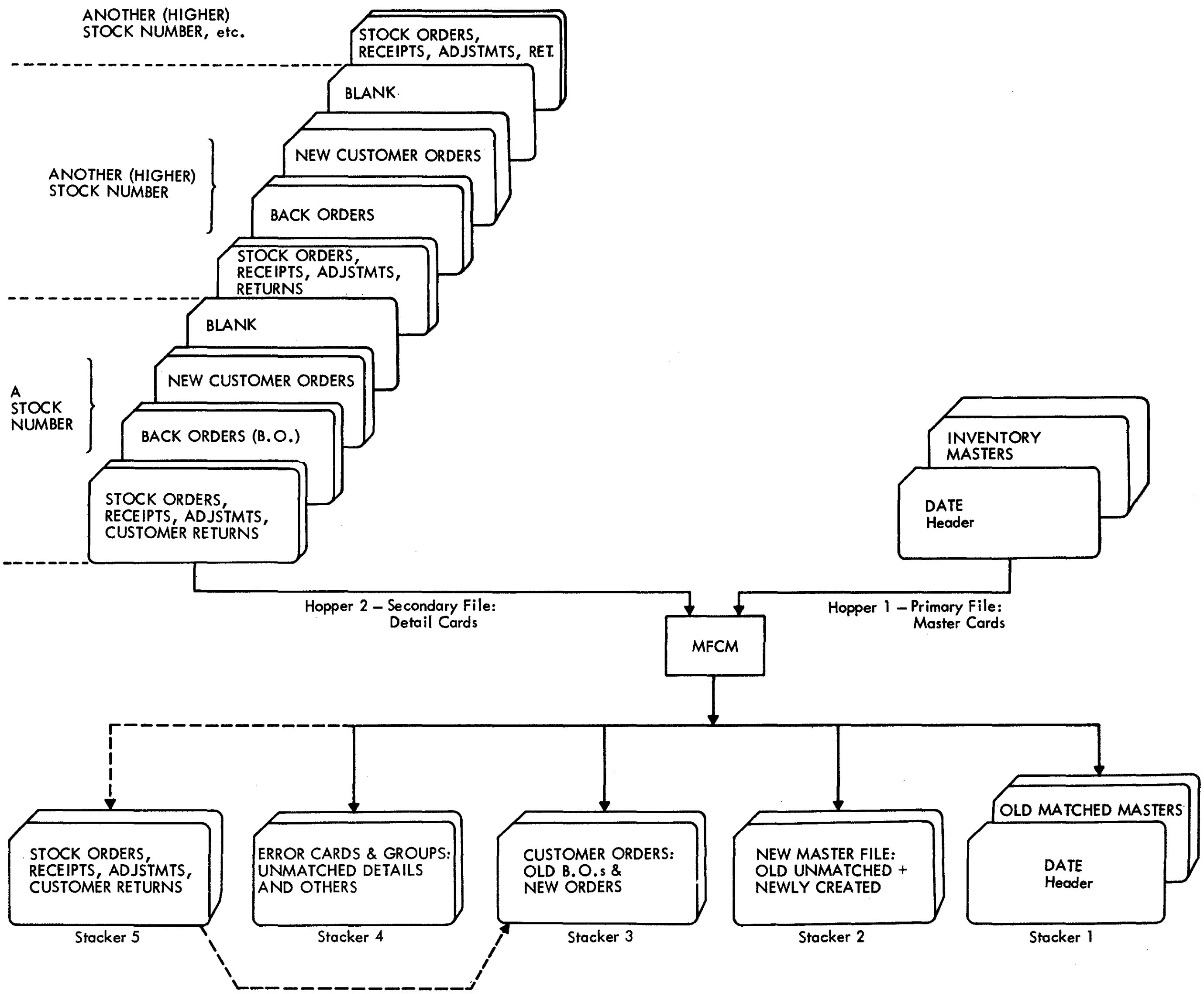


Figure 64.Pre-Billing with Inventory Control, Diagram of Card Flow

Study of figures 63 and 64 will clarify the details of the operation. The report has been laid out (see Figure 65) to fit within the 120-position print span of all IBM 2203 and 1403 Printers attachable to Model 20. Explanation of specifications sheets follows; Figure 70 (Assignment of Indicators) will also help in following the discussion.

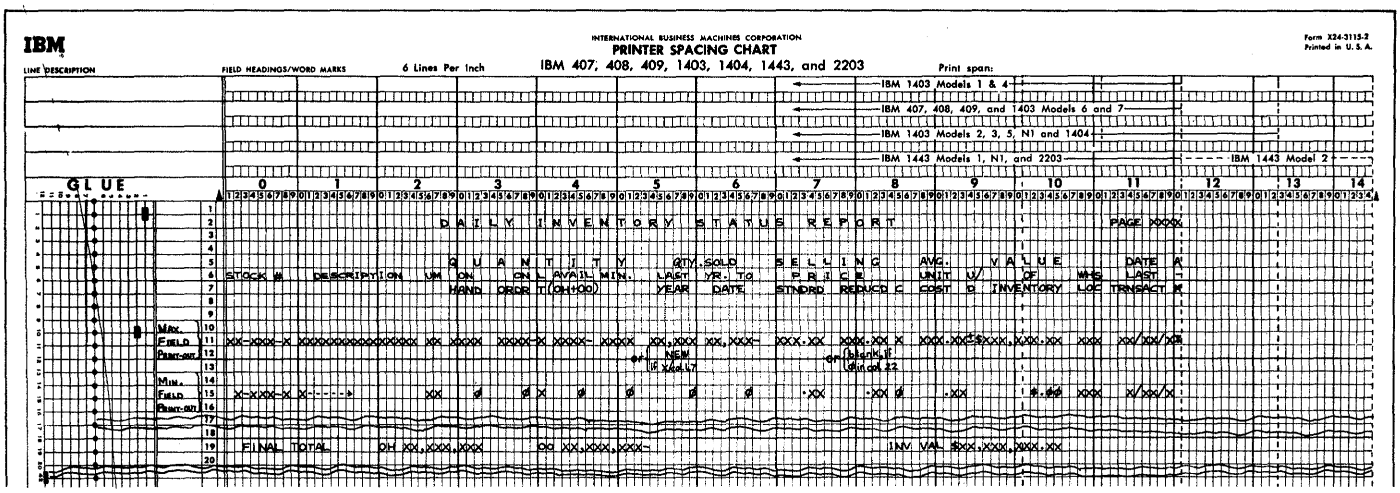
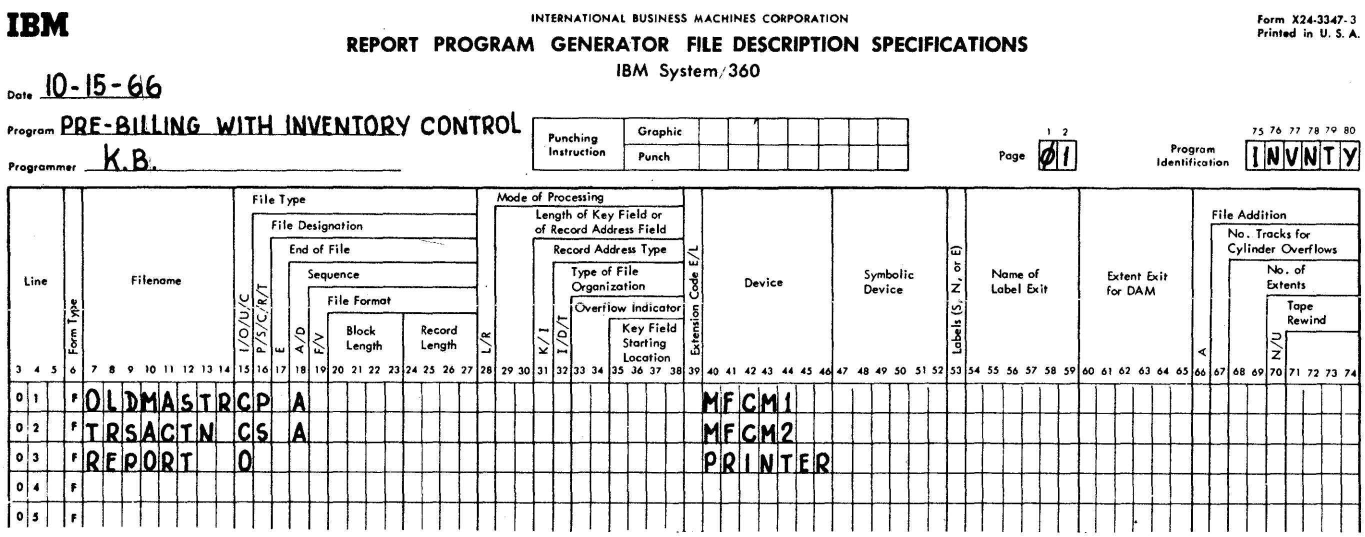


Figure 65. Pre-Billing with Inventory Control, Layout of Inventory Report



Line	Form Type	Filename	File Type (15)	File Designation (16)	End of File (17)	Sequence (18)	Device
01	F	OLDMASTR	C	P		A	MFCM1
02	F	TRSACTN	C	S		A	MFCM2
03	F	REPORT	O				PRINTER
04	F						
05	F						

Figure 66. Pre-Billing with Inventory Control, File Description Specifications

File Description Specifications (Figure 66)

The file of inventory master cards is named OLDMASTR, and associated with the primary hopper of the MFCM. It is defined as a combined file (C in col. 15) so that stacker selection may be performed via output specifications, and to allow punching of a code for "obsolete" at output time into those old masters that are replaced as a result of new transactions.

The detail transaction cards are assigned to the file named TRSACTN, and associated with the secondary hopper of the MFCM. Stacker selection is dependent on calculation operations; therefore--and because output is required to some customer-order item cards--TRSACTN is a combined file.

The input files are in ascending sequence (A in col. 18). A sequence is required, and must be uniform for the input files, when matching of records in two or more files is called for. If col. 17 is blank, or contains E, for all input files, the LR indicator does not turn on until all input files are exhausted.

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS

IBM System/360

Form X24-3350-1
Printed in U. S. A.

Date 10-15-66

Program PRE-BILLING WITH INVENTORY CONTROL

Programmer K. B.

Punching Instruction								
	Graphic	–	&	+	ƀ	#		
	Punch	11	12	12/4/8	blank	3/8		

Page Ø2

Program Identification INVNTY

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Resulting Indicator	Record Identification Codes 1 Position	Not (N)	C/Z/D	Character	2 Position	Not (N)	C/Z/D	Character	3 Position	Not (N)	C/Z/D	Character	Stacker Select	Packed (P)	Field Location From	To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field-Record Relation	Field Indicators Plus	Minus	Zero or Blank	Sterling Sign Position
3 4 5	6	7 8 9 10 11 12 13 14	15 16	17	18	19 20	21 22 23 24	25	26	27	28 29 30 31	32	33	34	35 36 37 38	39	40	41	42	43	44 45 46 47	48 49 50 51	52	53 54 55 56 57 58	59 60	61 62	63 64	65 66	67 68	69 70	71 72 73 74
0 1	I	OLDMASTR	IM			Ø1	1		C	Ø																					
0 2	I																				2	7	Ø	STKNO	L1	M1			97		
0 3	I																				8	11	Ø	ONHAND					99		
0 4	I																				12	16	2	PRICEA							
0 5	I																				17	21	2	PRICEB							
0 6	I																				22	22	Ø	CRITQT						2Ø	
0 7	I																				23	24		UNMEAS							
0 8	I																				25	39		DESCR							
0 9	I																				4Ø	42		WHSLOC							
1 0	I																				43	47	Ø	LASTYR							
1 1	I																				47	47		NEWITM							
1 2	I																				48	52	Ø	YRTDAT							
1 3	I																				53	56	Ø	ONORDR							
1 4	I																				57	57		LEADTI							
1 5	I																				58	61	Ø	MINQTY							
16	I																				62	66	2	AVGCST							
17	I																				67	74	2	INVVAL							
18	I																				76	8Ø	Ø	TRNSDA							
	I																														
	I																														

Card Electro Number

Figure 67 (Part I of II). Pre-Billing with Inventory Control, Input Specifications

The printer is associated with an output file named REPORT.

Input Specifications (Figure 67--Parts I and II)

Because the file OLDMASTR is specified ahead of the TRSACTN file, it is therefore the primary file; i.e., matching cards from the OLDMASTR file are processed ahead of their matching TRSACTN-file cards.

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS

IBM System/360

Form X24-3350-1
Printed in U.S.A.

Date 10-15-66

Program PRE-BILLING WITH INVENTORY CONTROL

Programmer K.B

Punching Instruction	Graphic	ƀ							
	Punch	Blank							

Page ϕ3 Program Identification INVNTY

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Resulting Indicator	Record Identification Codes 1 Position	Not (N)	C/Z/D	Character	2 Position	Not (N)	C/Z/D	Character	3 Position	Not (N)	C/Z/D	Character	Stacker Select	Packed (P)	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field-Record Relation	Field Indicators Plus	Minus	Zero or Blank	Sterling Sign Position
3 4 5	6	7 8 9 10 11 12 13 14	15 16	17	18	19 20	21 22 23 24	25	26	27	28 29 30 31	32	33	34	35 36 37 38	39	40	41	42	43	44 45 46 47	48 49 50 51	52	53 54 55 56 57 58	59 60	61 62	63 64	65 66	67 68	69 70	71 72 73 74
0 1	I		DA			ϕ9	1		C	-									1												
0 2	I																				76	8ϕ	ϕ	DATE							
0 3	I	TRSACTN	TR			11	1		C	5																					
0 4	I	O	R			12	1		C	6																					
0 5	I	O	R			13	1		C	7																					
0 6	I	O	R			15	1		D	9																					
0 7	I																				1	1	ϕ	BOCARD					21		
0 8	I																				2	7	ϕ	STKNO	L1	M1			98		
0 9	I																				8	11	ϕ	QTY					22		
1 0	I																				12	16	2	UNCOST							
1 1	I																				76	8ϕ	ϕ	ACCNT							
1 2	I		BL			19	1		C	ƀ																					
1 3	I																														
1 4	I																														
1 5	I																														
	I																														
	I																														
	I																														
	I																														
	I																														

Card Electro Number

Figure 67 (Part II of II). Pre-Billing with Inventory Control, Input Specifications

Inventory Master Cards--OLDMASTR File (Page 02)

The old Inventory Master cards are identified by 0 in col. 1, and assigned indicator 01. Since they are the only card type in the file--apart from the initial single Date card--an alphabetic code is specified in Sequence (cols. 15-16). (If any other--undefined--card types (besides the Date or Master card) appear in the file, the system halts.)

Lines 02-18 contain the normal specifications for reading these fields from the old Inventory Master cards that may be needed for processing of the application (see Figure 63 for greater detail on the card fields). Fields defined as numeric are used in calculations, edit operations, or numeric compare. Points of special interest are:

1. Stock No. is defined as numeric to allow formatting in the output by edit word, and to simplify detection of an obsolete master card (see 4, below).

2. The files are matched and sequence-checked on Stock No. (M1 in cols. 61-62 for Stock No.).

3. The L1 indicator is turned on for the first card of each stock-number group (L1 in Control Level for Stock No.).

   L1 is not used in this program for total printing or punching--it is used solely to recognize the first card of a group for error-control purposes. L-indicators have no inherent connection with matching of files, and L1 is not needed merely because M1 is assigned.

4. Whenever an old Master Card is replaced by a new one, to reflect transactions,

the old card is overpunched with an 11-punch in col. 7 at output time (page 07, line 06) to mark it as obsolete. If such a card is accidentally reentered next time, indicator 97 turns on--the 11-punch causes the Stock No. to read in as negative (a matching-field sequence error does not, however, arise because all zone punches are eliminated from the matching-fields operations of a numeric field).

5. Indicator 99 turns on if the Quantity On-Hand is negative in the old Inventory Master card. Such a card should never appear, because subsequent specifications (page 04) cause output to be bypassed if On-Hand turns negative.

6. Indicator 20 turns on if the Criterion Quantity field is zero. The zero code indicates that only Unit Price A applies, and that the Price-B field is to remain blank both in the report and in a new Inventory Master card.

7. Col. 47 appears twice among the input fields--the first time as part of a normal numeric field; the second time with another name and as a single-column alphameric field:
   If col. 47 is X-punched (11-punch), Quantity Sold Last Year does not apply because the item is new this year. The word NEW is then to appear in the report, and the field is only to contain an X-punch in a new Inventory Master card. But a numeric field that is blank or zero with an X-overpunch in the units position will set on a Field Indicator for Zero or Blank--not for Minus. Therefore, the column that contains the X-punch for "new" is separately defined as alphameric. It can then be tested for a Minus zone by a TESTZ calculation specification.

8. Stacker assignment is not known until calculations are performed. It must therefore be specified at output time.

Date Card--OLDMASTR File (Page 03)

The single Date card at the front of the file is identified by an X-punch in col. 1, and assigned indicator 09. The date is stored in a field given the name DATE. It is defined as numeric to allow editing.

No matching is specified for this card. It is therefore processed first.

The Date card is to enter the normal stacker for the MFCM primary hopper and, therefore, need not have stacker selection specified. However, when no output operation is to be performed on a combined-file card type, and the desired stacker number is known at input time, a stacker-selection specification--even for the normal stacker --should be given in the input specifications: this maximizes I/O overlap. (For a single card in an entire file, this is of course insignificant.)

The file Name need not be repeated where no others intervened.

Note: The Date card is specified after the old Master cards, although it occurs first, so that the program need not attempt a match against its record-identification code each time a card is read from the OLDMASTR file (see Nature of the Card-Type Sequence Check).

Transaction Cards (Except Blank Trailers)--TRSACTN File (Page 03)

The four types are identified, and assigned separate indicators. The customer-order or merchandise-return item card is checked for digit--rather than character--9, because back orders have an X-overpunch in col. 1.

Stacker selection is dependent on calculations, and is therefore assigned in the output specifications. In the case of card type 9 (indicator 15), output to the card is also required: this precludes stacker selection in the input specifications.

Points to note:

1. Indicator 21 is turned on for order-item cards that were previously back-ordered: 11/9 in the low-order, or sole, position of a numeric field indicates a negative value. (Back-order cards are so designated at output time --page 07, line 17.)

   The field BOCARD is not used in the program; it is assigned only so that a Field Indicator may be set. Alternatively, a separate card-type Resulting Indicator could have been assigned via an OR line.

2. The same name is assigned to Stock No. here as for the OLDMASTR file, to conserve core storage space. No harm is done because there is no situation in this program where the distinction needs to be preserved.

3. When an order-item cannot be filled, and is not to be back-ordered, col. 7 of the card is overpunched with an 11-punch (page 07, line 18) to designate "cancelled." If such a card is inadvertently reentered, indicator 98 turns on because the 11-overpunch causes Stock No. to be read as negative.

4. Indicator 22 distinguishes between order-item and merchandise-return cards--both card-type Resulting Indicator 15.

5. The fields UNCOST applies only to Receipt cards. No harm is done reading it also from card types with Resulting Indicators 12, 13, and 15, because utilization in the calculation specifications is confined to card type 11 (page 05, line 06). If it were necessary to restrict the input of this field to Receipt cards, the indicator number (11) would be entered in Field-Record Relation (cols. 63-64).

Blank Trailer Card--TRSACTN File (Page 03).

The trailer cards--destined to become new Inventory Master cards--are identified by absence of a punch in col. 1, and are assigned indicator 19.

The blank trailer card at the end of each stock-number group in the TRSACTN file is not matched (no entry in Matching Fields) against the OLDMASTR file; therefore, it is processed immediately after the card it follows in the same file, before the Inventory Master card for the next Stock No.

<u>Calculation Specifications (Figure 68--Parts I, II and III)</u>

In order to minimize the need for conditioning indicators (Indicators, cols. 9-17), branching (GOTO) over entire sections has been employed to bypass a series of inapplicable calculation specifications.

Where practical, specifications lines are discussed sequentially. In some areas, however, it is preferable, for clarity, to relate non-consecutive lines.

<u>Note</u>: In several instances, result fields are defined as smaller than the theoretically possible maximum. We assumed that knowledge of the particular business indicated that these field sizes are adequate for the actual figures that could occur.

Where such cases involve multiplication or division, the RPG program will, during object-program generation, cause printing of the message "RESULT FIELD MAY NOT BE LARGE ENOUGH", prefixed by the letters C C("Cautionary" message pertaining to "Calculation" specifications) and followed by the consecutive numbers of the relevant specifications cards. Generation will, however, proceed properly.

Date Card (Card-Type Resulting Indicator 09)--Page 04, Lines 01 and 02

No calculation operations are performed on this card. Indicator 93 is turned on (line 01) solely for use in a subsequent check on proper card-type sequence (line 05). The entries in line 02 cause branching to the end of the calculation specifications (page 06, line 20), so that N09 need not be specified in Indicators in subsequent lines.

Error Control--Page 04, Lines 03-18

Calculation specifications are employed to test for certain error conditions. Where an error is recognized that affects only the individual card, calculations are bypassed for that card, and the card will be selected (by output specifications) to stacker 4; where the effect pervades the entire stock-number group, all calculations for the group are bypassed from the point of error recognition, and those cards will be selected to stacker 4. For certain error situations, the system is also halted.

Indicator 90 is set on for all of the major error conditions tested for, and is used to specify the bypassing of calculations and the selection (see output specifications) of the group to stacker 4.

<u>Specifications line 03</u> clears indicator 90 at the beginning of each control group (i.e., stock-number group), so that the error actions do not carry through to the next group.

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-3351-1
Printed in U. S. A.

Date 10-15-66

Program PRE-BILLING WITH INVENTORY CONTROL

Programmer K.B.

Punching Instruction	Graphic							
	Punch							

Page Ø4 Program Identification INVNTY

	Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Plus / High 1>2	Minus / Low 1<2	Zero or Blank / Equal 1=2	Comments
DATE CARD	01	C		Ø9		SETON						93			IDENT NEXT CARD
DATE CARD	02	C		Ø9		GOTO	END								DATE BYPAS CALC
ERROR CONTROL	03	C		L1		SETOF						9Ø			NEW GRP-CLR ERR
	04	C		L1 N91 N92		SETON						94			AND W. NXT LINE
	05	C		94 N93		SETON						9Ø	H2		MISSING BLANK
	06	C				SETOF						92	93	24	CLR 1-CYCLE IND
	07	C		L1		SETOF						91	94		SEE LINE 14
	08	C		Ø1	STKNO	COMP	OLDNO						H1	H1	DUPL MASTR HALT
	09	C		Ø1		MOVE	STKNO	OLDNO	6	Ø					PREPAR DUP TEST
	10	C		H1		SETON						9Ø			DUPLC MASTR IND
	11	C		99		SETON						9Ø			NEGATV ON HAND
	12	C		NMR NØ1 N19		SETON						9Ø			UNMATCHD TRANSA
	13	C		97		SETON						9Ø			OBSOLETE MASTER
	14	C		91		SETON						9Ø	H2		BLANK MISPOSTND
	15	C		98		GOTO	END								CANCLD ITM-ORDR
	16	C		Ø1		SETON						92			IND FOLWG MASTR
	17	C		19		SETON						91			IND FOLWG BLANK
	18	C		9Ø		GOTO	END								GRP ERROR CONDT
	19	C	*	THIS COMPLETES GENERAL CHECKS FOR PRINCIPAL ERROR CONDITIONS											
BLK CARD	2Ø	C		19		GOTO	NEWMST								SKIP TO RELEVNT

Page Ø5 Program Identification INVNTY

	Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Plus / High 1>2	Minus / Low 1<2	Zero or Blank / Equal 1=2	Comments
OLD MASTER	01	C		Ø1		MOVE	AVGCST	OLDCST	5	2					SAVE OLD AVGCST
	02	C		Ø1 MR		GOTO	END								OLD MASTR COMPL
	03	C		Ø1 NMR		GOTO	NEWMST								SKIP TO RELEVNT
STOCK ORDERS, STOCK-ADJST, RCPTS.	04	C		N15 N12	ONORDR	ADD	QTY	ONORDR					H1		UPDATE ON-ORDER
	05	C		12	AVGCST	MULT	QTY	CSTEXT	8	2					VALUE OF ADJSTS
	06	C		11	UNCOST	MULT	QTY	CSTEXT							VALUE OF RECPTS
	07	C		N15 N13	INVVAL	SUB	CSTEXT	INVVAL							UPDT VAL-RCT&AJ
	08	C		N15 N13	ONHAND	SUB	QTY	ONHAND				26	9Ø		UPDT OH-RCT&ADJ
	09	C		9Ø		GOTO	END								ERROR CONDITION
	10	C		11 26	INVVAL	DIV	ONHAND	AVGCST			H				REVISE AVGCST
	11	C		N15		GOTO	END								CRDS 5/6/7 FINI
CUST. ORDER OR RETURN	12	C			ONHAND	SUB	QTY	ONHAND					23		APPLY ORDR/RTRN
	13	C		23	ONHAND	ADD	QTY	ONHAND							RESTOR -QTY/ORD
	14	C		23 N21	ONORDR	COMP	Ø					24			ESTAB.BO.INDCTR
	15	C		23		GOTO	END								ITEM UNAVAILABL
		C													
		C													
		C													
		C													
		C													

Figure 68 (Parts I and II of III). Pre-Billing with Inventory Control, Calculation Specifications

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-3351-1
Printed in U. S. A.

Date 10-15-66

Program PRE-BILLING WITH INVENTORY CONTROL

Programmer K.B.

Punching Instruction	Graphic								
	Punch								

Page (1 2) ØG

Program Identification (75 76 77 78 79 80) INVNTY

Line (3-5)	Form Type (6)	Control Level (LØ-L9, LR) (7-8)	Indicators (Not, And, Not, And, Not) (9-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field (43-48)	Field Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Plus / Compare High 1>2 (54-55)	Minus / Compare Low 1<2 (56-57)	Zero or Blank / Compare Equal 1=2 (58-59)	Comments (60-74)
01	c													
02	c		N22	CRITQT	SUB	QTY	CSTEXT					27	27	PRICE CRITERION
03	c		22	CRITQT	ADD	QTY	CSTEXT					27	27	PRICE CRITERION
04	c				Z-ADD	PRICEA	UNPRIC	5	2					TRSFR PRICEA
05	c		27N2Ø		Z-ADD	PRICEB	UNPRIC							SUPERSEDE BY B
06	c			QTY	MULT	UNPRIC	EXTPRI	6	2					PRICE EXTENSION
07	c			QTY	MULT	AVGCST	CSTEXT							PREPR GRS PROFT
08	c			EXTPRI	SUB	CSTEXT	GRSPRO	5	2					CALC PROF/LOSS
09	c			YRTDAT	ADD	QTY	YRTDAT							UPDATE SALES/YR
10	c				GOTO	END								ORDR/RTRN FINI
11	c			NEWMST	TAG									START NEW MASTR
12	c				TESTZ		NEWITM					3Ø		NEW THIS YEAR
13	c			ONHAND	ADD	ONORDR	AVAIL	4	Ø					AVAILABLE
14	c			MINQTY	COMP	AVAIL					31			MIN EXCDS AVAIL
15	c			AVGCST	COMP	OLDCST					32	33		COST TREND
16	c			ONHAND	MULT	AVGCST	INVVAL							UPDATE INVVAL
17	c			TOTOHD	ADD	ONHAND	TOTOHD	8	Ø					TOTAL ON-HAND
18	c			TOTOOR	ADD	ONORDR	TOTOOR	8	Ø					TOTAL ON-ORDER
19	c			TOTVAL	ADD	INVVAL	TOTVAL	1Ø	2					TOTAL INV. VALUE
2Ø	c			END	TAG									BEGIN DTL OUTPT

(Left margin, lines 01–10: CUST. ORDER OR RETURN; lines 11–19: NEW MASTER & ACTIVE OLD)

Card Electro Number

Figure 68 (Part III of III). Pre-Billing with Inventory Control, Calculation Specifications

Missing blank trailer card. Indicators 90 and H2 (which will halt the system) are turned on--see lines 04 and 05--when the first card of a (stock-number) control group (L1) was not preceded by:

(a) A blank trailer card: 91 is set on in the previous cycle (in line 17) if indicator 19 was then on; or
(b) A master card (legitimate case of two successive old Inventory Master cards without intervening matching transaction cards): 92 is set on in the previous cycle (in line 16) if indicator 01 was then on; or
(c) The initial Date card: 93 turned on (in line 01) if indicator 09 was on in the last cycle.
    (Note: 93 was set in a previous cycle, because the program branches to END--and does not proceed from line 02 to 03--when indicator 09 is on.)

If none of the conditions (a), (b) or (c) applies when the first card of a control group is processed, the blank (i.e., the new inventory master) card is missing.

Indicators 91, 92, 93, and 94 are reset appropriately in lines 06 and 07 so that error conditions are not spuriously signalled in a subsequent cycle. (The reset of indicator 24 each program cycle is related to its use in line 14 of page 05 and lines 17 and 18 of page 07.)

Excess blank trailer card. Indicator 91 is turned on in line 17 if indicator 19 (blank card) is on. Next program cycle, indicators 90 and H2 are turned on if indicator 91 is still on when the instructions in line 14 are reached by the program. However, if indicator L1 (first card of control group) is on when the instructions in line 07 are reached, indicator 91 is turned off.

Thus, an error is signalled (90 and H2 are turned on) if there is no control break (L1) following a blank card (91 turned on by 19): trailer card present but not at end of group.

Duplicate master or sequence step-down. In line 08, the stock number in the old Inventory Master card is compared algebraically with that of the previous old master card. If the number is the same (duplicate master) or lower, H1 is turned on to halt the system after the card has been processed. In line 09, the Stock No. is transferred to the field OLDNO to be available as the former number when the next master card is processed.

In line 10, indicator 90 is turned on if H1 was turned on in line 08, so that all processing for the remainder of the group will be bypassed, and the cards selected to stacker 4.

Note: Because the matching fields assigned in the input specifications were defined as numeric (line 02 of page 02, and line 08 of page 03), the sequence check performed as a result of the M1 specification ignores sign. For that reason, the H1 indicator is also turned on for a negative comparison result--otherwise a duplicate is not detected if one card is positive and one negative in the stock-number field. However, indicators 97 and 98 also signal a negative stock number, but without a halt.

Obsolete old Inventory Master card. As explained with Figure 67 (Input Specifications), indicator 97 turns on if the Stock No. in the old master card is negative, signalling reentry into the operation of a previously obsoleted card.

In line 13, indicator 90 is turned on if that situation exists.

Negative on-hand in old Inventory Master card. As explained in Figure 67, indicator 99 turns on if the On-Hand field is negative at input time of the old Master card.

In line 11, indicator 90 is set on for that condition.

Cancelled order-item card. As explained with Figure 67, indicator 98 turns on when a transaction card with a negative stock-number field is read. This signals reentry of a previously cancelled order-item card.

Indicator 98 is used to specify bypassing of calculations for that card only (see line 15), and its selection to stacker 4; but the remainder of the group is processed normally because it is not otherwise affected.

Unmatched transaction cards. The specifications in line 12 cause indicator 90 to be turned on for unmatched cards (NMR), other than Inventory Master cards (N01), and other than blank trailer cards (N19) which are always unmatched.

Bypassing calculations for the error group. In line 18, the program branches to END (line 20 on page 06) when indicator 90 is on. This makes detail output the next operation, omitting all calculations below line 17 on page 04.

Line 19 illustrates use of a Comments card (* in col. 7). It will be printed during generation as punched, but otherwise it does not enter the generation process. (It is checked for proper position, based on cols. 1-6.)

Bypassing Detail-Card Operations on Master Cards--Page 04, Line 20 and Page 05, Lines 01-03

Line 20 of page 04 provides program skipping past all the specifications lines that do not apply to the new Inventory Master card (i.e., the blank trailer card). This minimizes the need for N19 specifications in Indicators in subsequent lines.

In line 01 of page 05 the Average Unit Cost from the old Inventory Master card is saved for later determination of cost trend when compared with new merchandise costs.

In line 02, all calculations are terminated for old Inventory Master cards that will be replaced by new ones (i.e., there are matching transaction cards).

In line 03, the program skips--for old Master cards that are to be retained (i.e., there are no transactions)--to the same point at which calculations are resumed for new Inventory Master cards. This permits uniform preparation of report data for both situations.

Merchandise-Receipt, Stock-Adjustment, and Stock-Order Cards--Lines 04-11 of Page 05

In line 04, the On-Order quantity is revised to reflect merchandise Receipts, new purchase Stock Orders, and cancellation of Stock Orders. Cards 5 and 7 are so coded in col. 11 that addition provides the proper algebraic operation (see Figure 63). The system is halted if the operation results in a negative On-Order quantity. (Indicator 90 is not turned on, because such an error was not deemed of sufficient significance to require bypassing of the remainder of the group.)

Line 05 provides for extending the cost of a stock adjustment, based on last-known unit cost, so that the value of the inventory may be adjusted (in line 07). A new work field (CSTEXT) is set up for the product.

Line 06 provides for the same operation as line 05, but using the specific unit cost at which new merchandise was received.

In line 07, the extended cost of an Adjustment or merchandise Receipt is algebraically subtracted from the total Inventory value of the stock item. The signs in cards 5 and 6 are appropriately coded (see Figure 63).

In line 08, the On-Hand quantity is updated to reflect Receipts and Adjustments. Indicator 90 is turned on if On-Hand has become negative; further calculations are then bypassed for that stock-number group (by entry in line 09), and the cards from this point on are selected to stacker 4 (output specifications).

In Line 10, a new Average Unit Cost is established during processing of Receipt cards, because each of these cards contains unit cost. (In lines 06 and 07 we adjusted the Inventory Value to reflect the cost of the new Receipt proportionately.)

The quotient is half-adjusted.

Division by zero is not permitted, nor meaningful. Indicator 26 (turned on in line 08 if On-Hand was greater than zero) is therefore a conditioning indicator.

Line 11 causes termination of calculations for cards 5, 6, and 7.

Order-Item and Merchandise-Return Cards--Lines 12-15 on Page 05 and Lines 01-10 on Page 06

No conditioning indicators are needed to restrict these specifications to this card type: prior entries have branched past these lines for all other card types.

In line 12, the quantity in the customer-order card is subtracted from Quantity On-Hand. Merchandise-Return cards are automatically added because they are X-overpunched in col. 11.

A merchandise Return card cannot cause On-Hand to turn negative. If On-Hand was already negative, entries in lines 08 and 09 caused branching to END. Therefore, indicator 23 turns on only for a customer order-item card containing a quantity larger than the positive or zero On-Hand quantity.

Lines 13-15 are executed only to handle the insufficient-stock situation (i.e., indicator 23 is on). In accordance with our Basic Assumptions:

a. No order-item will be partially filled;
b. No item card will be back-ordered if it was previously back-ordered;
c. No item will be back-ordered unless merchandise is on order.

In Line 13, the quantity is added back to On-Hand, to restore the prior status.

In Line 14, indicator 24 is turned on if Quantity On-Order is greater than zero (COMP operation), provided the card was not previously back-ordered (N21--see page 03, line 07). Indicators 23 and 24 determine, in the output specifications, whether the card is to be identified as Back-Ordered or Cancelled (page 07, lines 17 and 18).

Indicator 24 is turned off each cycle (see page 04, line 06) before this point is reached, because line 14 is not executed each time. Incorrect card identification in col. 1 would otherwise be punched when non-backorder cards follow a back-order card.

Line 15 causes branching to the end of the calculation specifications for order-item cards that could not be filled. The specifications in lines 02-10 of page 06 will not be executed for these cases.

On page 06, lines 02 and 03, respectively, set on indicator 27 if the customer-order or merchandise-return quantity is equal to or greater than the Criterion Quantity that qualifies for Price B.

We are only interested in the Resulting Indicator--not the actual result quantity. However, an arithmetic operation requires a result field. In order not to waste core storage space, a field only temporarily needed elsewhere (page 05)--but now available--has been utilized. A numeric Compare operation is always algebraic; therefore, a more complex routine would have had to be substituted for the ADD operation in line 03 (where QTY is negative) if COMP were to be used instead.

Line 04 places Price A into a new field, UNPRIC, which will be used for the unit-price factor in the selling-price extension.

In line 05, Price B is substituted for Price A in the UNPRIC field--but only provided the quantity in the order-item or merchandise-return card satisfied the criterion (lines 02 and 03) and provided criterion Quantity was not 0 (N20--see page 02, line 06): zero in col. 22 indicates that Price A applies in all cases.

In line 06, the quantity in the item card is multiplied by the unit price previously selected (lines 02-05). The new field, EXTPRI, will be negative for a merchandise-return card, because quantity is negative.

In line 07, cost of the item sale or return is calculated, using the Average Unit Cost as updated during processing of any stock Receipt cards (page 05, line 10). Again, the same work field (CSTEXT) is utilized, because the product is not needed beyond line 08.

In line 08, gross profit is calculated for each item card. For merchandise returns, the sign is automatically reversed: -EXTPRI - (-) CSTEXT = -GRSPRO (unless selling price is less than Average Unit Cost).

In line 09, Quantity Sold This Year To Date is updated for this item card. Returns reduce the value, because quantity in these cards is negative. Because it is possible for returns early in a year to exceed sales, provision is made for a negative total (page 11, line 17--edit word).

Line 10 terminates calculations for card 9.

New Inventory Totals - Lines 11-19, Page 06

This section contains the specifications for completing the data needed (1) to punch the new Inventory Master cards for stock numbers with transactions, and (2) to print the Inventory Status Report for all stock numbers.

No conditioning indicators are required because the program has been instructed, in earlier lines, to branch past this section --to END (line 20)--for all card types except blank trailer cards or unmatched (i.e., no transaction) old Master cards.

Line 16 is not needed when there are no transactions; but there is no harm in executing it. Although there is no change in Average Unit Cost when there are no merchandise Receipt cards in the group, line 15 (in conjunction with line 01, page 05) provides a uniform method of determining cost trend that sets the indicators appropriately regardless of whether there has been a Receipt.

Line 11 is the destination point to which the program branched from page 04, line 20 (blank trailer card) or page 05, line 03 (unmatched old Inventory Master card).

Line 12 provides for determining whether the item is new this year (X-punch in col. 47 of old Master card--see line 11, page 02). Indicator 30 will be used in output specifications to control punching into cols. 43-47, and printing in print positions 54-59.

In line 13, the available quantity (On-Hand + On-Order) is calculated for the report.

In line 14, indicator 31 is turned on if the available quantity is less than the minimum specified in the old Master card, so that this condition can be signalled by a symbol in the report (print position 120).

In line 16, the updated Inventory Value is calculated after all transactions have been processed.

Lines 17-19 contain the specifications for summing quantity On-Hand, quantity On-Order, and Inventory Value for report grand totals.

The first two serve only as audit trails and control totals--to balance out former totals with control totals for Receipts, Adjustments, Stock Orders, merchandise Returns, Back-Orders, and Order-Item cards.

The Inventory Value total is also an important figure for management.

Line 20 represents merely the destination point to which the program branched from a number of previous lines when calculations were complete. It is followed by detail-time output.

Output-Format Specifications (Figure 69--Parts I-VI)

All output is at detail time (D or H in col. 15)--except for grand totals, based on LR indicator which terminates the job after total-output time.

Old Inventory Master Cards--OLDMASTR File (Page 07, Lines 01-06)

Lines 01-03 specify different stacker selection for card types in an OR relationship:

Cards with major errors (indicator 90 on--see page 04) are selected to stacker 4; the remainder (i.e., the bulk) are selected to stacker 2 if unmatched (NMR), or stacker 1 if matched (MR). (Stacker 1--the normal stacker--need not be specified.)

Thus, when a new Master card will be created because there were transactions, the matched old Master is directed to pocket 1; if no new Master is created, it is directed to stacker 2, which will also receive the newly punched Masters for groups with transactions.

Indicator 01 is needed:

1. To prevent old Master cards of the following stock-number groups being passed through output operations--without being read--in the program cycles during which matched secondary cards are being processed (MR on); and

2. To prevent an old Master card being passed through output operations--without being read--during the detail-time output preceding the reading of the first card (at 1P time. MR is then off; thus, NMR would apply)--see RPG Program Logic, Figure 6.

3. To prevent performance of this output for the Date card (during whose processing NMR applies). The Date card was specified as requiring input only, by the stacker selection designated for it in the input specifications.

Line 04 specifies that obsolete old Inventory Master cards (which are replaced by the trailer card of the matched transaction-card group) are to receive an 11-punch in col. 7. This is the safeguard against accidental reentry of these cards next time (see indicator 97: page 02, line 02 and page 04, line 13).

Note: Indicators in File-Identification lines of card types in an OR relationship are tested in sequence: if indicator 90 is on, line 01 is applied. Therefore, N90 is not needed in the next two lines. However, in line 04, N90 is necessary, because each Field-Description line is considered separately for all card types in an OR relationship.

Transaction Cards: Receipts, Adjustments, Stock Orders, and Errors--TRSACTN File (Page 07, Lines 07-12)

Cards of groups with major recognized errors are selected to stacker 4. A previously cancelled order-item card that was inadvertently reentered (indicator 98--see page 03, line 08) is also selected to stacker 4. 15 is specified in line 08 (with indicator 98) so that additional cards following a cancelled order-item card are not also selected: indicator 98, once on, is not reset until the next transaction card other than a blank is read.

Receipt, Stock-Adjustment, and Stock-Order cards are selected to stacker 5. They could instead be directed to stacker 3 with the order-item cards and subsequently sorted apart on Card No. (col. 1).

Order-Item and Merchandise-Return Cards--TRSACTN File (Page 07, Lines 13-19 and Page 08, Lines 01-09)

By the entries in lines 13-16, Merchandise-Return cards (indicator 22 on--see page 03, line 09) are directed to stacker 5, whereas order-item cards are selected to stacker 3. (The Returns cards could also, of course, be selected to stacker 3, and subsequently sorted apart by the X-overpunch in col. 11.)

The file name need not be repeated in line 13.

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System/360

Form X24-3352-1
Printed in U.S.A.

Date 10-15-66

Program PRE-BILLING WITH INVENTORY CONTROL

Programmer K.B.

Punching Instruction								
Graphic	–	&	+	б	#			
Punch	11	12	12/6	Res.	3/8			

Page Ø7   Program Identification INVNTY

	Line	Form Type	Filename	Type (H/D/T)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators (Not)	And (Not)	And (Not)	Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
OLD MASTERS	0 1	O	OLDMASTR	D	4					9Ø	Ø1								
	0 2	O		OR	2					NMR	Ø1								
	0 3	O		OR						MR	Ø1								
	0 4	O								MR	N9Ø					7		'-'	
	0 5	O																	
	0 6	O																	
STK, ORDERS, RCPTS, ADJ, CANCL	0 7	O	TRSACTN	D	4					9Ø	NØ1	NØ9							
	0 8	O		OR	4					98	15								
	0 9	O		OR	5					11									
	1 0	O		OR	5					12									
	1 1	O		OR	5					13									
	1 2	O																	
CUST. ORDERS & RETRNS	1 3	O		D	3					N9Ø	N98	N22							
	1 4	O		AND						15									
	1 5	O		OR	5					N9Ø	N98	22							
	16	O		AND						15									
	17	O								24						1		'-'	
	18	O								N24	23					7		'-'	
	19	O								N21			DESCR			39			
		O																	

Card Electro Number

Punching Instruction								
Graphic								
Punch								

Page Ø8   Program Identification INVNTY

	Line	Form Type	Filename	Type (H/D/T)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators (Not)	And (Not)	And (Not)	Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
CUST. ORDERS & RETURNS	0 1	O								N23			UNPRIC			16			
	0 2	O								N23			EXTPRI			22			
	0 3	O								N23			UNMEAS			24			
	0 4	O								N23			WHSLOC			42			
	0 5	O								N23			GRSPRO			47			
	0 6	O								N21			STKNO			11Ø		'Ø - - -'	
	0 7	O								N21			QTY			118		' CR'	
	0 8	O								N23			WHSLOC			13Ø			
	0 9	O								N21			ACCNT	Z		2Ø8			
NEW MASTER	1 0	O		D	2					N9Ø	19								
	1 1	O														1		'Ø'	
	1 2	O											STKNO			7			
	1 3	O											ONHAND			11			
	1 4	O											PRICEA			16			
	1 5	O								N2Ø			PRICEB			21			
	16	O											CRITQT			22			
	17	O											UNMEAS			24			
	18	O											DESCR			39			
	19	O											WHSLOC			42			
		O																	

Figure 69 (Parts I and II of VI). Pre-Billing with Inventory Control, Output-Format Specifications

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System/360

Form X24-3352-1
Printed in U.S.A.

Date 10-15-66

Program PRE-BILLING WITH INVENTORY CONTROL

Programmer K.B.

Punching Instruction	Graphic							
	Punch							

Page Ø9 — Program Identification INVNTY

NEW MASTER

Line	Form Type	Filename	Type (H/D/T)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators	And	And	Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
01	O								N3Ø			LASTYR			47			
02	O								3Ø			NEWITM			47			
03	O											YRTDAT			52			
04	O											ONORDR			56			
05	O											LEADTI			57			
06	O											MINQTY			61			
07	O											AVGCST			66			
08	O											INVVAL			74			
09	O								32						75		'+'	
10	O								33						75		'-'	
11	O											DATE			8Ø			
12	O											DESCR			216			
13	O											STKNO			11Ø		'Ø - -'	
14	O											ONHAND	Z		116			
15	O											ONORDR			123		' Ø -'	
16	O											MINQTY	Z		129			
17	O											LEADTI			13Ø			
18	O											YRTDAT			139		' Ø -'	
19	O											DATE			15Ø		' / / '	
	O																	

Punching Instruction	Graphic	#	&	+	(	/	)	$
	Punch	3/8	12	12/4/8	12/5/8	0/1	11/5/8	11/3/8

Page 1Ø — Program Identification INVNTY

INVENTORY REPORT

Line	Form Type	Filename	Type (H/D/T)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators	And	And	Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
01	O	REPORT	D			3	Ø1		1P									
02	O	OR							OF									
03	O														51		'D A I L Y I N V E N T'	
04	O														7Ø		'O R Y S T A T U S'	
05	O														84		'R E P O R T'	
06	O														115		'PAGE'	
07	O											PAGE	Z		12Ø			
08	O		D			1			1P									
09	O	OR							OF									
10	O														5Ø		'Q U A N T I T Y'	
11	O														64		'QTY.SOLD'	
12	O														91		'S E L L I N G AVG.'	
13	O														12Ø		'V A L U E DATE A'	
14	O		D			1			1P									
15	O	OR							OF									
16	O														22		'STOCK # DESCRIPTION'	
17	O														46		'UM ON ON L AVAIL'	
18	O														66		'MIN. LAST YR. TO'	
19	O														95		'P R I C E UNIT U/'	
2Ø	O														12Ø		'OF WHS LAST -'	

Figure 69 (Parts III and IV of VI). Pre-Billing with Inventory Control, Output-Format Specifications

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System/360

Form X24-3352-1
Printed in U.S.A.

Date 10-15-66

Program PRE-BILLING WITH INVENTORY CONTROL

Programmer K. B.

Page 11 Program Identification INVNTY

INVENTORY REPORT

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
01	O		D					Ø2	1P							
02	O		OR						OF							
03	O												47		'HAND ORDR T(OH+OO)'	
04	O												75		'YEAR DATE STNDRD'	
05	O												94		'REDUCD C COST D'	
06	O												12Ø		'INVENTORY LOC TRNSACT M'	
07	O		D			2			19 N9Ø							
08	O		OR						NMR Ø1 N9Ø							
09	O									DESCR			24			
10	O									UNMEAS			27			
11	O									ONHAND			32		' Ø '	
12	O									ONORDR			39		' Ø -'	
13	O									AVAIL			46		' Ø -'	
14	O									MINQTY	Z		51			
15	O								N3Ø	LASTYR			59		' , Ø '	
16	O								3Ø				59		'NEW'	
17	O									YRTDAT			67		' , Ø -'	
18	O									STKNO			8		'Ø - -'	
	O															
	O															

Card Electro Number

Page 12 Program Identification INVNTY

INVENTORY REPORT

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
01	O									PRICEA			75		' Ø. '	
02	O								N2Ø	PRICEB			83		' Ø. '	
03	O									CRITQT	Z		85			
04	O									AVGCST			93		' Ø. '	
05	O								32				94		'+'	
06	O								33				94		'-'	
07	O									INVVAL			1Ø5		' , $Ø. '	
08	O									WHSLOC			11Ø			
09	O								NØ1	DATE			119		' / / '	
10	O								31				12Ø		'*'	
11	O		T		2			Ø1	LR							
12	O												21		'FINAL TOTAL OH'	
13	O									TOTOHD			32		' , , Ø '	
14	O												41		'OO'	
15	O									TOTOOR			53		' , , Ø -'	
16	O												9Ø		'INV VAL'	
17	O									TOTVAL			1Ø5		'$, , Ø. '	
	O															
Ø91	O								Ø1	TRNSDA			119		' / / '	
	O															

Card Electro Number

Figure 69 (Parts V and VI of VI). Pre-Billing with Inventory Control, Output-Format Specifications

Line 17 provides for an 11-overpunch in col. 1 of order-item cards being back-ordered--see page 05, lines 12 and 14, for indicators.

Line 18 specifies an 11-overpunch in col. 7 for order-items to be cancelled--see page 05, lines 12 and 14, for indicators.

Line 19 on page 07 and lines 01-05 on page 08 provide for punching of the pertinent data. Description (line 19) is not punched into formerly back-ordered cards (indicator 21 on--see page 03, line 07) because it is punched the first time these cards are processed. The other fields (lines 01-05) are not punched into cards now being back-ordered or cancelled (indicator 23 on)--they will be punched into the back-ordered cards when they are reprocessed, if the order-item is then filled.

Lines 06-09 provide for document printing (interpreting) on the order-item and merchandise Return cards, on an MFCM Model A1.

Warehouse location (line 08) is printed only if the item was filled, because the goods could be at a different location when new merchandise is received and the back-orders are filled.

The other three items are interpreted the first time the card is processed (to facilitate card handling), and are therefore not printed again on previously back-ordered cards.

Stock No., Quantity, and Warehouse Location are printed by print head 1; Account No. is printed by print head 2.

Stock No. (line 06) is edited with hyphens between digit positions two and three, and between the fifth and sixth (the presumed self-check digit). The third hyphen in the edit word is in the status portion and identifies a cancelled card. All leading zeros, except the first, are preserved.

Zero Suppress is used to eliminate leading zeros in Account No. (line 09).

Note: These cards hereafter contain all the information needed to:

1. Run invoices;
2. Serve as warehouse picking tickets;
3. Run sales, cost-of-sales, and gross profit reports by stock number and merchandise class.

Punching New Master Cards (Blank Trailer Cards)--TRSACTN File (Page 08, Lines 10-19 and Page 09, Lines 01-11)

The pertinent fixed data from the old Master card and the updated variable information are specified for punching as per the card layout (Figure 63).

If Criterion Quantity was 0 (indicator 20 on--see page 02, line 06), the field for Price B (line 15) is left blank. If the item is new this year (indicator 30 is on--see page 06, line 12), the single-position alphameric NEWITM field (consisting of an 11-punch) is punched into col. 47 (line 02); if the item existed last year, the five-digit numeric field LASTYR is punched into cols. 43-47 (line 01). If cost trend is up (indicator 32 is on), a plus (12/6/8) is punched into col. 75 (line 09); if it is down (indicator 33 is on), a minus (11) is punched (line 10, page 09); if there was no change in merchandise cost (indicators 32 and 33 are off), col. 75 is left blank--see page 06, line 15 for setting of indicators 32 and 33.

Line 11 (page 09) provides for punching the new date from the Date card. Thus, new Inventory Master cards contain today's date, while retained (unmatched) old ones keep the old date. Each item Inventory Master card thus contains the date of the latest transaction--actual or attempted (i.e., unfilled order-item cards).

Interpreting New Master Cards--TRSACTN file (Page 09, Lines 12-19)

Note: The card document-printing special feature is available only for the 2560 MFCM Model A1.

Various fields were chosen to be interpreted by print head 1 or 2. Note in lines 15 and 18 (edit words) that one zero will be printed even if the entire field contains zeros. Since Minimum Quantity (line 16) needs no hyphen or slashes, cannot be negative, and cannot be completely zero, we elected to eliminate leading zeros by the Zero Suppress instruction rather than an edit word.

Heading the Inventory Status Report--REPORT File (Page 10; and Page 11, Lines 01-06)

Note: Figure 65 should be referenced while reading the description of the report specifications.

Lines 01-06 on page 10 provide for the general heading of the report. This heading is printed before the first card is read (indicator 1P is on) and during overflow-output time (OF on). The form is advanced to the next carriage-tape channel-1 punch before this heading is printed, and upspaced 3 lines after printing. (For printing at overflow-output time, T in col. 15 could be used in place of D or H; however, 1P is only on at detail time; therefore, detail output time is the simplest way to handle the operation.)

The heading consists of constants, with one exception: the output field PAGE is specified. This is the only field name (as contrasted with constants) that can provide other than blank or zeros before the first card has been read (i.e., when 1P is on). The page No. 1 will be printed in the first heading line of the first page (it is not possible to start with any other value before a card has been read); it will be incremented by 1 before printing on the first line of each succeeding page. Zero Suppress is specified to eliminate leading zeros and the units position zone (C zone).

Lines 08-13 contain the specifications for the first print line of column headings, 3 lines beneath the report heading. The form is single-spaced after printing.

The column headings, too, are to appear on each page (first and overflow pages). The file name need not be repeated.

Lines 14-20 contain the specifications for the second print-line of column headings. A single space follows printing.

Lines 01-06 on page 11 take care of the third print line of column headings. After printing, the form is advanced to the next channel-2 punch.

Printing the Item Lines--REPORT File (Page 11, Lines 07-18 and Page 12, Lines 01-10).

Lines 07 and 08 specify that the data is to be printed at detail-output time (D in col. 15) while processing either:

(a) A formerly blank trailer card (indicator 19 on) that does not belong to a recognized error group (N90--see page 04; and page 05, line 08); or
(b) An unmatched (NMR) old Inventory Master card (indicator 01 on) that does not belong to a recognized error group (N90).

Thus, one line will be printed per stock number, showing the original old Inventory Master card data for items without new transactions (NMR), and the updated information where transaction cards exist.

Points to note:

In lines 11, 12, 13, 15, and 17 on page 11, the edit word is designed so that one 0 is printed when the quantity is completely zero, and a minus sign is printed for negative values in fields that can be negative. In lines 01, 02, 04 and 07 on page 12, the edit word provides for printing of .00 when the amount field is all-zero. The edit word in line 07 of page 12 also provides for a floating dollar sign.

The maximum number of leading zeros (i.e., all but one) is preserved for the Stock No., in line 18 on page 11, and hyphens separate merchandise class from the remainder of the number, and the principal number from the self-check digit.

The dates--lines 09 and 091 on page 12--are edited to be printed with slashes between Month, Day, and Year. There is no point in placing a 0 in the edit word: the date can at most have one leading zero (months 01-09), and its suppression cannot be prevented by an edit-word entry.

Line 091 on page 12 illustrates insertion of a specifications line that had been forgotten initially, by assigning it a line number sequentially between two pre-printed numbers.

Lines 15 and 16 on page 11 cause the Quantity Sold Last Year to be printed (in print positions 54-59) if indicator 30 (see page 6, line 12) is off, but the word NEW to be printed instead (in print positions 57-59) if indicator 30 is on (i.e., new item this year).

Lines 05 and 06 on page 12 provide for printing a + symbol if the cost trend is up, a - if it is down, and leaving the print position blank if there has been no change in cost since the previous report. (See page 06, line 15, for setting of indicators 32 and 33.)

Lines 09 and 091 on page 12 determine whether today's date (DATE) from the Date card or the date (TRNSDA) from the old Inventory Master card is to be printed. If there were transactions (i.e., the report data is not printed while a Master card is being processed--N01), DATE is selected; if there were no transactions (i.e, the report is based on data in the old Inventory Master card--indicator 01), TRNSDA is selected.

Line 10 on page 12 provides for printing an asterisk in print position 120 when Quantity Available (i.e., On-Hand + On-Order) is less than the Minimum Stock Quantity (see page 06, lines 13 and 14, for setting of indicator 31).

Printing the Grand Totals--REPORT File (Page 12, Lines 11-17)

The line is printed at total-output time (T in col. 15), after the last data card has been processed (LR indicator on). It must be at total-output time, because the job is thereafter terminated if the LR indicator is on. The form is upspaced 2 lines before printing, providing 3 blank lines between the last detail line and the grand totals.

INDICATOR	WHERE ASSIGNED		IDENTIFIES:
	PAGE	LINE	
L1	02 03	02 08	First card of each stock-number group (excluding Date and Blank cards)
M1	02 03	02 08	Field to be matched between files (excluding Date and Blank cards)
01	02	01	Old Master card
09	03	01	Date (constant) card
11	03	03	Stock Receipt card
12	03	04	Stock Adjustment card
13	03	05	Stock Purchase Order card
15	03	06	Customer-order item or Merchandise Return card
19	03	12	Blank trailer card at end of each transaction-card group
20	02	06	Criterion quantity is 0: only PRICEA applies
21	03	07	Back-order card
22	03	09	Quantity field negative at input
23	05	12	On-hand less than order-item quantity
24	05	14	Order-item card to be back-ordered
26	05	08	Positive On-hand after Receipt or Stock-adjustment card
27	06	02 03	Quantity in Order-item or Return card sufficient for PRICEB to apply
30	06	12	Merchandise item new this year
31	06	14	Available stock (On-hand + On-order) less than Minimum Quantity
32	06	15	Updated Average Unit Cost greater than former
33	06	15	Updated Average Cost less than former

Figure 70 (part I of II). Pre-Billing with Inventory Control, Assignment of Indicators in Figures 67-68

INDICATOR	WHERE ASSIGNED PAGE	LINE	IDENTIFIES:	
90	04	05 10 11 12 13 14	Major error: calculation for all cards of the stock number group, from the point of error detection, is bypassed, and these cards are selected to Stacker 4	
	05	08		
91	04	17	Card following Blank trailer card	
92	04	16	Card following Old Master card	
93	04	01	Card following Date (constant) card	
94	04	04	First card of control group if not following Old Master or Blank	Part of error-detection routines
97	02	02	Obsolete Old Master card: should not have been reentered	
98	03	08	Cancelled unfilled Order-item card: should not have been re-entered	
99	02	03	Old Master with negative On-hand at input: should not exist	
H1	04	08	Duplicate Old Master cards, or obsolete Old Master card	
	05	04	On-hand negative after Stock Purchase or Adjustment card	
H2	04	05 14	Missing or mispositioned Blank trailer card	

Figure 70 (part II of II). Pre-Billing with Inventory Control, Assignment of Indicators in Figures 67-68

The form is advanced to channel 1 afterwards.

Constants describing the fields are printed preceding the values.

A fixed dollar sign is used in the edit word for Inventory Value.

INVOICING

This report utilizes the order-item cards processed in the previous program example (Pre-Billing with Inventory Control), in conjunction with sold-to and ship-to name-and-address cards. The same mail-order company is assumed, with modifications to illustrate more features.

The example is deliberately kept fairly simple, its main purpose being to provide an illustration of:

1. Printing sold-to and ship-to name and address side by side, each on three lines, and each from a single card;

2. Predetermined total line;

3. Summary punching.

   The summary cards can be used for:
   (a) Accounts Receivable,
   (b) Sales report by customer,
   (c) Sales report by salesman;

4. Card-type sequence check by Sequence entry (cols. 15-16, input specifications);

5. Table look-up.

Assumptions

1. The item cards from the preceding example serve as detail cards (customer order-item cards--card 9, excluding merchandise-return cards with 11-overpunch in col. 11). They are assumed to have been sorted by Warehouse Location and Account No. after the Pre-Billing operation.

2. The heading and detail cards have been previously match-merged, so that there are no missing masters or legitimate missing details. (This match-merging could have been done by an RPG program, or with the Punched-Card Utility COLAT Program, or on a collator.)

   The card with today's date and the starting invoice number (less 1) is placed ahead of this group of cards.

   The deck is placed in the primary hopper of the MFCM.

3. Name and address are confined to three lines from a single card.

   The presence of a ship-to card is optional. When it is present, it precedes the sold-to card; when there is no ship-to card, the sold-to name and address are to be printed in both positions.

   Placing the optional ship-to card ahead improves throughput: printing of name and address can proceed during processing of the sold-to card. If the sold-to card were placed first, printing of name and address could not be commenced, when there is no ship-to card, until the first detail card is being processed--only then can the program know that no further Name-and-Address card (namely, a ship-to card) must be awaited.

4. The blank cards, which are to become summary cards, are a separate file, in the secondary hopper of the MFCM.

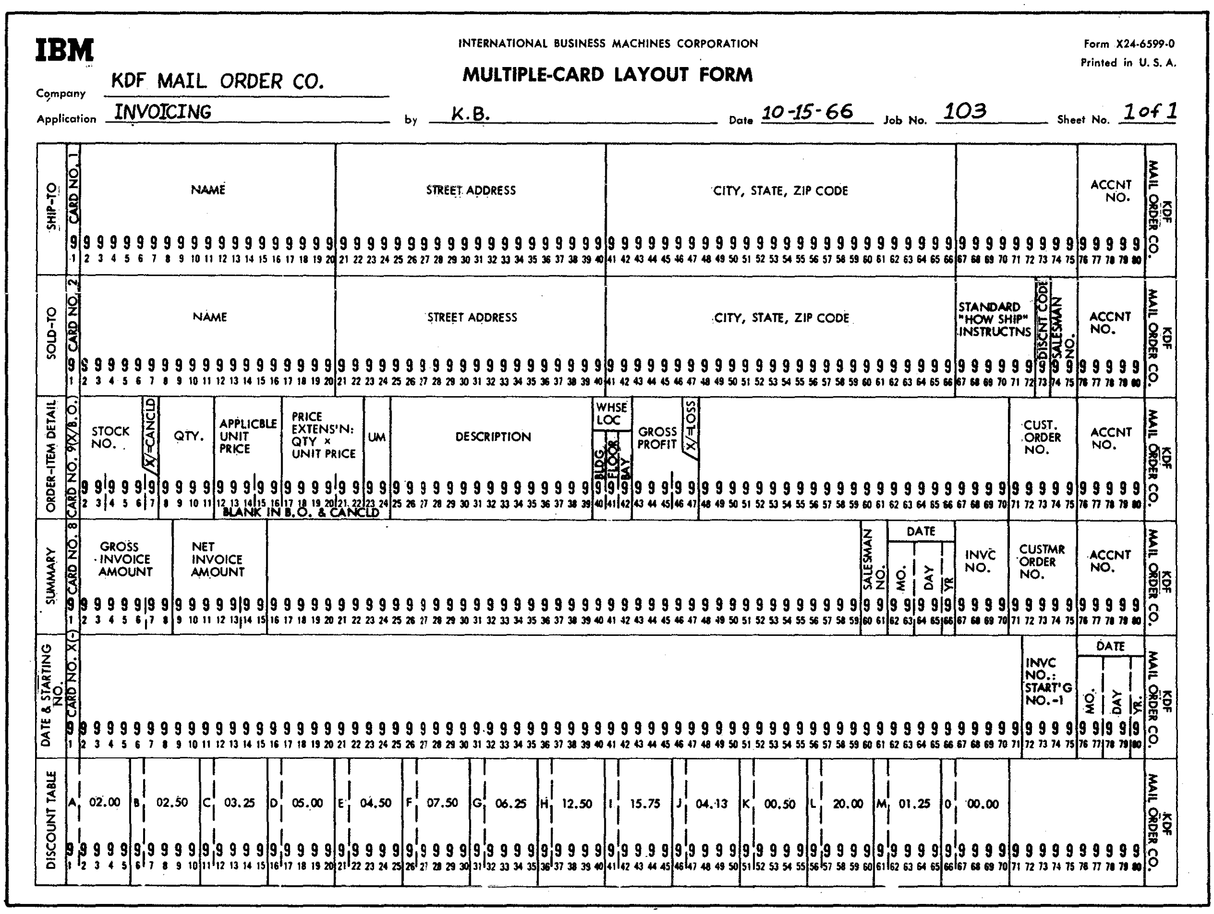


Figure 71. Invoicing, Card Layouts

5. Arbitrarily, the MFCM is used for the two files: other Model 20 I/O devices can be used if the File Description Specifications are changed.

6. Stacker Selection has been arbitrarily determined thus:

   Date and Invoice-No. heading card--stacker 1;
   Name- and Address cards--stacker 1;
   Detail cards--stacker 2;
   Summary cards--stacker 3.

7. A discount percentage is applied to the invoice total based on a customer-type code in the sold-to card. For this, table look-up is employed.

8. a. Certain identifying data is repeated on overflow pages.
   b. Invoice totals are to be printed at a predetermined point on the page.

Figure 71 presents the card layouts and Figure 72 portrays the layout of the report. Constant headings are not printed by the program, because use of a pre-printed invoice form is usual.

In the explanations that follow for the application example, most of the obvious points will be omitted, as the reader is by this time familiar with them.

File Description Specifications (Figure 73)

The input file, named INPCARDS, is associated with the primary hopper of the MFCM. It consists of one card containing the day's date and the starting invoice number (less 1) and, for each customer account number represented, contains--in this order:

One Ship-to Name-and-Address card (optional);
One Sold-to Name-and-Address card;
At least one Order-Item detail card.

A file of blank cards (named SUMCARDS), which will become the Invoice Summary cards, is to be placed in the secondary hopper of the MFCM.

The printer has been assigned the file named INVOICE.

Input Specifications (Figure 74--Parts I and II)

There is only one input file, named INPCARDS, constituted of four card types.

Date/Invoice-No. Card--Page 02, Lines 01-03

Sequence (cols. 15-16) is alphabetic, because the card appears only once, and does not fall into a sequence within each account-number group.

Stacker selection need not be specified, because 1 is the normal stacker for the primary hopper of the MFCM.

No card-type Resulting Indicator is needed: the card is never referenced, and all calculations are conditioned by indicators of the appropriate cards.

Ship-To Card--Page 02, Lines 04-08

The card, if present, is to precede all others of the group; therefore, it is Sequence number 01 (cols. 15-16). If present, only one is permitted; therefore, 1 is specified in col. 17. Its presence is optional; therefore, an O in col. 18.

Control Level 1 is assigned to customer account number--both (1) to perform end-of-invoice routines, and (2) to guard against cards out of sequence, or missing Sold-To card (see calculation specifications).

Stacker 1 is the normal stacker, and need not be designated.

Sold-To Card--Page 02, Lines 09-16

Exactly one card (1 in col. 17) of this type must be present (no O in col. 18), and it follows the Ship-To card (if this is present); therefore, cols. 15-16 contain a number higher than for the Ship-To card, but lower than for the detail cards (page 03, line 01).

Different field names are used for name and address in this card: the name-and-address data from the Ship-To card (if any) is to be printed alongside that from the Sold-To card, and must therefore be preserved at least until completion of output from the Sold-To card.

The same field name is used for ACNTNO in all cards, because the data should be the same from all cards within the group and therefore need not be saved from card to card: if it is not the same, a control break will occur (L1 is assigned to Account No.).

Indicator 20 is utilized to recognize the first detail card of each invoice--see page 06, lines 12 and 13.

Stacker 1 need not be specified.

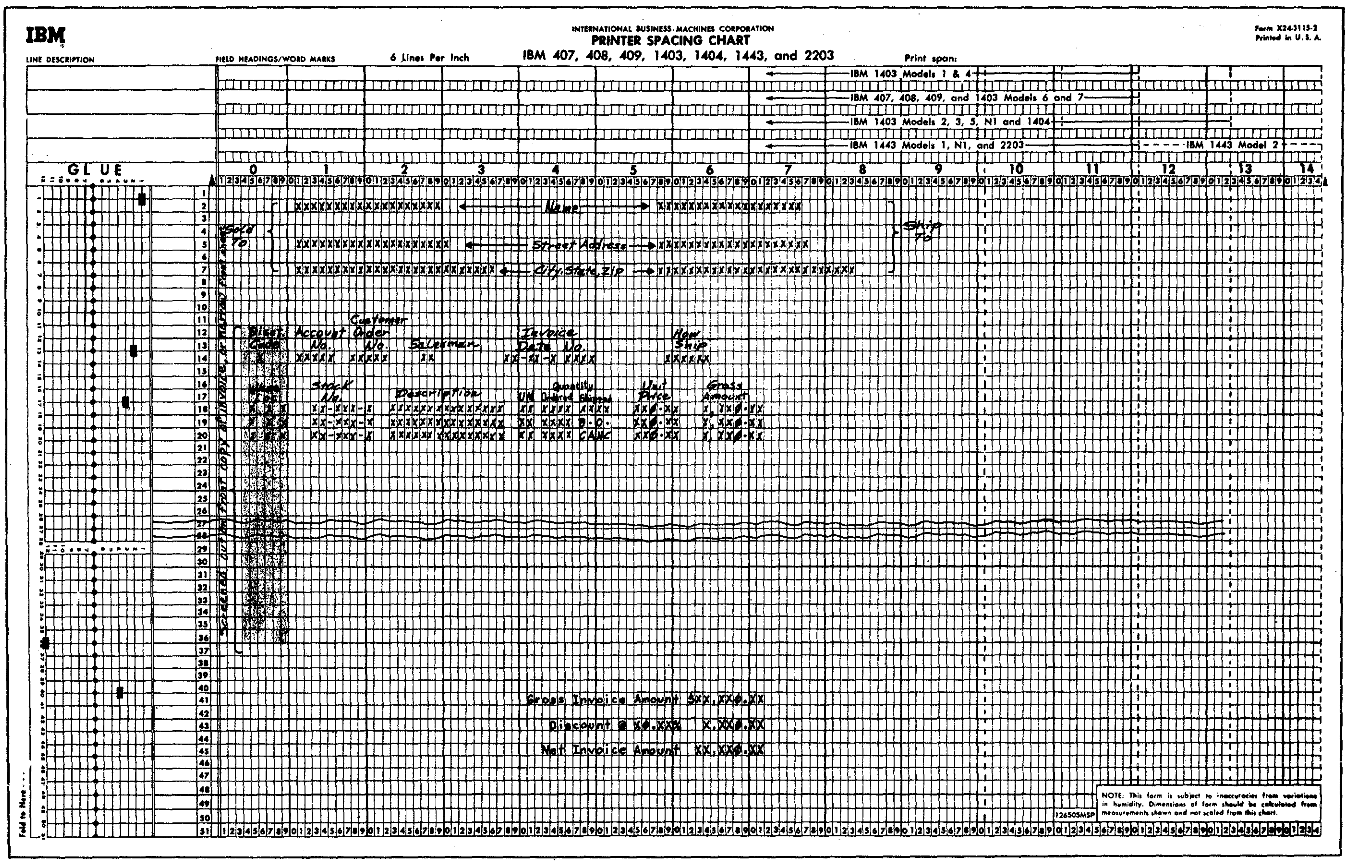
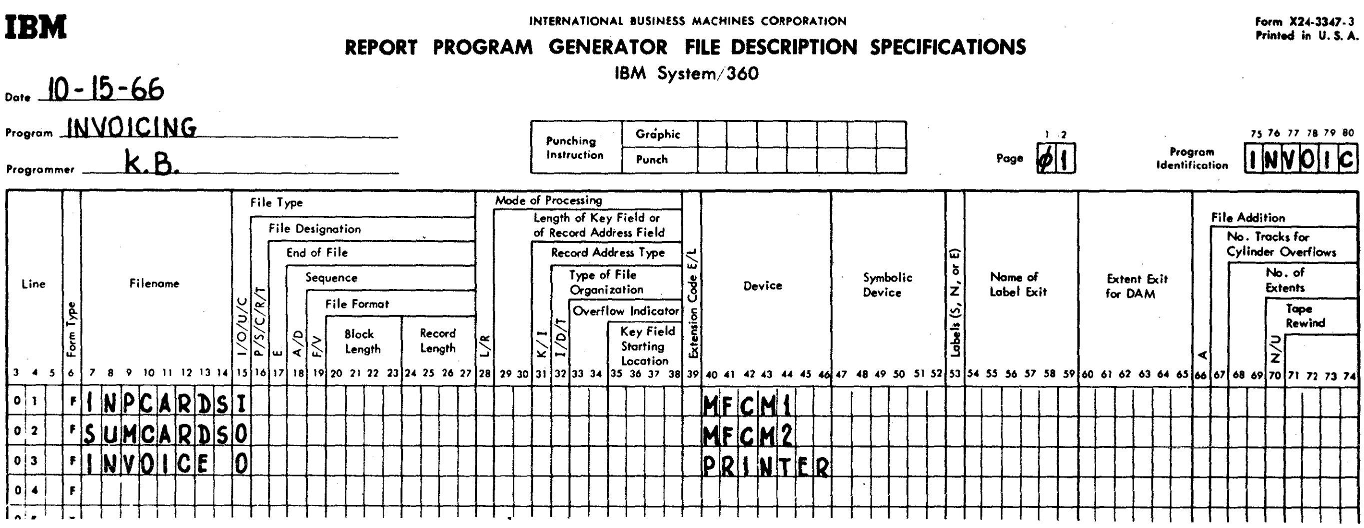


Figure 72. Invoicing, Invoice Layout



IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS

IBM System/360

Form X24-3347-3
Printed in U. S. A.

Date 10-15-66

Program INVOICING

Programmer K.B.

Punching Instruction: Graphic / Punch

Page ø1

Program Identification (75–80): INVOIC

Line (3–5)	Form Type (6)	Filename (7–14)	File Type I/O/U/C (15)	File Designation P/S/C/R/T (16)	End of File E (17)	Sequence A/D (18)	File Format F/V (19)	Block Length (20–23)	Record Length (24–27)	L/R (28)	Mode of Processing / Length of Key Field or of Record Address Field (29–30)	Record Address Type (31)	Type of File Organization I/D/T (32)	Overflow Indicator (33–34)	Key Field Starting Location (35–38)	Extension Code E/L (39)	Device (40–46)	Symbolic Device (47–52)	Labels (S, N, or E) (53)	Name of Label Exit (54–59)	Extent Exit for DAM (60–65)	A (66)	File Addition / No. Tracks for Cylinder Overflows (67–69)	No. of Extents / N/U (70)	Tape Rewind (71–74)
01	F	INPCARDS	I														MFCM1								
02	F	SUMCARDS	O														MFCM2								
03	F	INVOICE	O														PRINTER								
04	F																								

Figure 73. Invoicing, File Description Specifications

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS

IBM System/360

Form X24-3350-1
Printed in U. S. A.

Date 10-15-66

Program INVOICING

Programmer K. B.

Punching Instruction	Graphic	—
	Punch	11

Page Ø2 Program Identification INVOIC

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Resulting Indicator	Record Identification Codes 1 Position	Not (N)	C/Z/D	Character	2 Position	Not (N)	C/Z/D	Character	3 Position	Not (N)	C/Z/D	Character	Stacker Select	Packed (P)	Field Location From	To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field-Record Relation	Field Indicators Plus	Minus	Zero or Blank	Sterling Sign Position
01	I	INPCARDS	ND				1		C	-																					
02	I																				72	75	Ø	INVCNO							
03	I																				76	8Ø	Ø	DATE							
04	I		Ø1	1	O	Ø1	1		C	1																					
05	I																				2	2Ø		NAMEP							
06	I																				21	4Ø		ADDRSP							
07	I																				41	66		CTYSTP							
08	I																				76	8Ø	Ø	ACNTNO	L1						
09	I		Ø2	1		Ø2	1		C	2																					
10	I																				2	2Ø		NAMED							
11	I																				21	4Ø		ADDRSD							
12	I																				41	66		CTYSTD							
13	I																				76	8Ø	Ø	ACNTNO	L1						
14	I																				67	72		HOWSHP							
15	I																				73	73		DSCTCO						2Ø	
16	I																				74	75		SALSMN							

Figure 74 (Part I of II). Invcicing, Input Specifications

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS

IBM System/360

Form X24-3350-1
Printed in U. S. A.

Date 10-15-66

Program INVOICING

Programmer K. B.

Punching Instruction	Graphic	
	Punch	

Page Ø3 Program Identification INVOIC

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Resulting Indicator	Record Identification Codes 1 Position	Not (N)	C/Z/D	Character	2 Position	Not (N)	C/Z/D	Character	3 Position	Not (N)	C/Z/D	Character	Stacker Select	Packed (P)	Field Location From	To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field-Record Relation	Field Indicators Plus	Minus	Zero or Blank	Sterling Sign Position
01	I		Ø3	N		Ø3	1		D	9									2												
02	I																				1	1	Ø	BOCARD					21		
03	I																				2	7	Ø	STKNO					22		
04	I																				8	11	Ø	QTY							
05	I																				12	16	2	UNTPRI						24	
06	I																				17	22	2	GRSAMT							
07	I																				23	24		UNMEAS							
08	I																				25	39		DESCR							
09	I																				4Ø	42	Ø	WHSLOC							
10	I																				71	75		ORDRNO							
11	I																				76	8Ø	Ø	ACNTNO	L1						
12	I																														

Figure 74 (Part II of II). Invoicing, Input Specificaticns

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-3351-1
Printed in U. S. A.

Date 10-15-66

Program INVOICING

Programmer K. B.

Punching Instruction: Graphic = | Punch ≠

Page Ø4

Program Identification INVOIC

Line	Form Type	Control Level (L0-L9, LR)	Indicators	Indicators	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators Plus / High 1>2	Resulting Indicators Minus / Low 1<2	Resulting Indicators Zero or Blank / Equal 1=2	Comments
01	C		L1	Ø2			MOVE	NAMED	NAMEP							SHIP-TO=SOLD-TO
02	C		L1	Ø2			MOVE	ADDRSD	ADDRSP							SHIP-TO=SOLD-TO
03	C		L1	Ø2			MOVE	CTYSTD	CTYSTP							SHIP-TO=SOLD-TO
04	C		Ø3			INVGRS	ADD	GRSAMT	INVGRS	7	2					INVC GRS TOTAL
05	C		Ø2			DSCTCO	LOKUP	TABCOD	TABPRC						23	FIND DISCT PRCT
06	C		Ø2	N23			SETON						H1			NO D-CODE MATCH
07	C	L1	23			INVGRS	MULT	TABPRC	AMTDSC	6	2	H				CALC DISCT AMT
08	C	L1	23			INVGRS	SUB	AMTDSC	INVNET	7	2					CALC INVC NET
09	C	L1	23			TOTGRS	ADD	INVGRS	TOTGRS	8	2					CALC TOTL GRS
10	C	L1	23			TOTDSC	ADD	AMTDSC	TOTDSC	7	2					CALC TOTL DISCT
11	C	L1	23			TOTNET	ADD	INVNET	TOTNET	8	2					CALC TOTL NET
12	C	L1	NØ1	NØ2			SETON						H2			NO NAME/ADD CAR
13	C															
14	C															
15	C															
Ø31	C		L1			INVCNO	ADD	1	INVCNO							INCRMT INVCNO
	C															
	C															
	C															
	C															

Card Electro Number

Figure 75. Invoicing, Calculation Specifications

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR FILE EXTENSION SPECIFICATIONS

IBM System/360

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

Date 10-15-66

Program INVOICING

Programmer K.B.

Punching Instruction: Graphic | Punch

Page Ø5

Program Identification INVOIC

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table Name	Number of Table Entries Per Record	Number of Table Entries Per Table	Length of Table Entry	Packed (P)	Decimal Pos.	Sequence (A/D)	Table Name (Alternating Table)	Length of Table Entry	Packed (P)	Decimal Pos.	Sequence (A/D)	Comments
01	E					TABCOD	14	14	1				TABPRC	4		4		
02	E																	
03	E																	
04	E																	
05	E																	
06	E																	

Figure 76. Invoicing, File Extension Specifications

Order-Item Detail Cards--Page 03

Our assumptions called for selecting these cards to stacker 2: therefore, a 2 is entered in col. 42 of line 01.

The field BOCARD in line 02 is specified only to provide an indicator (21) for recognition of back-order cards (11-overpunch in col. 1, making the field negative).

If the item card was to be cancelled, because of unavailability, an 11-overpunch was punched in col. 7 in the previous application example. This makes the Stock No. (line 03) negative. Indicator 22 is utilized subsequently to control operations for cancelled items.

Unit Price, among other fields, was left blank in the previous operation whenever the item could not be filled. Indicator 24 is subsequently utilized to control operations for unfilled items.

Calculation Specifications (Figure 75--Page 04)

Lines 01-03 cause the Sold-To name-and-address data to be moved to the corresponding Ship-To fields whenever there was no Ship-To card (i.e., the first card of the control group is a Sold-To card). At output time, this will cause the same information to be printed in the Sold-To and Ship-To areas on the invoice.

Line 031 causes the Invoice No. to be incremented during processing of the first card of each Account-No. control group. (It was loaded with a value one less than the desired starting number.)

Line 04 specifies cumulation of the gross amount from each item card for an invoice total. If the item was not filled, the GRSAMT field is blank.

Line 05 causes a search through the argument table TABCOD for a code that exactly matches the Discount Code in the permanent customer Sold-To Name-and-Address card. When a match is found, indicator 23 turns on, and the discount percentage in the equivalent position of the function table TABPRC is stored and becomes available as a calculation factor and as output-field data.

The tables are defined in File Extension Specifications--see page 05.

In line 06, indicator H1 is turned on--to stop the system after this card--if no Discount-Code match was achieved.

Lines 07-12 provide for the following calculations during total time following the last detail card of each invoice:

1. The invoice gross total is multiplied by the table-supplied percent of discount to establish the discount amount (line 07)--note that half-adjustment is used, and 4 decimal positions are dropped (there are 2 decimals in INVGRS and 4 in TABPRC, since percentages less than 100 expressed as ratios fall to the right of the decimal point).

2. The discount amount is subtracted from the gross invoice amount to produce the net invoice amount (line 08).

3. The three invoice amount totals (gross, discount, net) are accumulated in three other fields, to provide grand totals (lines 09-11).

The operations in lines 07-11 are executed only when the Discount Code matched an entry in the argument table (indicator 23 on).

The specifications in line 12 set on indicator H2--and halt the system after this card--if the first card of a control group is not a Name-and-Address card (i.e., neither a Ship-To nor a Sold-To card).

Note: Since the test is made at total time (L1 in cols. 7-8), the first group will not be checked: total time is bypassed on the first card with Control Level specifications. (The test could have been programmed for detail time instead; but our approach offers the opportunity to remind the reader of the initial total-time bypass.)

File Extension Specifications (Figure 76--Page 05)

Two tables are used in this application--one as an argument table (TABCOD) and the other as a function table (TABPRC). For convenience, the two tables are punched alternately in the same card, but this has nothing to do with the manner in which they are employed (argument or function). The table cards (in this instance, a single card) must be loaded at program-generation time.

There are only 14 codes, and all fit in one card; therefore, both the number of table entries per card and per table are the same. The code is a single character (thus, 1 in col. 42), and the percentage is 4 digits long (format xx.xx %). Since the term "percent" means "per hundred", the decimal point must be moved two positions further to the left when multiplying by a percentage: thus, the field contains 4 decimal positions (not 2).

Output-Format Specifications (Figure 77 - Parts I-III)

(Refer also to Figures 71 and 72)

Note that all detail output is specified ahead of all total output.

Detail Printing on the Invoice--INVOICE File (Page 06; and Page 07, Lines 01-11).

This output is performed at detail time (D or H in col. 15). INVOICE was associated with the printer (see page 01, line 03).

Lines 01-04 on page 06 control the printing of the name on the first print line of the page. The Sold-To name (NAMED = Name solD) --read from card 2 (assigned card-type Resulting Indicator 02)--is printed in positions 11-29; the Ship-to name (NAMEP = Name shiP)--read from card 01 (indicator 01)--is printed in positions 58-76. Both are printed in the same line on the invoice form.

The printing at the beginning of each Account-No. group takes place as the Sold-To card is being processed (indicator 02 on); at that time, both the ship-to and sold-to information is available, and can be printed concurrently (if L1--instead of 02--were specified, only data from the first card of the group would be available).

The names are also repeated at the top of overflow pages, at overflow-output time (indicator OF). NL1 is specified, so that the old names are not printed at overflow time at the top of one new page--followed by the new names on the next page from card type 02--when the overflow point and the end of a group coincide.

In the calculation specifications (page 04, line 01), the Sold-To name was moved into the Ship-To name field if there was no

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System/360

Form X24-3352-1
Printed in U.S.A.

Date 10-15-66

Program INVOICING

Programmer K.B.

Punching Instruction	Graphic	/
	Punch	5/8

Page 06 Program Identification INVOIC

Line	Form Type	Filename	Type (H/D/T)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Not	Output Indicator	And Not	Output Indicator	And Not	Output Indicator	Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
01	O	INVOICE	D			3	01			OF	N	L1									
02	O		OR							02											
03	O														NAMED			29			
04	O														NAMEP			76			
05	O		D			2				02											
06	O														ADDRSD			30			
07	O														ADDRSP			77			
08	O		D			0				02											
09	O														CTYSTD			36			
10	O														CTYSTP			83			
11	O		D				02	03		OF	N	L1									
12	O		OR							03	N	20									
13	O														DSCTCO		B	6			
14	O														ACNTNO	Z		15			
15	O														ORDRNO			22			
16	O								N	OF					SALSMN			28			
17	O														DATE			44			
18	O														INVCNO	Z		49			
19	O								N	OF					HOWSHP			64			
	O																				

Card Electro Number

Figure 77 (Part I of III). Invoicing, Output-Format Specifications

Ship-To card; therefore, both names are the same in that case.

Note: If the Ship-To area on the invoice is to be left blank when there is no Ship-To card--rather than repeating the Sold-To card information--lines 01-03 on page 04 (calculation specifications) would be omitted. However, a B must then be placed in col. 39 (Blank After) of lines 04, 07, and 10 of page 06; otherwise, whenever there is no Ship-To card in a group, the data from the last preceding Ship-To card remains in storage, and will be printed.

Lines 05-07 and 08-10 provide the equivalent functions for the second and third lines of the addresses. However, the street addresses and city/state are not repeated on overflow pages.

No entry is required in cols. 17-22 of line 08, because spacing to the miscellaneous-data print line is specified in line 11. The 0 in col. 18 is entered only for compatibility with other RPGs (any entry 0-3 would satisfy that requirement).

Lines 11-12 (and, as explained below, line 13) control the conditions under which the miscellaneous data is printed above the first detail line on the invoice.

The form is skipped to the next channel-2 punch before the miscellaneous-data line is printed, and to the next channel-3 punch (first detail line) thereafter.

Note: Instead of utilizing Skip/Before in specification line 11 to reach the miscellaneous-data print line (the simplest way to program this), Skip/After--which is usually more efficient in terms of throughput--can be used in the name-and-address specifications lines. It requires several entries, however, because all three name-and-address lines are printed at the start of a new customer group, but only the name line is printed on overflow pages. The entries in cols. 17-22 (forms control)

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System/360

Form X24-3352-1
Printed in U. S. A.

Date 10-15-66

Program INVOICING

Programmer K.B.

Punching Instruction: Graphic & • , ; Punch 12

Page Ø7 Program Identification INVOIC

Line	Form Type	Filename	Type (H/D/T)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators	And	And	Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Sterling Sign Position
01	O		D			1			Ø3									
02	O											WHSLOC			9		'& &'	
03	O											STKNO			2Ø		' - - '	
04	O											DESCR			37			
05	O											UNMEAS			41			
06	O											QTY	Z		46			
07	O								N24			QTY	Z		51			
08	O								24	21	N22				51		'B.O.'	
09	O								22						51		'CANC'	
10	O								N24			UNTPRI			6Ø		' Ø. '	
11	O								N24			GRSAMT			71		' , Ø. '	
12	O	SUMCARDS	T	3					L1									
13	O														1		'8'	
14	O											INVGRS			8			
15	O											INVNET			15			
16	O											SALSMN			61			
17	O											DATE			66			
18	O											INVCNO			7Ø			
19	O											ORDRNO			75			
2Ø	O											ACNTNO			8Ø			

Card Electro Number

Figure 77 (Part II of III). Invoicing, Output-Format Specifications of specifications lines 01, 02, 08, and 11) should then read:

```
Line 01--bb  01  02
Line 02--b3  C1  bb
Line 08--bb  bb  02
Line 11--bb  bb  03
```

The miscellaneous-data line is printed after the name and address for a new group, and ahead of the first detail line. It is also printed in the same position on overflow pages (when overflow does not coincide with the end of a group); but some of the fields are not printed (NOF) on overflow pages.

Because Customer Order No. (ORDRNO) is not available until the first detail item card has been read, the miscellaneous-data line must be printed after the first detail item card has been read, yet above the regular detail data. Therefore, it is printed during processing of a detail card (indicator 03 in specifications line 12), yet before the print line for the regular detail data (see page 07, lines 01-11). But it is to be printed only before the first detail line (apart from overflow identification specified in line 11); therefore, the first detail card of a group must be identified. We chose to accomplish this as follows:

The data for the field DSCTCO is supplied by the Sold-To card, where it is never blank. When the first detail card is processed, the DSCTCO field, therefore, contains data. (One of the possible Discount Codes in this example is 0--see Discount Table in Figure 71--but 0 is treated as non-blank in an alphameric field. DSCTCO was defined as alphameric--see page 02, line 15.) Indicator 20 is on only when DSCTCO is blank (see page 02, line 15); it is therefore off when the first detail card is processed.

Specifying N20 with 03 in line 12 permits the output to be performed for the first detail card, because indicator 20 is off. As the data from the DSCTCO

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System/360

Form X24-3352-1
Printed in U.S.A.

Date 10-15-66

Program INVOICING

Programmer K.B.

Punching Instruction: Graphic $ / ; Punch

Page 08

Program Identification INVOIC

Line (3-5)	Form Type (6)	Filename (7-14)	Type (H/D/T) (15)	Stacker Select (16)	Space Before (17)	Space After (18)	Skip Before (19-20)	Skip After (21-22)	Output Indicators (Not, 23-25)	Output Indicators (And Not, 26-28)	Output Indicators (And Not, 29-31)	Field Name (32-37)	Zero Suppress (Z) (38)	Blank After (B) (39)	End Position in Output Record (40-43)	Packed Field (P) (44)	Constant or Edit Word (45-70)	Sterling Sign Position (71-74)
01	O											ACNTNO	Z		105			
02	O											SALSMN			110			
03	O											INVGRS			210		' 0. '	
04	O											INVNET			221		' 0. '	
05	O											DATE			121		' / / '	
06	O	INVOICE	T			2	04		L1									
07	O											INVGRS		B	71		'$, 0. '	
08	O		T			2			L1	23								
09	O											TABPRC			59		' 0. '	
10	O											AMTDSC			71		' , 0. '	
11	O		T			0			L1	23								
12	O											INVNET			71		' , 0. '	
13	O		T				01		LR									
14	O											TOTGRS			25		' , 0. '	
15	O											TOTDSC			45		' , 0. '	
16	O											TOTNET			65		' , 0. '	
	O																	
	O																	
	O																	
	O																	

Card Electro Number

Figure 77 (Part III of III). Invoicing, Output-Format Specifications field is transferred to the output-storage area, the Blank-After (B in col. 39) instruction causes the field to be cleared, and indicator 20 to turn on. The output controlled by the specifications in line 12 will thus never be performed again until another Sold-To card has preceded a detail card--because indicator 20 remains on until data is read into the DSCTCO field again. (The entries in line 11 provide for the output at overflow time.) The field DSCTCO was chosen because its data is not needed again in the remainder of the operations for a group.

Note: An alternate approach would be:
Change all L1 specifications to L2.
Then, specify Control Level L1 for ORDRNO (page 03, line 10).
In place of N20 on page 06, line 12, specify L1.
The B in line 13, page 06 is then not needed; nor is indicator 20 in line 15, page 02 then required.
This technique might be employed if the contents of all pertinent fields had to be preserved for summary punching.

Specifications lines 13-19 specify the data to be printed in the miscellaneous-data print line.

Although the field DSCTCO is not suppressed for overflow lines (no NOF entry), nothing will be printed from it, because it is blank at that time (see above).

Lines 01-11, page 07 contain the specifications for printing of the item detail lines.

The ampersand symbols in the edit word for WHSLCC provide blank spaces on the invoice between the three digits.

If the order item was not filled (i.e., it was back-ordered or cancelled), the Unit Price (UNTPRI) field was left blank (in the previous operation), and indicator 24 is on (see page 03, line 05). Output of Unit Price (UNTPRI) and Gross Amount (GRSAMT) is suppressed (N24) when these fields do not apply (i.e., they are blank, with UNTPRI used as the criterion to set indicator 24). Although the fields are blank at input, blank numeric fields are converted to zeros, and .00 would be printed if the output is not suppressed.

The QTY in line 06 pertains to Quantity Ordered; in line 07, it represents Quantity Shipped (see Figure 72), although the data is taken from the same field. The quantity in line 07 is therefore allowed to print only if the order item was filled (N24--UNTPRI field not blank)--it was part of the assumptions in the preceding application example that no partial fills would be made: either stock was sufficient to satisfy the quantity ordered, or the order item was not filled at all (it was then back-ordered or cancelled).

B.O. is printed in the Quantity-Shipped area on the invoice (see specifications line 08) if the order item was back-ordered and not cancelled: indicator 21 is on if the card is identified in col. 1 as a back-order card (see page 03, line 02); indicator 22 is on if the order item was cancelled (see page 03, line 03). All three indicators (24 21 N22) are needed to establish an active back order, because the item might have been previously back-ordered, and filled or cancelled in the most recent pre-billing pass (see preceding application example).

CANC is printed in the Quantity-Shipped area on the invoice (see specifications line 09) if the item was cancelled (indicator 22--see page 03, line 03).

Summary Punching--SUMCARDS File (Page 07, Lines 12-20 and Page 08, Lines 01-05)

This output is performed at total time (T in col. 15), at the end of an Account-No. control group (L1 in Output Indicators, line 12), when all totals accumulated from the cards of the group are available.

The file name SUMCARDS was associated with an output file in the secondary hopper of the MFCM (see page 01, line 02). The cards are directed to stacker 3.

Lines 13-20 on page 07 contain punch--rather than interpret--instructions, because col. 41 is blank or 0.

Lines 01-05 on page 08 contain interpreting instructions for selected fields--they are interpreting, rather than punching, specifications because col. 41 contains a print-head number (i.e., is not blank or 0).

Note 1: The interpreting feature is available only on the MFCM Model A1.

Note 2: Punching of the summary card was specified between detail and total printing to optimize throughput--generally, alternating forms printing and card punching tends to increase throughput.

Total Printing on the Invoice--INVOICE File (Page 08, Lines 06-16)

The form is first advanced to a predetermined total line (04 in cols. 19-20, specifications line 06). Three lines of totals are then printed at total time (T in col. 15) when the L1 indicator is on (i.e., after each Account-No. group). The form is double-spaced between the total lines. In specifications line 11, no entry is needed in col. 18, because forms advance before the grand-total line is specified in line 13--a zero is entered only for compatibility with other RPGs (for that purpose, any digit 0-3 is satisfactory).

Output for the second and third total lines (see specifications lines 08 and 11) is also subject to indicator 23 being on. This suppresses the discount and net amount lines when no match on Discount Code was achieved between the code in the Sold-To card and those in the argument table. While calculation of these amounts was suppressed in such case--see page 04, lines 07 and 08--.00 (not blank) would be printed for the two amount fields (because of the format of the edit words) if output were not suppressed, and a percentage figure from an earlier LOKUP operation would be printed from TABPRC.

Whenever the total in specification line 07 is transferred to the output-storage area, the field is cleared to zero (B in col. 39) to be ready for accumulation of the total for the next group. Note that the Blank-After instruction could not be entered on page 07 (SUMCARDS); otherwise, the field would be zero before output for printing.

In line 09, note the location of the decimal point in the edit word: in the file-extension specifications, TABPRC is defined as consisting of 4 decimal places, so that decimal alignment is correct when calculating the percentage amount. When printing the figure, however, it is to appear as a percentage again--the printing of a decimal point (like any other constant) has no connection with the location of the decimal point for arithmetic operations, as specified in the field definition.

Lines 13-16 control the printing of the grand totals at the end of the report (LR indicator on). The form is advanced to a new invoice page, and all three final totals are printed on the first line.

STORAGE REQUIREMENTS

The storage requirements, for both program generation and processing of the object program, depend upon the number and types of specifications used by the programmer in the source program. Approximations for the Model 20 card RPG program follow.

Program Generation

The RPG generator and the protected storage area require an approximate average of 1900 bytes. In addition, the requirements for each card punched from the specifications forms are:

1. File description card: 14 bytes

2. File extension card: 18 bytes

3. Input specifications card:
   Record identification — 7 bytes
   + 3 bytes for each card code

   Field description — 7 bytes
   + 4 bytes if the specification FIELD-RECORD RELATION and/or FIELD INDICATORS is used
   + 2 bytes if Sterling Field has an entry

4. Calculation specifications card: 5 bytes
   + 8 bytes if one or two of the fields FACTOR 1, FACTOR 2, and RESULT FIELD contain an entry
   or +12 bytes if all three of these fields are used
   + 3 bytes each time the entry in the INDICATORS field (cols. 9-17) differs from the corresponding entry in the preceding line
   + 3 bytes if resulting indicators are used
   +10 bytes for each literal whose overall length (including sign or apostrophes) is longer than six characters (See Note 1 at the end of Appendix A)

5. Output specifications card:
   File identification — 7 bytes
   + 3 bytes if output indicators are used

   Field description — 8 bytes
   + 4 bytes if the specification OUTPUT INDICATORS and/or BLANK AFTER is used
   + 4 bytes for each use of a constant or edit word
   + 1 byte for each position of a constant or edit-word field, excluding the enclosing apostrophes (See Note 1 at the end of Appendix A)
   + 2 bytes if Sterling Field has an entry

6. Defined fields:
   (See Note 1 at the end of Appendix A)
   For each field name defined in the input or calculation specifications — 8 bytes
   For each literal defined in the calculation specifications that does not exceed six characters — 8 bytes

Processing of Object Program

Nearly all available core storage can be used by the object program. The storage requirements for the object program are based upon four factors:

1. Basic routines

2. Input/output routines

3. Number of fields, literals, and indicators used

4. Processing routines

Basic Routines

The basic routines contain the general logic of the object program. Their approximate storage requirements are as follows:

Basic requirement, including the protected storage area — 1090 bytes
+ 40 bytes for using Matching Fields specifications
+120 bytes for multiple input files.

Thus, the maximum requirement for basic routines of one input file is 1130 bytes; for three input files, 1250 bytes.

Input/Output Routines

The storage requirements for the I/O routines depend upon the particular I/O units used in the program.

1. IBM 2560 Multi-Function Card Machine basic requirement	240 bytes
if both hoppers are used, add	30 bytes
for input using one hopper, add } or	140 bytes
for input using two hoppers, add	240 bytes
for punched output, add	150 bytes
for card printing, add + 64 bytes for each print head used	150 bytes
2. IBM 2520 Card Read-Punch, Model A1	
for input only	230 bytes
for input and output	390 bytes
for output only	190 bytes
3. IBM 2501 Card Reader	150 bytes
4. IBM 2520 Card Punch, Model A2 or A3	190 bytes
5. IBM 1442 Card Punch	160 bytes
6. IBM 1403 or 2203 Printer	100 bytes
for Dual-Feed Carriage, add	30 bytes

Number of Fields, Literals, and Indicators Used

Alphameric fields and literals require one byte for each position. The number of bytes for numeric fields and literals can be computed with the following formula:

$$\text{If N is odd:} \quad n = \frac{N + 1}{2}$$

$$\text{If N is even:} \quad n = \frac{N + 2}{2}$$

N = number of positions in the field or literal
n = number of bytes required for the numeric field or literal

Constants and edit words are always considered alphameric literals when determining storage requirements; but the actual length of an edit word exceeds the specified length by one or two bytes.

Core-storage space is required only once for each field, literal, constant, and edit word--regardless of how often it appears in the program.

Each entry in a table is treated like a literal: if it is alphameric, the number of bytes of core storage required is equal to the number of positions (N) in the entry; if it is defined as numeric, the number (n) of bytes required is determined by the above formula. For the entire table, the storage requirement is then:

$$S = L(K + 1) + 6$$

where
S = number of bytes needed to store entire table
L = length, in bytes, of one table entry (= N, if alphameric; = n, if numeric)
K = number of entries in the table

The 1 in (K + 1) represents the "hold" area for the value selected from the table (see LOKUP operation, under Calculation Specifications in the body of the manual).

The number of bytes required for each control level equals the total number of positions in the control field pertaining to this level. (See Note 2 at the end of Appendix A)

The number of bytes required for matching fields levels (M1, M2, M3) is computed by the following formula:

$$(N + 1) \ (M + 1)$$

where N stands for the total number of positions in the pertinent fields and M stands for the number of input files. (See Note 2 at the end of Appendix A).

The basic requirement for the special indicators (L0-L9, 1P, MR, H1, H2, OF, OV, LR) is 21 bytes total, regardless of whether they are used in the program. Any other indicators used in the program take up one byte each, once.

Note: At least 200 bytes are always reserved for indicators and fields.

Processing Routines

Processing routines contain the instructions created from the source specifications. Therefore, the storage requirements for these routines depend upon the degree of complexity of the program and the number of statements used. There are no hard-and-fast rules for the computation of these requirements.

The listing below shows the approximate requirements of the more important entries. The storage requirements for processing routines are obtained by adding up the requirements of all entries used.

1. Input Specifications
   (a) Record Identification Entries;
       Basic requirement for each main record — 22 bytes
       Basic requirement for each OR record — 14 bytes
       + 2 bytes for a non-sequential main record (alphabetic entries in column 15-16)
       + 8 bytes for test of record identification code "C"
       or +14 bytes for test of record identification code "D"
       or +12 bytes for test of record identification code "Z"
   (b) Field Description Entries
       Alphameric fields — 6 bytes
       Numeric fields — 12 bytes
       + 8 bytes for field-record relation if it differs from that in the previous line
       +18 bytes for first field indicator
       +12 bytes for second field indicator
       +12 bytes for third field indicator
   (c) Control Levels and Matching Fields:
       For each file with matching fields — 14 bytes
       For each control level used — 14 bytes
       For each record that contains split control fields — 4 bytes
       or
       For each record that contains split control fields with field-record relation — 12 bytes
       For each record that contains matching fields * — 4 bytes
       For each record that contains unsplit control fields * — 4 bytes
       For each control field or matching field entry * — 6 bytes
       * (See Note 3 at the end of Appendix A)

2. Calculation Specifications
   For ADD/SUB:
   If
   (a) the same name is used for one factor field and the result field,
       <u>and</u>
   (b) the length of the other factor is equal to or shorter than the length of the result field,
       <u>and</u>
   (c) the number of decimal places is equal for the fields — 6 bytes

   For three operands, other than (a) and (b), above — 36 bytes
       + 6 to 32 bytes if the number of decimal places differs between the fields.

   For Z-ADD:
   If the number of decimal places in the fields is
       equal — 6 bytes
       unequal — 24 to 44 bytes

   For Z-SUB:
   If the number of decimal places in the fields is
       equal — 18 bytes
       unequal — 30 to 50 bytes

   For MULT:
       without decimal alignment ($D_1 + D_2 = Dr$) — 30 bytes
       with decimal alignment ($D_1 + D_2 \neq Dr$) — 36 to 46 bytes
       (See Note 4 at the end of Appendix A)

   For DIV:
       without decimal alignment ($D_1 - D_2 - H = Dr$) — 36 bytes
       with decimal alignment ($D_1 - D_2 - H \neq Dr$) — 46 to 52 bytes
       (See Note 4 at the end of Appendix A).

   For MVR:
       without decimal alignment ($Dm = Dr$) — 12 bytes
       with decimal alignment ($Dm \neq Dr$) — 18 to 28 bytes
       (See Note 4 at the end of Appendix A).

For MOVE:

From alphameric to alphameric field	12 bytes
From numeric to numeric field	12 to 18 bytes
From numeric to alphameric field	12 bytes
From alphameric to numeric field	18 to 24 bytes

For MOVEL:

From alphameric to alphameric field	12 bytes
From numeric to numeric field	24 to 30 bytes
From numeric to alphameric field	12 to 18 bytes
From alphameric to numeric field	24 to 30 bytes

For MLLZO, MLHZO, MHLZO, MHHZO:

From alphameric to alphameric field	12 bytes
From numeric to numeric field	12 bytes
From numeric to alphameric field	18 bytes
From alphameric to numeric field	24 bytes

For COMP:

numeric fields--	
If the number of decimal places is equal for the fields	30 bytes
If the number of decimal places differs between the fields	36 to 40 bytes
alphameric fields--	
If both fields are of equal length	12 bytes
If field lengths are unequal	28 bytes

For EXIT:	4 bytes
For RLABL:	0 bytes
For GOTO:	4 bytes
For TAG:	0 bytes
For LOKUP:	60 to 108 bytes
For TESTZ:	24 to 54 bytes

If indicator(s) assigned to:

Plus, Minus, and Blank	54 bytes
Plus and Minus	46 bytes
Plus or Minus, and Blank	42 bytes
Plus or Minus	24 bytes
Blank	34 bytes

For SETOF, SETON:

basic	10 bytes
+ 4 bytes for second indicator	
+ 4 bytes for third indicator	

For half-adjusting	6 to 16 bytes
For each different resulting indicator within one line	12 bytes
Testing conditioning indicators assigned in INDICATORS (cols. 7 to 17)--each indicator	8 bytes

However, no storage at all is consumed if
(a) all the same indicators appear in the preceding line in INDICATORS, in identical form and order, and
(b) none of the same indicators are specified in RESULTING INDICATORS (cols. 54 to 59) in the preceding line.

3. Output-Format Specifications
(a) File Identification and Control:

Basic requirement for each main record	8 bytes
Basic requirement for each OR record	4 bytes

+ 4 bytes for each stacker select or space and skip entry, but not if the OR record was preceded by another record containing the same stacker select or space and skip entry
+ 6 bytes for card printing
+ 8 bytes for each output indicator
+ 2 bytes for card punching

(b) Field Description Entries:
Basic Requirement 6 bytes
+ 8 bytes for zero suppress
+ 8 bytes for editing
+ 8 bytes for each output indicator
+ 6 bytes for Blank After (numeric field)
+10 bytes for Blank After (alphameric field)
+ another 4 bytes for Blank After if a Zero-or-Blank indicator is involved
+ 6 bytes for each line with field name PAGE, and an additional 6 bytes if output indicators are specified in such line.

TIMING FOR THE RPG PROGRAM

Generation of Object Program

The time required for generating the object program is estimated by the number of lines written on the specifications sheets. The first 50 specifications lines require about:

3.5 minutes with the
2560 Multi-Function Card Machine, Model A1
2501 Card Reader, Model A1
2520 Card Read-Punch, Model A1
2.5 minutes with the
2501 Card Reader, Model A2
5.5 minutes with the
2560 Multi-Function Card Machine, Model A2

Each additional 25 specifications lines require about:

0.5 minutes with
input devices attached to an IBM System/360 Model 20, Submodel 1, 2, or 5, or
0.8 minutes with the
2560 Multi-Function Card Machine, Model A2.

The time required to punch out the object-program deck at the end of the generation run is:

60 seconds with the
2560 Multi-Function Card Machine, Model A1
80 seconds with the
2560 Multi-Function Card Machine, Model A2
12 seconds with the
2520 Card Read-Punch, Model A1 or
2520 Card Punch, Model A2
20 seconds with the
2520 Card Punch, Model A3
70 seconds with the
1442 Card Punch, Model A1

Note: The times given above refer to a core-storage capacity of 4096 bytes.

Processing of the Object Program

The time required to process the object program depends upon the complexity of the specifications and the particular I/O units involved. A precise timing calculation of a specific RPG object program requires detailed knowledge of the RPG generator. No simple rules for timing can be used.

Note 1

When determining the core-storage requirements for literals, constants, edit words, and field names, each is counted only once--regardless of how often it appears in the program.

Note 2

Fields used for control levels and/or as matching fields are stored separately for these purposes--apart from their storage for calculations, output, etc. The just-stated storage requirements refer only to the control-level and matching-fields operations. For these purposes, each position is counted, in numeric as well as in alphameric fields: numeric fields are not packed.

Note 3

Does not apply if the record was preceded by another record containing the same fields (unsplit control fields and/or matching fields) in the same columns.

Note 4

D1 = number of decimal places in Factor 1
D2 = number of decimal places in Factor 2
Dr = number of decimal places in Result Field
Dm = number of decimal places in Remainder
H = 1, if Half-Adjust specified;
0, if Half-Adjust not specified

APPENDIX B. MACHINE UNITS AND FEATURES REQUIRED AND SUPPORTED

The Report Program Generator requires a minimum of machine units to generate an object deck or to process an object program. These are called required units. Many features and units of the IBM System /360 Model 20 can be utilized in the Model 20 RPG, even though they are not required for object deck generation or for object program processing. These are called supported units and features. The required and supported units and features for the Model 20 card RPG are itemized below. Model 20 CPS RPG does not use 24,576 or 32,768 bytes of main storage.

MACHINE UNITS REQUIRED

Generation of Object Program

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

- 4096 bytes of core storage,
- One card-reading device (if the 2501 Card Reader is attached to the system, it must be used).

Processing of Object Program

The minimum machine requirements for execution of the RPG object program are as follows:

- 4096 bytes of core storage
- Input/Output devices as specified for the object program.

MACHINE UNITS AND FEATURES SUPPORTED

Generation of Object Program

The following machine units and features are supported for program generation, in addition to the required units:

- Additional 4096, 8192, or 12,288 bytes of core storage
- One printer with at least a 48-character set, for program listings
- A second card-reading device (if the 2501 is attached to the system, it must be used as one of the card-reading devices).
- A card-punching device, if the object program is to be punched.

Processing of Object Program

The following machine units and features can be utilized during the processing of object programs (the particular units that can be attached to the system depend on the submodel of the IBM System/360 Model 20 that is used).

- IBM 2020 Processing Unit with 4096 (minimum requirement), 8192, 12,288, or 16,384 core-storage bytes. Programs compiled for an IBM System/360 Model 20 Submodel 1, 2, 3, or 4, will run on a 24K or 32K Submodel 5.
- One printer with up to 144 print positions.
- Dual-Feed carriage for the IBM 2203 Printer.
- Card-Printing special feature for the IBM 2560 Multi-Function Card Machine, Model A1.
- One, two, or three input files.
  - a. One input file:
    2560 MFCM, hopper 1, or
    2560 MFCM, hopper 2, or
    2520 Card Read-Punch, or
    2501 Card Reader
  - b. Two input files:
    2560 MFCM, hoppers 1 and 2, or
    2560 MFCM, hopper 1 and 2501 Card Reader, or
    2560 MFCM, hopper 2 and 2501 Card Reader, or
    2520 Card Read-Punch and 2501 Card Reader
  - c. Three input files:
    2560 MFCM, hoppers 1 and 2, and 2501 Card Reader
- One, two, or three card output files:
  - a. One card output file:
    2560 MFCM, hopper 1 or 2, or
    2520 Card Read-Punch, or
    2520 Card Punch, or
    1442 Card Punch
  - b. Two card output files:
    2560 MFCM, hoppers 1 and 2, or
    2560 MFCM, hopper 1 and 1442 Card Punch, or
    2560 MFCM, hopper 2 and 1442 Card Punch, or
    2520 Card Read-Punch and 1442 Card Punch, or
    2520 Card Punch and 1442 Card Punch
  - c. Three card output files:
    2560 MFCM, hoppers 1 and 2, and 1442 Card Punch.

After the programmer has written the specifications, and before the source deck is keypunched from them, he should thoroughly "desk check" the program. Desk checking consists of a visual check of the specifications sheets for obvious mistakes, and may also include a "manual run" of data records through the program. Desk checking can eliminate many errors in a new program.

The following are suggestions of items to check which, experience indicates, tend to be sources of error. It is also an excellent idea to review two other appendices as reminders of points that must not be overlooked when writing a program:

Appendix G - Summary of RPG Specifications, and
Appendix H - RPG Program Listing (diagnostic messages)

FILE DESCRIPTION SPECIFICATIONS

1. File names must be left-justified.
2. File type must be I, O, or C.
3. DEVICE must contain a valid code.
4. SEQUENCE (A or D) must be assigned if MATCHING FIELDS in the input specifications contains an entry, and it must be the same for all input and combined files.

INPUT SPECIFICATIONS

1. The first line must be a record identification line.
2. Record identifications (cols. 7-42) and field descriptions (cols. 43-74) must not be specified in the same line.
3. File names must refer to input or combined files.
4. File and field names must be left-justified.
5. Every main record-identification line must have a SEQUENCE entry.
6. Any alphabetic SEQUENCE entry in cols. 15-16 must precede any numeric entry.
7. The first Numeric SEQUENCE must be 01 in cols. 15-16.
8. Numeric SEQUENCE entries must have 1 or N in NUMBER.
9. FIELD LOCATION--From and To--must be within the limits 1 to 80.
10. A field defined as numeric must not be specified as greater than 15 positions.
11. Field length, format (alphameric or numeric), and number of decimal places (if numeric) must be identical every place that the same field might be redefined, anywhere in the program.
12. The number of decimal places specified for a numeric field must not exceed the field length. (Exception: packed input.)
13. Field Indicators must not be specified in PLUS or MINUS for alphameric fields.
14. There must be an entry (M1, M2, or M3) in MATCHING FIELDS for at least one card type in each input and combined file when there is more than one input or combined file.
15. The highest level for Matching Fields is M3.
16. A Matching-Fields level (M1, M2, or M3) cannot be split.
17. The total number of positions for one Matching-Fields level or one Control Level must be uniform in all records with which it is used.
18. a. Control Levels must be specified in ascending sequence.

    b. The aggregate length of a split Control Level must be uniform.
19. Remember that Field Indicators assigned to ZERO-or-BLANK are on at the beginning of program execution, and until data is read into the field or the indicator is turned off by a calculation specification. They also turn on when the field is cleared by a Blank-After instruction in output specifications.
20. Do not specify stacker selection on input for a combined-file card type for which punching or document-printing is to be performed.

21. If PAGE is specified as an input field, it must be defined as numeric, and 4 positions long. It cannot be read in packed format.

22. Note that card punch-combination 12-6-8 = + for literals--not 12 (=&).

CALCULATION SPECIFICATIONS

1. All detail-time calculation specifications must precede those for total time (Lx in cols. 7-8).

2. Operation codes must be left-justified, and worded exactly as described in the manual.

3. Field names and literals must be left-justified.

4. Alphameric literals must be enclosed in apostrophes, and numeric literals must not be.

5. Field length, format (alphameric or numeric) and number of decimal places (if numeric) must be identical every place that the same field might be redefined, anywhere in the program.

6. A field defined as numeric must not be specified as greater than 15 positions.

7. The number of decimal places specified for a numeric field must not exceed the field length.

8. An alphameric field must never exceed 256 positions. An alphameric field used in a LOKUP operation is limited to a length of 80 characters; in a COMP operation, it must not be longer than 40 characters.

9. An operation (except certain Move operations) must not involve both alphameric and numeric fields. (However, the function table in LOKUP may differ in format from the argument table.)

10. All arithmetic operations require numeric fields, and the data must consist of valid digits (0-9 only, plus a sign in the low-order position).

11. TESTZ requires an alphameric field.

12. An entry in RESULTING INDICATORS is mandatory in COMP, LOKUP, SETON, SETOF, and TESTZ operations.

13. Factor 1 and Factor 2 must have the same field length in LOKUP operations.

14. There is no automatic decimal alignment in LOKUP or Move operations.

15. The field names in Factor 2 and in Result Field (if used) in a LOKUP operation must start with TAB.

16. A table name in an RLABL line must be exactly four characters long, the first three being TAB.

17. A RESULT FIELD name may not begin with TAB unless the operation code is LOKUP or RLABL.

18. RESULTING INDICATORS must be blank for all Move operations, and for GOTO, TAG, EXIT, and RLABL operations.

19. Half-Adjust must not be used with a DIV operation if an MVR operation follows.

20. The pertinent DIV-operation specifications must be in the line immediately preceding an MVR operation.

21. An MVR-operation line must have identical entries in INDICATORS (cols. 7-17) as the preceding DIV-operation line.

22. Remember that total-time calculations are bypassed on the first card and--if control levels are specified--until after the first card of a type with Control Level specified.

FILE EXTENSION SPECIFICATIONS

1. Table names must begin with TAB, and must be four to six characters long. If the table name is to be referenced in a B.A.L. subroutine, it must be exactly four characters long, including TAB as the first three.

2. PACKED (cols. 43 and 55) must be left blank.

3. If table LOKUP involves a RESULTING INDICATOR in HIGH or LOW, SEQUENCE (A or D) should be specified.

OUTPUT-FORMAT SPECIFICATIONS

1. All detail time output (D or H in col. 15) must be specified ahead of all total-time output (T in col. 15).

2. The first line must be a file-identification line.

3. Each main file-identification line must have an entry in TYPE (col. 15).

4. File-identification and field-description must not be specified in the same line.

5. File names must refer to output or combined files.

6. File and field names must be left-justified.

7. Each field-description line must have an entry in END POSITION IN OUTPUT RECORD.

8. ZERO SUPPRESS must be blank if CONSTANT-or-EDIT WORD contains an entry, and vice versa.

9. ZERO SUPPRESS can only be specified for a numeric field.

10. Constants and edit words must be left-justified, and enclosed in apostrophes.

11. An edit word can only be specified for a numeric field.

12. A constant (cols. 45-70) and a field name (cols. 32-37) are mutually exclusive.

13. An edit word can cause a field to consume more space in the output record than the defined field length. Therefore, when specifying END POSITION IN OUTPUT RECORD, make sure not unintentionally to overlay portions of successive fields.

14. a. Each time PAGE is used in output, it is first incremented by 1; therefore, do not use it in several output file-identification groups unless the number is supposed to advance each time.
    b. PAGE is always 4 positions long, numeric, and the units position is zoned. Normally, therefore, Zero Suppress or an edit word should be used.
    c. Entries in OUTPUT INDICATORS of a line with field name PAGE do not condition the output; instead, when the indicator conditions are satisfied, they cause the contents of the PAGE field to be reset to 0, before the usual 1 is added prior to output.

15. Output to a file occurs each program cycle (at detail or total time--depending on the entry in col. 15), unless indicators are entered in OUTPUT INDICATORS, and the specified indicator conditions are not satisfied (i.e., an indicator preceded by N is on, or one not preceded by N is off). Therefore, three cautions are in order:

a. Output will occur at detail-output time <u>before</u> the first card has been read (see Figure 6, RPG Program Logic) unless conditioned by the negative status (Nxx) of an indicator that is then on (e.g., 1P), or by the positive status (bxx) of an indicator that is then off (e.g., a card-type Resulting Indicator). A common error is the conditioning of an output file by only NMR--and, of course, the MR indicator is off in the beginning. (Indicator Hierarchy--Figure 11--shows which indicators are on in the beginning.)

   Printing before the first card has been read is normal for constants used as report headings, and for page number (PAGE) if it is to start with 1; but output from data fields would be blank or zero. Usually, the desired printing at that time is conditioned by indicator 1P.

   Punching of combined-file cards before the first card has been read will completely disrupt the program.

b. Indicators assigned to ZERO-or-BLANK in FIELD INDICATORS (input specifications) or in RESULTING INDICATORS (calculation specifications--arithmetic and TESTZ operations) are on at the beginning of program execution, and after execution of field-description specifications which include Blank-After (B in col. 39).

   Care is therefore required not to affect output inappropriately because of initial or premature ON status of an indicator.

c. In a file-matching operation, a card will be processed (for output only) from each file during the same program cycle when the files match, if the only Output Indicator specified is MR.

   This means, for instance, that--when a matched secondary-file card is being processed--a primary-file card belonging to the <u>next</u> group is also processed for output; but it is never read.

Therefore, the output for all card files being matched should always also be conditioned in OUTPUT INDICATORS by their card type (or by the negative--N--of the other card types), besides the MR indicator condition.

(See <u>Program Examples, Pre-Billing with Inventory Control.</u>)

16. When there are AND or OR lines, each file-identification line in the group must have Output Indicators specified.

17. A Blank-After instruction (B in col. 39) causes the field to be cleared (to blank if alphameric, to zeros if numeric) as soon as the data has been transferred to the output-storage area.

    If the same field is used for output in several field-description lines, be careful to place the B in the last line executed. Normally, lines are executed--within detail-, total-, and overflow-output time--in the order in which they appear. However, when punching and card document-printing (interpreting) are both specified under the same file-identification line, transfer to the output-storage area for punching precedes transfer for interpreting--regardless of which specification appears first.

18. Stacker selection must not be specified for the same card type (of a combined file) in both the input and output specifications.

    If an output operation is to be performed, stacker selection--if any--must be specified in the output specifications.

19. Remember that overflow-output time is distinct from detail- or total-output time. If OF (or OV) appears in OUTPUT INDICATORS of a file-identification line, and OF (or OV) is on after total-time output, the output specified under that file-identification line is performed at overflow-output time.

    Therefore, be careful--if an OR line calls for the same output under another condition (e.g., L1)--that the output does not occur twice in the same cycle. (Usually, the OF line is also made contingent upon the negative of the other condition--say, NL1).

20. Bear in mind that total-time output is bypassed in the first program cycle and--if Control Levels are specified--until after the first card of a type for which Control Level is specified.

21. On run-in, a specified Control Level indicator does not turn on until the first card with non-zero (alphameric field) data, or non-zero and non-blank (numeric field) data, in the control field has been read.

22. Forms movement is suppressed when cols. 17-22 in the file-identification line are blank (except in OR lines when a preceding line specifies forms control).

Although this appendix may be of general interest, the programmer need be familiar with it only if:

1. Input or output data is in packed format; or

2. An operation is dependent on sequence, and the cards are in a non-standard sequence; or

3. Uncommon characters or zones are involved in the data and operations.

The information provided here is in condensed and non-technical form. Greater detail, together with more technical data, can be found in the SRL publication *IBM System/360 Model 20 Functional Characteristics* , Form A26-5847.

CODE STRUCTURE

The Basic Character Unit

Characters are normally stored and manipulated in the System/360 Model 20 central processor in basic units called bytes. A byte consists of eight bits (binary digits).

While a decimal digit can assume ten different values (0-9), a binary digit can only take on two alternate states: 0 and 1. Accordingly, eight bits provide for a maximum of 256 (i.e., 2^8) unique entities, which is the number of different characters that System/360 can recognize and handle.

This set of 256 unique characters representable by a single byte is termed the Extended Binary-Coded-Decimal Interchange Code (EBCDIC).

Hexadecimal Notation

A byte may be thought of as being subdivided into two half-bytes: an upper and a lower half-byte. Each half-byte consists of four bits. Four bits permit 2^4 permutations, so that a half-byte can represent 16 different characters.

This offers several benefits--among them:

1. A half-byte is adequate for the storage of a digit (0-9)--see *Data Formats*, below; and

2. Any EBCDIC code can be referenced (e.g., in machine-language programming) by no more than two characters.

The symbols chosen to express the 16 permutations of bits in a half-byte in System/360 are the digits 0-9 plus the letters A-F. This is known as a hexadecimal code--forming "sixteen" from the Greek for "six" and the Latin for "ten". (The system itself converts between hexadecimal and bit representation.)

Since each upper half-byte value may be associated with every lower half-byte value, a 16 x 16 matrix results, yielding 256 unique bytes.

Comparison of Hexadecimal and Decimal Notations

In the decimal notation, carry-over to the next position occurs at each multiple of ten; in hexadecimal notation, it occurs at each multiple of sixteen. The difference is illustrated below:

Decimal	Hexadecimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F
16	10
17	11
.	.
.	.
.	.
31	1F
32	20
.	.
.	.
.	.
255	FF

This shows how two hexadecimal characters can represent the decimal range from 0 to 255.

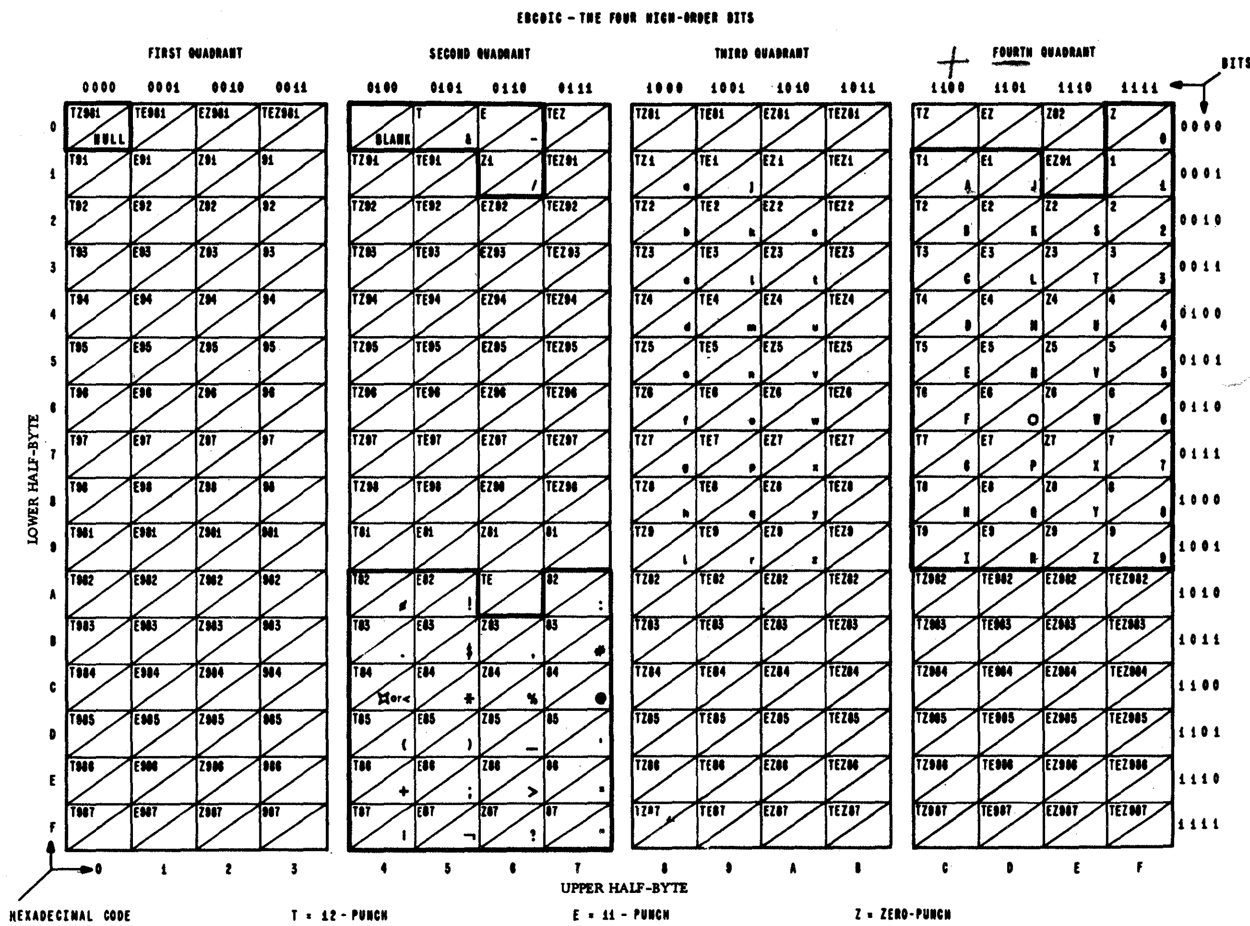


Figure D1. Hexadecimal Codes, Bit Structure, Card Codes, and Assigned Standard Graphics

<u>The EBCDIC Table</u>

Figure D1 presents the 256 unique EBCDIC characters. It is organized as a matrix, with the column labels pertaining to the upper half-bytes, and the row labels applying to the lower half-bytes.

Along the top and down the right side appear the sets of four bits that represent each half-byte internally in the system. The equivalent single-character hexadecimal codes are shown along the bottom and down the left side.

The rectangle at the intersection of each column and row indicates

(a) Above the diagonal line within the rectangle: The card punch-combination that corresponds to the particular EBCDIC character. (T = 12-punch, E = 11-punch, Z = 0-punch.)

The transformation between card punches and internal binary representation is performed automatically by the central processor of the system.

(b) Below the diagonal line: The graphic that is normally associated with that EBCDIC character.

To date, only the more common graphics have been assigned as standard for specific EBCDIC characters.

COLLATING SEQUENCES

Refer to the EBCDIC table (Figure D1) in connection with the discussion below.

<u>Standard Collating Sequence</u>

The system assumes the standard ascending collating sequence to run from hexadecimal code 00 to FF (and the converse for descending sequence). The sequence extends through all rows of a table column before proceeding to the top of the next column, thus:

column 0, row 0; column 0, row 1; 02, 03, ...; column 0, row F;

column 1, row 0; column 1, row 1; 12, ..., 1F, 20, ..., FD, FE, FF.

Hence, for example, in ascending sequence:

1. The period symbol (.) precedes (i.e., is lower in sequence than) the exclamation mark (!)

2. The exclamation mark (!) is lower in sequence than the 11-punch (-).

3. Punch combination EZ987 precedes digit 0.

4. All special-character graphics, in standard assignments, precede the alphabet.

5. The alphabet precedes (i.e., is lower in sequence than) the decimal digits.

(If descending sequence is specified, the converse applies.)

Thus, the standard ascending collating sequence for the most commonly used characters is:

blank, &, -, /, A-Z, 0-9.

(The converse applies to descending sequence.)

Altered Collating Sequence

RPG permits the user to substitute any desired collating sequence for the IBM System/360 standard collating sequence (see above). Among likely reasons for such a substitution could be the use of ASCII (American Standard Code for Information Interchange).

Operations to Which Applicable

When a user-specified sequence has been substituted, it applies during object-program execution to the collating sequence in:

1. Alphameric COMP (Compare) operations; and

2. Matching Fields operations (matching and sequence checking).

Note: When numeric Matching Fields are specified, characters in hexadecimal column F should not be converted to characters in hexadecimal columns 0-E.

When an altered collating sequence has been provided to the program, it applies uniformly to all of the above operations--it cannot be confined to an individual specification.

Collating sequence cannot be altered for:

1. COMP operations on fields defined as numeric;

2. Table LOKUP operations; or

3. Control Level data.

The standard collating sequence is always applicable to these operations.

Steps in Altering Collating Sequence

1. Punch the letter A into col. 17 of RPG processor-deck card J010001. (RPG processor-deck cards contain J in col. 73, and are numbered in cols. 74-79.)

   Alternatively, duplicate processor card J010001, and punch A into col. 17 of one of the two copies. Re-insert the altered card in the processor deck, saving the other one for jobs with standard collating sequence.

2. Punch a translation table into a series of four cards, as described below.

3. Insert the four translation-table cards and the modified processor-deck card (J010001) into the complete RPG input deck each time an object program that utilizes the altered collating sequence is to be generated by RPG.

   (The correct insertion of those cards is described in the SRL publication IBM System/360 Model 20, Report Program Generator for Punch-Card Equipment, Operating Procedures, Form C26-3800.)

Format of Translation-Table Cards

Columns	Entries
1-5	Card sequence number Any numeric characters may be used, so long as the sequence is ascending for the four cards (described below). (The same format as for RPG specifications cards--i.e., page number and card number--is suggested.)

6	Card identification S = mandatory entry
7-70	Translation table See explanation below.
71-80	Unused May be used for any desired identification. RPG ignores entries in these columns.

Translation Table

The collating sequence is punched into a set of four cards. If any deviation from the standard collating sequence is desired, all four cards must be prepared, even though the change might be confined to only one quadrant of the EBCDIC table .

Each of the four cards provides for assigning positions in the collating sequence to the 64 punch combinations in one of the four quadrants (blocks of 64 punch combinations) in the EBCDIC table (Figure D1).

The first card assigns sequence positions to the punch combinations in the first quadrant--table columns labeled with hexadecimal codes 0, 1, 2, 3. For example:

Table Column 0, Row 0 corresponds to card column 7.
Table Column 0, Row 1 corresponds to card column 8.
Table Column 1, Row 0 corresponds to card column 23.
Table Column 3, Row F corresponds to card column 70.

The second card assigns sequence positions to the punch combinations in the second quadrant--table columns labeled 4, 5, 6, 7.

The third card for the third quadrant--table columns labeled 8, 9, A, B.

The fourth card for the fourth quadrant --table columns labeled C, D, E, F.

For characters that are to retain their standard position in the collating sequence, punch the character as it appears in the EBCDIC table, into the card and column that corresponds to that position in the table.

The procedure for assigning a particular character (say, alpha) to a non-standard position in the sequence is:

1. Determine the card and column occupied by that character (alpha) in the standard collating sequence.

2. Determine the character (say, beta) that occupies the position in the standard sequence to which character alpha is to be transposed.

3. Punch character beta into the standard column for character alpha.

It is permissible to assign a character to a new position in the collating sequence, and also retain the character normally in that position in its standard position in the sequence, by also punching it in its standard column. The two characters are then treated as equal in Matching Fields and alphameric COMP operations.

Representative examples are presented further below.

The following rules apply to the assignment of a non-standard collating sequence:

1. All four cards must be used.

2. Only those of the columns 7-70 in each card need be punched whose corresponding punch combinations in the standard collating sequence--as shown in the EBCDIC table--occur in the input data.

3. If a character-column is left blank, but the punch combination corresponding to that column in the standard collating sequence does occur in the input data, that punch combination is converted to the sequence position of "blank" (hexadecimal 40).

Examples of Translation

1. A character is to retain its standard collating-sequence position.

   To retain E83 ($) in its standard collating sequence position:
   Punch 11-8-3 ($) into column 34 of the second card--the column that corresponds to the standard-sequence position of the $ symbol.

   (Note that translation cards are required at all only if there is any deviation from the standard collating sequence somewhere in the EBCDIC character set.)

2. A character is to be transferred to a non-standard collating-sequence position.

   To transfer E83 ($) to the sequence position immediately after digit 9:

   Punch 12-11-0-9-8-2--the standard punch combination following digit 9--into column 34 of the second card--the column that corresponds to the standard-sequence position of the $ symbol.

If nothing is punched in column 65 of the fourth card--the standard position for 12-11-0-9-8-2--the punch combination 12-11-0-9-8-2, if it occurs, is converted to the sequence position of "blank".

3. Several characters are to be transferred to the same non-standard collating-sequence position.

   To transfer E82 (!) and E84 (*) to the sequence position immediately preceding the alphabet, and thus cause both to be treated as equal in sequence:
   Punch 12-0--the standard punch combination preceding the letter A--into columns 33 and 35 of the second card--the columns that correspond to the standard-sequence positions of the (!) and (*) symbols, respectively. If nothing is punched in column 7 of the fourth card--the standard-sequence position of 12-0--the punch combination 12-0, if it occurs, is converted to the sequence position of "blank".

4. The sequence positions of two characters are to be interchanged.

   To interchange the relative sequence positions of the letter B and the digit 2:
   a. Punch 2--the character for the sequence position to be assumed by letter B--into column 9 of the fourth card--the column that corresponds to the standard-sequence position of the letter B.
   b. Punch B--the character for the sequence position to be assumed by the digit 2--into column 57 of the fourth card--the column that corresponds to the standard-sequence position of the digit 2.

5. A character in its standard position is to be combined with another character. The example chosen will illustrate how signed positive values (12-overpunched) can be combined with unsigned positive values, without sacrificing the identity of negative values (11-overpunched). (The example assumes that the field is defined as alphameric, and therefore not used in arithmetic or edit operations.)

   To combine the letter A (12-1 with the digit 1, in the position occupied by the digit 1 in the standard collating sequence:
   a. Punch 1--the character for the sequence position to be assumed by the letter A--into column 8 of the fourth card--the column that corresponds to the standard-sequence position of the letter A.
   b. Also punch 1 into column 56 of the fourth card--its standard-sequence position.

DATA FORMATS

Alphameric Fields

Data for fields defined as alphameric is stored in the IBM 2020 central processing unit with one character per byte (see Code Structure, above).
The byte bas the bit structure shown in the EBCDIC table (Figure D1). For example:

1. The letter A occupies one byte composed of the bits 1100 0001--equivalent to hexadecimal code C1 and card punch-combination 12-1.

   The C may be considered the zone portion of the character.

2. The asterisk (*) occupies one byte composed of the bits 0101 1100--equivalent to hexadecimal code 5C and card punch-combination 11-4-8.

3. The numeral 4 occupies one byte composed of the bits 1111 0100--equivalent to hexadecimal code F4 and card punch code 4.

   The F can be considered the zone portion of the character. An F zone is tantamount to no zone for a digit (0-9) in an alphameric field--the absence of a zone punch over a digit at input causes an F zone to be assigned; at output, an F does not cause a zone to be punched over a digit (0-9).

Numeric Fields

Because four bits are adequate to represent any decimal digit (0-9), only a half-byte is needed in storage for each position of a field defined as numeric, except:
(a) A half-byte is reserved for the sign--all numeric fields are signed, although the sign may be hexadecimal F (tantamount to no sign); and
(b) Each field must consist of full bytes--a half-byte must, therefore, be wasted in fields with an even number of digit positions.

The method of storing two digits in one byte is termed packed-decimal format, and is illustrated below.

Packed-Decimal Format

All fields defined as numeric are stored by RPG in packed-decimal format. Normally, the conversion to packed format is performed by RPG after the data has been read in; but, if "Packed" is specified in the input specifications for the field, the packing routine is bypassed, the data then being assumed already to be in packed format. Core storage in packed format can be portrayed as follows:

If the number of digit positions in the field is odd --

4 bits	4 bits	4 bits	4 bits	4 bits	4 bits
digit	digit	digit	digit	digit	sign
byte		byte		byte	

If the number of digit positions in the field is even --

4 bits	4 bits	4 bits	4 bits	4 bits	4 bits
0000	digit	digit	digit	digit	sign
byte		byte		byte	

The maximum size for a numeric field is 8 bytes (i.e., 15 digits and sign).

Digit positions (i.e., all but the low-order half-byte) must not contain bits 1010 through 1111 (i.e., hexadecimal A-F) if the field is to be used in arithmetic, COMP (compare), or edit (including Zero-Suppress) operations. These operations are based on decimal arithmetic; therefore, the digit values must lie within the decimal range (0-9).

Data Input and/or Output in Packed Format

Reasons and the proper specifications for reading numeric data into or out of the System in packed form were given under Input Specifications (Packed--col. 43) and under Output-Format Specifications (Packed Field--col. 44), together with formulas for equating field lengths in packed and unpacked formats.

Figure D2 gives some examples of numbers, the format in which they are stored in the processor, and the corresponding card punch-combinations if packed format is specified for input from or output to cards. Reference to Figure D1 (EBCDIC table) will clarify the examples.

NUMBER	STORED IN CORE AS (Expressed in Hexadecimal Notation)								CORRESPONDING PUNCHES IN CARD COLUMNS IF PACKED FORMAT IS SPECIFIED							
Max. = 15 digits	upper half- lower half- Byte	upper half- lower half- Byte	upper half- lower half- Byte	upper half- lower half- Byte	upper half- lower half- Byte	upper half- lower half- Byte	upper half- lower half- Byte	upper half- lower half- Byte								
1234						01	23	4F						12/9/1	0/9/3	12/8/7
+1234						01	23	4C						12/9/1	0/9/3	12/8/4
-1234						01	23	4D						12/9/1	0/9/3	12/8/5
97531						97	53	1F						12/11/7	12/11/9/3	11/9/8/7
+97531						97	53	1C						12/11/7	12/11/9/3	11/9/8/4
-97531						97	53	1D						12/11/7	12/11/9/3	11/9/8/5
29486670				02	94	86	67	0F				12/9/2	12/11/4	12/0/6	11/0/9/7	12/9/8/7
+29486670				02	94	86	67	0C				12/9/2	12/11/4	12/0/6	11/0/9/7	12/9/8/4
572768911504329	57	27	68	91	15	04	32	9F	12/11/9/7	0/9/7	11/0/9/8	12/11/1	11/9/5	12/9/4	9/2	12/11/8/7
-572768911504329	57	27	68	91	15	04	32	9D	12/11/9/7	0/9/7	11/0/9/8	12/11/1	11/9/5	12/9/4	9/2	12/11/8/5
+57276891150432	05	72	76	89	11	50	43	2C	12/9/5	12/11/0/9/2	12/11/0/9/6	12/0/9	11/9/1	12	12/0/9/3	0/9/8/4
-57276891150432	05	72	76	89	11	50	43	2D	12/9/5	12/11/0/9/2	12/11/0/9/6	12/0/9	11/9/1	12	12/0/9/3	0/9/8/5

Figure D2. Examples of Packed-Decimal Data in Core and Cards

TIPS FOR MINIMIZING CORE-STORAGE REQUIREMENTS

These are generalized suggestions: not every one necessarily holds true under all combinations of conditions, nor are they always practicable because of other factors involved. (Refer also to Storage Requirements, Appendix A.)

1. (a) If all pertinent input fields in several card types are in the same columns, and the format (e.g., alphameric, or numeric and number of decimal places) corresponds between the types, define all such cards as one type.
   (b) If the majority of the input fields are identical, use OR lines and Field-Record Relation entries rather than completely separate file identifications.

2. (a) Use the same field name for different input fields in different card types when field size and format (alphameric, or numeric and number of decimal places) are identical, if the data need not be preserved from one card type to the other.
   (b) Use the same field name repeatedly in calculations as result field for various different items whenever the result need no longer be retained; i.e., employ one work field in several calculations. (See CSTEXT in Figure 68--Parts II and III.)

3. (a) Use the minimum number of Record-Identification Codes to identify a card type.
   (b) Use C for record-identification code in preference to D or Z.

4. When there are several card types per input (or combined) file, two control levels (say, L1 and L2) usually take less core than one split level.

5. Employing Matching-Fields specifications solely for purposes of sequence-checking a single file on one or two fields requires more core than sequence checking by comparing (COMP) in the calculation specifications. For alphameric fields, this is still true for three fields. (Figure 68--Part I illustrates COMP for sequence-checking. Note that the compare data from one card is saved for the next card.)

   Sequence-checking by COMP operation also permits detection of duplicates (shown in Figure 68--Part I), which is not part of the Matching-Fields sequence-check function.

Note: Sequence-checking a single file by COMP operation, rather than by Matching-Fields entries, also tends to improve throughput: a Matching-Fields specification delays the reading of the next card.

6. Try to keep the source columns for each Control-Level or Matching-Fields input field uniform in all card types.

7. Use Field Indicators in input specifications in preference to COMP operation in calculations.

8. (a) Where feasible, group together calculation specifications conditioned by the same indicators (cols. 7-17). (In this context, "same" also implies identical status criterion for an indicator: either on or Not on in all lines.)
   and
   (b) When successive specifications lines are conditioned by the same indicators, specify the indicators in the same order (cols. 9-11, 12-14, 15-17).

   When the same indicator is assigned as Resulting Indicator (cols. 54-59) in a specifications line, and as conditioning indicator (cols. 7-17) in that and the next line, there is no core saving if the two lines contain identical conditioning indicators.

9. (a) Try to define the number of decimal places to be uniform for all numeric fields or literals used in one arithmetic or compare operation.
   (b) For ADD/SUB operations, also try to have fields or literals of equal length.
   (c) For comparing (COMP) alphameric fields, try to have fields or literals of equal length.

10. When a number of fields are to be edited for output, define as many of them as possible with the same length, and edit them uniformly. This requires storage of only a single edit word.

11. (a) In ADD and SUB operations, significant core space is saved when the

Result Field and Factor 1 and/or Factor 2 are the same, provided the other Factor is of equal or shorter length and all three have the same number of decimal places.

(b) In Z-ADD and Z-SUB, significant core space is saved if the two fields have the same number of decimal places.

(c) In MULT, DIV, and MVR, core space is saved if no decimal alignment is required.

12. To clear a numeric field to zero, use SUB, with field name in Factor 1 = Factor 2 = Result Field.

13. To turn indicators on or off (as Resulting Indicators) in calculation specifications, based on whether numeric-field contents are plus, minus, or zero, add a literal zero rather than comparing with a zero.

For the core savings to be realized:

(a) The Result Field must be the same as one of the two Factors; and

(b) The literal zero must be specified with the same number of decimal places as in the other Factor (e.g., 0.00 or .00).

14. Specify one SETON (or SETOF) with two or three indicators in preference to separate SETON (or SETOF) for each indicator.

15. Use GOTO to:

(a) Bypass a group of inapplicable calculation specifications--rather than conditioning each by indicators, particularly when several conditioning indicators would be required therefor. (See illustration in Figure 68--Parts I, II, and III.)

(b) Utilize the same series of calculation instructions, applicable under different circumstances at different points in the calculation specifications,--rather than repeating identical, or near-identical, specifications. (See Square Root illustration below--Figure E12.)

16. Punch constant data (such as date, for example) into the input card in edited format--rather than using an edit word in output specifications.

17. When testing for specific numeric codes in a field, define the field as alphameric and compare (COMP) to alphameric literals.

TIPS ON SPECIAL PROGRAMMING REQUIREMENTS

This section consists of suggestions for satisfying some common, yet non-routine, programming requirements. Most of the hints are presented in the form of illustrative RPG specifications, each preceded by a statement of the problem and followed by a brief explanation; but some hints lend themselves best to narrative format. (Two examples are included that involve branching to B.A.L. subroutines.) Those tips already included in illustrations in the body of the manual are not repeated here.

An attempt has been made to group the pointers by the type of specifications with which the problem is most closely associated.

Input-Oriented Pointers

Group Indication on "Run-In"

Problem: At the beginning of the deck, a Control-Level L-indicator does not turn on until detail-calculation time for the first card of a type for which Control Level is specified and which is not zero (if alphameric field), or not blank or zero (if numeric field) in the control field.

Figure E1 illustrates a method for assuring group indication at the beginning of each group, even when the control field in the first such card is blank or zero.

Assumption: Control Level 1 was assigned in the input specifications (cols. 59-60) to the pertinent card type and field.

Explanation: The L1 indicator is turned on, by a SETON operation, in the detail-time calculations during the first program cycle. This assures that L1 is on for group indication during detail-time output for the first card. The program itself turns off L1 after each detail-time output. Thereafter, normal Control Level operation turns on L1 at the end of each control group.

The SETON instruction also turns on another indicator (say, 90). By conditioning execution of the SETON specification by N90 (cols. 9-11), L1 is never again turned on by the SETON instruction--only at the beginning of program execution, when indicator 90 is off.

If desired, additional L-indicators (say, L2) could also be SETON for the first card. Remember, however, that SETON of L2 does not automatically set on L1--both must then be specified.

Checking That Output-Card Fields are Blank Before Output

Problem: The user sometimes desires assurance that the card fields into which output will be punched are blank; i.e., were not inadvertently punched in a prior operation.

Solution:

1. Define the file as a combined file.

2. Define the file in the input as well as the output specifications.

3. Define the area to be output-punched as a single continuous alphameric input field (if practical--otherwise, as several fields).

4. Assign an indicator (01-99) to Field Indicators, Blank (cols. 69-70). In the detail-calculation specifications, SETON H1 or H2 if the indicator assigned to Blank is not on. The system will halt when the card has been processed; punching can be suppressed by Output Indicator NH1 (or NH2).

Note:
1. The approach assumes a read-punch I/O device.

2. If the area is broken into several fields, do not assign the same indicator to more than one field--otherwise it can be turned on again by a later-specified field even if turned off by an earlier one.

3. Punching is assumed to be at detail time, because Field Indicator status is not available until detail time.

Control Levels Initiated by Card Type

Problem A: A Control Level (say, L1) is specified in the normal manner in the input specifications. In addition, a higher level (say L2) is to be initiated preceding the processing of a card of a certain type.

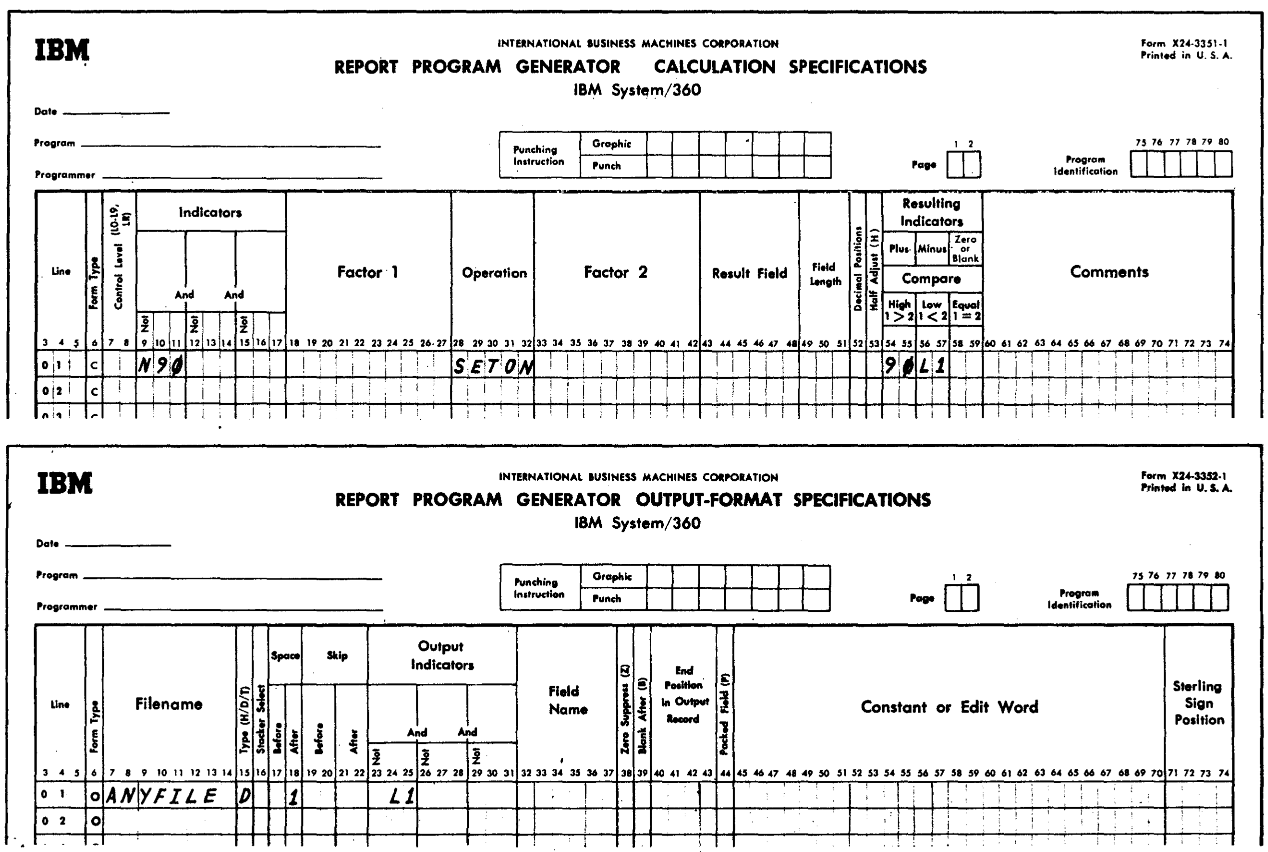


Figure E1. Group-Indication, Even When Control Field is Zero in First Card of Deck

IBM — INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS
IBM System/360 — Form X24-3351-1, Printed in U. S. A.

Line	Form Type	Control Level	Indicators	Operation	Resulting Indicators	Comments
01	C			Detail-Time		
02	C			Specs		PART I
03	C	L0	80	SETON	L1 L2	
04	C					
05	C			Detail-Time		
06	C			Specs		PART II
07	C	L0	81	SETOF	L1 81	
08	C	L0	80	SETON	L1 L2 81	
09	C					

IBM — INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS
IBM System/360 — Form X24-3351-1, Printed in U. S. A.

Line	Form Type	Control Level	Indicators	Operation	Resulting Indicators	Comments
01	C			SETOF	90	
02	C		80	SETON	90	PART III
03	C	L0	90	SETON	L1	
04	C					
05	C					

Figure E2. Control Levels Initiated by Card Type--Regular Control Level Also Specified

Assumption: A separate card-type Resulting Indicator (say, 80) has been assigned to the card type which is to initiate the higher-level control break.

Figure E2 (Parts I and II) shows two of several possible programming methods. Part I assumes that the L1 control field also exists in the special L2-level card. Part II assumes that the special L2-level card contains no other control field, and it is necessary to prevent the occurrence of a spurious L1 break after it--when the L1 field of the next regular card is compared with that in the last preceding regular card.

Explanation--Part I: The first total-time instruction turns on L2. L1 must also be SETON, because lower levels do not automatically turn on when a higher one is turned on by SETON. All L-indicators are turned off by the program itself after the next card has been read, and before its control fields are compared.

The specifications must be at total time (L0 in cols. 7-8, because another L-indicator is not necessarily on), and must be the first ones at that time so that others are properly conditioned by the status of L1 and L2.

Note: Unless there is a special reason for wishing to utilize L2, the same results can be accomplished by simply conditioning the operations concerned by indicator 80 itself--remember that totals, too, can be conditioned by any indicator, and are not dependent on a Control-Level indicator.

Explanation--Part II: As Part I; but, in addition, indicator 81 is turned on whenever L2 is turned on. 81 is used to turn off L1 next cycle, when it may have been spuriously turned on by the regular next card following the special card; at the same time 81 is turned off, so that L1 turned on by subsequent (legitimate) control breaks is not artificially turned off.

Problem B: A Control Level (say, L1) is to be initiated following the processing of a card of a particular type (say, identified by card-type Resulting Indicator 80).

Figure E2--Part III shows one of several solutions.

Explanation--Part III: During detail-time processing of a card of the relevant type (indicator 80), another indicator (say, 90) is SETON (see second line). Indicator 90 is then on at the beginning of processing of the next card (total time) and until SETOF at detail time of the next cycle (see first specifications line). As the first total-time calculation specification (see third line), L1 is SETON if 90 is on--thus: L1 is turned on at the beginning of processing for the card following a type-80 card; it remains on until turned off by the program itself after the card has been completely processed.

Note: If the purpose is merely to calculate or output totals following a card of a particular type--not to utilize data from, or punch into, the next card in some selective manner--none of the special calculation specifications shown in Part III are necessary.

The indicator (80) of the card that is to be followed by the special operations is on throughout its processing. Totals can be calculated or output at detail time as well as at total time. The simplest solution then is to specify all such calculations and/or output as the last detail-time calculations and output, respectively--conditioned by indicator 80. Detail-time operations on the data in the type-80 card have then been completed before the operations that are to follow a card-type 80. (See also Figure E5 for another approach.)

Problem C: No Control Level is specified in cols. 59-60 of the input specifications; but Control Level L1 is to turn on preceding the processing of a card of a certain type.

Solution (one of several): Assign Resulting Indicator L1 in cols. 19-20 in the card-type identification specifications for the pertinent card type.

Control On A Signed Field--No Break Between Unsigned and Plus-Signed Cards

Problem: Occasionally, a Control Level must be assigned to a numeric field that contains positive and negative values. If the sign is to be ignored, no problem exists: defining the field as numeric strips the zones from control. If the sign must be considered, and positive fields are never or always signed (12-overpunch), no problem exists either: defining the field as alphameric retains all zones for control. (It is assumed that negative values contain an 11-overpunch in the low-order position.)

It is, however, possible that positive values in some cards are signed (12-overpunch), and others are not. This situation may arise, for instance, if some cards were created as unedited output in a previous Model 20 operation, while others were keypunched.

Figure E3 shows how it is possible to control on such a field, distinguishing between positive and negative values--yet not breaking control between identical positive values, some of which are 12-overpunched in the sign position while others are not.

IBM	INTERNATIONAL BUSINESS MACHINES CORPORATION REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS IBM System/360	Form X24-3350-1 Printed in U.S.A.

Line	Form Type	Filename	Sequence	Resulting Indicator	Position (1)	Not (N)	C/Z/D	Character	Position (2)	Not (N)	C/Z/D	Character	From	To	Decimal Positions	Field Name	Control Level (L1-L9)
01	I	ANYFILE	AB	11	80		C	1	10		Z	-					
02	I	OR		12	80		C	1	10	N	Z	-					
03	I												5	10	0	DISTRI	L1
04	I																

IBM	INTERNATIONAL BUSINESS MACHINES CORPORATION REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS IBM System/360	Form X24-3351-1 Printed in U.S.A.

Line	Form Type	Control Level (L0-L9, LR)	Indicators	And	And	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators High 1>2 (Plus)
01	C						Detail-Time			
02	C						Specs.			
03	C	L0	11	N21			SETON			L1
04	C	L0	12	21			SETON			L1
05	C	L0					SETOF			21
06	C	L0	11				SETON			21
07	C									

Figure E3. Control on a Signed Field, No-Zone and 12-Zone Combined

Explanation: Cards with a negative value in the field (11-punch in col. 10) are assigned a different card-type Resulting Indicator (11) from those that are not negative (no 11-punch in col. 10 --indicator 12). No distinction is made between the non-negative cards with and without a 12-overpunch in the low-order position: thus, indicator 12 represents all positive-value cards.

The field (cols. 5-10) is defined as numeric, and a Control-Level indicator (L1) is assigned in the normal manner (cols. 59-60 of the input specifications). This provides normal control on the absolute numeric value in the field, ignoring sign: if the absolute value differs in two successive cards, L1 turns on before total-calculation time.

The calculation specifications shown are irrelevant when L1 has already been turned on by a change in absolute value; but they do not interfere, because they do not cause L1 to turn off. However, if L1 is not on--i.e., the absolute value did not change from the previous card--they cause L1 to turn on if the sign of the value changed:

The last line causes indicator 21 to turn on if the value is negative. On the next program cycle, L1 is turned on if either (1) the value is negative (indicator 11) and that in the previous card was not (21 not on)--see the first line--or (2) the value is not negative (indicator 12) but that in the previous card was negative (21 on)--see the second line. The third line merely turns off indicator 21 so that it does not remain on following a card with a positive value.

In order to condition other total-time operations properly based on the status of L1, these specifications must be the first ones at total time. Since, by definition of the problem, these specifications are significant only when L1 is not already on, L0 is required in cols. 7-8 to define them as pertaining to total time.

Note that Field Indicators or Compare on the field contents--instead of OR lines and card-type Resulting Indicators--could not be used, because that information from the new card is not available at total time (see Figure 6, RPG Program Logic).

Note: See also Altered Collating Sequence, Appendix D, for combining C-zone and F-zone characters for Matching Fields operations on alphameric field.

Indexing--Analyzing and Forming Fields Position-by-Position

Problem: A card type contains a variable number of fields of variable length, and these fields are to be printed separately.

A common application involves name-and-address cards in which the user has not assigned a field of fixed length to each portion (name, street address, city-state), and the number of fields (data print lines) is allowed to vary (usually from two to four).

Figure E4 (Parts I and II) shows RPG calculation specifications that offer one solution (in conjunction with appropriate input and output specifications). Part II of Figure E4 suggests a method of eliminating unnecessary print cycles when the card contains less than four fields.

Assumptions:

1. Each name-and-address card contains one to four name-and-address fields.
2. Each field allows for 1-18 positions of data.
3. The last data character in each utilized field is followed by a special symbol ($ was arbitrarily chosen).
4. The last field used is followed by two $ symbols.
5. The group of fields is defined in the input specifications as a single 77-position alphameric field, named SOURCE (4 x 18 = 72 +5 possible $ symbols = 77).

Solution: Figure E4--Part I--should be studied line by line, and the operation simulated on a piece of paper. Basically, a source field is shifted left one position at a time, repeatedly (lines 19 and 06), until it is exhausted (indicator 80 on). Each time, the leftmost position is examined to see whether the $ symbol, terminating a field, has been reached (line 10). A test is also made for two $ symbols in succession, signalling the end of data (lines 07, 08, 10, and 11). The characters (line 08) up to a dollar symbol are assembled, one at a time, in the field RESULT (lines 14, 16, and 17).

The shifting is accomplished by moving between a work field (WORK1) and the SOURCE field (lines 12 and 13), and between WORK2 and RESULT (lines 16 and 17). The work fields differ in length by one position from the corresponding SOURCE and RESULT fields which, in conjunction with appropriate use of MOVE and MOVEL operations, accomplishes the desired end.

Stepping the data to the left in the RESULT field is continued even when the data field contained less than 18 positions, so that the output will be left-aligned (lines 18-20, and 15). For this reason, a counter (CHRCTR = character counter) is initialized at 17 for each field, and decremented for each shift (lines 03, 04, 05, 18, and 31). Figure E4 -part II- shows the specifications required when the data is to be right-justified in the output field.

Four output fields (PRINT1, 2, 3, and 4--see lines 22, 24, 26, and 28) are set up to receive the data which has been assembled in the field named RESULT from the respective portions of the consolidated input field (SOURCE). A counter (FLDCTR = field counter) keeps track of which data field in the consolidated field SOURCE is being processed; it is reset to zero at the beginning of the entire operation, so that it is always cleared when processing of the next name-and-address card starts. (See line 30, and lines 01 and 02--we have assumed that the user specified GOTO START at the appropriate point in the calculation specifications for a name-and-address card, and branches around this section on other card types.) The COMP operations in lines 21, 23, 25, and 27 determine, based on the contents of FLDCTR, to which of the four output fields the data in the field RESULT belongs.

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System 360

Form X24-3351-1
Printed in U. S. A.

Date

Program

Programmer

Punching Instruction	Graphic							
	Punch							

Page 1 2

Program Identification 75 76 77 78 79 80

Line (3 4 5)	Form Type (6)	Control Level (L0-L9, LR) (7 8)	Indicators: Not (9 10 11)	And: Not (12 13 14)	And: Not (15 16 17)	Factor 1 (18–27)	Operation (28–32)	Factor 2 (33–42)	Result Field (43–48)	Field Length (49 50 51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators: Plus / Compare: High 1 > 2 (54 55)	Resulting Indicators: Minus / Compare: Low 1 < 2 (56 57)	Resulting Indicators: Zero or Blank / Compare: Equal 1 = 2 (58 59)	Comments (60–74)
01	C					START	TAG									
02	C					FLDCTR	SUB	FLDCTR	FLDCTR	1	Ø					
03	C					SETCTR	TAG									
04	C						Z-ADD	17	CHRCTR	2	Ø					
05	C						SETOF						81			
06	C					INDEX	TAG									
07	C						SETOF						9Ø			
08	C		8Ø				SETON						9Ø			
09	C						MOVEL	SOURCE	CHARAC	1						
10	C					CHARAC	COMP	'$'							8Ø	
11	C		8Ø	9Ø			GOTO	END								
12	C						MOVE	SOURCE	WORK1	76						
13	C						MOVEL	WORK1	SOURCE							
14	C		N8Ø				MOVE	CHRAC	RESULT	18						
15	C					SHIFT	TAG									
16	C		N81				MOVEL	RESULT	WORK2	19						
17	C		N81				MOVE	WORK2	RESULT							
18	C		N81			CHRCTR	SUB	1	CHRCTR						81	
19	C		N8Ø				GOTO	INDEX								
2Ø	C		N81				GOTO	SHIFT								
21	C					FLDCTR	COMP	Ø							85	
22	C		85				MOVE	RESULT	PRINT1	18						
23	C					FLDCTR	COMP	1							86	
24	C		86				MOVE	RESULT	PRINT2	18						
25	C					FLDCTR	COMP	2							87	
26	C		87				MOVE	RESULT	PRINT3	18						
27	C					FLDCTR	COMP	3					H1		88	
28	C		88				MOVE	RESULT	PRINT4	18						
29	C						MOVE	BLNK	RESULT							
30	C					FLDCTR	ADD	1	FLDCTR							
31	C						GOTO	SETCTR								
32	C					END	TAG									
33	C						RLABL		BLNK	18						
34	C															
35	C															
	C															
	C															
	C															
	C															
	C															

Card Electro Number

Figure E4 (Part I of III). Indexing: Analyzing and Forming Fields Position by Position, Calculation Specifications

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-3351-1
Printed in U.S.A.

Date ______

Program ______

Programmer ______

Punching Instruction	Graphic								
	Punch								

Page 1 2   Program Identification 75 76 77 78 79 80

Line (3-5)	Form Type (6)	Control Level (LO-L9, LR) (7-8)	Indicators (Not) 9-11	And (Not) 12-14	And (Not) 15-17	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field (43-48)	Field Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators Plus / Compare High 1>2 (54-55)	Minus / Low 1<2 (56-57)	Zero or Blank / Equal 1=2 (58-59)	Comments (60-74)
01	C					START	TAG									
02	C					FLDCTR	SUB	FLDCTR	FLDCTR	1	Ø					
03	C					INDEX	TAG									
04	C						SETOF						9Ø			
05	C		8Ø				SETON						9Ø			
06	C						MOVEL	SOURCE	CHARAC	1						
07	C					CHARAC	COMP	'$'							8Ø	
08	C		8Ø	9Ø			GOTO	END								
09	C						MOVE	SOURCE	WORK1	76						
10	C						MOVEL	WORK1	SOURCE							
11	C		N8Ø				MOVE	CHARAC	RESULT	18						
12	C		N8Ø				MOVEL	RESULT	WORK2	19						
13	C		N8Ø				MOVE	WORK2	RESULT							
14	C		N8Ø				GOTO	INDEX								
15	C		8Ø				MOVEL	WORK2	RESULT							
16	C					FLDCTR	COMP	Ø							85	
17	C		85				MOVE	RESULT	PRINT1	18						
18	C					FLDCTR	COMP	1							86	
19	C		86				MOVE	RESULT	PRINT2	18						
2Ø	C					FLDCTR	COMP	2							87	
21	C		87				MOVE	RESULT	PRINT3	18						
22	C					FLDCTR	COMP	3					H1		88	
23	C		88				MOVE	RESULT	PRINT4	18						
24	C						MOVE	BLNK	RESULT							
25	C					FLDCTR	ADD	1	FLDCTR							
26	C						GOTO	INDEX								
27	C					END	TAG									
28	C						RLABL		BLNK	18						
	C															
	C															
	C															
	C															
	C															
	C															
	C															
	C															
	C															
	C															
	C															
	C															

Card Electro Number ______

Figure E4 (Part II of III). Indexing: Analyzing and Forming Fields Position by Position, Calculation Specifications

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System/360

Form X24-3352-1
Printed in U.S.A.

Date ______

Program ______

Programmer ______

Punching Instruction: Graphic / Punch

Page 1 2   Program Identification 75 76 77 78 79 80

Line (3-5)	Form Type (6)	Filename (7-14)	Type (H/D/T) (15)	Stacker Select (16)	Space Before (17)	Space After (18)	Skip Before (19-20)	Skip After (21-22)	Output Indicators (23-25)	And (26-28)	And (29-31)	Field Name (32-37)	Zero Suppress (Z) (38)	Blank After (B) (39)	End Position in Output Record (40-43)	Packed Field (P) (44)	Constant or Edit Word (45-70)	Sterling Sign Position (71-74)
01	O	INVOICE	D			1	01		01									
02	O											PRINT1		B	24			
03	O		D			1			01	N85								
04	O											PRINT2		B	24			
05	O		D			1			01	N85	N86							
06	O											PRINT3		B	24			
07	O		D						01	88								
08	O											PRINT4		B	24			
09	O		D				02		05									
10	O																	

Figure E4 (Part III of III). Indexing: Analyzing and Forming Fields Position by Position, Output-Format Specifications

Because no more than four fields are permitted, H1 (line 27) is turned on (to halt the system), if the field-counter value exceeds 3. (3--not 4--is the criterion, because we start with 0, and do not increment until the first field has been processed--see line 30.) When the two successive $ symbols are encountered at the end of an earlier field (i.e., less than four fields appear in the card), the entry in line 11 prevents further processing.

The field RESULT must be blanked (line 29) after the processing of each field of a card: because only eight blanks can be moved as a literal, an 18-position blank alphameric field (BLNK) is set up for the purpose (line 33). (See explanation below: Clearing an Alphameric Field.)

Unnecessary print cycles can be prevented, as shown in Part III, when there are less than four fields, because the indicator-setting operations in lines 21, 23, 25, and 27 were so designed that the indicator (85, 86, 87, or 88) on at output time represents the number of fields in the card, the other three indicators being off. Forms advance after the last name-and-address line should then be specified as Skip/Before in the first output line that follows, because of variable execution of preceding lines. (If unnecessary print cycles are suppressed, as shown in Part III, Blank-After need not be specified and the conditioning indicators in cols. 10-11 of lines 22, 24, 26, and 28 are not required.)

Note: The technique can also be adapted to translating certain characters in a field to others.

Calculations-Oriented Pointers

Clearing an Alphameric Field

Problem: It may be necessary to clear (i.e., blank) an alphameric field by a method other than a Blank-After instruction in the output specifications.

Solution I--field no larger than eight positions: Specify an alphameric literal of "blank" in Factor 2, and MOVE this literal to the pertinent field (specified as Result Field).

Solution II--field larger than eight positions: With the RLABL pseudo-operation, define a new alphameric field of the desired length; the field will be blank. MOVE this field to the relevant field to be blanked (specified as Result Field in the MOVE operation).

Figure E4--Part I illustrates the method--see lines 33 and 29. (Note that RLABL field names must not exceed four characters.)

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

Form X24-3351-1
Printed in U. S. A.

Date
Program
Programmer

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 Plus / High 1>2	Minus / Low 1<2	Zero or Blank / Equal 1=2	Comments
01	C				Detail-Time									
02	C				Specs									
03	C		1Ø		SETON						2Ø			
04	C	Lx												
05	C	Lx	2Ø		Total-Time									
06	C	Lx	N2Ø		Specs									
07	C													
08	C	LØ			SETOF						2Ø			
09	C													

Figure E5. Total-Time Calculations Based on Type of Last Preceding Card

Card-Type Resulting Indicator During Total-Time Calculations

During detail-time calculations, the status of card-type Resulting Indicators reflects the card type being processed. During total time, the indicator for the next card--whose data is not yet available to RPG--is on, and that representing the just processed card is off (unless both cards are of the same type); thus, total-time calculations can very easily be based on the type of card that follows (L0 can be specified in cols. 7-8 if none of the indicators L1-L9 are desired as a condition).

Problem: Calculations at total time are to be conditioned by the type of the last preceding card.

Figure E5 shows a simple solution. We have assumed that the criterion card type was assigned indicator 10 in the input specifications.

Explanation: An indicator (say, 20) is SETON at any point during detail-time calculations, when the particular card type (indicator 10) is being processed. That indicator is then used at total time in conjunction with any L-indicator (Lx = L0-L9, as desired). It must be SETOF again, either by the end of total time, or during detail time before it is SETON on the basis of indicator 10--otherwise it would remain on even when card type 10 did not precede total time. If SETOF during total time, L0 must be in cols. 7-8, because other L indicators may not be on for every program cycle.

INTERNATIONAL BUSINESS MACHINES CORPORATION

IBM **REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS**

IBM System/360

Form X24-3351-1 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	Control Level	Indicators	And	And	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Plus / High 1>2	Minus / Low 1<2	Zero or Blank / Equal 1=2	Comments
01		C						SETOF						9Ø			
02	Part I	C		Ø1	N15	22		SETON						9Ø			
03		C		9Ø	Ø5	N14	(Any operation in this line executed subject to Ø1&N15&22&Ø5&N14)										
04																	
05		C	a	Ø1	N22		ALPHA	ADD	BETA	OMEGA	5	Ø					
06		C		NØ1	22		ALPHA	ADD	BETA	OMEGA							
07		C															
08		C	b					SETOF						9Ø			
09		C		Ø1	N22			SETON						9Ø			
10		C		NØ1	22			SETON						9Ø			
11	II	C		9Ø			ALPHA	ADD	BETA	OMEGA	5	Ø					
12	Part	C															
13		C		Ø1	N22			GOTO	COMRTN								
14		C		NØ1	22			GOTO	COMRTN								
15		C	c					other specs									
		C						GOTO	END								
		C					COMRTN	TAG									
		C					ALPHA	ADD	BETA	OMEGA	5	Ø					
		C					END	TAG									

Card Electro Number ______

Figure E6. AND and OR Lines in Calculation Specifications

AND and OR Lines in Calculation Specifications

Figure E6--Part I presents a technique for conditioning a calculation by more than three indicators in an AND relationship. Part II of Figure E6 shows three alternatives for establishing OR relationships.

Explanations: The illustrations are largely self-explanatory. It is important to remember that (with certain exceptions, like L-indicators) indicators that are SETON remain on unless turned off by SETOF or as Resulting or Field Indicator.

Part I: When three indicator conditions are satisfied, a fourth is SETON. This is then used in conjunction with further indicators to condition execution of other specifications. Each additional "AND" line, using this approach, permits adding two more indicators to the AND relationship.

Part II(a): The same spcifications are executed when the indicator conditions in either line are satisfied.

Part II(b): The same indicator (90) is SETON when the indicator conditions in either line are satisfied. Execution of the calculation operations is then based on the status of that indicator (90).

Part II(c): The program branches to the same routine when the indicator conditions in either line are satisfied.

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-3351-1
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	Control Level (L0-L9, LR)	Indicators (Not, And, Not, And, Not)	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators Plus / Compare High 1>2	Resulting Indicators Minus / Compare Low 1<2	Resulting Indicators Zero or Blank / Compare Equal 1=2	Comments
01	C				TESTZ		CODE				12	11	9Ø	
02	C			CODE	COMP	'S'					1Ø		1Ø	
03	C		1Ø	CODE	COMP	'Z'						1Ø	1Ø	
04	C			CODE	COMP	' '							99	
05	C													

Figure E7. Distinguishing 12-, 11-, 0-, No-Zone, and b̸ in Input Fields

Distinguishing Zone Punches in Input Fields

Problem: Alphabetic codes are sometimes structured so that the so-called unit-record zone punches (12 = A-I, 11 = J-R, 0 = S-Z) represent groups. TESTZ provides for convenient determination of 12- and 11-punches in the high-order or sole position of alphameric fields; but it does not distinguish between 0 and the remaining (not 12- or 11-) zone punches (i.e., it lumps all hexadecimal codes other than 50, C, 60, or D as "no zone").

Figure E7 shows how to test a single-position alphameric input field (named CODE) for 12-zone, 11-zone, 0-zone, no zone punch (digits 0-9), or blank. We have made the assumption that the code consists only of A-Z, 0-9, and blank. The COMP-operation technique can, of course, be applied to identifying characters within any EBCDIC range (for example, all of hexadecimal zone E).

Explanation: Indicator 12 will be on for A-I (or any character with hexadecimal 50 or zone C); indicator 11 is on for J-R (or any character with hexadecimal 60 or zone D); indicator 10 is on for letters S-Z; and indicator 99 is on when the column is blank. No zone (i.e., digits 0-9) is represented by 90 N10 N99.

IBM — INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-3351-1
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	Control Level (L0-L9, LR)	Indicators		Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators Plus / High 1>2	Minus / Low 1<2	Zero or Blank / Equal 1=2	Comments
01	C			Part I		Z-ADD	FIELD1	COMFLD	6	2			21		
02	C		21			Z-SUB	COMFLD	COMFLD							
03	C				COMFLD	COMP	125Ø.ØØ					11	12	13	
04	C														
05	C			Part II											
06	C					MLLZO	5	FIELD3							
07	C				FIELD3	COMP	125Ø.ØØ					11	12	13	
08	C														
09	C			Part III	FIELD1	ADD	FIELD2	FIELD3	6	2			21		
10	C		21			Z-SUB	FIELD3	FIELD3							
11	C				FIELD3	COMP	125Ø.ØØ					11	12	13	
12	C														

Figure E8. Absolute Compare, Sign Reversal, and Sign Removal

Absolute Numeric Compare--Including Illustration of Sign Reversal and Sign Removal

Problem: A Compare (COMP) operation for numeric fields is always algebraic; i.e., the sign in the low-order position of the field is taken into consideration. For example: -8 is smaller than (+) 1.

Figure E8--Part I presents a method for comparing the absolute value (ignoring the sign) in a field to a constant, without changing the sign in the original field itself. Two of several alternate techniques are shown in Parts II and III--this time on the assumption that the sign in the field need not be preserved.

Assumptions: FIELD1, FIELD2, and FIELD3 represent numeric data fields; COMFLD is a new field created so as not to disturb the sign in a data field.

Explanations: Part I: As the data field (FIELD1) is transferred to a work field (COMFLD), it is tested for sign. If negative, the sign is reversed, so that comparison is always with a positive value.

The second line, incidentally, illustrates the simplest method for reversing the sign in a field.

Part II: The zone of the digit 5 (hexadecimal zone F--see EBCDIC table)--equivalent to "no sign"--replaces any other zone in the low-order position of FIELD3. Any other character in the EBCDIC-table column labelled F would do as well as a 5.

This illustrates a simple technique for removing a sign. (See also Figure E19 for removal of only a plus sign.)

Part III: The first line is assumed to represent any normal arithmetic operation. Advantage is taken of an operation that has to be performed anyway to ascertain the sign in the pertinent field. If negative, the sign is reversed by the Z-SUB operation.

The examples in Parts II and III sacrifice the original sign in the field.

In all three examples, the literal could have been specified simply as 1250 instead of a 1250.00. However, the format employed minimizes the core requirements for the operation by obviating decimal alignment (see Storage Requirements, Appendix A and Tips on Minimizing Core-Storage Requirements, in this appendix).

Arithmetic Overflow

Problem: During object-program execution, no indication is given when an arithmetic result exceeds the capacity of the Result Field: the excess high-order positions are lost.

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-3351-1
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	Control Level (L0-L9, LR)	Indicators: Not	Indicators	And: Not	Indicators	And: Not	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators Plus / Compare High 1>2	Minus / Low 1<2	Zero or Blank / Equal 1=2	Comments
3 4 5	6	7 8	9	10 11	12	13 14	15	16 17	18–27	28–32	33–42	43–48	49 50 51	52	53	54 55	56 57	58 59	60–74
01	C								FIELDA	MULT	FIELDB	AOWORK	15	Ø					
02	C									MOVE	AOWORK	PRODCT	12	Ø					
03	C								AOWORK	SUB	PRODCT	AOWORK				H1	H1		
04	C								PRODUCT	ADD	ADDEND	AOWORK							
05	C									MOVE	AOWORK	FINAL	12	Ø					
06	C		N	H1					AOWORK	SUB	FINAL	AOWORK				H1	H1		
07	C																		

Figure E9. Testing for Arithmetic Overflow

Figure E9 illustrates a method for recognizing such cases, when the Result-Field length is less than 15 positions.

Explanation: A work field (AOWORK = arithmetic overflow work field) is set up, long enough to accommodate the largest result that could occur in any arithmetic operation that is to be checked for overflow and larger than the fields ultimately destined to receive the results of the pertinent operations. The same work field can be used each time, provided the number of decimal places required is uniform. Two operations employing the same work field are shown in Figure E9. (NH1 is a conditioning indicator in the second example; otherwise, a correct second result would turn off H1 even if the first result overflowed.)

The result of the arithmetic operation is moved, right-aligned, to the ultimate Result Field (PRODCT or FINAL), and then subtracted back into the work field. If the result of this operation is not zero, overflow has occurred.

The work field must be large enough to contain a significant digit in the overflow portion (the excess length, to the left, beyond the ultimate result fields), even when part of the overflow portion could contain zero--for example: initial result of operation = 10045678. If the ultimate result field contains (say) six positions, overflow will not be detected unless the work field (containing the initial result) allows for at least eight positions--because the result happens to be 0 in the seventh position.

Function Table Containing Several Functions per Field

Problem: In a table look-up operation, it is sometimes desired to locate more than one function per argument.

Solution: Place the several functions for one argument next to each other so as to form one continuous field. In the file extension specifications and the LOKUP operation, treat the multiple function fields as a single field of a size that exactly accommodates the several functions. After the LOKUP operation, separate the individual functions by MOVE and MOVEL operations.

Note: If the functions are numeric, all subfunctions for one function field should have the same sign--which should only be punched in the table-input cards in the low-order position of the rightmost subfield.

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Plus / Compare High 1>2	Minus / Low 1<2	Zero or Blank / Equal 1=2
ARGUMT	LOKUP	TABARG	TABFUN						40
	MOVEL	TABFUN	FUNC#1	4					
	MOVE	TABFUN	FUNWRK	11					
	MOVEL	FUNWRK	FUNC#2	6					
	MOVE	FUNWRK	FUNC#3	5					

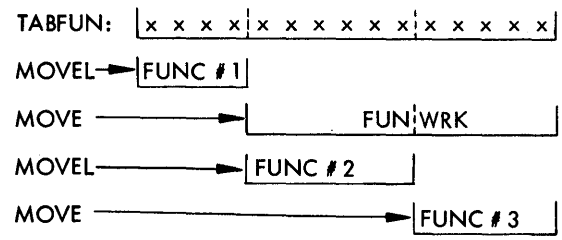


Figure E10. Multiple Functions from a Single LOKUP Operation

Figure E10--Part I illustrates the specifications to handle three functions for one argument, and Part II graphically portrays the Move operations.

Assumptions: 15-position function field, defined appropriately (and, in this example, as alphameric) in the file-extension specifications. It consists of three functions--from left to right: FUNC#1 (4 positions), FUNC#2 (6 positions), and FUNC#3 (5 positions).

Converting Units to Dozens

Problem: Input quantities are expressed in units; but output is to be expressed in dozens, and units less than a dozen.

Figure E11 shows two simple conversion routines.

Assumptions: UNITS field has been defined elsewhere, as numeric with 0 Decimal Positions.

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-3351-1
Printed in U. S. A.

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	High 1>2	Low 1<2	Equal 1=2	Comments
01	C			DZNRTN	TAG									
02	C			UNITS	SUB	12	UNITS					91	92	
03	C	OPTION I	N91	DOZENS	ADD	1	DOZENS	3	0					
04	C		N91 N92		GOTO	DZNRTN								
05	C		91	UNITS	ADD	12	UNITS							
06	C													
07	C	OPTION II		UNITS	DIV	12	DOZENS	3	0					
08	C				MVR		UNITS							
09	C													
10	C													

Figure E11. Converting Units to Dozens

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-3351-1
Printed in U. S. A.

Date ______ Program ______ Programmer ______

Punching Instruction: Graphic / Punch Page Program Identification

Line	Form Type	Control Level (L0-L9, LR)	Indicators (Not, And, Not, And, Not)	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Plus / Compare High 1>2	Minus / Low 1<2	Zero or Blank / Equal 1=2	Comments
01	C				Z-ADD	3ØØØØØØ	SQROOT	8	Ø					
02	C			SQRRTN	TAG									
03	C				Z-ADD	SQROOT	OLDROO	8	Ø					
04	C			SQUARE	DIV	SQROOT	WRKFLD	9	1					
05	C			WRKFLD	ADD	SQROOT	WRKFLD							
06	C			.5	MULT	WRKFLD	SQROOT			H				
07	C			SQROOT	COMP	OLDROO							9Ø	
08	C		N9Ø		GOTO	SQRRTN								
09	C													
10	C													

Figure E12. Square Root

Explanation--Option I: As long as 12 can be subtracted from UNITS without the contents of the field UNITS turning negative, 1 is added to DOZENS. The routine is repeated as long as the contents of UNITS is greater than zero after 12 has been subtracted. The last specifications line provides for restoring the UNITS value to what it was before subtraction of 12 caused it to turn negative.

Note: Indicator 92 (second and fourth lines) may be removed: this saves some core-storage space, but causes the program to execute one extra circuit through the loop whenever UNITS happens to be an integral multiple of 12.

Explanation--Option II: The number of units is divided by 12 and the result is stored in a new field named DOZENS. Since there are no decimal positions in dividend, divisor, or quotient, the remainder also is an integer. The remainder is moved to UNITS by the MVR operation, which first sets UNITS to zero; thus, excess positions beyond the two-digit size of the remainder are zero, and no longer contain the original UNITS value. (Alternatively, a new field can be assigned for the units less than a dozen, if it is desired to preserve the original total units.)

Square Root

Figure E12 presents one of several simple RPG routines for calculating a square root. The Newton-Raphson successive-approximation formula is applied:

$$\sqrt{S} = R_{i+1} = .5\left(\frac{S}{R_i} + R_i\right), \text{ where}$$

R_i = successive approximations of square root, and
S = the square (radicand) of which the square root is to be extracted.

Assumptions: Radicand (SQUARE) ≤ 14 positions, with 0 decimal places. As the example is designed, a 7-position square root is calculated; 8 positions are allowed for intermediate values (field SQROOT), but the high-order position will always be zero in the final result.

The user may change the sizes of the SQUARE and SQROOT fields, and may introduce decimal places into the radicand--provided he maintains the proper mathematical and RPG relationships for sizes and decimal places in all the pertinent fields.

Explanation of Figure E12: Relating the calculation specifications to the formula should clarify the operation, with these additional comments--

1. An extra position (decimal place) is assigned to the quotient (fourth line--WRKFID) for greater accuracy; it is then half-adjusted and dropped after the last calculation in the loop (the sixth line).

2. The routine is repeated until two successive half-adjusted results are identical (see seventh and eigth lines, and indicator 90). To permit the comparison of succesive results, the old result is stored at CLDROO (see third line).

3. An arbitrary initial value (3000000--first line) has been used for the first approximation. Optimally, to minimize the number iterations, the initial value should be of the same order of magnitude as the ultimate square root. If the magnitude of the radicands is fairly homogeneous, the user should substitute another, more appropriate, first-approximation value.

If square roots are to be extracted for values in a substantial number of cards, the following is a good technique for minimizing the number of iterations, if other aspects of the job do not preclude it:

> On a sorter, sort the data-card deck into sequence on the radicand field--at least on the first few high-order positions. Initialize SQROOT with an arbitrary value (see first line) only for the first card; for subsequent cards, always enter the routine at SQRRTN. This uses the square root from the preceding card as the first approximation. Since the radicand values are in sequence, the previous square root will be relatively close to the one for the new radicand value, and convergence will be rapid.

Output-Oriented Pointers

Repetitive Output

Problem: it is sometimes desired to repeat the same output (forms printing or card punching) a fixed or variable number of times. One common application is the printing of shipping labels.

Figure E13 presents a method for printing the same name and address a number of times from a single input card.

Assumptions:

1. The Name-and-Address cards include a field (NUMBER) which contains the number of times the data is to be printed.

2. The data from each card is to be printed at least once, and not more than nine times. (If more than nine times must be allowed for, simply increase the size of the fields NUMBER and CONTRL.)

3. Input consists of a single file composed solely of Name-and-Address cards.

4. No calculation specifications are required, except those that control the output iterations.

Note: If the user's application does not conform to the restrictions in items 3 and 4, above, he must make suitable modifications in the assignment and use of indicators. Caution will be required: the user must bear in mind that (1) in this example, the last output per input card occurs after the next card has been read, and the new card-type Resulting Indicator is on and (2) whenever (in each loop) the program advances from total-time output to detail-time calculations, it also passes through overflow output time--thus, if other operations in the job utilize the OF indicator, they will occur in each pass through the loop.

IBM — INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS

IBM System/360

Form X24-3350-1
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	Filename	Sequence	Field Location From	Field Location To	Decimal Positions	Field Name
01	I	NAMEADDR	AA				
02	I			1	1	Ø	NUMBER
03	I			11	3Ø		NAME
04	I			31	5Ø		ADDRSS
05	I			51	7Ø		CTYSTA
06	I						
07	I						

IBM — INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-3351-1
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	Control Level (L0-L9, LR)	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Resulting Indicators Plus	Comments
01	C		N9Ø		Z-ADD	NUMBER	CONTRL	1	Ø		
02	C			CONTRL	SUB	1	CONTRL			9Ø	
03	C		9Ø		GOTO	TOTOUT					
04	C	LØ		TOTOUT	TAG						
05	C										

IBM — INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System/360

Form X24-3352-1
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	Filename	Type (H/D/T)	Space After	Skip After	Field Name	End Position in Output Record	Constant or Edit Word
01	O	PRINT	T	1				
02	O					NAME	25	
03	O		T	1				
04	O					ADDRSS	25	
05	O		T		Ø1			
06	O					CTYSTA	25	
07	O							

Figure E13. Repetitive Output

Explanations for Figure E13 (Figure 6, RPG Program Logic will be a helpful reference): No card-type Resulting Indicator is needed in the input specifications, because there is only one card type. Output need not be conditioned by an indicator, because (1) it is at total time (T in col. 15); therefore, no spurious output occurs at detail time before the first card has been read (1P time), and (2) the data is to be printed every program cycle, except at the beginning before data from the first card read is available for output--and, at that time (the first cycle), total time is bypassed by the program.

The first line of the calculation specifications provides for a new field (CONTRL) to which the contents of NUMBER are transferred. This is done only the first time the program passes through detail-time calculations on each card (when indicator 90 is off). It is necessary because, each time before detail calculations are repeated for the same card, the program again transfers the input data to the process area: thus, the same data from col. 1 is repeatedly placed into the location for the field NUMBER. To control the number of iterations, a separate field is therefore needed.

In each pass through detail calculations, 1 is subtracted (second line) from the controlling number. As long as the result is greater than zero (indicator 90 on) the program branches (from the third line) to total-time calculation. When the contents of CONTRL are no longer greater than zero, the program follows its normal sequence: detail time is followed by the reading of a new card, which is in turn followed by total-time calculation and output.

It will be noted that indicator 90 turns off when the data has been printed once less than desired. However, after detail time has been completed, and the next card read, total-time calculations and output follow. The total-time output at that time is based on the data from the previous card, because the new input data is not transferred to the process area until just prior to detail time: thus, this output cycle supplies the data for the last time for each card. No spurious output is created after the first card has been read --whose data is not yet available at the first total-time output of that program cycle--because total time is bypassed at the beginning of program execution; however, when branching from detail time to total time, total-time operations are executed.

Note: During program generation, a cautionary message--"GOTO and TAG are not in the same calculation time"--will be printed. Since the branching from detail time to total time is intentional, the message should be ignored.

IBM — INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS
IBM System/360
Form X24-3351-1 Printed in U. S. A.

Line	Form Type	Control Level	Indicators		Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions
01	C			Option I	PGETOT	ADD	VALUE	PGETOT	6	0
02	C			Option I	CTRGRP	ADD	VALUE	CTRGRP	6	0
03	C			Option I	FINTOT	ADD	VALUE	FINTOT	8	0
04	C									
05	C			Option II	PGETOT	ADD	VALUE	PGETOT	6	0
06	C	L0	OF	Option II	CTRGRP	ADD	PGETOT	CTRGRP	6	0
07	C	L1	NOF	Option II	CTRGRP	ADD	PGETOT	CTRGRP		
08	C	L1		Option II	FINTOT	ADD	CTRGRP	FINTOT	8	0
09	C									

IBM — INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS
IBM System/360
Form X24-3352-1 Printed in U. S. A.

Line	Form Type	Filename	Type (H/D/T)	Space Before	Space After	Skip Before	Skip After	Output Indicators	Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record
01	O	REPORT	D		1			N1P				
02	O								VALUE	Z		10
03	O		T			04	01	OF NL1				
04	O	OR		2		04		L1				
05	O								PGETOT	Z	B	10
06	O		T				01	L1				
07	O								CTRGRP	Z	B	10
08	O		T					LR				
09	O								FINTOT			10
10	O											

Figure E14. Page Totals

Page Totals

Problem: Sometimes it is desired to list values and--when the forms-overflow point (channel 12) is reached--to print a total of the listed values at the bottom of the page before forms advance to the next page, although a control break may not have occurred.

Figure E14 presents two solutions: only the calculation specifications differ between the two. Option I cumulates the individual values directly into the three total fields (page total, control group total, and final total), whereas Option II employs total transfer from one total to the next higher. Option II uses more instructions and core-storage space, but minimizes the amount of calculation time.

Assumptions for this example:

1. The contents of an input field named VALUE are to be listed at detail time.

2. Totals of VALUE are desired at the bottom of each page, at the end of each

Control-Level-1 group, and at the end of the report.

3. Carriage tape channel 4 serves for the page-total location.

4. When there is a control break, a page total is to be forced, followed by the Control-Level total.

Explanations for Figure E14--calculation specifications: Option I shows straightforward addition of the individual values to the three totals; Option II accomplishes the same by total transfer to progressively higher-level totals. Noteworthy in Option II:

1. The second line provides for transfer--at total time--of the page total (PGETOT) to the Control-Level total (CTRGRP), provided the overflow point had been reached (OF on) during the preceding detail-time printing.

2. The third line accomplishes the same as the second whenever a control break occurs at a point on the page other than the overflow point. This is necessary so that the control-group total includes the values from a page that is only partially filled (OF not on) when a control break occurs.

Output-format specifications: The first two lines specify printing at detail time for each card, except that output is suppressed (N1P) before the first card has been read.

The third, fourth, and fifth lines provide for the printing of page totals, clearing the page total (B in col. 39) each time in anticipation of accumulation for the next page. When the overflow point coincides with a control break, the output must be performed at total-output time--not at overflow-output time; otherwise, the page total would be printed under the control-group total (see next two specifications lines), and not at all if LR is on--because overflow-time occurs later than total-output time. NL1 is specified with OF in Output Indicators so that the output does not occur at overflow time, but at total-output time (L1 in CR line). Also, if a control break has occurred, the form should not be advanced to a new page until the control-group total has also been printed on the old page: therefore, the different forms-control specifications in the main and OR lines.

The sixth and seventh lines provide for the control-group total beneath the page total, and for clearing of the control-group field so that the next group's total can be accumulated. The form is advanced to the top of a new page after printing of a group total.

The last two lines contain normal specifications for printing the grand total.

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-3351-1 Printed in U. S. A.

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Plus (High 1>2)	Minus (Low 1<2)	Zero or Blank (Equal 1=2)	Comments
01	C		Option I	SUMANY	ADD	ANYTHG	SUMANY	6	2				9Ø	
02	C		OF		SETON						99			
03	C		L1		SETOF						99			
05	C		Option II	SUMANY	ADD	ANYTHG	SUMANY	6	2					
06	C													
07	C													

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System/360

Form X24-3352-1 Printed in U. S. A.

Line	Form Type	Filename	Type (H/D/T)	Space Before	Space After	Skip Before	Skip After	Output Indicators	Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record	Constant or Edit Word
01	O	REPORT (Option I)	D		1								
02	O								ANYTHG			2Ø	' Ø. '
03	O		T	1	3			L1 NOF N99					
04	O		OR	1			Ø1	L1 99					
05	O		OR					L1 OF					
06	O							N9Ø	SUMANY		B	2Ø	' Ø. '
08	O	REPORT (Option II)	D		1								
09	O								ANYTHG	Z		2Ø	
10	O		T	1	3			L1 NOF					
11	O		OR	1			Ø1	L1 OF					
12	O								SUMANY	Z	B	2Ø	
13	O												
14	O												

Figure E15. Delayed Forms Cverflow

Delayed Forms Overflow

Problem: Forms overflow normally occurs after total-output time in the same program cycle in which the overflow point was reached. However, with small control groups, it may be desired always to complete the listing of cards of one group on the same page.

Figure E15 presents two techniques for delaying forms overflow until a control break has occurred at or beyond a carriage-tape channel-12 punch. Option I is based on a single channel-12 punch. Option II is based on succesive punches in channel 12 from the earliest desired overflow point through the last possible point--assuming that a group total might have been printed three lines above the first channel-12 punch.

Assumptions:

1. One or more fields are listed at detail time.

2. A group total is to be printed at total time at the end of each control group--Control Level L1 has been assigned to some control field in the input specifications.

3. Forms overflow is desired only after a control-group total (i.e., a group is not to be split between two pages)--and only provided a punch in channel 12 has been sensed.

4. The (first) channel-12 punch is high enough to allow space on the same page for printing the largest possible control group (+ 4 lines, because of the specific Space/Before and Space/After instructions in this example).

5. For Option II, channel 12 is punched for every line from the first overflow line (as explained in 4, above) through the last possible print line on the page.

6. Channel-1 punch represents the top of a page.

Explanations for Option I: The first line in the calculation specifications presents the normal manner of summing an amount field for control-group total; in addition, an indicator (90) is turned on whenever the total field (SUMANY) is zero (90 is used in the output specifications). The second line causes an indicator (99) to be set on at detail-calculation time, if the overflow indicator turned on after the preceding detail-time or total-time output--99 serves to "remember" that the overflow point has been passed. The third line causes 99 to be turned off during detail-time calculations for the first card of a control group because, if the overflow point had been reached before or during the preceding total-output time, the form was advanced to a new page (see output).

The first two lines of the output-format specifications take care of normal detail printing; lines 03-05 control the output of the L1 group total (preceded by an extra space), and forms overflow, as follows:

1. Line 03 causes printing at total time if channel 12 had not been passed during detail-time output--in the same (indicator OF) or a previous (indicator 99) program cycle. The form is advanced 3 spaces--but not to a new page--after the total is printed.

2. Line 04 causes printing at total time--followed by forms advance to the next page--if the overflow point had been passed during a previous program cycle (indicator 99).

3. Line 05 provides for printing at overflow-output time--followed by forms advance to the next page (forms control from the previous specifications line applies)--if the overflow point was passed during detail-time or total-time output of the same program cycle (indicator OF).

   This takes care of the situation where OF turned on after listing of the last card of the group.

   It is also possible that OF turned on only as a result of printing the group total on the basis of the specifications in line 03--the output is then performed twice: first at total-output time (per line 03), and again at overflow-output time (per line 05). In such case, the second output to the file must be performed, in order to get the forms advance to channel 1. However, as the field SUMANY is transferred to the output-storage area, it is reset to zeros (B in col. 39): this turns on indicator 90 (see first line of calculation specifications), and output from the field is suppressed (N90). (Note that this method assumes that the L1-level data total cannot be zero. If Zero Suppress, or an edit word that does not preserve .00 for an all-zero field, were used, output for the field need not be suppressed since nothing would then be printed when the field contents are zero--see Option II.)

Explanations for Option II: The calculation specifications consist only of the normal entries for summing an amount field. The first two lines (lines 08 and 09) of the output-format specifications take care of normal detail printing. Lines 10 and 11 control the output of the L1 group total (preceded by an extra space), and forms overflow, as follows:

1. Line 10 causes printing at total time if a channel-12 punch has not been previously sensed. The form is advanced 3 spaces--but not to a new page--after the total is printed.

2. Line 11 provides for printing at overflow output time--followed by forms advance to the next page--if a channel-12 punch was passed during output in the same program cycle. Since channel 12 is repeatedly punched, the initial overflow point could have been passed

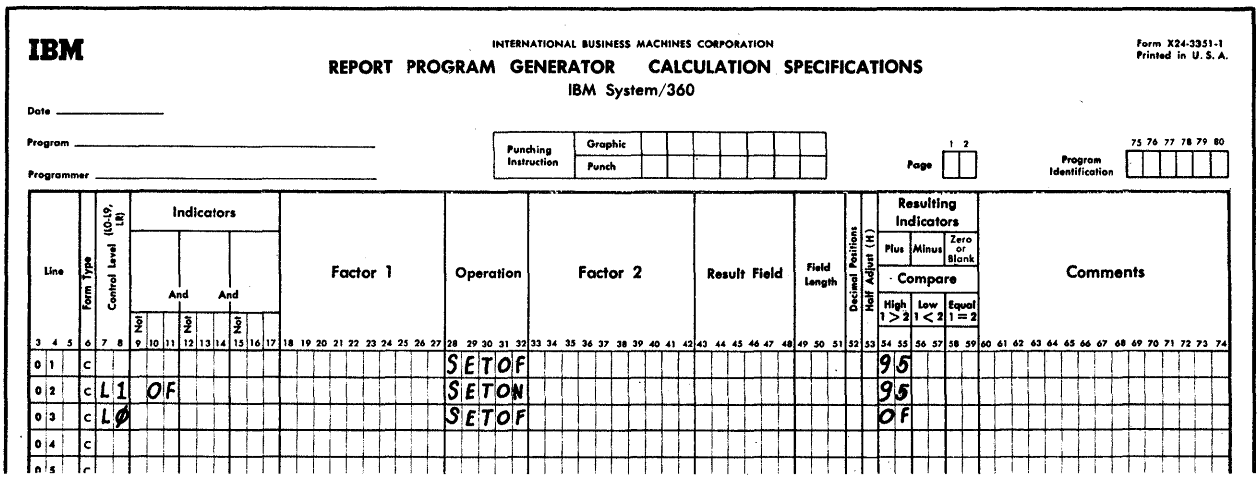
in a prior cycle when there was no control break.

Note that--if the first channel-12 punch is sensed at the line on which the L1 NOF total (spec. line 10) is printed at total-output time--the output will be repeated at overflow-output time, because OF is then on for the first time. The output to the file must be performed in such case in order to get forms advance to the next page. The field SUMANY is reset to zero after the first output (B in col. 39); nothing will be printed the second time, because Zero Suppress (Z in col. 38) eliminates the entire field contents (zeros).

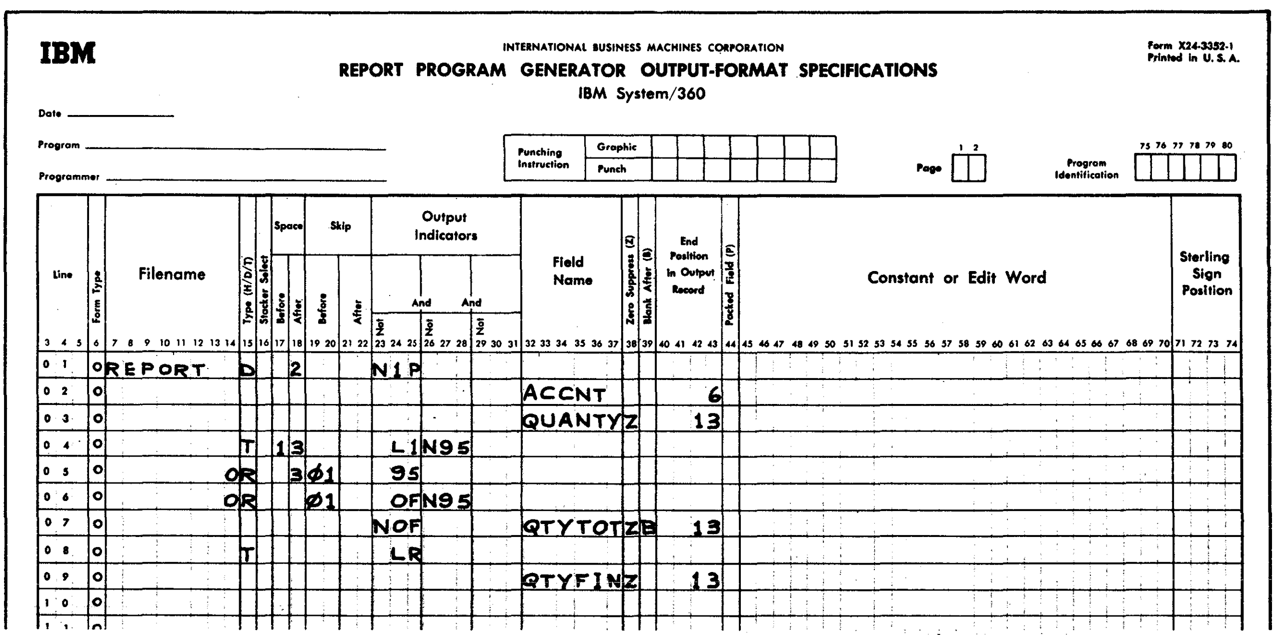
Overflow Forms Advance Before Totals

Problem: Normally, totals are printed on the same page as the last detail data, even if the overflow point (channel-12 punch) was passed during detail-time output of that program cycle. This is so because overflow time, or automatic forms advance to channel 1, occurs after total-time output.

Figure E16 shows how to advance the form to the next page before the printing of totals, if the overflow point was passed during detail-time output in the same program cycle.


IBM INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS
IBM System/360
Form X24-3351-1 Printed in U. S. A.

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust	Resulting Indicators Plus (High)	Minus (Low)	Zero (Equal)	Comments
01	C				SETOF						95			
02	C	L1	OF		SETON						95			
03	C	L0			SETOF						OF			
04	C													
05	C													


IBM INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS
IBM System/360
Form X24-3352-1 Printed in U. S. A.

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
01	O	REPORT	D			2			N1P							
02	O									ACCNT			6			
03	O									QUANTY	Z		13			
04	O		T		1	3			L1 N95							
05	O	OR				3	01		95							
06	O	OR					01		OF N95							
07	O								NOF	QTYTOT	Z	B	13			
08	O		T						LR							
09	O									QTYFIN	Z		13			
10	O															
11	O															

Figure E16. Forms Overflow Before Totals

Explanations: Output specifications in lines 01-03 are normal for detail printing when there is only one card type.

If overflow was signalled at detail-output time, indicator 95 is SETON at total-calculation time--provided there is also an L1 control break. (95 is SETOF each detail-calculation time so as not to affect output in subsequent cycles.)

Output-specifications line 04 provides for total-time output when there is a control break (L1 on), but the overflow point had not been passed during detail-time output (95 is off). The form is advanced 3 spaces after the total is printed.

Line 05 provides for total-time output when both a control break and an overflow signal have occurred--the only condition under which indicator 95 is on. This output is preceded by forms skipping (01 in cols. 19-20): thus, a total that was preceded by an overflow signal at detail-output time is printed on the next page.

Line 06 provides for output--preceded by forms skipping to the next page, but without any forms movement after output--to cover these situations:

1. Overflow is signalled at detail-output time, but no control break follows; or

2. Overflow is signalled as the L1 total is printed at total-output time on the old page (see line 04).

Specifying forms skipping at overflow-output time under these two conditions is necessary because automatic overflow forms skipping--which takes place when OF does not appear in any file-identification line --must be prevented. Otherwise--after the output specified by line 05 has taken place on a new page (see Skip/Before in line 05) --the form is again automatically skipped at overflow time, because the overflow indicator is then still on.

When output is performed at overflow time as a result of the execution of the specifications in line 06, output from the data fields must be suppressed--because either (1) no control break has occurred, or (2) the total was already printed (by specs. in line 04 or 05). To accomplish this, output from the field(s) is conditioned by NOF (see line 07). During each total-time calculation, OF is SETOF (see third calc. spec. line). This permits the output from the field to take place at total-output time (when OF is always off because it was set off), if the output conditions in line 04 or 05 are met. However, output from the field is always suppressed when the specifications in line 06 are executed: the overflow indicator is always turned on by the program itself before overflow-output time, if overflow was signalled during the preceding detail- or total-time output--and this supersedes any setting by a prior calculation instruction. (See CF, OV, under Indicators, in the chapter Programming For RPG--General Information.)

Single-Card Total Elimination

Problem: It is often desired not to print the lowest-level total when it would be identical with the quantity or amount listed immediately above it, because the control group consisted of a single card.

Figure E17 presents two methods for accomplishing this. Option I does not, however, save the actual total-print operation; whereas Option II does not go through the total-print operation at all for a single-card group. While not shown, the techniques can also be adapted to elimination of higher-level (say, L2) totals that consist of a single lower-level (say, L1) group.

Assumptions:

1. An asterisk is to be printed next to the total; when the total is suppressed for a single-card group, the asterisk is to be printed next to the listed value.

2. When the total is eliminated, the same extra spaces as after a total should appear after the single item line.

General explanation: The earliest point at which it is known whether a group consists of only a single card is at total time following the last card's detail time. The calculation and output specifications are based on this fact.

Explanation--Option I: Normal listing at detail time. Space/Before must be used because, when the group consists of a single card, the total identifying asterisk--printed at total time--must be printed next to the listed item: therefore, there must be no forms movement until total-time output or the next detail time.

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-3351-1
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	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators Plus (High 1>2)	Minus (Low 1<2)	Zero or Blank (Equal 1=2)	Comments
01	C				SETOF						81	82		
02	C		L1		SETON						81			
03	C		L1	SUMB	SUB	SUMB	SUMB							
04	C													
05	C													
06	C			SUMB	ADD	FLDB	SUMB	6	2					
07	C													
08	C													
09	C	L1	81		SETON						82			
10	C													
11	C													

Options I and II

Omit Line 03 With Option II

Detail-Time Specs

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System/360

Form X24-3352-1
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	Filename	Type (H/D/T)	Space Before	Space After	Skip Before	Skip After	Output Indicators	Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record	Constant or Edit Word	Sterling Sign Position
01	O	REPORT	D	1				N1P						
02	O								FLDA			8		
03	O								FLDB	Z		19		
04	O		T	1	2			L1 N82						
05	O	OR			2			82						
06	O							N82	SUMB	Z		19		
07	O											20	'*'	
~~08~~	~~O~~													
09	O	REPORT	T		1			N82						
10	O	OR			2			82						
11	O								FLDA			8		
12	O							N82	FLDB	Z		19		
13	O							82	SUMB	Z	B	19		
14	O							82				20	'*'	
15	O		T		2			L1 N82						
16	O								SUMB	Z	B	19		
17	O											20	'*'	
	O													

Option I

Option II

Figure E17. Single-Card Total Elimination

If there is a control break (L1 on), output is also performed at total time. The total-identifying asterisk is then printed; but the contents of the total field (SUMB) are printed only if the group consisted of more than one card (N82)--otherwise, the output from FLDB and SUMB would be identical.

If the group consists of more than one card (N82)--i.e., SUMB wil be printed--the form is spaced before the total is printed (otherwise, SUMB data overprints FLDB data), and after; for a single-card group (82 on), it is advanced only after printing, so that the asterisk for total identification appears next to the FLDB data.

The calculation specifications are routine, except

1. Indicator 82 is turned on at total time (line 09) when there is a control break (L1 in cols. 7-8)--provided there was also a control break on the previous card (81 on). (L1 on at detail time turns on 81--line 02.)

   Indicators 81 and 82 are set off (line 01) before these criteria are tested.

2. Because SUMB is not printed for single-card groups (see N82 in line 06, output), Blank-After cannot be used in the output specifications to clear the field. Therefore, it is cleared to zero in the calculations specifications (line 03) at the beginning of each control group, before the FLDB data from the first card of the group is added in.

Explanation--Option II: Because it is known by total time whether the last card represented a single-card group, it is possible to avoid the output operation for the total sum altogether for single-card groups if the listed output from card fields is also performed at total time (the data from the last card is then still available). The second output (line 15) is not performed at all unless the group contained more than one card (N82).

For multiple-card groups (N82), FLDB is printed (line 12) from each card, and SUMB is printed (see lines 15 and 16) beneath the last FLDB value. For single-card groups, FLDB is suppressed (N82, line 12), and SUMB (82, line 13) is substituted at the same point in the cycle; the second output (lines 15-17) is not performed.

FLDB and SUMB contain the same value for single-card groups; but this approach permits use of Blank-After (B in col. 39) to reset SUMB , since SUMB is always printed--either via line 13 or line 16.

The asterisk is printed next to SUMB for either multiple- or single-card groups (line 14 or 17).

The spacing instructions in the OR line (line 10) and in the second output file-identification line (line 15) provide for a uniform extra space before the first card of a new group, regardless of whether it follows a multiple- or single-card group.

Because Blank-After can be used in Option II, line 03 in the calculation specifications is not needed.

Note: At total time, the card type Resulting Indicator for the new card is already on. Thus , if the operation in Option II differs for various card types--and the user therefore assigns card-type indicators in Output Indicators and in calculation Indicators--he must be careful to preserve appropriate card identification (by setting indicators at detail time in the calculation specifications to reflect the pertinent card type at total time).

Eliminating Excess Control Breaks (Major-Minor Control)

Problem: If a deck of cards consists of detail cards with two control fields (assigned level L1 and L2)--but each L2-level group is preceded by a single card of another type (say, to print a heading) which contains only the L2 control field--then a spurious L1 control break may occur between the L2-level heading card and the first detail card that follows. This wastes printer time, causes spacing as specified for L1 output and--if all leading zeros are not suppressed--causes printing of some zeros and (possibly) symbols (depending on the edit word--see edit word in output specifications for L1 total in Figure E18).

IBM — INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-3351-1
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	Control Level (L0-L9, LR)	Indicators			Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Plus (High 1>2)	Minus (Low 1<2)	Zero or Blank (Equal 1=2)	Comments
01	C	LØ	5Ø				SETOF						5Ø	L1		
02	C	L2	Ø1				SETON						5Ø			
03	C															

IBM — INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System/360

Form X24-3352-1
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	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
01	O	REPORT	D			2			Ø1									
02	O											DESCR			25			
03	O		D			1			Ø2									
04	O											GROUP			5			
05	O											AMOUNT			15		' Ø. '	
06	O		T			2			L1									
07	O											GRPTOT			15		' Ø. '	
08	O		T					Ø1	L2									
09	O											MAJTOT			15		' , Ø. '	
10	O														16		'*'	
11	O																	

Figure E18. Eliminating Excess Control Breaks

Figure E18 shows how to eliminate the false control break.

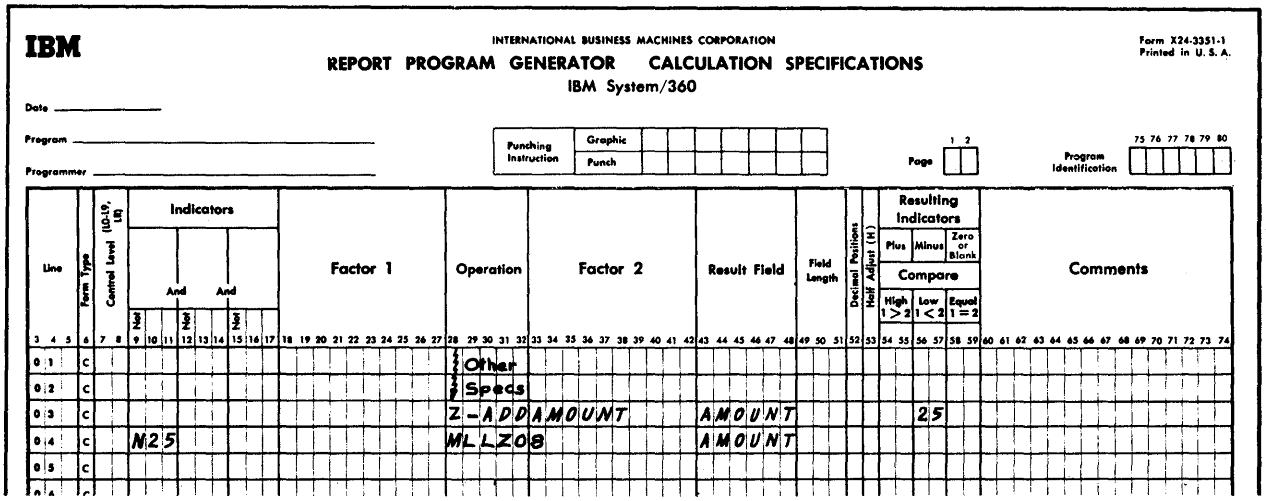
Assumptions:

1. Master cards are assigned card-type Resulting Indicator 01; details, indicator 02.

2. L2 control is assigned to both card types; L1 only to details.

Explanation: The output-format specifications are normal ones for the situation described.

The second calculation specification sets on an indicator (50) at total time if an L2 break has occurred--but only provided the new card is a heading master (to guard against a missing master and cards out of order). Next program cycle (at total time), when the first detail card following the master may have caused an L1 control break, L1 is turned off. 50 is also turned off, so that subsequent minor control breaks remain valid . Since L1 has been turned off before total-time output, the L1 output (GRPTOT) is not performed following a master header card.


INTERNATIONAL BUSINESS MACHINES CORPORATION
IBM REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS
IBM System/360
Form X24-3351-1
Printed in U. S. A.

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust	Resulting Indicators Plus	Minus	Zero or Blank	Comments
01	C				Other									
02	C				Specs									
03	C				Z-ADD	AMOUNT	AMOUNT					25		
04	C		N25		MLLZO	8	AMOUNT							
05	C													
06	C													

Figure E19. Preventing 12-Overpunch

Preventing 12-Overpunch

Problem: When punching into a card from a non-negative numeric field that was read in with a 12-overpunch in the low-order position, or to which Field Indicators were assigned in the Input Specifications, or which served as Result Field in an arithmetic operation, a 12-punch will be punched over the digit in the low-order position if no specifications are provided to prevent this. (See Zone Elimination, under Rules for Forming Edit Words, for a full list of situations that cause a field to become zoned).

Zero Suppress (Z in col. 38) of the output-format specifications eliminates the 12-overpunch; but it also eliminates any 11-overpunch (minus) and leading zeros, which is usually not acceptable in a card. An edit word, too, removes the 12-overpunch; but it also eliminates at least one leading zero, and requires an additional card column for the 11-punch for a minus sign.

Figure E19 presents a technique for removing the 12-overpunch without eliminating a possible 11-overpunch or any leading zeros.

Explanation: The sign of the pertinent field (AMOUNT) is ascertained; if it is not negative, an F-zone (tantamount to a "no-zone") is moved to its low-order position (instead of an 8, any other character in EBCDIC column F could also be used). This removes the 12-zone placed there by the program if the result was positive or zero. The example shows use of the cheapest operation (Z-ADD)--in terms of core usage--to test the sign (of course, this extra operation can be obviated by assigning an indicator to Minus, or to Plus and Zero, the last time the field serves as Result Field in an arithmetic operation that is required anyway as part of the application).

Punch output is assumed to be without Zero Suppress or edit word.

Note: Figures 43 and E8 illustrate removal of 12- or 11-overpunch.

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-3351-1
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	Control Level (L0-L9, LR)	Indicators (Not, And, Not, And, Not)	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators Plus / Compare High 1>2	Minus / Low 1<2	Zero or Blank / Equal 1=2	Comments
01	C			Method A	Z-ADD	AMOUNT	AMT	7	Ø					
02	C				Z-ADD	SOCSEC	SS*	1Ø	Ø					
03	C			Method B	MOVEL	SOCSEC	SS1	3	Ø					
04	C				MOVE	SOCSEC	SSWORK	6	Ø					
05	C				MOVEL	SSWORK	SS2	2	Ø					
06	C				MOVE	SSWORK	SS3	4	Ø					
07	C			Part C	Z-ADD	CENTS	CENTS						2Ø	
08	C				MLLZO	1	CENTS							
09	C													

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System/360

Form X24-3352-1
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	Filename	Type (H/D/T)	Stacker Select	Space Before	Space After	Skip Before	Skip After	Output Indicators (Not, And, Not, And, Not)	Field Name	Zero Suppress (Z)	Blank After (B)	End Position in Output Record	Packed Field (P)	Constant or Edit Word	Examples of Printed Output
01	O															
02	O								Method A	AMT			1Ø		'$Ø , . '	ƀ$Ø,ØØØ.ØØ
03	O									SS#			3Ø		'Ø - - '	ƀØ95-14-ØØ36
04	O								Method B	SS1			22			Ø95
05	O												23		'-'	-
06	O									SS2			25			14
07	O												26		'-'	-
08	O									SS3			3Ø			ØØ36
09	O															
10	O								N2Ø Part C				4Ø		'.'	ƀ or .
11	O								N2Ø	CENTS			42			ƀƀ or Ø5
12	O															

Figure E20. Editing Pointers

Editing Pointers

Problem 1: An edit word is to be assigned for printout, (1) to prevent zoning of the low-order position, and/or (2) to insert characters (symbols) between digits, or to precede the amount with fixed $ symbol--but all leading zeros, and constants among them, are to be printed.

Solution A: Increase the size of the field by one position. Place a zero in the extreme left position of the edit-word body so that, when the minimum of one leading zero is suppressed, a field of the correct size--retaining all leading zeros for the original field size--is printed.

Of course, the size of the field in the output-storage area is then one position larger than the original field would have been, even though the high-order position is always blank. Care must be taken, therefore, to allow for the greater length in End Position in Output Record, so as not to overlay two fields. Alternatively, if the extra (blank) position cannot be spared in the output record: Specify output for the artificially enlarged field ahead of specifications for the field that is to appear to its immediate left in the output record; the rightmost position of the later-specified output field is then overlaid over the excess blank leftmost position of the enlarged field--data is transferred to the output area in the sequence in which output specifications occur (except for punch data always being transferred ahead of card-print data for the same card).

Solution B: Split the field by MOVE and MOVEL operations, then treat it as several fields. Do not use an edit word. Treat symbols as constants.

Figure E20--Methods A and B--illustrates the two approaches.

Assumptions:

1. Field SOCSEC defined in input specifications as nine digits long--contents are 095140036.

2. Field AMOUNT defined in input specifications as six digits long, including two decimal places--contents are 000000.

Problem 2: Printing a two-digit field (say, consisting of cents only) so that it is preceded by a decimal point when there are significant digits in the field, but eliminating the printout altogether when the field contains only zeros. E.g.: If field contents are 05--print .05; if field contents are 00--leave output area blank.

Solution: An edit word cannot be used, because a character (symbol) preceding the first digit is never printed (except for a fixed $ sign), and because a leading zero would be replaced by blank. Therefore:

1. Specify the field for output without an edit word;

2. Specify a period (.) as a constant for the preceding print position;

3. Suppress output of the field and the constant when the field contents are zero.

Figure E20--Part C--illustrates this approach.

Assumption: The two-digit field CENTS was defined in the input specifications, or elsewhere in the calculation specifications, as a numeric field.

Comments on Part C: The first calculation specifications line (Z-ADD, to test for zero is unnecessary if an indicator can be assigned to Zero (or to Plus and Minus) in Field Indicators, or in any other arithmetic operation using CENTS as Result Field which may be required for the application (if the contents of CENTS are not changed thereafter, before output).

The second calculation specification (MLLZO) is necessary because result fields of arithmetic operations (or, alternatively, fields used with Field Indicators in input specifications) are signed (hexadecimal C or D). The output is not to be edited; therefore, the equivalent of the 12-punch or 11-punch zone must be removed by calculation specifications. (If a minus or CR symbol is to be printed for a negative amount, it can be assigned as a constant in the output specifications, and its output controlled by an indicator assigned to Minus in the calculation specifications.)

Date on Same Print Line as Constant Heading Data

Problem: The date--read from a lead card--is to be printed on the same line as the report heading, which is specified as a constant. The date cannot be printed during the first detail-output time (1P time) --when constant report headings are usually printed--because the Date lead card has not yet been read at that time.

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS

IBM System/360

Form X24-3350-1
Printed in U. S. A.

Date ______ Program ______ Programmer ______

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

Line (3-5)	Form Type (6)	Filename (7-14)	Sequence (15-16)	Number (1-N) (17)	Option (O) (18)	Resulting Indicator (19-20)	Record Identification Codes 1: Position (21-24)	Not (N) (25)	C/Z/D (26)	Character (27)	2: Position (28-31)	Not (N) (32)	C/Z/D (33)	Character (34)	3: Position (35-38)	Not (N) (39)	C/Z/D (40)	Character (41)	Stacker Select (42)	Packed (P) (43)	Field Location From (44-47)	Field Location To (48-51)	Decimal Positions (52)	Field Name (53-58)	Control Level (L1-L9) (59-60)	Matching Fields or Chaining Fields (61-62)	Field-Record Relation (63-64)	Field Indicators Plus (65-66)	Minus (67-68)	Zero or Blank (69-70)	Sterling Sign Position (71-74)
01	I	INPUT	AA			99	8ø		C										2												
02	I																				1	5	ø	DATE							
03	I		AB			ø1	8ø		C	1																					
04	I																				1	6	ø	ACCNT#	L1						
05	I																				7	12	2	AMOUNT							
06	I																														
07	I																							etc.							
08	I																														
09	I																														
10	I																														
11	I																														

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System/360

Form X24-3352-1
Printed in U. S. A.

Date ______ Program ______ Programmer ______

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

Line (3-5)	Form Type (6)	Filename (7-14)	Type (H/D/T) (15)	Stacker Select (16)	Space Before (17)	Space After (18)	Skip Before (19-20)	Skip After (21-22)	Output Indicators Not (23)	(24-25)	And Not (26)	(27-28)	And Not (29)	(30-31)		Field Name (32-37)	Zero Suppress (Z) (38)	Blank After (B) (39)	End Position in Output Record (40-43)	Packed Field (P) (44)	Constant or Edit Word (45-70)	Sterling Sign Position (71-74)
01	O	DSREPORT	D			3	ø1			L1												
02	O	O	R							OF	N	L1										
03	O																		65		'DAILY SALES REGISTER'	
04	O														Option I	DATE			75		' / / '	
05	O		D			1				ø1												
06	O									L1						ACCNT#	Z		6			
07	O															AMOUNT			15		' ø. '	
08	O																					
09	O																					
10	O	DSREPORT	D			3	ø1			99												
11	O	O	R							OF												
12	O														Option II				65		'DAILY SALES REGISTER'	
13	O															DATE			75		' / / '	
14	O		D			1				ø1												
15	O															ACCNT#	Z		6			
16	O															AMOUNT			15		' ø. '	
	O																					
	O																					

Figure E21. Date on Same Print Line as Constant Heading Data

Figure E21 presents two of several available solutions.

Explanation of Option I: If the heading is to be printed after each control break (say, Control Level L1) and at the top of each overflow page, report heading and date are simply specified as the first printed output at detail-output time for the first card of each control group and at overflow-output time. This takes care, too, of heading the first page (otherwise done at 1P time), because Control-Level indicators are also on at detail time of the very first card for which Control Levels are designated (see main text for special situation when control fields are zero or blank in first card).

Explanation of Option II: This method is independent of Control Levels--we have assumed that the form is not skipped to a fresh page after each control break (or that there are no Control Levels assigned).

The report heading and date for the first page are printed as the only output at detail time during processing of the Date lead card; they are repeated at the top of each overflow page.

Selecting the Last Card of Each Control Group

The PLACE specification card in the PCU Collate Program offers the simplest method of selecting the last card of each control group, using the IBM 2560 MFCM attached to the Model 20; an IBM collator provides a simple method of accomplishing the same, offline, without tying up the Model 20 System.

If it is desired to select the last card of each control group as an incidental to the RPG processing of a complete application with the Model 20, this can be achieved by branching to a Basic Assembler Language subroutine.

Figure E22 illustrates the necessary program instructions.

Assumption: The B.A.L. program is assembled separately.

Restrictions:

1. The file must be in the MFCM.
2. The file must be defined as a combined file.
3. No stacker selection--not even to the normal stacker--may be designated in the input or output specifications for the relevant card type.
4. The last card of each group must be directed to a non-normal stacker, with the others allowed to enter the normal stacker for the hopper involved.
5. An output operation (punching and/or interpreting) must be performed for the card that is to be selected. (This could be merely the "punching" of a blank constant in col. 1.)
6. Output (punching) must be at detail time.
7. The pertinent field(s) must be punched into all cards of that type; i.e., it is not possible to punch only the last card of the group.

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS

IBM System/360

Form X24-3347-3
Printed in U. S. A.

Date ______ Program ______ Programmer ______

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

Line (3-5)	Form Type (6)	Filename (7-14)	File Type (15)	File Designation (16)	Device (40-46)
01	F	DATACARD	C	P	MFCM1
02	F				
03	F				

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS

IBM System/360

Form X24-3350-1
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	Filename	Sequence	Resulting Indicator	Record Identification Codes 1: Position	C/Z/D	Character	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)
01	I	DATACARD	AA	Ø1	8Ø	C	1					
02	I							1	5		FLDA	L1
03	I							6	12	2	FLDB	
04	I											

IBM INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-3351-1
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	Control Level (L0-L9, LR)	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions
01	C				ANY DETAIL SPECS				
02	C								
03	C		Ø1	SUMFLD	ADD	FLDB	SUMFLD	8	2
04	C	L1			EXIT	SSEL			
05	C	L1		SUMFLD	SUB	SUMFLD	SUMFLD		
06	C								

Figure E22A. Selecting Last Card of Each Control Group--Part I

IBM | INTERNATIONAL BUSINESS MACHINES CORPORATION
REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS
IBM System/360 | Form X24-3352-1 Printed in U. S. A.

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

Line (3-5)	Form Type (6)	Filename (7-14)	Type (H/D/T) (15)	Stacker Select (16)	Space Before (17)	Space After (18)	Skip Before (19-20)	Skip After (21-22)	Output Indicators: Not (23)	(24-25)	And Not (26)	(27-28)	And Not (29)	(30-31)	Field Name (32-37)	Zero Suppress (Z) (38)	Blank After (B) (39)	End Position in Output Record (40-43)	Packed Field (P) (44)	Constant or Edit Word (45-70)	Sterling Sign Position (71-74)
0 1	O	DATACARD	D							ø1											
0 2	O														SUMFLD			2ø			
0 3	O																				
0 4	O																				
0 5	O																				
0 6	O																				
0 7	O																				

IBM | IBM System/360 Basic Assembler Short Coding Form | X28-6506-2 Printed in U.S.A.

PROGRAM ______ PROGRAMMER ______ DATE ______ | PUNCHING INSTRUCTIONS: GRAPHIC / PUNCH | PAGE OF | CARD ELECTRO NUMBER

Name (25-30)	Operation (32-36)	Operand (38-71)	Comments	Identification-Sequence (73-80)
SSEL	START	ø		
	USING	*,ø		
	CIO	2,X'22'		
	BCR	15,14		
	END	SSEL		

Figure E22B. Selecting Last Card of Each Control Group--Part II

<u>Comments on Figure E22:</u> In the calculation specifications, only line 04 is directly germane to the stacker-selection technique. However, to relate the example to a realistic problem, we have assumed that a detail field (FLDB) is to be summed (line 03), and the sum (SUMFLD) is to be punched.

It is not possible to recognize the last card of a control group in time to control punching into it only; therefore, SUMFLD is punched into each card of the pertinent type. The total punched is thus a cumulative one: in the first card of the group it equals FLDB; in the last card, it represents the group total. By stacker-selecting the last card of each control group, a card deck is formed in the select stacker (say, stacker 2) with each card containing the total of one control group.

Blank-After cannot be used, because punching from SUMFLD occurs each detail time--not only at the end of a group. Line 05 of the calculation specifications provides for resetting SUMFLD at the end of each group.

<u>Note</u>: The same principle may be applied to determine stacker selection for a card based on the type of card that follows it.

Exit to the subroutine is then conditioned by the card-type Resulting Indicator for the new (following) card and that of the preceding card. The preceding card's type is "remembered" by setting an indicator (SETON) during its processing cycle. <u>(remember to SETOF that indicator again at an appropriate point).</u>

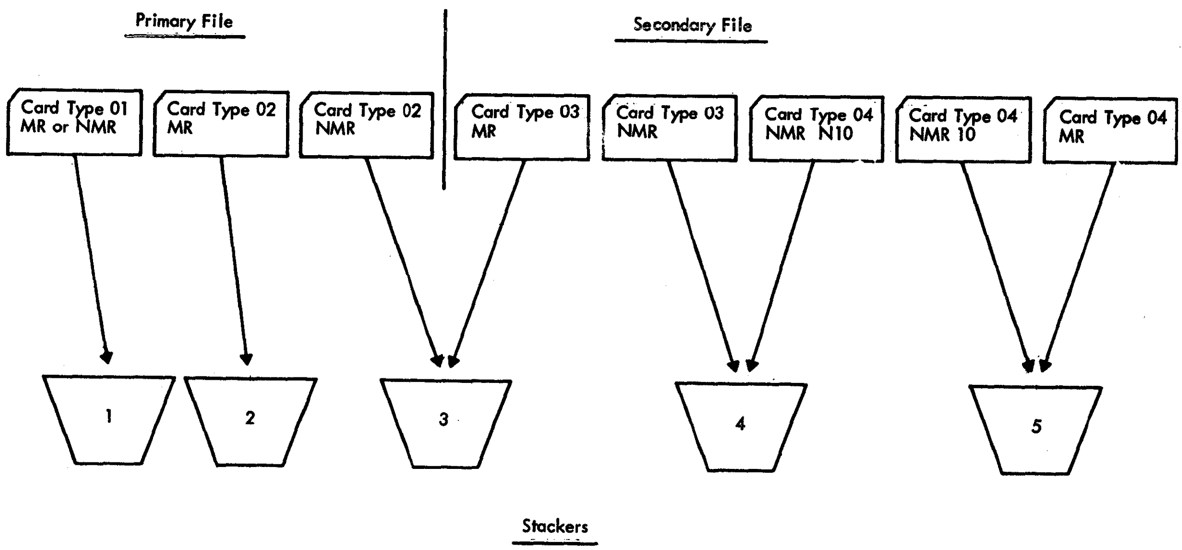


Figure E23A. Stacker Selection of Input-File Cards Based on File-Matching and/or Calculation Results--Schematic of Card-Type Input Hoppers and Destination Stackers

Stacker Selection of Input-File Cards Based on Matching of Files and/or Calculation Results

Problem: In some applications, it is necessary to direct input cards to different stackers based on their matching status (MR on or not on) in a file-matching operation and/or based on the results of calculations--yet no output operation (punching or card document-printing) is required for the job. To accomplish this with RPG, the file(s) must be defined as (a) combined file(s) and the pertinent File Identification specifications, including Stacker Select, must be entered.

Figure E23 presents a method, involving BAL subroutines, that accomplishes the desired stacker selection while maximizing throughput.

Assumption: The BAL routines are assembled separately before generation of the RPG object program.

Restrictions:

1. The relevant file(s) must be in the MFCM.

2. The relevant file or files must be defined as input files, and no punching can be performed in cards of such file(s).

3. No document-printing is to be performed on cards in any file--not even in a different file from the one involved in the BAL stacker-selection subroutines.

Note: It is permissible to designate stackers for some card types in a file in the input specifications and for other types by BAL subroutine.

IBM — INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS

IBM System/360

Form X24-3347-3
Printed in U. S. A.

Date ______ Program ______ Programmer ______

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

Line (3-5)	Form Type (6)	Filename (7-14)	File Type I/O/U/C (15)	File Designation P/S/C/R/T (16)	End of File E (17)	Sequence A/D (18)	File Format F/V (19)	Block Length (20-23)	Record Length (24-27)	L/R (28)	Mode of Processing / Length of Key Field or of Record Address Field (29-30)	Record Address Type K/I (31)	Type of File Organization I/D/T (32)	Overflow Indicator (33-34)	Key Field Starting Location (35-38)	Extension Code E/L (39)	Device (40-46)	Symbolic Device (47-52)	Labels (S, N, or E) (53)	Name of Label Exit (54-59)	Extent Exit for DAM (60-65)	File Addition A (66)	No. Tracks for Cylinder Overflows (67-68)	No. of Extents (69)	Tape Rewind N/U (70)	(71-74)
01	F	MASTER	I			A											MFCM1									
02	F	TRSACTN	I			A											MFCM2									
03	F																									

IBM — INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS

IBM System/360

Form X24-3350-1
Printed in U. S. A.

Date ______ Program ______ Programmer ______

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

Line (3-5)	Form Type (6)	Filename (7-14)	Sequence (15-16)	Number (1-N) (17)	Option (O) (18)	Resulting Indicator (19-20)	Record Identification Codes 1: Position (21-24)	Not (N) (25)	C/Z/D (26)	Character (27)	2: Position (28-31)	Not (N) (32)	C/Z/D (33)	Character (34)	3: Position (35-38)	Not (N) (39)	C/Z/D (40)	Character (41)	Stacker Select (42)	Packed (P) (43)	Field Location From (44-47)	To (48-51)	Decimal Positions (52)	Field Name (53-58)	Control Level (L1-L9) (59-60)	Matching Fields or Chaining Fields (61-62)	Field-Record Relation (63-64)	Field Indicators Plus (65-66)	Minus (67-68)	Zero or Blank (69-70)	Sterling Sign Position (71-74)
01	I	MASTER	MA			01	80		C	1																					
02	I	OR				02	80		C	2																					
03	I																				1	4	0	STKNO		M1					
04	I	TRSACTN	TR			03	80		C	3																					
05	I	OR				04	80		C	4																					
06	I																				1	4	0	STKNO		M1					
07	I																				5	6	0	FLDA							
08	I																				7	8	0	FLDB							
09	I																														
10	I																														

Figure E23B. Stacker Selection of Input-File Cards Based on File-Matching and/or Calculation Results--File Description and Input Specifications

Comments on Figure E23: Figure E23A shows the stackers to which the different cards are directed by the specifications in Figure E23C; any other arrangement is equally feasible if appropriate modifications are made in the calculation-specifications entries. To minimize the number of BAL subroutines, no stacker-select instructions are given for cards that are to enter the normal stackers (card type 01; card types 04, MR; and 04, NMR, 10); it is unnecessary (though permissible) to specify the normal stackers.

Summary Punching Matching-Group Totals into Primary Trailer Cards

Problem: Usually, summary punching of totals for groups of matched primary and secondary (and, perhaps, also tertiary) cards is accomplished by first merging a blank card behind each secondary (or tertiary) group. (The application example Pre-Billing with Inventory Control, Figures 63 to 69, utilizes this standard approach.)

Occasionally, it may be necessary to design a job so that the blank card--into which data is to be punched at the end of each complete group--is merged behind the last card of each primary group. However--as explained in the earlier section Matching of Files--blank cards are always processed immediately after the previous card in the same file. It is, therefore, not possible to delay the processing of (i.e., punching into) a blank card in the primary file until secondary- (or tertiary-) file cards of the matching group have been processed.

IBM — INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-3351-1 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	Control Level (L0-L9, LR)	Indicators (Not, And Not, And Not)	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators Plus (High 1>2)	Minus (Low 1<2)	Zero or Blank (Equal 1=2)	Comments
01	C		ø4	FLDA	SUB	FLDB	FLDC	3	ø			1ø		
02	C		ø2 MR		EXIT	SS2								
03	C		ø2 NMR		EXIT	SS3								
04	C		ø3 MR		EXIT	SS3								
05	C		ø3 NMR		EXIT	SS4								
06	C		ø4 NMR N1ø		EXIT	SS4								
07	C													
08	C													

IBM System/360 Basic Assembler Short Coding Form

X28-6506-2 Printed in U.S.A.

PROGRAM ______ PROGRAMMER ______ DATE ______ PUNCHING INSTRUCTIONS: GRAPHIC / PUNCH — PAGE OF — CARD ELECTRO NUMBER

Name	Operation	Operand	Comments	Identification-Sequence
SS2	START	ø		
	USING	*,ø		
	CIO	2,X'22'		
	BCR	15,14		
	END	SS2		
SS3	START	ø		
	USING	*,ø		
	CIO	3,X'22'		
	BCR	15,14		
	END	SS3		
SS4	START	ø		
	USING	*,ø		
	CIO	4,X'22'		
	BCR	15,14		
	END	SS4		

Figure E23C. Stacker Selection of Input-File Cards Based on File-Matching and/or Calculation Results--Calculation and BAL Specifications

Solution: As a blank card is merged behind each primary group, in preparation for the job, it is punched with the same Matching Fields data contained in the primary group. In addition, a constant (say, 1) is punched into all blank cards, in any column that will not be needed. The extra column is then used as the lowest (M1) matching field. The "blank" card then is always higher in sequence than the group to which it belongs, but lower than the next group, and is thus processed after the last card of the matching group.

Summary-punching into the "blank" card is at total time when--although the "blanks" are unmatched (on M1)--the MR indicator is still on from the last card of a matched group, but the new card-type indicator is already on.

Restrictions:

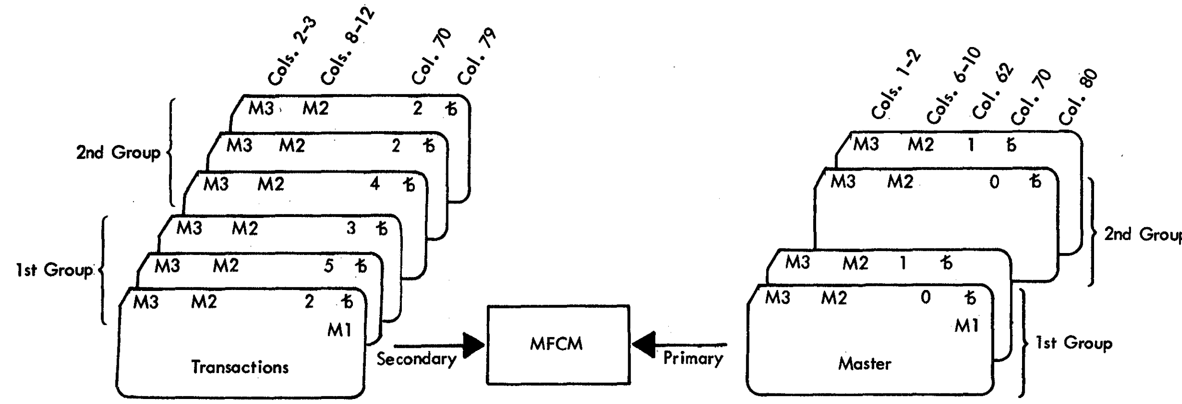
1. There must be a column in all cards (other than the "blank" card) that is always blank.

   (Alternatively, it may, instead of being blank, contain the same entry in all such cards if the value of that entry is lower--if ascending sequence applies--than the value of the constant punched into the blank cards. "Lower" refers to the applicable collating sequence: EBCDIC or user-altered collating sequence; if numeric is specified for the field, only hexadecimal column F applies.)

2. Not more than two Matching-Fields levels (M2 and M3) can be needed for the job, so that M1 is available to control the movement of the "blank" cards.

Note: If descending sequence applies, a blank column serves as M1 in the "blank" cards, and a uniform value higher than blank must be punched (or must already exist) in all other cards.

Figure E24 illustrates the problem.



IBM INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR FILE DESCRIPTION SPECIFICATIONS

IBM System/360

Form X24-3347-3
Printed in U. S. A.

Date
Program
Programmer

Punching Instruction: Graphic / Punch

Page 1 2 Program Identification 75 76 77 78 79 80

Line (3-5)	Form Type (6)	Filename (7-14)	I/O/U/C (15)	P/S/C/R/T (16)	E (17)	A/D (18)	F/V (19)	Block Length (20-23)	Record Length (24-27)	L/R (28)	Mode of Processing (29-30)	K/I (31)	I/D/T (32)	Overflow Indicator (33-34)	Key Field Starting Location (35-38)	Extension Code E/L (39)	Device (40-46)	Symbolic Device (47-52)	Labels (S, N, or E) (53)	Name of Label Exit (54-59)	Extent Exit for DAM (60-65)	A (66)	File Addition (67-69)	N/U (70)	(71-74)
01	F	INVNTRY	C	P		A											MFCM1								
02	F	TRANSACTIS	I	S		A											MFCM2								
03	F																								

Figure E24A. Summary Punching Matching-Group Totals into Primary Trailer Cards--Part I

Comments on Figure E24: The input specifications re-emphasize (1) that the Matching Fields need not be in the same column in different card types and (2) that stacker selection--even to the normal stacker--should preferably be given in the input specifications for combined-file card types for which no output operation is required. (The input-only file of transaction cards is directed to the normal stacker for the secondary hopper without the need for a stacker-select specification.)

IBM — INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR INPUT SPECIFICATIONS

IBM System/360

Form X24-3350-1 Printed in U.S.A.

Line	Form Type	Filename	Sequence	Number (1-N)	Resulting Indicator	Position (1)	Not (N)	C/Z/D	Character	Position (2)	Not (N)	C/Z/D	Stacker Select	From	To	Decimal Positions	Field Name	Matching Fields or Chaining Fields
01	I	INVNTRY	Ø1	1	Ø1	7Ø		C	Ø				1					
02	I													1	2	Ø	DIVSN	M3
03	I													6	1Ø	Ø	STOCK#	M2
04	I													8Ø	8Ø	Ø	SEQFED	M1
05	I														Other			
06	I														Fields			
07	I		Ø2		Ø2	7Ø		C						1	2	Ø	DIVSN	M3
08	I													6	1Ø	Ø	STOCK#	M2
09	I													62	62	Ø	SEQFED	M1
10	I	TRANSACT	AZ		11	7Ø	N	C	Ø	7Ø	N	C						
11	I													2	3	Ø	DIVSN	M3
12	I													8	12	Ø	STOCK#	M2
13	I													79	79	Ø	SEQFED	M1
14	I														Other			
15	I														Fields			

IBM — INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR OUTPUT-FORMAT SPECIFICATIONS

IBM System/360

Form X24-3352-1 Printed in U.S.A.

Line	Form Type	Filename	Type (H/D/T)	Stacker Select	Output Indicators (Not)	Output Indicators	And (Not)	And	Field Name	End Position in Output Record
01	O	INVNTRY	T	2		Ø2		MR		
02	O	OR		3		Ø2	N	MR		
03	O					MR			ONHAND	15
04	O					MR			Other	
05	O					MR			Fields	
06	O									
07	O									

Figure E24B. Summary Punching Matching-Group Totals into Primary Trailer Cards--Part II

Calculation specifications have been omitted because they are not affected by the particular Matching-Fields approach under discussion.

Output to the trailer card must be at total time (T in col. 15): the trailer cards are always unmatched (because the M1 field was so designed); but, at total time,the MR indicator is still on if the preceding primary-file card matched the secondaries (which were then processed ahead of the "blank" trailer card). On the other hand, at total time, the indicator for the new card is already on, so that punching into the trailer card can also be conditioned by its indicator (02).

Trailer cards of matched groups are directed to stacker 2; of unmatched groups, to stacker 3.

General

Object Program Register Usage

In some applications in which the programmer specifies branching from RPG to a B.A.L. subroutine (by the EXIT operation code), it is important to know, for programming of the subroutine, what function each of the eight general registers performs--(1) so that the contents of certain registers can be preserved before other data is placed in those registers during the subroutine, and (2) to make use of the data in the registers for the subroutine.

Figure E25 itemizes the functions of the general registers.

Register	Output Routines	Calculation Routines	Input Record Routines	Input Data Routines
08	Base Register for Sterling Subroutines		Input Control	Base Register for Sterling Subroutines
09	Link Register		Input Control	Link Register for Sterling Subroutines
10	Working Register	Address of Table LOKUP Argument	Data Address	Working Register
11	Working Register	Address of Table LOKUP Function		Working Register
12				Base Register for Sign-Test Subroutine
13			File Control	
14	Link Register for Output Fields, Sterling Edit, Space/Skip Routines	Link Register	Input Parameter and Linkage for Matching Fields Sequence Control	
15	Link Register for Output Record Routine	Link Register for Sign and Test Zone Subroutines	Link Register for Input Records	Link Register for Sign-Test Subroutine

Figure E25. Object Program Register Usage

APPENDIX F. SUMMARY OF INDICATORS - SEE ALSO FIGURE 6 (RPG PROGRAM LOGIC) FOR RELATION TO POINTS IN PROGRAM CYCLE

	Indicators	Where Located	Where Normally Used as Conditioning Indicators	Normally Turned On: By	Normally Turned On: When	Normally Turned Off: By	Normally Turned Off: When
Specifications-Forms Locations	Card-Type Resulting Indicator	Input Specs.- Cols. 19-20	Input: Field-Record Relation (cols. 63-64) Calc: Indicators (cols. 9-17) Output: Output Indicators (cols. 23-31)	Record Identification	Before total-time calcs.	Different new record. (Only one such indicator on at one time)	Before total-time calcs.
	Field Indicators: Plus/Minus	Input Specs.- Cols. 65-68: Num. only	Calc: Indicators (cols. 9-17) Output: Output Indicators (cols. 23-31)	Plus/Minus (but not ±0), respectively, in field.	Before detail-time calcs.	Non-Plus / Minus data read in from field	Before detail-time calcs.
	Zero/Blank	Cols. 69-70		0/b̸ (num., incl. ±0); or b̸ (alph.) in field.	Initially; upon Blank-After; & before detail calcs. when field 0/b̸	Non-0/b̸ data read in from field	
	Control Level (L1-L9)	Input Specs Cols. 59-60	Calc: Control Level (cols. 7-8) Calc: Indicators (cols. 9-17) Output: Output Indicators (cols. 23-31)	Control break of that or higher level	Before total-time calcs.	Program itself	After detail-time output
	Matching Records (MR) -- based on Matching Fields	Input Specs. - cols. 61-62: M1, M2, M3 Control MR	Calc: Indicators (cols. 9-17) Output: Output Indicators (cols. 23-31)	Matching of primary with secondary &/or tertiary record	Before detail-time calcs.	Non-match between primary and other records	Before detail-time calcs.
	Calculation Resulting Indicators	Calc. Specs.- Columns 54-59	Calc: Indicators (cols. 9-17) Output: Output Indicators (cols. 23-31)	*Note: Zero-or-Blank indicators for arith. and TESTZ ops. are on initially and following Blank-After		Failure to satisfy the assigned condition when the specifications in the line are executed	Immediately
	Arith. Ops. Plus / Minus / Zero			Plus result / Minus result / Field contents zero*	Immediately, when the specified condition is met upon execution of the operation.		
	COMP High / Low / Equal			Factor 1 > Factor 2 / Factor 1 < Factor 2 / Factor 1 = Factor 2			
	TESTZ Plus / Minus / Blank			12 = hex. 50 or Cx / 11 = hex. 60 or Dx / other: hex ≠ 50, 60, Cx, Dx*			
	LOKUP High / Low / Equal			Factor 1 < Factor 2 / Factor 1 > Factor 2 / Factor 1 = Factor			

	Indicators	Where Normally Specified to be Turned On or Off	Turned On by Program Itself	Turned Off by Program Itself
Name/Number	L1-L9 (Control Level)	Control Level: Cols. 59-60 - Input Specs.	Before total time upon control break	After each detail output time
	L0 (Universal Total)	Nowhere	Initially & after each detail-output time	Never
	LR (Last Record Total)	Nowhere	Before total time following last data card (before/*)	After detail output time (unless LR terminated job)
	1P (First Page)	Nowhere	At beginning of program execution	After each detail-time output
	OF/OV (Overflow)	Nowhere	After outputs following printing at/past channel 12 (but not 1)	After next detail-time output
	H1/H2 (Halt)	Field and Resulting Indicators	Never - but, if on at detail-output time, halts system thereafter	When system is restarted after halt
	01-99 (General)	Field and Resulting Indicators	Never	Never

NOTE: 1 Any indicator may be turned on by a SETON specification, or turned off by a SETOF specification.
2 Any indicator except L0 may be assigned as Field or Resulting Indicator — but, for unconventional assignments, Indicator Hierarchy and RPG Program Logic should be consulted. (Indicator L0 can only be assigned in calculation Resulting Indicators.)
3 An indicator of one name or number is the same indicator no matter where or how often assigned.

This brief column-by-column description of each of the five RPG specifications forms is intended to provide a concise reference guide for the normal, common entries. Abbreviated phraseology is employed (symbol ƀ = blank). For details, and less conventional specifications, the body of the manual should be consulted.

Figures 6 (RPG Program Logic) and 36 (Fields Pertinent to Each Operation Code) are repeated at the end of this chapter, for the user's convenience. A new chart (Figure G1: Calculation Operations Summary) is also appended.

Note: The numbers that follow the underscored item name represent the specifications-card column for the item. The (R) or (O) after the card-column numbers indicate that an entry is required or optional respectively.

ALL SPECIFICATION FORMS

Page 1-2 (O)
Line 3-5 (O)

Any EBCDIC characters (see Appendix D) may be entered, and any position may be left blank. Recommended: Ascending numeric sequence, to number cards in the order in which they are to be entered into the system. The specifications types must be in the following order for program generation:

- File Description Specifications (code F in col. 6)
- File Extension Specifications (code E in col. 6)
- Input Specifications (code I in col. 6)
- Output-Format Specifications (code O in col. 6)
- Calculation Specifications (code C in col. 6)

The specifications types are summarized here, however, in the same sequence in which they appear in the body of this manual, which was thought to be the most likely order of writing an RPG program.

The first two line-number digits are preprinted for convenience, on the first 15 lines of each page; the third is available to identify additional lines to be inserted.

Form Type 6 (R)

The code appropriate to each specifications type is preprinted on the form, and must be punched. (The pertinent codes are shown above.)

Comments Card 7 (O)

An asterisk (*) identifies a comments line for which no generation takes place.

Program Identification 75-80 (O)

Any information desired to identify the specifications card.

FILE DESCRIPTION SPECIFICATIONS (REQUIRED)

File Name 7-14 (R)

Enter a name once for each file used in the program. Left-justify. One to eight characters: first alphabetic, remainder alphabetic or numeric; no special characters or embedded blanks.

File Type 15 (R)

I = Input: File name appears in input--but not output--specifications.

O = Output: File name appears in output--but not input--specifications.

C = Combined: File name appears in input and output specs.--file contains cards to be read, and cards to be punched and/or interpreted. (A file with cards to be read, and cards to be stacker-selected on any basis besides card type, must be defined C.)

File Designation 16 (C--this RPG; R--other RPGs)

P = First (primary) or sole input or combined file

S = Second or third (secondary/tertiary) input or combined file
Note: Enter P ahead of S, and secondary ahead of tertiary.

ƀ = Output file

End of File 17 (O)

E = LR turns on when last data card of all input or combined files for which E is specified has been processed.

b̸ = Turning on of LR determined by E entered for other files; or, if b̸ for all input and combined files,

= E entry for all input and combined files.

= LR turns on when last data card of all input or combined files has been processed.

Leave blank for output files.

Sequence 18 (R: multiple input or combined files)
(O: single input or combined file)

b̸ = Output file; or

b̸ = Single input or combined file which is not to be sequence-checked by Matching-fields entry.

A = Ascending } sequence of single or of all multiple
D = Descending } input and combined files.

Multiple input and combined files must all have same sequence.

Columns 19-39 Leave blank.

Device 40-46 (R)

Code--left-justified--of I/O device to be associated with file name. Do not assign same device to more than one file, or vice versa.

CRP20	=	IBM 2520 Card Read-Punch
MFCM1	=	IBM 2560 MFCM, Hopper 1
MFCM2	=	IBM 2560 MFCM, Hopper 2
PRINTER	=	IBM 1403 Printer, or IBM 2203 Printer--Standard or Lower Feed
PRINTLF	=	PRINTER (see immediately above)
PRINTUF	=	IBM 2203 Printer--Upper Feed (Dual-Feed Carriage special feature)
PUNCH20	=	IBM 2520 Card Punch
PUNCH42	=	IBM 1442 Card Punch
READ01	=	IBM 2501 Card Reader

Columns 47-65 Leave blank.

Comments 66-74 (O)

Any information that is to be printed next to the specifications at object-program generation time without affecting the program.

INPUT SPECIFICATIONS (REQUIRED)

Input Specifications contain entries only for input and combined files. At least one input or combined file is required. For multiple input or combined files, record files in order of priority: Primary, Secondary, Tertiary (if applicable).

File and Card-Type Identification 7-42 (R)

File Name 7-14 (R)

Enter name once per file--left-justified. One to eight characters: first alphabetic, remainder alphabetic or numeric, no special characters or embedded blanks.
Same name must appear as input or combined file in file-description specifications.

AND Relationship 14-16 (O)

AND = Record Identification Codes (col. 21-41) in this line must be considered with those from the preceding line to establish identification of the card type.

OR Relationship 14-15 (O)

OR = The card type defined in this line is to be in an OR relationship to that defined in the preceding line. It may or may not be assigned a separate Resulting Indicator (cols. 19-20).

Sequence 15-16 (R)

To check relative sequence of card types within a file:

Record such card types in the order in which they should appear in the file;
Assign 01 to the first such card type within a file;
Assign any two-digit numbers (higher than 01) to succeeding card types, in ascending sequence. (Leading zeros must be recorded.)

Note: Sequence does not recognize Control-Level breaks.

For card types that may appear in any relative position or sequence in the file:

Record any two alphabetic characters.

When both card types with numeric and alphabetic Sequence codes exist within a file, those with alphabetic codes must appear first.

Do not enter a Sequence code in AND or OR lines.

Number 17 (R: cols. 15-16 numeric)
(ƀ: cols. 15-16 alphabetic)

ƀ = Sequence code is alphabetic
1 = One such card per group } Sequence code is numeric
N = One or more such cards per group } Sequence code is numeric

Option 18 (O: cols. 15-16 numeric)
(ƀ: cols. 15-16 alphabetic)

ƀ = Sequence code is alphabetic; or

ƀ = Card type must be represented in each group } Sequence code is numeric
O = Card type need not be represented in each group } Sequence code is numeric

Resulting Indicator 19-20 (O)

01-99, H1, H2 = Indicator number to represent the card type defined by the Record Identification Codes (cols. 21-41). (Do not drop a leading zero.)
H1, H2 cause halt after next card is read.

ƀ = No indicator to be assigned to this card type; or
= OR line to which same indicator is to apply as in the preceding line with indicator; or
= AND line

Indicator turns on, and previous card-type Resulting Indicator turns off, before total time following reading of card. (Other indicators--besides 01-99, H1, H2--are also permitted, but require detailed understanding of time relationships and indicator hierarchy.)

Record Identification Codes 21-41 (O)

Three identical subfields: cols. 21-27, 28-34, 35-41. Subfields are in AND relationship; additonal AND subfields, and OR relationships, available through AND and OR lines (see cols. 14-16). Only the first subfield is described here. If entire area (cols. 21-41) blank, all cards tested against this line are identified as this card type. (See Nature of the Card-Type Sequence Check for order in which identification lines are tested.)

Position (cols. 21-24). Number of card column containing an identifying code. Right-justify; leading zeros unnecessary.

Not (col. 25).

N = The code (col. 27) must be absent } to satisfy criteria for this card type
ƀ = The code (col. 27) must be present } to satisfy criteria for this card type

C/Z/D (col. 26).

C = Match entire character in data-card column against entire code in specification col. 27.

D = Match digit portion of character in data-card column against digit portion of code in specification col. 27.

Z = Match zone portion of character in data-card column against zone portion of code in specification col. 27.

Character (col. 27). Identifying code (any EBCDIC) for which to test. If D or Z in col. 26--with other than A-Z, 12, 11, 0-9, +0, -0, or ƀ--see C/Z/D in body of manual.

Stacker Select 42 (C)

Number of stacker to which card type is to be directed (subject to stackers available in particular I/O unit).

ƀ = Input-file card type to normal or only stacker of I/O device; or
= Combined-file card type requiring output operation; or
= AND line

1-5 = Input-File card type to specific stacker of I/O device; or
= Combined-file card type requiring no output stacker selection, card punching, or card document-printing.

Field Descriptions 43-74 (O)

Field-description lines for a card type (or group of OR-relation card types)--one line per input field used in the program--follow immediately beneath the File and Card-Type Identification line(s) for the card type(s). The entries for card-type identification and field description must not be in the same line.

Note: R means that the entry is required when there is a Field-Description line.

Packed 43 (O)

b = Input data in standard (non-packed) format; or
 = Input data not defined as numeric.

P = Input data, from field defined as numeric (0-9 in col. 52), already in packed format--limitations:
Maximum length of field = 8 columns (= 15 digit positions, plus sign);
No Control Level (cols. 59-60) or Matching Fields (cols. 61-62) entry permitted.

Note: In subsequent operations, a packed input field must be considered of length L = 2n-1, where n = number of columns.

Field Location 44-51 (R)

From (cols. 44-47). First (leftmost) / To (cols. 48-51). Last (rightmost) } card column of input field--range: 1-80

Right-justify; leading zeros unnecessary.
Max. permissible field sizes where less than 80: Numeric field--15 digit positions (plus sign); Alphameric field used in COMP operations--40 positions.

Decimal Positions 52 (O)

b = Alphameric field; or
 = Numeric field that need not be defined as numeric (see immediately below).

0-9 = Number of decimal places in field hereby defined as numeric. Field must be defined as numeric if:

a. Used in arithmetic calculations; or

b. To be used in numeric (as contrasted with logical) COMP operation; or

c. To be formatted for output by edit word or zero suppress; or

d. To act as search argument (LOKUP operation) for an argument table defined as numeric; or

e. P (packed) is specified in col. 43.

Definition as numeric:

a. Strips all zones--except in the low-order position--for general use of the field (i.e., the program packs the field); and

b. Strips all zones for use of the field in Control-Level or Matching-Fields operations.

c. Limits the field size to 15 digits (plus sign).

Field Name 53-58 (R)

Left-justify. One to six characters (four-character limit if used with RLABL op. code):
first alphabetic, remainder alphabetic or numeric.
No special characters or embedded blanks.
ALTSEQ, CONTD(x), TAB(x)(x)(x), and PAGEx(x) prohibited--see body of manual for use of PAGEbb.

Control Level 59-60 (O)

L1-L9 = Control-Level indicator(s) for that and lower levels turn on when field contents in successive cards differ (L1 is lowest level, L9 highest.)

Multiple Control Levels for one card type must be specified in ascending sequence--lowest level used appears in uppermost line for which a Control Level is designated. The lowest level assigned need not be L1, and there may be gaps in levels used. A Control Level may be assigned to some card types and not others.

Same Control Level may be split among several fields, but no other Control Level may intervene between the portions. The sum of the number columns for all portions of a split Control Level, and for unsplit control fields, must be uniform for all card types where that level is specified.

If any field to which a Control Level is assigned is defined as numeric, all control of that level is numeric (all zones stripped).

Packed input fields cannot have Control Level assigned.

Matching Fields 61-62 (O)

A primary file can be matched to a secondary file, or to a secondary and a tertiary file; and files can be sequence checked.

M1-M3 = Successive cards of single or multiple input and combined files are sequence-checked on the field(s); also

= Multiple input or combined files are matched on the field(s)--this determines status of MR indicator, and sequence in which cards from the several files are processed, within the priorities (primary, secondary, tertiary) established by the file order in the input specifications.

M1 must be assigned to lowest-level or only Matching Field, M2 to next higher, M3 to highest.

A Matching-Fields level cannot be split between several fields in one card type.

Matching Fields may be assigned to some card types and not others; but

a. At least one card type in each of multiple input or combined files must have Matching Fields assigned; and

b. The same number of Matching-Fields levels must be specified for all applicable card types, and the length of a Matching Field of one level must be uniform.

If any field to which a Matching-Fields level is assigned is defined as numeric, all matching and/or sequence-checking for that level is numeric (all zones stripped).

Packed input fields cannot have Matching Fields assigned.

Matching Fields have no inherent relationship to Control Levels, nor need Control Levels be specified solely because Matching Fields are specified.

<u>Field-Record Relation</u> 63-64 (O)

To relate a field description to a particular one of several card types in an OR relationship, enter here the card-type Resulting Indicator assigned to the pertinent card type. Fields without indicator are associated with all card types in the OR relationship, regardless of whether they also appear with indicator--except for Control-Level and Matching-Fields operations:

If the same Control Level or Matching-Fields level appears both with and without an indicator in Field-Record Relation, the indicator-conditioned entry takes precedence when the pertinent indicator is on.

Any Control-Level or Matching-Fields entry without indicator must precede any entry with indicator for the same level. Do not condition portions of one split Control Level differently. Do not vary the <u>number</u> of Matching Fields levels applicable to different OR card types beyond the choice of whether or not Matching Fields apply.

It is advantageous to group field-description lines by indicator, preceded by those without indicator.

<u>Field Indicators</u> 65-70 (C)

Enter any indicator code (normally: 01-99, H1, H2) to cause that indicator to turn on if the field contents conform to the assignment of the indicator; any indicators assigned to the other two conditions turn off--the settings occur before detail-calculation time. (Do not drop a leading zero.) The same indicator assigned to more than one condition turns on if one of these is satisfied.

If H1 or H2 is turned on, system halts after next card is read.

<u>Plus (cols. 65-66)</u>. Tests for value in <u>numeric</u> field greater than zero (low-order position 12- or no overpunch--but excluding all-zero value signed plus).

<u>Minus (cols. 67-68)</u>. Tests for value in <u>numeric</u> field less than zero (low-order position 11-overpunch--but excluding all-zero value signed minus).

<u>Zero or Blank (cols. 69-70)</u>. Alphameric field: tests for blank. Numeric field: tests for absence of significant digits (1-9); i.e., turns on if blank, zero, ±0, or zones alone. Indicator also turned on at beginning of program execution and following each Blank-After output specification--until changed by data read from card of pertinent type.

<u>Sterling Sign Position</u> 71-74 (O)

Leave blank unless processing Sterling-currency amounts. (Refer to SRL publication <u>IBM System/360 Model 20, Sterling Currency Processing Routines</u>, Form C26-3605.)

CALCULATION SPECIFICATIONS (OPTIONAL)

See Figures G1 and G2 at end of chapter for supplementary details.

Note: Operations occur in the order specified, within grouping of detail time or total time (see cols. 7-8).

Control Level 7-8 (O)

ƀ = Detail-time operation
L1-L9, LR, L0 = Total-time operation, and performed only if the specified indicator is then on. L0 normally always on (exceptions: see text). L1-L9 on when control break of that or higher level. LR on after processing last input card (also turns on L1-L9).

Note: All detail-time calculation specifications must precede those for total time.

Indicators 9-17 (O)

Three identical subfields in AND relationship: cols. 9-11, 12-14, 15-17. One to three indicators may be entered to condition performance of the operation. Status of the indicators not affected by specification here.

Entire field blank = Execute operation each program cycle

ƀxx (xx = any indicator) = Execute operation only if that indicator is on.

Nxx = Execute operation only if that indicator is off.

Note: An indicator in cols. 7-8 is in AND relationship to indicators in cols. 9-17.

Factor 1 18-27
Factor 2 33-42

Field names or literals used in calculation Enter left-justified.

Field Name

Must be defined as Result Field in calculation specifications or--if it is an input field--in input specifications.

Literal

Numeric. Maximum length: ten characters. May consist only of digits (0-9) and--optionally:
One decimal point (or comma, if European notation specified in RPG Control Card--Card H, col. 21), and/or one leading sign (+ or -).

Number assumed positive if no sign, and integer if no decimal point.

Note: + is punch combination 12-6-8

Alphameric. Maximum lenght : eight EBCDIC characters, plus enclosing apostrophes.

Factor 1 used with operation codes ADD, SUB, MULT, DIV, COMP, TAG, LOKUP.

Factor 2 used with all operations except MVR, TESTZ, SETON, SETOF, TAG, RLABL. (Field name limited to four characters if used with EXIT operation.) Also see Figures G1 and G2 at end of this chapter.

Operation 28-32 (R)

Entry of an RPG operation code, left-justified, required (except in a Comments line--asterisk in col. 7).

Decimal alignment automatic in arithmetic operations and numeric COMP. Sign control automatic in arithmetic operations. Results of arithmetic operations always signed; zero result always +0. See Figures G1 and G2 at end of chapter for operations summary.

Operations with Reduced Field-Size Limits

COMP (alphameric): 40 positions
LOKUP (alphameric): 80 positions
Arithmetic ops. require numeric fields; TESTZ requires alphameric field.

Result Field 43-48

Name (left-justified) representing location where result of operation is to be stored. One to six characters (four-character limit if used with RLABL op. code); first alphabetic, remainder alphabetic or numeric. No special characters or embedded blanks. PAGE has special use; INxx restricted in RLABL lines; TAB(x)(x)(x) confined to function table.

See Figures G1 and G2 at end of chapter for operations requiring Result-Field entry.

Defining Result Field 43-48, 49-51, 52

Every Result Field must be defined once, either in a calculation specification or--if it is an input or table field--in the input or file extension specifications, respectively. If defined more than once (unnecessary), all definitions (length, decimals) must be uniform.

A field is defined by assigning field name (cols. 43-48), field length (cols. 49-51), and--if numeric--decimal positions (col. 52). Result Field is used with arithmetic, move, TESTZ, and RLABL operations (i.e., all except COMP, SETON, SETOF, GOTO, TAG, EXIT); used with LOKUP only if function table.

Field Length 49-51 (O)

ƀ = No Result Field with this operation (cols. 43-48 blank); or Result Field not defined here.
1-15 = Numeric field (if too short for result, high-order digits lost).
1-40 = Alphameric field used in COMP op.
1-80 = Alphameric field used in table LOKUP op.
1-256= Alphameric field, not used in COMP or LOKUP op.
Leading zeros unnecessary.

Decimal Positions 52 (O)

ƀ = Alphameric field (or numeric field that need not be defined as such); or
= No Result Field with this operation (cols. 43-51 blank); or
= Result Field not defined here.
0-9 = Number of decimal places in field (hereby) defined as numeric (total field length in cols. 49-51).
Field must be defined as numeric if
(a) Used in arithmetic ops.; or
(b) Used in numeric (as contrasted with logical) COMP op.; or
(c) To be formatted for output by edit word or zero suppress; or
(d) To act as search argument (LOKUP op.) for argument table defined as numeric; or
(e) Output to be in packed format.

Half Adjust 53 (O)

H = Half-adjust result of arithmetic operation before dropping excess decimal position(s).

Resulting Indicators 54-59

Any indicators may be specified--normally: 01-99, H1, H2(if H1 or H2 is on at end of detail-calculation time, system halts after next card is read). Changes in indicator status are effective as soon as operation has been performed. Up to three indicators may be assigned in operations that permit any Resulting Indicators (except LOKUP). The same indicator can be specified in several lines.

Arithmetic Operations (O)

All fields must be numeric. Enter indicator code (if desired) to cause that indicator to turn on if the result conforms to the assignment of the indicator; any indicators assigned to the other two conditions turn off. The same indicator assigned to more than one condition turns on if one of these is satisfied.

Plus (cols. 54-55). Result greater than zero (excludes+0).
Minus (cols.56-57). Result less than zero.
Zero (cols. 58-59). Result zero (always +0). Also on initially, and following Blank-After (output specs.)

Compare (COMP) Operation (R--at least one)

An indicator whose assignment reflects the operation result turns on; others turn off. One indicator assigned to more than one condition turns on if any of these is satisfied.

High (cols. 54-55). Factor 1 > Factor 2
Low (cols. 56-57). Factor 1 < Factor 2
Equal (cols. 58-59). Factor 1 = Factor 2

Comparison algebraic for numeric fields, logical (EBCDIC sequence) for alphameric fields.

Test Zone (TESTZ) (R--at least one)

Alphameric field only. High-order position of field is tested. An indicator whose assignment reflects the operation result turns on; others turn off. One indicator assigned to more than one condition turns on if any of these is satisfied.

High (cols. 54-55). 12-zone (hex.50 or Cx)
Low (cols. 56-57). 11-zone (hex.60 or Dx)
Blank (cols. 58-59). No zone (not hex. 50, 60, Cx, Dx). Also on initially, and following Blank-After (Output specs.).

Set Indicators (SETON, SETCF) (R--at least one)

The indicators specified are turned on or off, respectively.

Table Lock-Up (LOKUP) (R--one or two)

One indicator may be assigned to one of the three conditions; or one indicator, or two different indicators, may be assigned to Equal and High, or to Equal and Low. This causes search of the argument table (Factor 2) for an entry that bears the designated relationship to the search argument (Factor 1).

If indicators are assigned to two conditions, an Equal match takes precedence. If a table entry meets a specified condition, the corresponding indicator turns on; a different indicator assigned to another condition, turns off--if it is the same indicator, it will be on. If the condition(s) cannot be satisfied, the indicator(s) will be off.

	Argument Table		Search Argument
High (cols. 54-55).	Factor 2	>	Factor 1
Low (cols. 56-57).	Factor 2	<	Factor 1
Equal (cols. 58-59).	Factor 2	=	Factor 1

Note that High and Low significance is the reverse of form-column heading.

Other Operation Codes

No Resulting Indicators permitted.

Comments 60-74 (O)

Permits any comments to be printed at generation time next to specifications in the same line, without affecting program generation.

FILE EXTENSION SPECIFICATIONS (OPTIONAL)

Required if table lock-up (LOKUP) used in Calculation Specifications.

Columns 7-26

Leave blank.

First or Only Table in a Table-Input Deck 27-45 (R)

Table Name 27-32 (R)

Left-justify. Four to six characters (four-character limit if used with RLABL op. code):
first three TAB, remainder alphabetic or numeric.

No special characters or embedded blanks.

If alternating-table input, this is name of table to which first entry in each card belongs.

Number of Table Entries per Record 33-35 (R)

Number of entries in one card of this table. Right-justify; leading zeros unnecessary.

Number of Table Entries per Table 36-39 (R)

Exact total number of entries in this table. Right-justify; leading zeros unnecessary.

Length of Table Entry 40-42 (R)

Number of columns per entry for this table (named in cols. 27-32). Right-justify; leading zeros unnecessary.

Maxima: Alphameric--80 columns
Numeric--15 columns

Packed 43

Leave blank: Packed-data table input not permitted

Decimal Positions 44 (O)

ƀ = Alphameric
0-9 = Number of decimal places in numeric-table field.
N = 0

Sequence 45 (O)

ƀ = Table search only for Equal match
A = Table values in ascending sequence } High or Low search specified
D = Table values in descending sequence }

Second Table, Alternating-Table Formats 46-57 (O)

Table Name 46-51 (R)

Name of table to which second entry in each card belongs. Entries equivalent to those described under col. 27-32.

Length of Table Entry 52-54 (R)

Equivalent to cols. 40-42, but for second table.

Packed 55

Leave blank.

Decimal Positions 56 (O)

Equivalent to col. 44

Sequence 57 (O)

Equivalent to col. 45

Comments 58-74 (O)

Permits any comments to be printed at generation time next to specifications in the same line, without affecting program generation.

OUTPUT-FORMAT SPECIFICATIONS (OPTIONAL)

Output specifications contain entries only for output and combined files. All detail- (or heading-) time output specified ahead of total-time output. Output (within grouping of detail and total time) occurs in the order specified--except (1) separate overflow-output time and (2) card-print transfer to output-storage area after card-punch transfer for same File-Identification group.

(Preferably, alternate punching and forms-printing when more than one of either during same cycle segment.)

File Identification and Control 7-31 (R)

File Name 7-14 (R)

Enter once for each output operation to the file (but card punching and card-printing to the same card specified under a single file-identification entry).

Exception: When output to same file specified repeatedly, file name need not be repeated if no other file name intervenes.

Left-justify. One to eight characters: first alphabetic, remainder alphabetic or numeric; no special character or embedded blanks. Same name must appear as output or combined file in file-description specifications.

Type 15 (R)

D (or H) = Detail-time output
T = Total-time output.

During overflow output, T output precedes D.

Stacker Select 16 (O)

ƀ = Normal stacker; or
= Single-stacker device; or
= Card type (combined file) stacker-selected in input specifications

1-5 = Output- or combined-file card to specific stacker of I/O device (combined file not stacker-selected in input specifications).

AND Relationship 14-16 (O)

AND = Output Indicators (cols. 23-31) in this line must be considered with those from the preceding line to establish the conditions under which the output is performed.

OR Relationship 14-15 (O)

OR = The output conditions defined in this line are in an OR relationship to those defined in the preceding line. If either set of conditions (Output Indicators, cols. 23-31) is satisfied, the output is performed at the appropriate time.

Forms control from the preceding line is applied unless cols. 17-22 contain an entry. (0 is considered an entry.)

Space (Before/After) 17-18 (O)

0-3 = Advance form the specified number of spaces before and/or after printing, respectively.
ƀ = 0 (see exception for OR lines, above)

Skip (Before/After) 19-20, 21-22 (O)

01-12 = Advance the form to the next carriage-control-tape punch in the specified channel before and/or after printing, respectively.
00=ƀƀ = No skip (see exception for OR lines, above)

NOTES on Forms Control

1. Leave blank in AND line
2. Space/After or Skip/After has throughput advantage over Space/Before or Skip/Before.
3. If Space/Before and Skip/Before both specified: Skip is executed, followed by Space.

4. If Space/After and Skip/After both specified: Only the Space/After operation is executed.
5. For compatibility with other RPGs: One entry in cols. 17-22 required.
6. Printing at or below channel 12, but above next channel 1, turns on OF indicator at end of cyle segment (detail or total output respectively)--or OV indicator if upper feed of Dual-Feed Carriage. Advance to channel 1 automatic if OF (or OV) not specified in any file-identification Output Indicators.

<u>Output Indicators</u> 23-31 (O)

Three identical subfields in AND relationship: cols. 23-25, 26-28, 29-31. Only the first subfield is described here. One to three indicators may be entered to condition performance of the output operation; additonal AND subfields, and OR relationships, available through AND and OR lines (see cols. 14-16). Status of the indicators not affected by specification here.

Entire field blank	= Execute operation each program cycle.
ƀxx (xx = any indicator)	= Perform this output only if that indicator is on.
Nxx	= Perform this output only if that indicator is off.
ƀOF (or ƀOV)	= Perform this output only at overflow time; and provided the OF (or OV) indicator is then on, as well as any other Output Indicators specified for this (or an AND) line.

CAUTION: If no conditioning indicator is specified, or only 1P, or only Nxx for indicators that are off initially, or ƀxx for indicators that are on initially (Zero-or-Blank), the output is performed before the first card has been read.

<u>Field Description and Control</u> 23-47 (O)

One line describes one output field within the file-identification group --separate line required for card punching and card document-printing. Field-description lines follow immediately beneath the pertinent file identification--field description must not be on same line as file identification.

<u>Output Indicators</u> 23-31 (O)

As described under Output Indicators, above (File Identification), with these differences:

1. No AND lines
2. No OR lines
3. Entries condition only output of the field or constant described in that line, and the output is also subject to Output Indicators in the file identification.
4. Entry of OF (or OV) does not transfer the output to overflow time--merely makes it subject to the status of the overflow indicator at output time.
5. Used with field PAGE, do not condition its output, but cause initialization to 1 before output.

<u>Field Name</u> 32-37 (O)

Any field name defined in Input, File-Extension, or Calculation Specifications. Field name PAGE for automatic consecutive numbering. Left-justify. Fields need not be listed in the order in which they are to appear in the output file. Prohibited names: ALTSEQ, CONTD(x), PAGEx(x). Leave blank if constant specified (cols. 45-70). Either field name or constant required.

<u>Zero Suppress</u> 38 (O)

ƀ = Alphameric field; or
= Constant specified (cols. 32-37 blank); or
= Edit word specified (cols. 45-70); or
= Packed Field (P in col. 44); or
= Output to include any leading zeros and low-order-position zone.
Z = Any zone or leading zeros removed from output.
Permissible only for numeric fields.

<u>Blank After</u> 39 (O)

ƀ = Field contents (or constant--cols. 45-70) undisturbed after output
B = Field (or constant) cleared as data moved to output-storage area. (Numeric field: 0; alphameric: ƀ.) Field or Resulting Indicator assigned to Zero-or-Blanks turns on.

End Position in Output Record 40-43 (R)

Enter number of rightmost position in output file (forms print position, card column, or card-interpret position). Right-justify entry; leading zeros unnecessary (col. 40 always ƀ or 0).

CAUTION: Allow for expansion of field by edit word.

Output to card file. Col. 41 establishes whether punching or interpreting:

ƀ = 0 = Punching

1-6 = Print head to be used for interpreting.

Packed 44 (O)

ƀ = Data to be output in standard (non-packed) format; or
= Field not defined as numeric; or
= Constant

P = Output of data, from field defined as numeric, to be in packed format. The output data is then of length
L = (n+1+E)/2, where
n = digit capacity of field; and
E = 0, if n is odd, and
= 1, if n is even

Constant 45-70 (O)

Left-justify. Any EBCDIC characters that are to appear in the output record are specified here, enclosed in apostrophes. (Two successive apostrophes within the constant appear as one apostrophe in the output.) Distinguished from edit word by absence of field name (i.e., cols. 32-37 are blank). Zero Suppress (col. 38) and Packed Field (col.44) must be blank.

Edit Word 45-70 (O)

Left-justify. Any EBCDIC characters, enclosed in apostrophes. Some of the characters appear as such in the output record; others serve special functions. Distinguished from constant by presence of field name (cols. 32-37). The field named in the line must be defined as numeric, and must contain valid digits (no hex. A-F, except in sign position). Zero Suppress (col. 38) and Packed Field (col. 44) must be blank.

Abbreviated rules for edit words.

1. Body of edit word = Exact number of positions allowed for data digits, spaces, and any constant data desired among--but not following--them, plus 1 if $ symbol. Blank is replaced by digit--with possible exception of leading 0--from corresponding field position.
   0 = Leading zeros, and any constant edit-word characters preceding first significant digit, suppressed through this point. At least one leading zero is suppressed if an edit word is specified, all 0's and constant characters (exc. $) suppressed if field all-zero and no 0 or * in edit-word body.
   & = Space in output record
   $ at extreme left = $ in that output record position, regardless of handling of leading zeros.
   $0 = Floating $ symbol = $ replaces rightmost leading 0 through this (0-entry) point.
   * = Asterisk protection = * replaces leading zeros through this point, and any constant edit-word characters to the first significant digit. Precludes $.

   Any EBCDIC character (except ƀ or &) in the edit-word body--including comma and decimal point--to right of first significant digit or end of zero suppression, appears identically in the corresponding location of the output record; to the left of that point, it is suppressed.

   Presence of edit word removes low-order-position zone from output.

2. Status portion of edit word = Positions to right of body through CR or - symbol, used to identify negative values. If no CR or - to right of body, there is no status portion.
   ƀ or & = ƀ in output record
   CR or - = CR or -, respectively, in output record when data negative
   = ƀ in output record, when data not negative.

3. Expansion of edit word = Positions (if any) to right of status portion; if no status portion, then to right of body. Any EBCDIC character (incl. & and space) appears identically in output record.

Sterling Sign Position 71-74 (O)

Leave blank unless processing Sterling-currency amounts. (Refer to SRL publication IBM System/360 Model 20, Sterling Currency Processing Routines, Form C26-3605.)

Op. Class	Nature of Operation	Factor 1	Operation Code	Factor 2	Result Field	Comments: Possibilities Presented Algebraically	Comments
Arithmetic Operations: Decimal alignment automatic	Add Factor 2 to Factor 1	a (N)	ADD	b (N)	c (N)	$a+b=c/a+a=2a/b+b=2b/a+c=c'/c+b=c'/c+c=2c$	
	Set Result Field to zero; then add Factor 2		Z-ADD	b (N)	c (N)	$0+b=c'/0+c=c$	Factor 2 may be c
	Subtract Factor 2 from Factor 1	a (N)	SUB	b (N)	c (N)	$a-b=c/a-a=0/b-b=0/a-c=c'/c-b=c'/c-c=0$	c–c=0 is the most efficient method to set a field to zero
	Set Result Field to zero; then subtract Factor 2		Z-SUB	b (N)	c (N)	$0-b=c'/0-c=-c$	0–c=–c: best method to reverse sign
	Multiply Factor 1 by Factor 2	a (N)	MULT	b (N)	c (N)	$a\times b=c/a\times a=a^2/b\times b=b^2/a\times c=c'/c\times b=c'/c\times c=c^2$	
	Divide Factor 1 by Factor 2	a (N)	DIV	b (N)	c (N)	$a\div b=c/a\div a=1/b\div b=1/a\div c=c'/c\div b=c'/c\div c=1$	$b\neq 0$ No Half-Adjust if MVR follows
	Move Remainder of preceding DIV to a Result Field		MVR		c (N)	Must follow immediately after DIV without Half-Adjust and with same Indicators.	
Move Operations: Resulting Indicators not allowed	Move Factor 2 into Result Field – right-justified		MOVE	b (A/N)	c (A/N)	No decimal alignment performed. If Factor 2 shorter: Left positions of result field unchanged. Excess left positions of a longer Factor 2: Not moved.	
	Move Factor 2 into Result Field – left-justified		MOVEL	b (A/N)	c (A/N)	No decimal alignment performed. If Factor 2 shorter: Right positions of result field unchanged. If Factor 2 longer: Excess right positions not moved, except for the sign if the result field is numeric.	
	Move Zone: from low-order position of Factor 2 to low-order Result position		MLLZO	b (A/N)	c (A/N)		
	Move Zone: from high-order position of Factor 2 to high-order Result position		MHHZO	b (A)	c (A)		
	Move Zone: from low-order position of Factor 2 to high-order Result position		MLHZO	b (A/N)	c (A)		
	Move Zone: from high-order position of Factor 2 to low-order Result position		MHLZO	b (A)	c (A/N)		
COMP & TESTZ: At least one Resulting Indicator	Compare Factor 1 to Factor 2	a (A-N)	COMP	b (A-N)		Numeric fields decimal-aligned; missing positions treated as zeros. Alphameric fields left-aligned; missing positions treated as blank. Numeric compare is algebraic; alphameric is logical EBCDIC.	
	Identify zone in high-order position of alpha. Result Field		TESTZ		c (A)	A Resulting Indicator assigned to Blank is on at start & following Blank-After	
Setting Indicators	Turn on one, two, or three specific indicators		SETON			Specify one, two, or three Resulting Indicators	
	Turn off one, two,or three specific indicators		SETOF				
Branching: No Resulting Indicators allowed	Branch to another RPG calculation specifications line		GOTO	name		One to six characters	First character alphabetic; remainder alphabetic or numeric. RLABL may also be TABx or INxx.
	Identify line as destination point of GOTO instruction(s)	name	TAG			One to six characters	
	Branch to an external B.A.L. routine		EXIT	name		One to four characters	
	Identify RPG field for use in external routine		RLABL		name	One to four characters	
Table Look-up	Search argument table for value that bears to search argument the relationship specified by Resulting Indicator(s). If a function table is specified,corresponding function is located.	argument a (A-N)	LOKUP	argum't table TABx(x)(x) (A-N)	function table (optional) TABx(x)(x) (A/N)	No decimal alignment performed. At least one Resulting Indicator needed. Do not assign Resulting Indicator to both High and Low. Table should be in sequence if Indicator in High or Low.	

LEGEND: (A) = Alphameric; (N) = Numeric; (A/N) = Alphameric or Numeric; (A-N) = Alphameric or Numeric, but both fields must be alike; a or b = Field or literal; c = Field

Figure G1. Calculation Operations Summary

IBM

INTERNATIONAL BUSINESS MACHINES CORPORATION

REPORT PROGRAM GENERATOR CALCULATION SPECIFICATIONS

IBM System/360

Form X24-3351-1
Printed in U. S. A.

Date ______________

Program ______________

Programmer ______________

Punching Instruction	Graphic								
	Punch								

Page 1 2 Program Identification 75 76 77 78 79 80

Line (3-5)	Form Type (6)	Control Level (L0-L9, LR) (7-8)	Indicators: Not, And, Not, And, Not (9-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field (43-48)	Field Length * (49-51)	Decimal Positions * (52)	Half Adjust (H) (53)	Resulting Indicators: Plus / Compare High 1>2 (54-55)	Minus / Low 1<2 (56-57)	Zero or Blank / Equal 1=2 (58-59)	Comments (60-74)
01	C			(N)	ADD	(N)	(N)							
02	C				Z-ADD	(N)	(N)							
03	C			(N)	SUB	(N)	(N)							
04	C				Z-SUB	(N)	(N)							
05	C			(N)	MULT	(N)	(N)							
06	C			(N)	DIV	(N)	(N)							
07	C				MVR		(N)							
08	C				MOVE	(A/N)	(A/N)							
09	C				MOVEL	(A/N)	(A/N)							
10	C				MLLZO	(A/N)	(A/N)							
11	C				MHHZO	(A)	(A)							
12	C				MLHZO	(A/N)	(A)							
13	C				MHLZO	(A)	(A/N)							
14	C			(A-N)	COMP	(A-N)								
15	C				TESTZ		(A)							
	C				SETON									
	C				SETOF									
	C				GOTO	(NAME)								
	C			(NAME)	TAG									
	C				EXIT	(NAME)								
	C				RLABL		(NAME)							
	C			(A-N)	LOKUP	TAB(A-N)	TAB(A/N)							

Resulting Indicators not related to column headings; at least one required

Legend: ——— Entry Required, - - - - Entry Optional (Max. Length of Entry)

(A) = Alphameric
(N) = Numeric
(A/N) = A or N
(A-N) = Either, but both same

Resulting Indicators: Plus, Minus, Zero/Blank — At least one required; - - - Optional

Plus/High, Minus/Low, Blank/Equal

* NOTE: The entries designated as required in Field Length and Decimal Positions are necessary only if the associated Result Field is not defined elsewhere.

Figure G2. Fields Pertinent to Each Operation Code

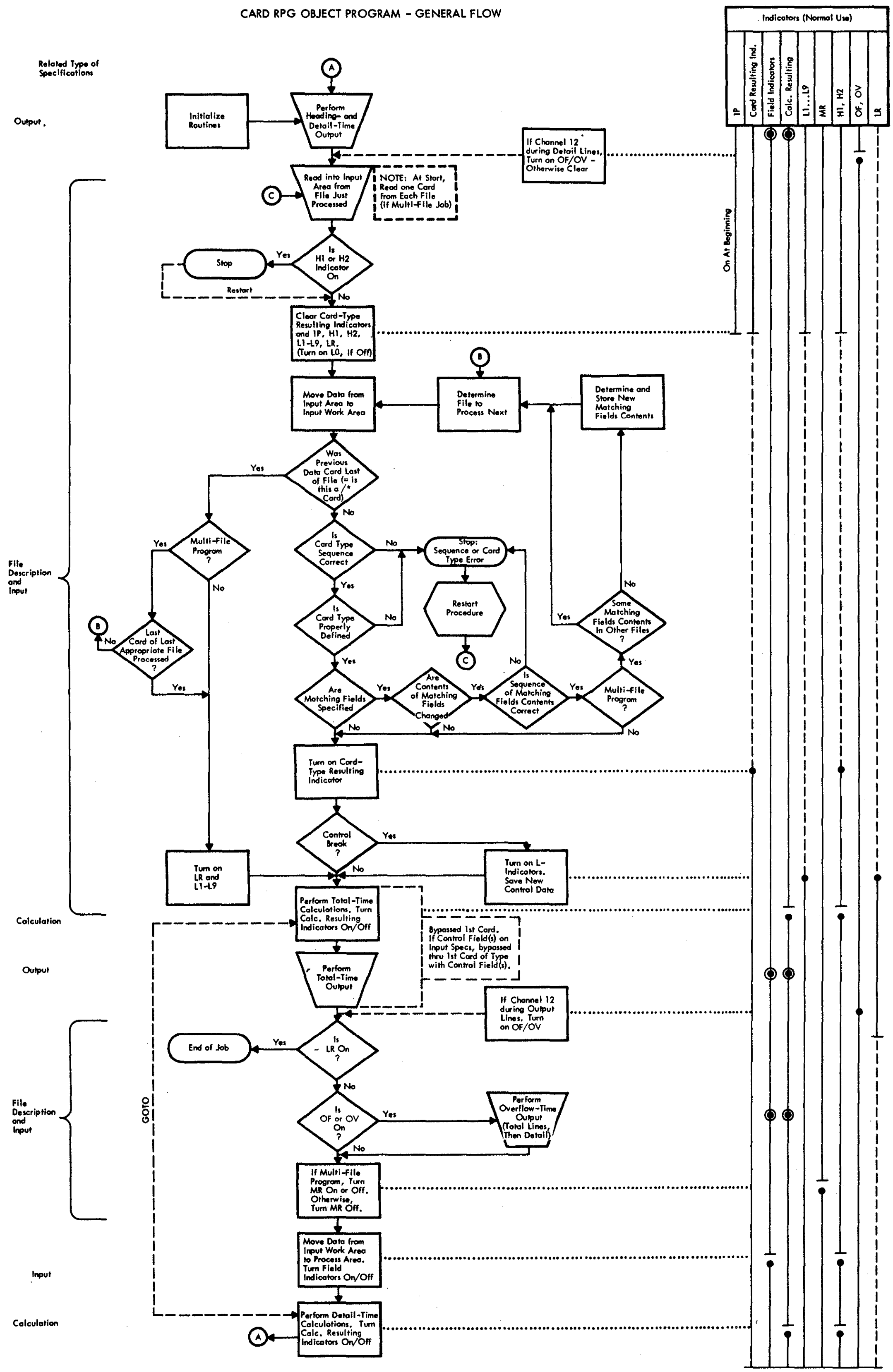


Figure G3. RPG Program Logic

During generation of an object program, RPG prints a listing that contains:

- The complete image of every specification card in the RPG source-program deck, one card to a line.
- A consecutive number, as counted by RPG during reading of the source deck--printed to the immediate left of each source-card image.
- A signal--the letter "S"--to the left of the consecutive number, whenever the contents of columns 1-5 (page and line sequence number) of a specification card represent a repetition or step-down in EBCDIC sequence from the preceding card. This does not interfere with continuation of program generation. ("S" is not printed the first calculation specifications line.)
- Messages that reflect the status of object-program generation, signal all kinds of errors, and document core-storage assignments.

A specimen of an RPG program listing is included in the SRL publication IBM System/360 Model 20, Report Program Generator for Punch-Card Equipment, Operating Procedures, Form C26-3800.

MESSAGES DURING RPG GENERATION OF OBJECT PROGRAM

Message Identification

Each message is preceded by a unique Message-Identification Code. From left to right, the code represents:

Program Identification = RG
Message Serial Number--three digits
Significance Code--one letter
Type of specification card to which the message primarily applies--one letter (not pertinent to all messages).

Figure H1 portrays the message-identification format, and lists the related codes with their meanings.

Card Number

The words CARD NUMBER are printed at the end of most messages, followed by the consecutive number(s) of the specification card(s) to which the message refers. (A consecutive card number is assigned by RPG to each source card. In the program listing, it precedes the card sequence number assigned on the programmer's specification form.)

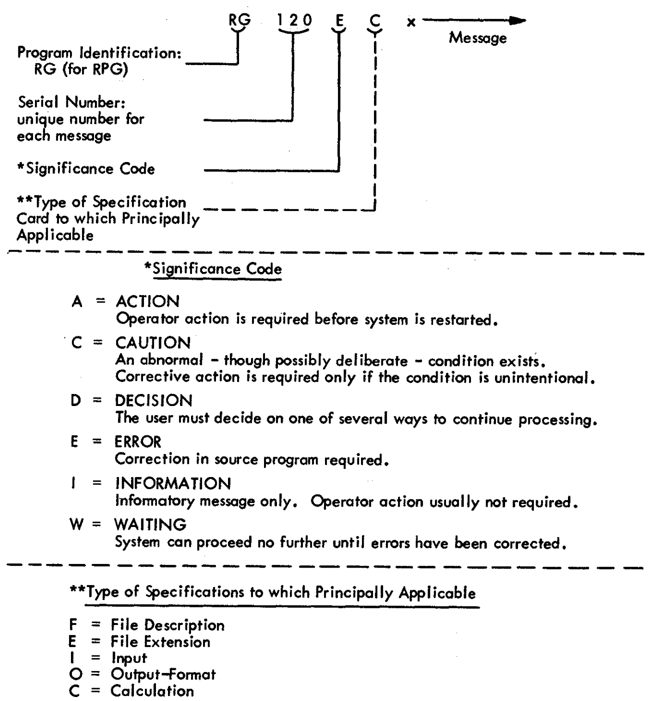


Figure H1. Format of Message-Identification Code

Messages

The informational and diagnostic messages are listed below, in message-serial-number order. Each message is preceded by its Identification code.

The phrases shown in upper-case characters represent the actual RPG message.
Sentences in lower-case letters give supplementary information that is not printed in the program listing.

RG001 I 360/20 RPG LISTING.
RPG prints this heading if the RPG source deck starts with the RPG Control Card.

RG002 I STORAGE POSITICNS.
This message is fcllowed by the core storage capacity specified in the RPG Ccntrol Card (Card H) for the systems used to generate or execute the program. If different capacities are specified, the smaller is used.

RG003 A ERROR IN STORAGE MAGNITUDE.
This message is associated with halt D12. It indicates invalid entries in cols. 7-9 or 12-14 of the RPG Ccntrol Card (Card H).

RG004 A NO CCNTROL CARD.
This message is associated with halt D11. It indicates that the source deck is not preceded by an RPG Ccntrol Card (Card H.)

RG005 C COL. 6 INCORRECT, CARD IS BYPASSED.
This message fcllows the image of the RPG source card to which it refers.

RG006 W PROGRAM TOC BIG.
This message is associated with halt D13

RG010 C F THE FILE NAME IS NOT REFERENCED.

RG011 E F FILE DESCRIPTICN SPECIFICATIONS ARE MISSING.
File descripticn cards must precede all other specification cards in the RPG source deck.

RG012 E F THE FILE NAME IS INCORRECTLY SPECIFIED OR IS MISSING.

RG013 E F FILE TYPE ENTRY (COL. 15) IS NOT I, C, OR C, OR IS MISSING.

RG014 E F FILE TYPE (COL. 15) AND DEVICE (COLS. 40-46) ARE INCCMPATIBLE.

RG015 E F SEQUENCE (COL. 18) DOES NOT CCNTAIN A, D, OR BLANK.

RG016 E F SEQUENCE (CCL. 18) IS NOT THE SAME FOR ALL INPUT FILES.
When there are multiple input or ccmbined files, all must have the same sequence specified.

RG017 E F COLUMNS 40-46 CONTAIN AN INVALID DEVICE.

RG018 E F FILE NAME OR DEVICE IS MULTIPLY DEFINED.
The same file name (cols. 7-14) is assigned to more than one device code (cols. 40-46), or the same device code is associated with more than one file name.

RG031 E E THE TABLE NAME IS INCORRECTLY SPECIFIED OR IS MISSING.
The table name in columns 27-32 and/or columns 46-51
a) is missing, or
b) includes special characters or embedded blanks, or
c) does not begin with TAB

RG032 E E THE TABLE NAME HAS NOT BEEN REFERENCED.
The table name defined in the file extension specifications is not referenced in the calculation specifications.

RG033 E E INCORRECT SPECIFICATION OF NUMBER CR LENGTH OF TABLE ENTRIES, OR THE PRODUCT OF BOTH IS LARGER THAN 80.

RG034 E E THE SAME TABLE IS DEFINED IN FILE EXTENSICN AND CALCULATION SPECIFICATIONS WITH DIFFERENT FIELD LENGTHS AND/OR DECIMAL POSITICNS

RG041 E I INPUT SPECIFICATIONS ARE MISSING.
Input specification cards must precede output specification cards.

RG042 E I MATCHING FIELD SPECIFIED, BUT NO SEQUENCE IN FILE DESCRIPTION.

RG043 E I THE FILE NAME IS MISSING, NOT DEFINED, CR NOT LEFT-JUSTIFIED.

RG044 E I THE FILE NAME DOES NOT REFER TO AN INPUT CR COMBINED FILE.

RG045 E I CARD-TYPE SEQUENCE (COLS. 15-16) BLANK CR INVALID.
Under cne common file name, a card type with numeric sequence specification is followed by a card type

with alphabetic sequence specification, or the numeric sequence specification for card types is not in ascending order, or the field is blank.

RG046 E I ENTRIES IN NUMBER AND/OR OPTION (COLS. 17-18) ARE INCORRECT.
Card-type sequence checking is specified in columns 15-16, but columns 17-18 contain an incorrect entry; or, the entries in cols. 15-16 are alphabetic, but cols. 17-18 are not blank.

RG047 E I ERROR IN RECORD IDENTIFICATION CODES (COLS. 21-26, 28-33, 35-40).

RG048 E I RECORD IDENTIFICATION AND FIELD DESCRIPTION APPEAR IN THE SAME SPECIFICATION CARD.
An input specification card contains entries in columns 7-42 and columns 43-70.

RG049 E I A RECORD IDENTIFICATION SHOULD PRECEDE THIS SPECIFICATION.

RG051 E I FIELD LOCATION ENTRIES (COLS. 44-51) ARE INVALID, OR THE LENGTH OF A NUMERIC FIELD EXCEEDS 15 POSITIONS.

RG052 E I THE FIELD WAS PREVIOUSLY DEFINED WITH DIFFERENT LENGTH OR DECIMAL POSITIONS.

RG053 E I THE FIELD NAME IS MISSING, NOT LEFT-JUSTIFIED, BEGINS WITH TAB OR CONTD, CONSISTS OF ALTSEQ, OR CONSISTS OF PAGE FOLLOWED BY ONE OR TWO CHARACTERS.

RG054 E I PLUS OR MINUS INDICATORS ARE SPECIFIED FOR ALPHAMERIC FIELD.

RG055 E I THE MATCHING FIELD SPECIFICATION (COLS. 61-62) IS INVALID.

RG056 E I INDICATOR SPECIFIED IN FIELD-RECORD RELATION (COLS. 63-64) IS NOT A RESULTING INDICATOR OF AN ASSOCIATED CARD TYPE.

RG057 E I STERLING FIELD SPECIFICATION IS INCORRECT.
The entries are invalid; or the Sterling field (cols. 71-74) contains specifications, but the RPG Control Card is blank in cols. 17 and 18.

RG058 E I STERLING FIELD IS SPECIFIED WITH INCORRECT DECIMAL LENGTH.
Either the field is defined as alphameric, or Decimal Positions is greater than 4.

RG059 E I STERLING FIELD IS SPECIFIED TO BE IN PACKED FORMAT.

RG060 E I INCORRECT INDICATORS.
Resulting and/or Field Indicators are invalid.

RG061 E I AN ALPHAMERIC FIELD IS SPECIFIED TO BE IN PACKED FORMAT.

RG071 E O THE FILE NAME IS MISSING, NOT DEFINED, OR NOT LEFT-JUSTIFIED.

RG072 E O THE FILE NAME DOES NOT REFER TO AN OUTPUT OR COMBINED FILE.

RG073 E O FILE IDENTIFICATION AND FIELD DESCRIPTION APPEAR IN THE SAME SPECIFICATION CARD.

RG074 E O TYPE (COL. 15) IS NOT H, D, OR T.

RG075 E O FILE IDENTIFICATION AND FIELD DESCRIPTION SPECIFICATIONS ARE NOT IN CORRECT SEQUENCE RELATIONSHIP, OR ALL DETAIL OUTPUT DOES NOT PRECEDE ALL TOTAL.

RG076 E O INCORRECT INDICATORS.

RG077 E O INVALID FORMS CONTROL SPECIFICATIONS (COLS. 17-22).

RG078 E O MULTIPLE FILE-IDENTIFICATION LINES FOR ONE OUTPUT DO NOT HAVE INDICATORS SPECIFIED. CHECK THIS AND PRECEDING CARD.
There are AND and/or OR lines for the output, but not all the File-Identification Specifications lines include Output Indicators.

RG081 E O THE FIELD NAME IS MISSING, IS NOT LEFT-JUSTIFIED, BEGINS WITH CONTD, CONSISTS OF ALTSEQ, OR CONSISTS OF PAGE FOLLOWED BY ONE OR TWO CHARACTERS.

RG082 E O THE FIELD IS UNDEFINED.

RG083 E C INCORRECT APOSTROPHES IN CONSTANT OR EDIT WORD.

RG084 E O END POSITION (COLS. 40-43) IS MISSING OR INVALID, IS SMALLER THAN THE SIZE OF THE FIELD, CONSTANT, OR EDIT WORD, OR IS INCOMPATIBLE WITH THE OUTPUT FILE OR DEVICE.

RG085 E C BOTH ZERO SUPPRESS AND A CONSTANT OR EDIT WORD ARE SPECIFIED.

RG086 E O ZERO SUPPRESS OR AN EDIT WORD SPECIFIED FOR ALPHAMERIC FIELD.

RG087 E O STERLING FIELD IS INCORRECTLY SPECIFIED.
The entries are invalid; or the Sterling field (cols. 71-74) contains specifications, but the RPG Control Card is blank in cols. 19 and 20.

RG088 E O STERLING FIELD IS SPECIFIED WITH INCORRECT DECIMAL LENGTH.
Either the field is defined as alphameric, or Decimal Positions is greater than 4.

RG089 E O STERLING FIELD IS SPECIFIED TO BE IN PACKED FORMAT.

RG090 E O EDIT WORD HAS INCORRECT LENGTH OR WAS PREVIOUSLY USED WITH A FIELD OF DIFFERENT LENGTH.

RG101 E C THIS FIELD IS UNDEFINED OR INCORRECTLY SPECIFIED.
This message is followed by the headings
FACT.1 FACT.2 RESULT F.
and the consecutive number of the affected card is printed below the appropriate heading.

RG102 E C THE FOLLOWING FIELDS SHOULD BE BLANK FOR THE OPERATION.
This message is followed by the headings
FACT.1 FACT.2 RESULT F.
and the consecutive number of the affected card is printed below the appropriate heading.

RG103 E C THE FOLLOWING ENTRIES ARE REQUIRED, BUT MISSING OR NOT LEFT-JUSTIFIED.
This message is followed by the headings
FACT.1 FACT.2 RESULT F.
and the consecutive number of the affected card is printed below the appropriate heading.

RG104 E C DETAIL CALCULATION IS ENCOUNTERED AFTER TOTAL CALCULATION.

RG105 E C OPERATION CODE INVALID.

RG106 E C RESULTING INDICATOR IS REQUIRED, BUT NOT SPECIFIED.
A COMP, LOKUP, TESTZ, SETON or SETOF operation has been specified, but a resulting indicator has not been specified in columns 54-59.

RG107 E C RESULTING INDICATORS ARE SPECIFIED, BUT NOT PERMITTED.
A Move, Move Zone, EXIT, RLABL, GOTO, or TAG operation has been specified with resulting indicators.

RG108 E C RESULT FIELD CONTAINS A LITERAL.
The specification in columns 43-48 is incorrect.

RG109 E C AN IMPERMISSIBLE OPERATION IS SPECIFIED BETWEEN ALPHAMERIC AND NUMERIC FIELDS.

RG110 E C FIELD LENGTH OF FACTOR 1 AND FACTOR 2 DIFFERS IN A LOKUP OPERATION.

RG111 E C A LOKUP OPERATION IS SPECIFIED AND THE NAME IN FACTOR 2 OR RESULT FIELD DOES NOT BEGIN WITH TAB.

RG112 C C GOTO AND TAG ARE NOT IN THE SAME CALCULATION TIME.
GOTO is specified for detail time and TAG is specified for total time, or vice versa.
This is only a warning message.

RG113 E C A TAG IS SPECIFIED WITH INDICATORS OR RESULTING INDICATORS.

RG114 E C THE SAME LABEL APPEARS IN MORE THAN ONE TAG STATEMENT.

RG115 E C THE RESULT FIELD WAS PREVIOUSLY DEFINED WITH DIFFERENT LENGTH OR DECIMAL POSITIONS.

RG116	E C	SPECIFIED LENGTH OF ALPHAMERIC RESULT FIELD EXCEEDS 256 POSITIONS, OR NUMERIC RESULT FIELD EXCEEDS 15 POSITIONS, OR NUMBER OF DECIMAL POSITIONS SPECIFIED EXCEEDS FIELD LENGTH.
RG117	C C	RESULT FIELD MAY NOT BE LARGE ENOUGH. The nature of the operation can produce a result larger than size of result field. This is only a warning message.
RG118	C C	THE FIELDS IN THESE OPERATIONS DO NOT OBEY SIZE RESTRICTIONS.
RG119	E C	INCORRECT INDICATORS. Indicators and/or Resulting Indicators are invalid.
RG120	E C	THE DIVIDE OPERATION IS SPECIFIED WITH HALF ADJUST AND FOLLOWED BY A MOVE REMAINDER, OR THE MOVE REMAINDER IS NOT PRECEDED BY A DIVIDE WITH THE SAME INDICATORS.
RG121	E C	RESULT FIELD NAME BEGINS WITH TAB, BUT THE OPERATION IS NOT LOKUP OR RLABL.
RG122	C C	HIGH/LOW INDICATOR FOR LOKUP BUT NO TABLE SEQUENCE SPECIFIED. This is only a warning message.
RG123	C C	FUNCTION TABLE SHORTER THAN ARGUMENT TABLE. This is only a warning message

----

RG131	E	MULTI-FILE PROGRAM, BUT NO MATCHING FIELDS SPECIFIED.
RG132	E I	THE OVERALL LENGTH OF CONTROL OR MATCHING FIELDS IS LARGER THAN 144.
RG133	E I	THE OVERALL LENGTH OF MATCHING FIELDS AND/OR CONTROL FIELDS FOR ONE LEVEL IS NOT CONSTANT FOR ALL PERTINENT CARD TYPES.
RG134	E I	A MATCHING FIELD IS MULTIPLY DEFINED WITHIN A RECORD-TYPE GROUP, BUT THE ENTRIES IN FIELD-RECORD RELATION ARE THE SAME.
RG135	E I	CONTROL AND/OR MATCHING FIELDS INCORRECTLY SPECIFIED
RG136	E C	THE FOLLOWING NAMES ARE USED AS FIELD NAMES AND IN TAG OR GOTO STATEMENTS. This message is followed by the name(s).
RG137	C	UNREFERENCED FIELD NAMES. The message is followed by the names of all fields that are defined but not referenced. This is only a warning message.
RG138	C I	THE CONTROL FIELDS OF ONE LEVEL ARE SPECIFIED BOTH AS NUMERIC AND ALPHAMERIC. This is only a warning message.
RG141	C	SAME FIELD WITH DIFFERENT BLANK/ZERO INDICATORS, AND BLANK-AFTER IS SPECIFIED. This message is followed by the field name together with the blank/zero indicator that is turned on by Blank-After. This in only a warning message.
RG142	E	UNDEFINED INDICATORS. Indicators are used as conditioning indicators but are nowhere assigned as Resulting or Field Indicators. This message is followed by a list of all undefined indicators.
RG143	E C	INCORRECTLY DEFINED INDICATOR LABELS. INxx in RLABL does not identify a valid indicator. This message is followed by a list of all incorrectly specified indicator labels.
RG144	C	UNREFERENCED INDICATORS. The message is followed by a list of all indicators that are defined but not referenced. This is only a warning message.

RG150	I	END OF DIAGNOSTICS NO ERRORS

RG151 D END OF DIAGNOSTICS
REVIEW CAUTIONS

This message is associated with halt D01. It indicates that--while there were no known errors in the source deck--abnormal conditions, as defined by Caution messages (Significance Code C), exist.

RG151 W END OF DIAGNOSTICS
ERRORS IN SOURCE DECK

This message is associated with halt D14. It indicates that known errors, as defined by Error messages (Significance Code E), exist in the source deck.

RG161 I INDICATORS.

This message is followed by a list of all indicators in which each indicator is preceded by its core storage address in hexadecimal notation.

RG162 I DATA FIELDS
ADDRESS|NAME|LENGTH IN BYTES|
DEC. POS.|TYPE

This message is followed by a list of all defined fields in the following order:

Alphameric fields, Numeric fields, Tables. In the column ADDRESS, the core storage addresses of the data fields are listed in hexadecimal notation. The column TYPE indicates whether the field is alphameric or numeric.

RG163 I CONTROL FIELDS
ADDRESS|CONTROL LEVEL|
LENGTH IN BYTES

This message is followed by a list of all assigned control levels and matching fields. The column ADDRESS contains the addresses of the control fields in hexadecimal notation.

RG164 I LITERALS AND CONSTANTS
ADDRESS|LENGTH IN BYTES|
DEC. POS.|TYPE|ACTUAL

This message is followed by a list of all constants and literals defined in the program. Their core storage addresses are listed in the column ADDRESS in hexadecimal notation. The remark NUMERIC, ALPHAMERIC, or EDIT WORD appearing in the column TYPE signifies the type of the constant or literal. The column ACTUAL contains the image of the constant or literal as defined in the specification forms. (Numeric literals are represented in hexadecimal notation.)

RG165 I STARTING ADDRESSES OF RPG ROUTINES

ADDRESS	NAME
xxxx	INPUT RECORDS
.	INPUT FIELDS
.	DETAIL CALCULATIONS
.	TOTAL CALCULATIONS
.	HEADING AND DETAIL LINES
.	TOTAL LINES
.	OVERFLOW LINES
.	OUTPUT FIELDS
.	USER'S SUBROUTINES
.	TRANSLATE SUBROUTINE
.	TRANSLATE TABLE
.	STERLING SUBROUTINES
.	INPUT AREA FOR IBM 2560 HOPPER 1 OR IBM 2520
.	INPUT AREA FOR IBM 2560 HOPPER 2
.	PUNCH AREA FOR IBM 2560 OR IBM 2520
.	CARD PRINT AREA FOR IBM 2560
.	FIRST INPUT AREA FOR IBM 2501
.	SECOND INPUT AREA FOR IBM 2501
.	PUNCH AREA FOR IBM 1442
.	1442 PUNCH ROUTINE
.	PRINT ROUTINE
.	2560/2520 ROUTINES
.	2501 INPUT ROUTINE
xxxx	LAST STORAGE POSITION USED (EXCLUDING I/O AREAS)

Only those routines and/or data fields appear in the program listing that have a core storage address assigned by RPG. The addresses are represented in hexadecimal notation.

RG166 I GENERATION COMPLETED.

This message is associated with halt DFF.

RG167 I TABLES.

Contents of table cards exactly as loaded.

APPENDIX I: FORMAT OF THE CPS RPG CONTROL CARD

THE CPS RPG CONTROL CARD

The format of the CPS RPG control card is as follows:

Column	Entries
1-5	Unused by the RPG program. RPG ignores entries in these columns. However, any entries will be printed in the program listing together with the image of the CTL card. The programmer may therefore use these columns for additional identification. Any valid EBCDIC characters (including blanks) may be used.
6	RPG control card identification H = mandatory entry.
7-9	Core storage capacity of the system used to generate the object program. blank or 004 = 4K (4,096 bytes of core storage) 008 = 8K (8,192 bytes of core storage) 012 = 12K (12,288 bytes of core storage) 016 = 16K (16,384 bytes of core storage)
10	Controls object program punching. blank = the object program is not to be punched into cards after generation. C = the object program is to be punched by a 2520 Card Read-Punch or a 2520 Card Punch. M = the object program is to be punched by a 2560 MFCM. (Blank cards must be in hopper 2). P = the object program is to be punched by a 1442 Card Punch Model 5.
11	Controls halts and printing of the program listing during generation of object programs. blank = RPG halts if it detects any type of programming errors during generation. The program listing is printed. B = RPG halts only if it detects serious programming errors that do not permit continuation of object program generation. The printing of the program listing is completely bypassed. (This feature may be applied to avoid using the printer during generation of a previously tested program).
12-14	Core storage capacity of the system used to execute the generated object program. blank = the core storage capacity of the system used for the execution is assumed to be equal to the capacity of the system used during generation as defined in column 7-9. 004 = 4K (4,096 bytes of core storage) 008 = 8K (8,192 bytes of core storage) 012 = 12K (12,288 bytes of core storage) 016 = 16K (16,384 bytes of core storage) NOTE: The core storage capacity of the executing system may exceed the core storage capacity of the generating system. In this case however, only a part of the executing system's core storage capacity is used during object program execution.
15	Must be left blank.
16	Affects stacking sequence of processed cards during object program execution of the following prerequisites are met: 1) A 2560 MFCM is attached to the system. 2) The programmer specified in the primary and secondary hopper of the 2560 MFCM: • an input file, or a combined file with cards stacker selected under input specifications, and • an output file.

3) The programmer selected cards from both files to one common stacker.

I = The input cards read are stacked just ahead of their associated output cards.
O = The output cards processed are stacked just ahead of their associated input card.

If no 2560 MFCM is attached, RPG ignores entries in this column. If this column is left blank, RPG takes better advantage of the concurrent processing capability of the Model 20. Therefore entries in this column should appear only if required by the programmer.

17-20 Define format of the Sterling fields used in Sterling currency routines.

Columns 17-20: blank = format not specified. The program contains no Sterling currency routines.

Column 17:

1 = The input-shillings field is in the IBM format.
2 = The input-shillings field is in the BSI format.

Column 18:

1 = The input-pence field is in the IBM format.
2 = The input-pence field is in the BSI format.

Column 19:

0 = The output-shillings field is to be printed only.
1 = The output-shillings field is to be punched in the IBM format. (The field may also be printed.)
2 = The output-shillings field is to be punched in the BSI format. (The field may also be printed.)

Column 20:

0 = The output-pence field is to be printed only.
1 = The output-pence field is to be punched in the IBM format. (The field may also be printed.)
2 = The output-pence field is to be punched in the BSI format. (The field may also be printed.)

21 Specifies use of the decimal point or the decimal comma in numeric literals.
(A numeric literal is a fixed numeric value used in calculation specifications by the programmer.)

blank = The programmer uses the decimal point in numeric literals (e.g., 183.55).
I = The programmer uses the decimal comma in numeric literals (e.g., 183,55).

22 Reserves a storage area as an extra input buffer if a 2501 Card Reader is attached to the system. This generally increases the speed of execution, especially if a 2501 Card Reader Model A2 is attached. If no 2501 Card Reader is attached, RPG ignores entries in this column.

B = An extra input buffer for the 2501 Card Reader is established.
blank = No extra input buffer for the 2501 Card Reader is established. (May be used to reduce core storage requirements.)

23-25 Specify the number of print positions used during object program execution.

This number must not exceed the number of available print positions on the attached printer, but it may be smaller. Note that additioanl print positions, whether specified here or not, require additional printing time. To reduce printing time, the entered number should therefore not exceed the actual requirements during object-program execution.

If not printer is used, the system ignores entries in these columns.

blank = 100 print positions are used during object program execution.
100 =
120 =
132 =
144 = Number of print positions per line used during object program execution

26-74 Must be left blank.

75-78 <u>The contents of these columns appear as program identification</u> in columns 73-76 of each punched, object program card. Any valid EBCDIC character may be used.

If these columns are left blank, RPGO will be punched into columns 73-76 of each object program card. (Columns 77-80 of the object program cards are used for consecutive numbering.)

GC26-3600-7

System/360 Model 20 CPS Report [illegible]am Generator (S360(Mod.20)-28) Printed in U.S.A. GC26-3600-7

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

IBM System/360
Card Programming Support
Report Program Generator

Order No. GC26-3600-7

READER'S COMMENT FORM

This sheet is for comments and suggestions about this manual. We would appreciate *your* views, favorable or unfavorable, in order to aid us in improving *this* publication. This form will be sent directly to the author's department. Please include your name and address if you wish a reply. Contact your IBM branch office for answers to technical questions about the system or when requesting additional publications. Thank you.

Name

Address

What is your occupation?

How did you use this manual?

As a reference source

As a classroom text

As a self-study text

Your comments* and suggestions:

*** We would especially appreciate your comments on any of the following topics:**

Clarity of the text	Accuracy	Index	Illustrations	Appearance	Paper
Organization of the text	Cross-references	Tables	Examples	Printing	Binding

GC26-3600-7

YOUR COMMENTS, PLEASE . . .

This manual is part of a library that serves as a reference source for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

CUT ALONG THIS LINE

Fold

Fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N. Y.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
112 East Post Road
White Plains, N.Y. 10601

Attention: Department 813 U

Fold

Fold

System/360 Model 20 CPS Report Program Generator (S360(Mod.20)-28) Printed in U.S.A. GC26-3600-7

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
(USA Only)

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