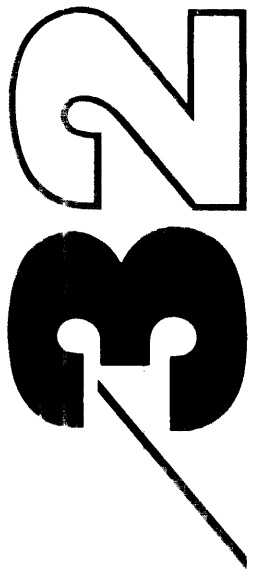
SYSTEM/32

IBM System/32
RPG II
Reference Manual



IBM System/32
RPG II
Programming Information

SC21-7595-0
File No. S32-28

Program Number
5725-RG1

IBM System/32
RPG II
Reference Manual

First Edition (January 1975)

This edition applies to version 01, modification level 00 of IBM System/32 RPG II (Program Product 5725-RG1), and to all subsequent versions and modification levels until otherwise indicated in new editions or technical newsletters.

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

A Reader's Comment Form is at the back of this publication. If the form is gone, address your comments to IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901.

Preface

This reference manual is for programmers writing RPG II specifications for the IBM System/32.

The manual describes the program documentation you can use to help write, test, and maintain RPG II programs. It also contains information common to all RPG II coding sheets, and describes the types of RPG II specifications in the order the RPG II compiler requires them.

Specifications for each coding sheet are described column by column as a programmer would write them. Information in every column description is presented in this order:

1. List of possible entries
2. General discussion of the use of the column and considerations for all possible entries
3. Specific discussion of each entry
4. Charts and examples

This manual has four major parts.

Part 1. RPG II Programming Specifications

This section contains an introduction to the RPG II language, RPG II program logic, machine requirements, summary of RPG II specifications (for quick reference by the programmer), and a column-by-column description for the specification sheets.

Part 2. RPG II Programmer's Guide

This section further explains topics which were introduced, but not fully explained in Part 1.

Part 3. Supplementary Information

This section contains sample programs, programming aids and tips, bytes of generated code for calculations, IBM written subroutines, OCL (operation control language) and command statements, and RPG II program halts.

Part 4. RPG II Auto Report Function

This section contains information for using the auto report function and a sample program for auto report.

Appendix A

Appendix A is a glossary of RPG II terms.

Related Publications

- *IBM System/32 Operator's Guide,* GC21-7591
- *IBM System/32 Messages Guide—RPG II,* SC21-7617
- *IBM System/32 System Control Programming Reference Manual,* GC21-7593
- *IBM System/32 Introduction,* GC21-7582
- *IBM System/32 RPG II Telecommunications Programming Reference Manual,* SC21-7597

Titles and abstracts of other related publications are listed in the *IBM System/32 Bibliography,* GC20-0032.

Contents

List of Abbreviations

ADDROUT	Address output
ASCII	American national standard code for information interchange
BSCA	Binary synchronous communications adapter
EBCDIC	Extended binary coded decimal interchange code
EOF	End of file
IDE	Interactive data entry
K	1,024 bytes
LR	Last record
MIC	Message identification code
MR	Matching record
OCL	Operation control language
SCP	System control program
SIAM	Shared input/output access method
TP	Teleprocessing

Part 1
RPG II Programming Specifications

Chapter 1. Introduction

USING RPG II

RPG II consists of a symbolic programming language and a compiler program. Figure 1 shows general operations involved in preparing a report using RPG II. The circled numbers in Figure 1 refer to the numbers in the following text:

1. The programmer analyzes the report requirements to determine the format of the input files and the layout of the finished report. For example, the programmer determines what fields in the input records are to be used, what calculations are to take place, where the data comes from, where the data is to be located in the output records, and how many and what kind of totals must be accumulated.

2. After analyzing the requirements for the report, the programmer provides the RPG II compiler with information about these requirements. This information is coded on RPG specification sheets. The specification sheets are designed so that one specification line represents one statement in the source program.

   a. Control card (header) specifications. The programmer provides special information about the program and describes the system to the RPG II compiler by making entries on this sheet.

      Since IBM System/32 does not support card input, the word *card* will not be used in this manual when referring to this specification sheet. The word *card* appears on the specification sheet since this sheet is used by all systems that support RPG.

   b. File description specifications. The programmer describes all files (input files, output files, table or array files, etc) the object program uses through entries on this sheet.

   c. Extension specifications. If the programmer uses record address files, tables, or arrays in an object program, the programmer furnishes information about them through entries on the extension specifications sheet.

   d. Line counter specifications. A programmer using printer files in an object program can provide information about the number of lines to be printed on the forms that are used by making entries on the line counter specifications sheet.

   e. Telecommunication specifications. A programmer using BSCA in a program must provide information about each BSCA file on the telecommunications specifications sheet.

   f. Input specifications. The programmer describes input records (record layout, fields used, etc) entered from all input devices except the keyboard by making entries on the input specifications sheet.

   g. Calculation specifications. The programmer states what processing is to be done (add, subtract, multiply, divide, etc) by means of entries on a calculation specifications sheet. The calculation sheet is also used to describe fields entered from the keyboard, and to control certain input and output functions related to those fields.

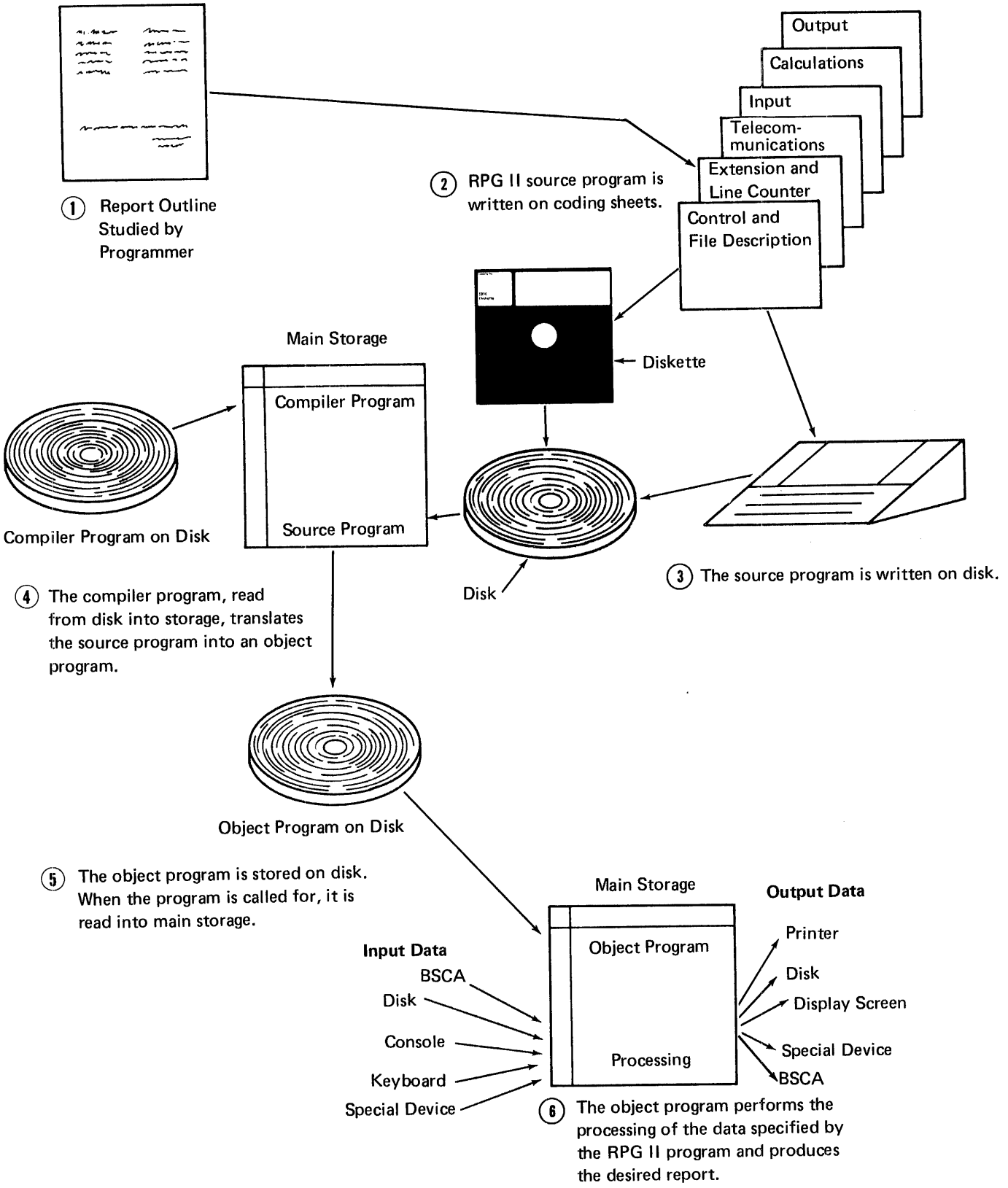
   h. Output specifications. The programmer defines the layout of the desired report (print positions, headings, etc) by making entries on the output specifications sheet.

3. After the specifications have been written on the appropriate forms, the source program is placed in the library on the disk. This can be done in the following ways:

   - Use the source entry utility program of System/32 utilities program product and enter the source program from the keyboard. See the *IBM System/32 Utilities Program Product Reference Manual—Source Entry Utility*, SC21-7605, for information about the source entry utility.

   - Place the source program on a diskette. (The source program on the diskette must use the basic interchange mode.) The source program can be transferred to the disk library by copying to a file using the TRANSFER command and from the file to the source library using the TOLIBR command. See the *IBM System/32 System Control Programming Reference Manual*, GC21-7593, for more information.

   - Use the reader-to-library copy function of the utility program $MAINT and enter the source program from the keyboard. See *IBM System/32 System Control Programming Reference Manual*, GC21-7593, for more information.



● **Figure 1. Preparation of a Report Using RPG II**

Each line of a specification is considered one input statement in your source program. Figure 2 shows the proper arrangement of statements in a source program.

4. These source program specifications should begin with RPG II control specifications. The source program specifications are entered into the system and the RPG II compiler processes them under control of the system control program. At the end of the processing run (compilation), the object program is produced and stored on the disk. This object program contains all the machine instructions required to prepare the desired report.

5. The programmer can now leave the object program on the disk to be processed later, or he can proceed directly to processing the object program.

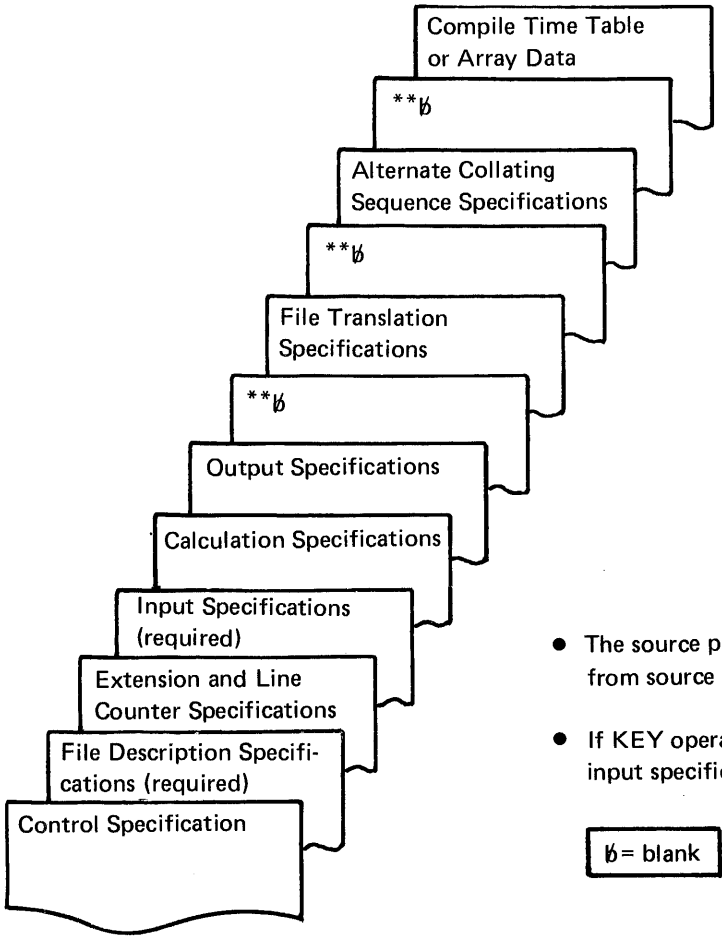
6. When processing is done, the input data files and the object program are read into the system. This final operation (object run) results in the desired report.

SYSTEM CONFIGURATION

The IBM System/32 RPG II Compiler runs on all models of System/32 and supports all available features of this system.

GENERAL RPG II OBJECT PROGRAM LOGIC

Knowledge of RPG II logic is essential for anyone writing RPG II programs. For relatively simple jobs involving a single input file, knowledge of the general logic presented here is sufficient. For more complex jobs, a more detailed



• **Figure 2. Arrangement of the RPG II Source Program**

understanding is required. See Part 2, *RPG II Programmer's Guide, Detailed RPG II Program Logic* for a detailed flowchart and explanation of the program logic.

Each object program the RPG II Compiler generates uses the same general program cycle (program cycle refers to all the logic functions performed for each data record read).

Every program cycle involves three basic logic steps:

1. Reading information (input)
2. Performing calculations (processing)
3. Recording results (output)

Within a program cycle, these basic logic steps can be divided into numerous substeps in which the programmer uses the input data to determine when calculation and output operations will occur. According to RPG II program logic, calculation and output operations are performed at two different times in a cycle: total time and detail time.

Total calculations are specified by placing an L indicator (L0-L9) in columns 7-8 of the calculation specifications sheet. Total output operations are specified by placing a T in column 15 of the output specifications sheet. The appropriate control level indicator should be entered in columns 23-31 of the output specifications sheet to distinguish between output operations performed for different control levels.

Total calculation and total output operations are normally performed on data accumulated for a group of related records which form a control group. Such operations are normally done only after a control break has occurred. A control break occurs when the control field of the record just read is different from the control field of the previous record. Whenever a record is read, a check determines if information in a control field (if a control field is specified) is different from the control field information on the previous record.

A change in the control field information indicates that all records from a particular control group have been read and a new group is starting. When all records from a group have been read (shown by control level indicators being turned on), calculation and output operations are done using information accumulated from all records in that group. Information on the record that started the new control group is not used in these total operations; only information from records in the previous group is used.

Those calculations not conditioned by L indicators in columns 7-8 of the calculation specifications sheet are detail calculations. Detail output operations are specified by placing an H or D in column 15 of the output specifications sheet. Detail calculation and detail output operations are normally performed for individual data records. These operations are done for each record, provided all conditioning indicators are satisfied. When any one of the following conditions is met, detail time calculation and output operations are done:

1. All total calculation and total output operations are completed, but the last record is not read.
2. No total operations are to be done (the information in the control field has not changed).

Total operations are performed before detail operations. This prevents data from the first record in a new control group from being accumulated in the totals for the previous group. Total operations are performed only on data accumulated from previous records. Detail operations on the record that caused the control break are done after total operations are finished.

Figure 3 shows specific steps in the general flow of RPG II program logic. A program cycle begins with step 1 and continues through step 10, then begins again with step 1. Steps 6 and 7 are known as total time; steps 1 and 10 are known as detail time. The following statements describe each step:

Step 1. If all conditioning indicators are satisfied, the program prints all heading or detail lines (those having an H or D in column 15 of the output specifications sheet).

Step 2. All control level and record identifying indicators are turned off.

Step 3. A record is read and identified by the object program. The appropriate record identifying indicator is turned on.

Step 4. The record just read is examined to determine whether or not a control break has occurred. (A control break occurs when the control field of the record just read is different from the control field of the previous record.)

Step 5. If a control break has occurred, the proper control level indicator and all lower control level indicators are turned on, except L0 which is always on.

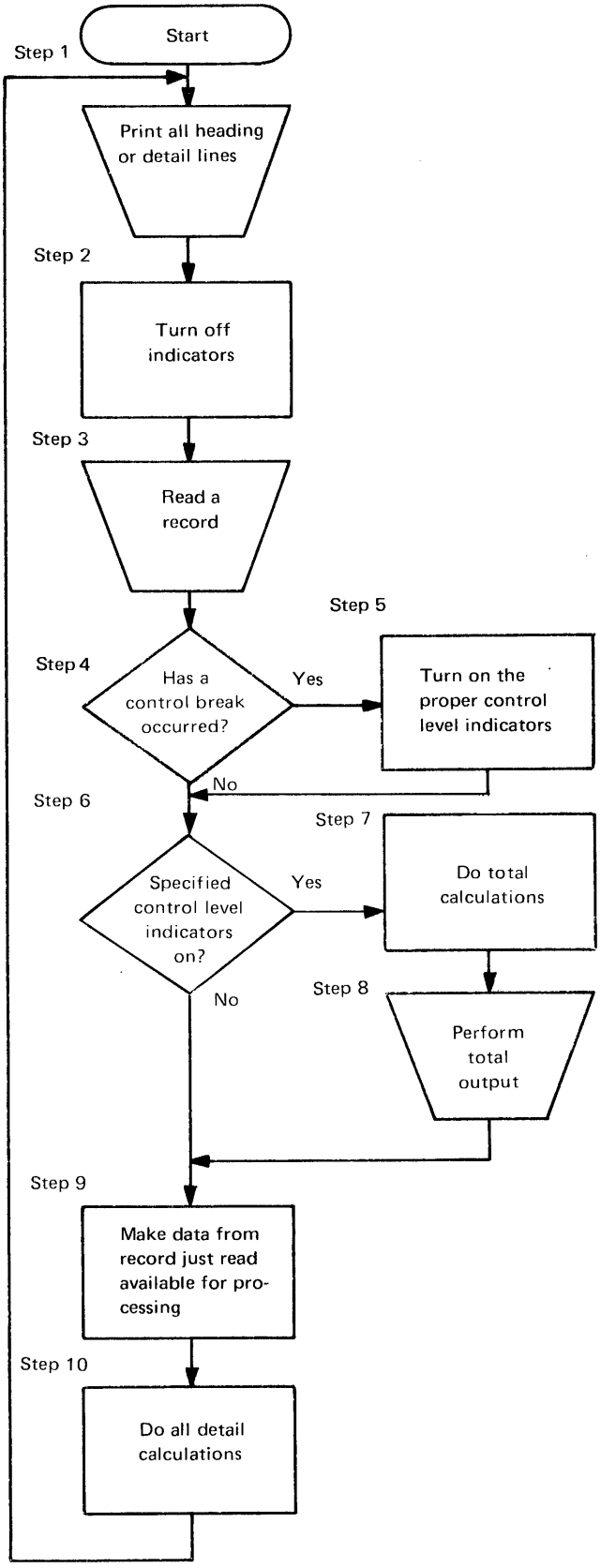


Figure 3. RPG II Program Logic Cycle

Step 6. A check determines if any of the control level indicators that are on are used in columns 7-8 to condition total calculations.

Step 7. Total calculation operations (those conditioned by control level indicators used in columns 7-8 of the calculation specifications sheet) are performed if the appropriate control level indicator is on.

Step 8. Total lines (those having a T in column 15 of the output specifications sheet) are written according to output specifications.

Step 9. Data from the record read at the beginning of the cycle (step 3) is now made available for use in detail calculation and output operations.

Step 10. All detail calculation operations (those not conditioned by control level indicators specified in columns 7-8 of the calculation specifications sheet) are performed on the data from the record read at the beginning of the cycle.

The first and last program cycles of a job are somewhat different from the normal cycle just described. Before the first record is read, lines conditioned by the first page (1P) indicator are printed. Any heading lines having no conditioning or all negative conditioning indicators are also printed at this time. Heading lines printed before the first record is read consist of constant or page heading information or fields for reserved words, such as PAGE and UDATE. In addition, total operations are bypassed until after the first record has been completely processed, even though a control break may occur. Steps 6 and 7 in the program are not done.

When the last record to be processed is read, the last record (LR) indicator is turned on. This automatically causes all control level indicators to turn on. Total operations are performed and the job ends; steps 3-8 in the program cycle are done.

SUMMARY OF RPG II SPECIFICATIONS

This section contains a brief column-by-column description of each of the RPG II specification sheets. It is intended as a quick reference for programmers who are acquainted with RPG II for the System/32. For a complete description of each entry, refer to the applicable section of this manual.

For telecommunications descriptions, see *IBM System/32 RPG II Telecommunications Programming Reference Manual,* SC21-7597.

Control Specifications

Columns	Name	Entry	Explanations
1 and 2	Page	Page number	Use to assign a page number of each specification sheet. See *Common Entries*.
3 through 5	Line	Line number	Use to number the specification lines. See *Common Entries*.
6	Form type	H	Identifies this as the control or header specification. See *Common Entries*.
7		*	If an asterisk is placed in this column, it identifies this line as a comment line. See *Common Entries*.
7 through 9	Size to compile	Blank	
10	Object output	Blank or D	Blank entry causes system halt for terminal errors only. Entering a D causes system halts for both warning messages and severe errors.
11	Listing options	Blank B P	Program listing is produced. No program listing is produced. Partial program listing is produced.
12 through 14	Size to execute		
12		Blank or 0 Q H T	No additional 256-byte increments are needed. One additional 256-byte increment is needed. Two additional 256-byte increments are needed. Three additional 256-byte increments are needed.
13 and 14		Blank 01 through system size - 2K	Main storage size available for program execution defaults to system size minus the SCP resident nucleus size (2K). Main storage size available for program execution. Entry is the number of K available.
15	Debug	Blank 1	DEBUG operation is not used. DEBUG operation is used.
16 through 18		Blank	
19	Date format (UDATE)	Blank or M D Y	Month/Day/Year format. If column 19 is blank and column 21 contains a D, I, or J, the day/month/year (ddmmyy) format is used instead of the month/day/year (mmddyy) format. Day/Month/Year (ddmmyy). Year/Month/Day (yymmdd).

Control Specifications (Continued)

Columns	Name	Entry	Explanations
20	Date edit (Y edit code)	Any of the 256 characters	Characters to be used with the Y edit code in the edit pattern for the date. A blank entry defaults to / if column 19 contains an M, or if column 21 contains a D or blank and column 19 is blank. A blank entry defaults to . if column 19 contains D or Y, or if column 21 contains I or J and column 19 is blank.
21	Inverted print	Blank	Numeric fields use decimal point (.). Date format is mmddyy if column 19 is blank.
		D	Numeric fields use decimal point (.). Date format is ddmmyy if column 19 is blank.
		I	Numeric fields use decimal comma (,). Date format is ddmmyy if column 19 is blank.
		J	Numeric fields use decimal comma (,) and leading zero remains for zero balance. Date format is ddmmyy if column 19 is blank.
22 through 25		Blank	
26	Alternate sequence	Blank	Normal collating sequence is used.
		S	Alternate collating sequence is used.
27 through 36		Blank	
37	Inquiry	Blank or I	Program is not interruptable.
		B	Program recognizes inquiry requests. (For an explanation of inquiry, see Part 1, Chapter 2.)
38 through 40		Blank	
41	1P forms positions	Blank	First 1P line is printed only once.
		1	First 1P line can be printed repeatedly to allow forms positioning.
42		Blank	
43	File translation	Blank	No file translation needed.
		F	Input, output, update, or combined files are translated.
44		Blank	
45	Nonprint characters	Blank	Program halts if unprintable character was in last line printed.
		1	Program does not halt for unprintable characters.
46 and 47		Blank	
48	Shared I/O	1	All disk files share a single input/output area.
		Blank	All disk files use a separate input/output area.
49 through 74		Blank	
75 through 80	Program identification		Use to assign a name to your object program. See *Common Entries.*

File Description Specifications

Columns	Name	Entry	Explanations
1 and 2	Page	Page numbers	Use to assign a page number of each specification sheet. See *Common Entries*.
3 through 5	Line	Line number	Use to number the specification lines. See *Common Entries*.
6	Form type	F	Identifies this as a file descriptions specification. See *Common Entries*.
7		*	If an asterisk is placed in this column, it identifies this line as a comment line. See *Common Entries*.
7 through 14	Filename	Filename	Enter a name for each file. The filename can be from one to eight characters long, must begin in column 7, and must be a valid RPG II name.
15	File type	I	Input
		O	Output
		U	Update
		C	Combined (SPECIAL device only)
16	File	P	Primary
		S	Secondary
		C	Chained
		R	Record address
		T	Table or array
		D	Demand
		Blank	Leave blank for all output files except chained output files
17	End of file	E	All records from the file must be processed before the program ends. An E can only be specified here if column 15 contains I or U, and column 16 contains a P, S, or R.
		Blank	The program can end whether or not all records from this file are processed. *Note:* If column 17 is blank or contains E for all files, all records from every file must be processed before the program can end.
18	Sequence	Blank	No sequence checking is to be done.
		A	Sequence checking is done. Records are in ascending sequence.
		D	Sequence checking is done. Records are in descending sequence. *Note:* Sequence checking is required when matching fields are used. Column 18 applies only to primary and secondary files.
19	File format	F	Indicates fixed length record format.

File Description Specifications (Continued)

Columns	Name	Entry	Explanations
20 through 23	Block length	1-4096	Disk (record length or multiple of record length)
		6-4096	Console
		1-40	Keyboard
		1-132	Printer
		1-40	Display screen
		1-4096	SPECIAL (record length or multiple of record length)
		1-4096	BSCA (record length or multiple of record length)
		Blank	These columns can be left blank for any file when default values are acceptable
24 through 27	Record length	1-4096	Disk
		4-160	Console
		1-40	Keyboard
		1-132	Printer
		1-40	Display screen
		1-4096	SPECIAL
		1-4096	BSCA
28	Mode of processing	Blank	- Sequential by key - Consecutive *Note:* This column must be blank for nondisk files.
		L	Sequential within limits
		R	- Random by relative record number - Random by key - By ADDROUT file - Direct file load (random load)
29 and 30	Length of key field or record address field	1-8	Length of record keys in packed format (for indexed files)
		1-29	Length of record keys in unpacked format (for indexed files and record address files)
		3	Length of disk address in ADDROUT files
31	Record address type	P	Indexed file with packed keys
		A	Indexed file
		I	ADDROUT file or processed by ADDROUT file
		Blank	Sequential or direct file *Note:* Column 31 applies to disk files specified as input, update, or chained output files
32	File organization or additional I/O area	I	Indexed organization
		T	ADDROUT file
		1-9	Sequential or direct file, use two I/O areas for the file
		Blank	Sequential or direct file, use one I/O area for the file
33 and 34	Overflow indicator	OA-OG, OV	Overflow indicator used to condition records in the file
		Blank	No overflow indicator used

File Description Specifications (Continued)

Columns	Name	Entry	Explanations
35 through 38	Key field starting location		For indexed files, enter the beginning position of the key field in the record. This entry must end in position 38.
39	Extension code	E	The file described on this line is a table file, array file, or record address file further described by extension specifications.
		L	The file described on this line is a printer file further described by line counter specifications.
40 through 46	Device	DISK	Disk
		KEYBORD	Display screen – keyboard
		PRINTER	132-position printer
		CONSOLE	Interactive data file. Uses the keyboard and display screen
		CRT	Display screen
		SPECIAL	Used for devices not supported directly by RPG II
		BSCA	Binary synchronous communications adapter
47 through 53		Blank	
54 through 59	Name of label exit	SUBRxx	Name of the user-written subroutine which performs the I/O operation for a SPECIAL device (x = any alphabetic character).
		SRyzzz	Name of the IBM-written subroutine (6 character name in library is #$yzzz) which performs the I/O operation for a device supported by SPECIAL (y = any of the following: B, C, D, F, G, H, I, L, M, O, P, R, S, T, or U; z = any of the following: A, B, C, D, F, G, H, I, L, M, O, P, R, S, T, or U).
		Blank	No SPECIAL device is being used.
	Continuation line option	Array name	Name of array to be used by user-written subroutine.
60 through 65	Storage index	6-9999	Number of bytes reserved for storage index.
		Blank	No storage index is kept in storage.
66	File addition/ unordered load	A	New records will be added to the file.
		U	Records are to be loaded into an indexed file in unordered sequence. *Note:* This column applies to sequential and indexed disk files.
67 through 70		Blank	
71 and 72	File condition U1-U8	U1-U8	The specified external indicator conditions the file.
		Blank	An external indicator does not condition the file. *Note:* These columns apply to output files, primary and secondary input files, and update files. A record address file can be conditioned by an external indicator if its associated primary or secondary file is conditioned either by the same indicator or by no indicator.
73 and 74		Blank	
75 through 80	Program identification		Use to assign a name to your object program. See *Common Entries.*

Extension Specifications

Columns	Name	Entry	Explanations
1 and 2	Page	Page number	Use to assign a page number of each specification sheet. See *Common Entries*.
3 through 5	Line	Line number	Use to number the specification lines. See *Common Entries*.
6	Form type	E	Identifies this as an extension specification. See *Common Entries*.
7		*	If an asterisk is placed in this column, it identifies this line as a comment line. See *Common Entries*.
7 through 10		Blank	
11 through 18	From filename	Filename	Enter, left-justified, the name of the table or array input file loaded at preexecution time or the name of the record address file defined on the file description specifications sheet.
19 through 26	To filename	Filename	If the file named in columns 11-18 is a record address file, enter the name of the primary or secondary input or update file containing the data records to be processed. If the file named in columns 11-18 is a table or array file, enter the name of the output file to which the table or array is written at end of job.
		Blank	Leave blank if the table or array is not written at end of job.
27 through 32	Table or array name	Table or array name	Enter the name of a table or array used in the program. If alternating tables or arrays are described, enter the name of the table or array whose entry is first on the input record. Entries must be left-justified and must be valid RPG II names. Table names must begin with TAB; array names must not begin with TAB.
33 through 35	Number of entries per record	Number of table or array entries per record	Enter, right-justified the number of entries on each table or array input record. These columns must contain an entry for compile and preexecution time tables and arrays.
		Blank	These columns must be blank for execution time arrays.
36 through 39	Number of entries per table or array	Maximum number of table or array entries	Enter, right-justified, the maximum number of entries in the table or arrays, corresponding items are considered one entry.
40 through 42	Length of entry	Length of table or array entries	Enter, right-justified, the length of each table or array entry. The maximum length is 256 for alphameric entries and 15 for numeric entries. For packed or binary numeric data, enter the number of digits required to represent the data in unpacked format.

Extension Specifications (Continued)

Columns	Name	Entry	Explanations
43	Packed or binary field	Blank	Alphameric or unpacked numeric data
		P	Packed numeric data
		B	Binary numeric data
44	Decimal positions	Blank	Alphameric table or array
		0-9	Number of positions to the right of the decimal
45	Sequence	Blank	No particular sequence
		A	Ascending sequence
		D	Descending sequence
			Note: This column describes the sequence of data in a table or array. Column 45 must contain an entry if high or low look-up is used.
46 through 57			Use these columns when describing a second table or array entered in alternating format with the table or array named in columns 27-32. These entries have the same significance as the corresponding entries in columns 27-45.
58 through 74	Comments		Enter any information that helps you understand what you are doing in each specification line.
75 through 80	Program identification		Use to identify your object program or for comments. See *Common Entries*.

Line Counter Specifications

Columns	Name	Entry	Explanations
1 and 2	Page	Page number	Use to assign a page number of each specification sheet. See *Common Entries*.
3 through 5	Line	Line number	Use to number the specification lines. See *Common Entries*.
6	Form type	L	Identifies this as the line counter specification. See *Common Entries*.
7		*	If an asterisk is placed in this column, it identifies this line as a comment line. See *Common Entries*.
7 through 14	Filename	Filename	Enter the name of a printer file for which you want to specify a form size and overflow line.
15 through 17	Line number—number of lines per page	6-84	Number of lines available for printing on the printer form. If an entry of less than 6 is made, the results are unpredictable.
18 and 19	Form length	FL	Indicates the previous entry is the form's length.
20 through 22	Line number—overflow line	1-84	Number of the overflow line.
23 and 24	Overflow line	OL	Indicates the previous entry is the overflow line.
25 through 74		Blank	
75 through 80	Program identification		Use to identify your object program or for comments. See *Common Entries.*

Telecommunications Specifications

Columns	Name	Entry	Explanations
1 and 2	Page	Page number	Use to assign a page number of each specifications sheet. See *Common Entries.*
3 through 5	Line	Line number	Use to number the specifications lines. See *Common Entries.*
6	Form type	T	Identifies this as a telecommunications specification. See *Common Entries.*
7		*	An asterisk in this column identifies this line as a comment line. See *Common Entries.*
7 through 14	Filename	Filename	Enter a valid filename for every BSCA file your program uses. This must be the same filename specified by the file description specifications.
15	Configuration	P or blank	Point-to-point, nonswitched network.
		M	Multipoint network, where the control station selects the tributary station through polling or addressing. System/32 cannot be the control station.
		S	Point-to-point switched network.
16	Type of station	T	This station transmits messages from the file named in columns 7 through 14. The file must be designated as an output file by file description specifications and must appear on the output specifications sheet.
		R	This station receives messages into the file named in columns 7 through 14. The file must be designated as an input file by file description specifications and must appear on the input specifications sheet.
17	Type of control	T	This is a tributary station on a multipoint network. System/32 cannot be the control station and transmit the polling supervisory sequence. Column 17 must contain a T if column 15 contains an M (multipoint network).
		Blank	Polling is not used.

Telecommunications Specifications (Continued)

Columns	Name	Entry	Explanations
18	Type of code	A or U	ASCII data link control characters are used. When ASCII is used, each station must provide file translation when it is required.
		E or blank	EBCDIC data link control characters are used.
19	Transparency	Y	EBCDIC transparency is used. The data being transferred may contain data link control characters.
		N or blank	EBCDIC transparency is not used. Unpacked numeric or alphameric data is transmitted and received. The data being transferred cannot contain data link control characters.
20	Switched	M	The computer operator at this station makes the connection between stations by dialing the number (manual dial).
		A	This station uses autoanswer.
		B	This station uses manual answer.
		Blank	This is not a switched network.
21 through 31	Blank		
32	Location of identification — this station	S	Switched network. This station's identification is at the position specified by the symbolic name in columns 33 through 39.
		E	Switched network. The entry in columns 33 through 39 is this station's identification.
		Blank	This is a nonswitched network or a switched network where no ID is desired for this station.
33 through 39	Identification — this station	Alphameric	When column 32 contains an E, this entry is the actual identification sequence of this station (from 2 to 15 characters). The station identification must not contain a control character sequence. When column 32 contains an S, this entry is the symbolic name of the location of this station's identification. The symbolic name must not be an array name. If the BSCA file is primary or secondary, this symbolic name must refer to the first element of a table.

Telecommunications Specifications (Continued)

Columns	Name	Entry	Explanations
40	Location of identification — remote station	S	Switched network. The remote station's identification is at the position specified by the symbolic name in columns 41 through 47.
		E	Switched network. The entry in columns 41 through 47 is the remote station's identification.
		Blank	This is a nonswitched network or a switched network where no ID is desired for the remote station.
41 through 47	Identification — remote station	Alphameric	When column 40 contains an E, this entry is the actual identification sequence of the remote station (from 2 to 15 characters). A station identification must not contain a control character sequence. When column 32 contains an S, this entry is the symbolic name of the location of the remote station's identification. This symbolic name must not be an array name. If the BSCA file is a primary or secondary file, this symbolic name must refer to the first element of a table.
48 through 51		Blank	
52	ITB	I	Intermediate block check (ITB) is used. ITB can be used only if records are blocked.
		Blank	ITB is not used. *Note:* Both ITB and EBCDIC transparency cannot be specified for a BSCA output file.
53 and 54	Permanent error indicator	01-99, L1-L9, LR, H1-H9	A permanent error indicator can be specified for every BSCA file. If you are using more than one BSCA file, each file can have a permanent error indicator. The indicator does not have to be unique for each file, however.
		Blank	No permanent error indicator is specified. If a permanent error occurs, a system halt occurs. The program cannot be restarted.

Telecommunications Specifications (Continued)

Columns	Name	Entry	Explanations
55 through 57	Wait time	Number 1-999	The length of time in seconds (1-999) that BSCA waits with no messages being sent or received before a permanent error occurs.
		Blank	The system convention for timeout, 180 seconds, is used.
58 and 59	Record available indicator	01-99, L1-L9, LR, H1-H9	A record available indicator must be assigned to every BSCA file that is to be reopened. (If a file is used again after end of file has been reached, the file is reopened.)
		Blank	No record available indicator is specified. The file cannot be used again.
60	Last file	L	This BSCA input file is processed only after all other primary and secondary input files have been processed.
		Blank	This BSCA input file is not the last input file processed.
61 and 62	Polling characters	Alphameric	The polling identification of this station is needed if this station is part of a multipoint network and the BSCA file is a transmit (output) file.
		Blank	This station is not transmitting on a multipoint network.
63 and 64	Addressing characters	Alphameric	The addressing identification of this station is needed if this station is part of a mutipoint network and the BSCA file is a receive (input) file.
		Blank	This station is not receiving on a multipoint network.
			Note: Enter polling and addressing characters in System/32 code; the compiler converts the characters to the form required by the code specified in column 18. (Enter uppercase addressing characters, and they are converted to lowercase ASCII characters.)
65 through 74		Blank	
75 through 80	Program identification	See *Common Entries*	Used to assign a name to your object program. See *Common Entries.*

Columns	Name	Entry	Explanations
1 and 2	Page	Page number	Use to assign a page number of each specification sheet. See *Common Entries*.
3 through 5	Line	Line number	Use to number the specification lines. See *Common Entries*.
6	Form type	I	Identifies this as an input specification. See *Common Entries*.
7		*	If an asterisk is placed in this column, it identifies this line as a comment line. See *Common Entries*.
7 through 14	Filename	Filename	Enter a valid RPG II filename for every input, update, and combined file your program uses.
14 through 16	AND/OR	AND or OR	Enter AND in columns 14-16 on the next line of the input specifications sheet if more than three record identification code subfields are needed to identify the record. Enter OR in columns 14-15 if either one of the codes can be present to identify the record. A maximum of 20 AND or OR lines in any combination can describe the record identifying code. *Note:* AND lines are not allowed with interactive data entry files.
15 and 16	Sequence	Numeric	Enter a 2-digit number to assign a special sequence to record types in a file and to request that the record type sequence be checked by the program.
		Alphabetic	Enter two alphabetic characters to indicate that record type sequence is not being checked. Alphabetic characters must be used for a chained file. Within a file, record types with an alphabetic sequence entry must be described before record types with a numeric sequence entry.
17	Number	Blank	Columns 15-16 contain alphabetic characters (record type sequence is not being checked).
		1	Columns 15-16 contain numeric characters; only one record of this type is present in each sequenced group.
		N	Columns 15-16 contain numeric characters; one or more records of this type can be present in the sequenced group.
18	Option	Blank	Record type must be present.
		O	Optional; record type may or may not be present. *Note:* Column 18 is used when record types are being sequence checked (columns 15-16 contain a numeric entry).

Input Specifications (Continued)

Columns	Name	Entry	Explanations
19 and 20	Record identifying indicator	01-99	Record identifying indicator. (Interactive data entry can use indicators 01-10 only.)
		L1-L9	Control level indicator used as a record identifying indicator when a record type rather than a control field signals the start of a new control group.
		LR	Last record indicator.
		H1-H9	Halt indicator used as a record identifying indicator when checking for a record type that causes an error condition.
		**	Look-ahead fields (not valid with interactive data entry).
21-41	Record identification codes		*Note:* Columns 21-41 are divided into three identical subfields that are described separately: (1) columns 21-27, (2) columns 28-34, and (3) columns 35-41. An AND relationship exists between these three fields.
21 through 24, 28 through 31, or 35 through 38	Position	Blank	No record identification code is needed.
		1-9999	Record position of the record identification code.
25, 32, or 39	Not (N)	Blank	Either the record identification code is present in the specified record position, or no record identification code is needed.
		N	Record identification is being used, but the identification code is not present in the specified record position.
26, 33, or 40	C/Z/D	C	Entire character
		Z	Zone portion of character
		D	Digit portion of character
27, 34, or 41	Character		Any alphabetic character, special character, or digit identifying the character used in the record as the record identifying code.
42		Blank	
43	Packed or binary field	Blank	Input field in unpacked decimal format
		P	Input field in packed decimal format
		B	Input field in binary format
44 through 47 and 48 through 51	Field location	Numeric field	Enter two 1- to 4-digit numbers to identify the beginning of a field (From) and the end of a field (To) in the input record. These entries are identical for a 1-position field.
52	Decimal position	Blank	Alphameric field.
		0-9	The number of decimal positions in the numeric field named in columns 53-58. This column must contain an entry for numeric fields.

Columns	Name	Entry	Explanations
53 through 58	Field name	Field name	A valid RPG II field name for each field defined in columns 44-51. An array name or array element. If an array name is entered, columns 59-64 must be blank. PAGE, PAGE1, or PAGE2 special words.
59 and 60	Control level	L1-L9 Blank	Field described on this line is a control field. Field described is not a control field. These columns must be blank for chained or demand files.
61 and 62	Matching fields	M1-M9	Enter a match value (M1-M9) to indicate matching fields and sequence checking on primary and secondary files with match fields. When you have just one input, update, or combined file with match fields, this entry causes only sequence checking.
63 and 64	Field record relation	01-99 L1-L9 MR U1-U8 H1-H9	Record identifying indicator assigned to a record type Control level indicator Matching record indicator External indicator Halt indicator
65 through 70	Field indicators	01-99 H1-H9	Field indicator. Halt indicator (when checking for an error condition in the data). *Note:* An indicator used in these columns is turned on if the condition tested for is true. For numeric fields, more than one condition may be tested at a time, but only the indicator which reflects the result of the test is turned on, the others are turned off. If a field is alphameric, an indicator can only be specified in columns 69-70.
71 through 74		Blank	
75 through 80	Program identification		Use to identify your object program or for comments. See *Common Entries.*

Calculation Specifications

Columns	Name	Entry	Explanations
1 and 2	Page	Page number	Use to assign a page number of each specification sheet. See *Common Entries*
3 through 5	Line	Line number	Use to number the specification lines. See *Common Entries*.
6	Form type	C	Identifies this as a calculation specification. See *Common Entries*.
7		*	If an asterisk is placed in this column, it identifies this line as a comment line. See *Common Entries*.
7 and 8	Control level	Blank	Calculation operation is done at detail time.
		L0	Calculation operation is done at total time (always on).
		L1-L9	Calculation operation is done when the appropriate control break occurs or an indicator is set on.
		LR	Calculation operation is done after the last record is processed or after LR has been set on.
		SR	Calculation operation is part of a subroutine.
		AN, OR	Indicates that indicators specified on this list are in either an AND relationship or in an OR relationship with indicators on the preceding line. A maximum of seven AN, OR, or mixed AN and OR lines are allowed to condition an operation.
			Note: Control level entries must be in the order listed.
9 through 17	Indicators	Indicators	Enter one to three indicators. Use any indicators except 1P and L0. Columns 9, 12, and 15 may contain blank or N. An AND relationship exists between indicators on a line. Additional lines may be used containing indicators in columns 9-17 which are in an AND or OR relationship with those on the first line by entering AN or OR in columns 7-8.
18 through 27	Factor 1		Name of any field that is defined. Alphameric or numeric literal. Subroutine, table or array name, or array element. Data field name (UDATE, UMONTH, UDAY, UYEAR). Special name (PAGE, PAGE1, PAGE2). Label for a TAG, BEGSR, or ENDSR operation.
28 through 32	Operation	Operation code	Must be left-justified.
31 and 32		01-99	Message identification code (MIC) to be displayed from the user message member during SET or KEY operations. (Entries are neglected by the compiler when factor 1 is also present on the same SET or KEY operation.)

Calculation Specifications (Continued)

Columns	Name	Entry	Explanations
33 through 42	Factor 2		Name of any field that is defined. Alphameric or numeric literal. Subroutine, table or array name, or array element. Date field name (UDATE, UMONTH, UDAY, UYEAR). Special name (PAGE, PAGE1, PAGE2). Label for a GOTO or EXSR operation. Filename for a CHAIN, DEBUG, READ, FORCE, or SET operation.
43 through 48	Result field	ERASE Field name, table names, array element INxx (xx= any RPG II indicator)	To erase an interactive data entry buffer. These entries hold the results, or are the object of, the operation specified in columns 28-32. The indicator to be transferred to an external subroutine in an RLABL operation.
49 through 51	Field length	Blank 1-256	Field defined elsewhere. Result field length. Maximum length of a numeric result field is 15 digits; maximum length of an alphameric result field is 256 characters. The entry must be right-justified.
52	Decimal position	Blank 0-9	Alphameric field or numeric field described elsewhere. Number of decimal places in a numeric result field.
53	Half adjust	Blank H	Do not half adjust (round) the result field. Half adjust (round) the result field. Half adjust is allowed only with arithmetic operations.
54 through 59	Resulting indicator	01-99 H1-H9 L1-L9 LR OA-OG, OV KA-KN KP, KQ U1-U8	Columns 54-59 are used: • To test the value of the result field after an arithmetic operation. • To check the outcome of a CHAIN, LOKUP, COMP, TESTB, or TESTZ operation. • To specify which indicators to SETON or SETOF. • To indicate end of file for the READ operation code. • To allow command keys to be pressed using the SET operation code. • To test the value of the result field after a KEY operation. • To condition which files are to be used by a specific job. *Note:* Command key KA is entered by pressing the CMD key, the 1 key, and then the ENTER key. Command key KB is entered by pressing the CMD key, the 2 key, and the ENTER key.
60 through 74	Comments		Enter information to help you understand what each specification line does.
75 through 80	Program identification		Used to identify your object program or for comments. See *Common Entries.*

Output Specifications

Columns	Name	Entry	Explanations
1 and 2	Page	Page number	Use to assign a page number of each specification sheet. See *Common Entries*.
3 through 5	Line	Line number	Use to number the specification lines. See *Common Entries*.
6	Form type	O	Identifies this as an output specification. See *Common Entries*.
7		*	If an asterisk is placed in this column, it identifies this line as a comment line. See *Common Entries*.
7 through 14	Filename	Filename	Enter a valid RPG II filename for each output and update file used by your program. Each filename need be specified only once on the first line describing that file.
14 through 16	AND/OR relationship	AND OR	Enter AND if output records are in an AND relationship. Enter OR (columns 14-15) if output records are in an OR relationship. *Note:* A maximum of 20 AND, OR, or mixed AND and OR lines are allowed to condition an output record.
15	Type	H D T E	Heading records. Detail records. Total records. Exception records (records to be written during calculation time).
16 through 18	Add a record	ADD	Enter ADD in these columns if records are added to an input, update, or output disk file. An A must also be entered in column 66 of the file description specifications sheet for the file to which a record is added.
16	Fetch overflow	F	Fetch overflow. The overflow routine is fetched when overflow occurs, before the usual time in the cycle.
17 through 22	Space/skip	See columns 17-18 and 19-22	If these columns are blank, single spacing occurs after each line is printed.
17 and 18	Space	0-3	Enter a number (0-3) under the appropriate column to indicate the number of lines spaced before or after a line is printed.
19 through 22	Skip	Blank 01-84	No skipping. Lines 1-4. Enter one of the 2-digit numbers listed above to indicate the position of the next line printed. All line numbers between are bypassed. Enter the number in the Before or After columns, depending on whether you want skipping to occur before or after the line is printed.

Columns	Name	Entry	Explanations
23 through 31	Output indicators	1 to 3 indicators	Enter one to three indicators. Any indicators may be used. Columns 23, 26, 29 may contain blank or N. The letter N preceding an indicator means the output operation is done only if the indicator is not on. An AND relationship exists between indicators on a line. Additional lines of indicators in an AND or OR relationship may be used by entering AND in columns 14-16 or OR in columns 14-15 of each additional line (up to 20).
32 through 37	Field name	Field name	Enter one of the following to name every field written out: • Any field name defined in this program. • The special words, PAGE, PAGE1, PAGE2, *PLACE, UDATE, UDAY, UMONTH, and UYEAR. • A defined table name, array name, or array element. These columns must be blank if a constant is entered on columns 45-70 of the line. If an entry is made in columns 32-37, columns 7-22 must be blank.
38	Edit codes	Edit codes	Enter an edit code in column 38 when you want to: • Suppress leading zeros for a numeric field. • Omit a sign from the lower order position of a numeric field. • Punctuate a numeric field without setting up your own edit word. A table summarizing the edit codes that can be used is printed above columns 45-70 on the output specifications sheet.
39	Blank after	B	Alphameric field is reset to blank or numeric field is reset to zero after writing.
		Blank	Field is not reset after writing. This column must be blank for look-ahead and UDATE fields. *Note:* If the field name specified with Blank After is a table name, the element of the table looked up last is blanked or zeroed.
40 through 43	End position in output record	Number	Columns 40-43 indicate the location on the output record of the field or constant written. Enter the number of the position occupied by the rightmost character of the output field. The end position entry must not be greater than the record length.
44	Packed or binary field	Blank	Field is unpacked numeric or alphameric blank. This column must be blank for *PLACE fields.
		P	Field is packed decimal numeric data.
		B	Field is in binary format. *Note:* Packed and binary fields can only be written on disk; they cannot be printed.
45 through 70	Constant or edit word	Constant	Enter a constant enclosed in apostrophes.
		Edit word	Enter an edit word, enclosed in apostrophes, to specify editing of numeric fields. Edit words are not used with edit codes.
71 through 74		Blank	
75 through 80	Program identification		Use to identify your object program or for comments. See *Common Entries.*

COMMON ENTRIES

This section defines in more detail entries that are common to all RPG coding sheets. Each coding sheet contains the following entries:

- Columns 1-2 (Page)
- Columns 3-5 (Line)
- Column 6 (Form Type)
- Column 7 (Comments)
- Columns 75-80 (Program Identification)

Columns 1-2 (Page)

Entry	Explanation
01-99	Page number
Blank	No page number is used

Use columns 1-2 in the upper right corner of each sheet to number the specifications sheets for your job. You can use more than one of each type of sheet, but keep all sheets of the same type together. When all specifications sheets are filled out, arrange them in the following order:

1. Control and file description specifications
2. Extension and line counter specifications
3. Telecommunications specifications
4. Input specifications
5. Calculation specifications
6. Output specifications

Number the sheets in ascending order.

Columns 3-5 (Line)

Entry	Explanation
Any numbers	Line numbers
Blank	No line number is used

Columns 3-5 are used to number the lines on each page. Columns 3-4 are preprinted on each sheet, so in most cases, line numbering is already done for you. For instance, the control and file description specifications sheet contains line numbers for lines 01-07. If you need more than seven lines on one sheet, enter 08 in columns 3-4 below line 07; then 09 can be entered if it is required. The blank areas below the preprinted numbers can also be used to insert a line between two lines you have completed (Figure 4).

The control specifications line is always line 01. Any other lines on the sheets can be skipped. The line numbers you use need not be consecutive, but should be in ascending order.

Example

Figure 4 shows the insertion of a line. To show that a line belongs between line 02 and line 03, a 1 is placed in column 5 (any number 1-9 can be used). Line 021 should be inserted between 02 and 03. All lines inserted between existing lines should be written after the last line with a printed line number. A maximum of nine lines can be inserted between two specification lines.

Column 6 (Form Type)

Entry	Explanation
H	Control (header) specifications
F	File description specifications
E	Extension specifications
L	Line counter specifications
T	Telecommunications specifications
I	Input specifications
C	Calculation specifications
O	Output specifications

Column 6 contains a preprinted letter on all sheets. The letter identifies the type of specifications for each line of coding. The H in column 6 of the control specifications stands for header or control record. The control record (control specification) must always be the first record in the RPG II source program (Figure 2).

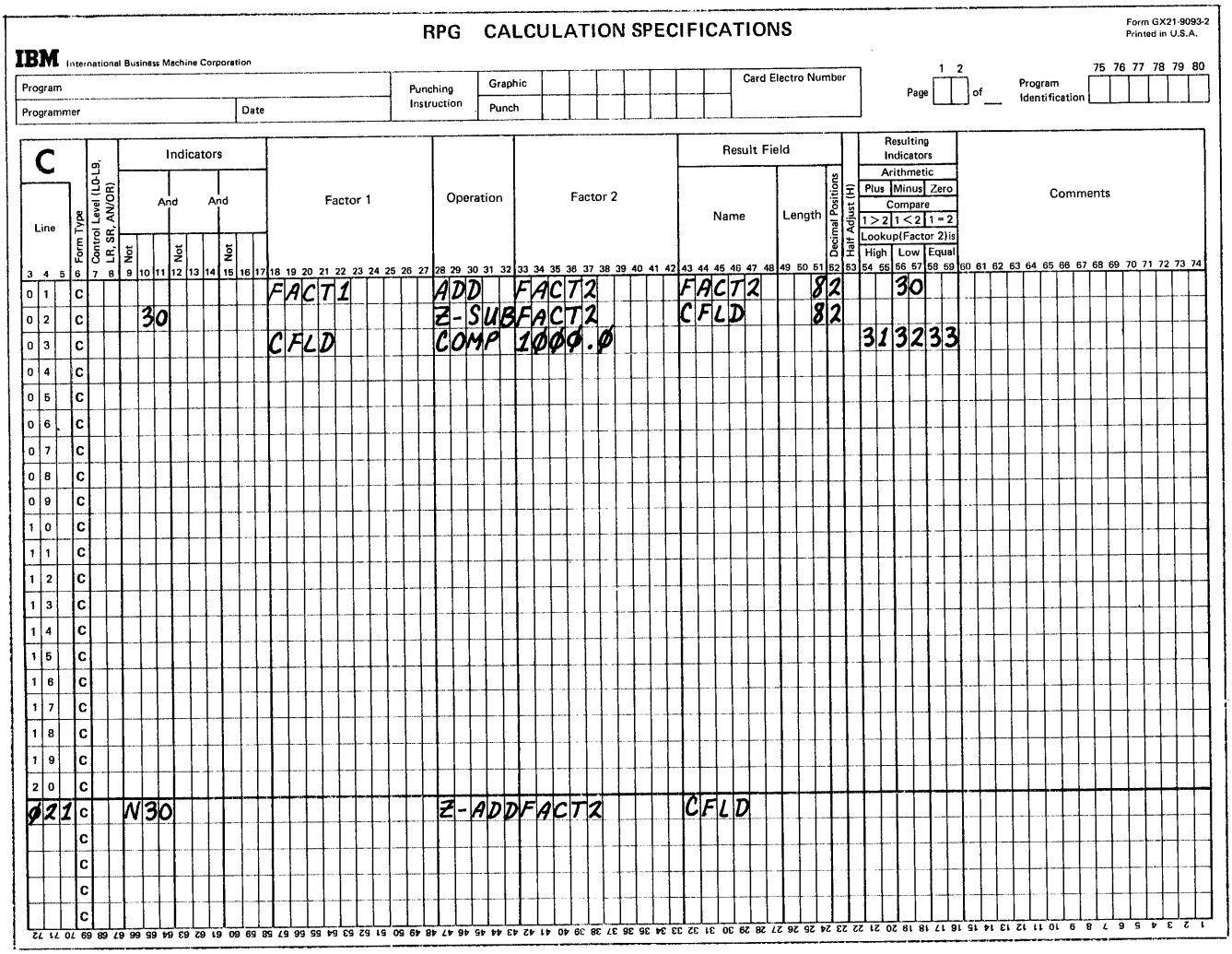


Figure 4. Insertion of Coding Lines

Column 7 (Comments)

Entry	Explanation
*	Comment line

You often want to write comments that help you understand what you are doing in a certain section of coding. RPG II allows you to use an entire line for these comments. The comment line is identified by placing an asterisk in column 7. Any character set may be in a comment line. Comments are not instructions to the RPG II program. They serve only as a means of documenting your program. A comment line cannot be written in the control specification line.

Note: To be compatible with other RPG systems, the specification sheets show only 80 record positions used for RPG II coding.

Columns 75-80 (Program Identification)

Entry	Explanation
Any valid program name	Program identification (the first character cannot be #, $, or @). No special character can be used in the name.
Blank	RPGOBJ is the program identification assigned.

Control Specifications

Columns 75-80 of the control specifications sheet are used to name your object program and to identify each record in the object program. The program name (columns 75-80) is used in a program directory that contains the location of your program on disk. In addition, the compiler places the first four characters from the control specifications (columns 75-78) in positions 89-92 of each record in your object program.

You can use any combination of alphabetic and numeric characters in columns 75-80; however, the first character must be alphabetic. Blanks must not appear between characters in the program name. The program name you specify should be unique. If columns 75-80 are left blank, the compiler assumes the entry is RPGOBJ. (The compiler uses positions 93-96 of each object program record for consecutive numbering of the records.)

All Other Source Program Specifications

Columns 75-80 of all source program specifications sheets, except the control specifications sheet, can contain any characters. These columns may use the program name in the control specifications or they can contain any other characters to identify a certain portion of the program. These entries are ignored by the compiler but appear in the source program listing.

One control specification is required for every program. It provides special information about your program and describes your system to the RPG II Compiler. If the control specification is omitted, a blank control specification is assumed.

To help you code the control specifications, one coding line is provided on the control and file description specifications sheet (Figure 5).

COLUMNS 1-2 (PAGE)

See *Common Entries*.

COLUMNS 3-5 (LINE)

See *Common Entries*.

COLUMN 6 (FORM TYPE)

A control record with an H in position 6 must be entered for every program to provide a unique name for each program. If columns 75-80 are blank or no control record is provided, the program name defaults to RPGOBJ.

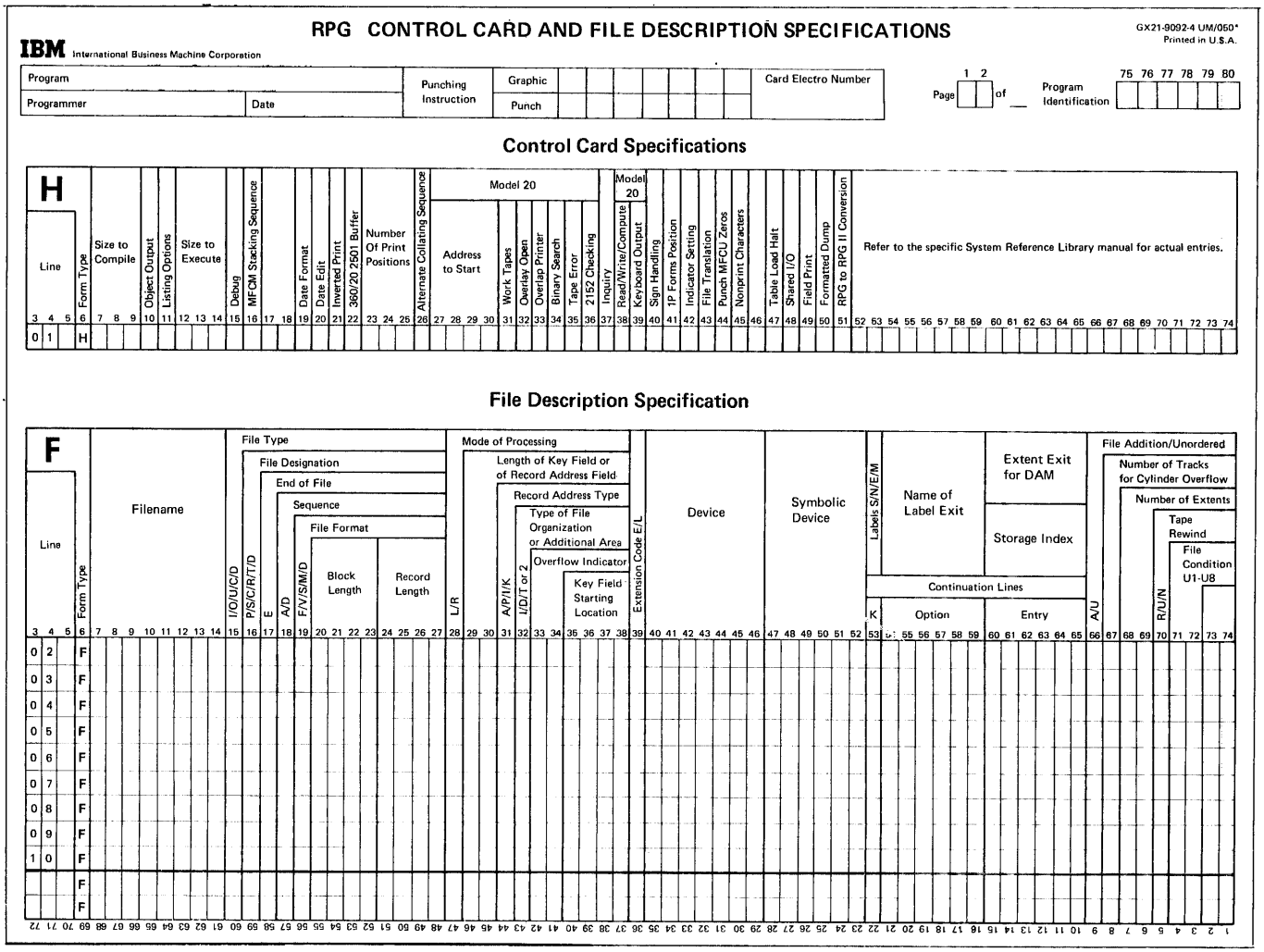


Figure 5. Control and File Description Specifications Sheet

COLUMNS 7-9 (SIZE TO COMPILE)

Columns 7-9 are not used. Leave them blank. Any entry in these columns is ignored by the compiler. The program is compiled in the available storage.

COLUMN 10 (OBJECT OUTPUT)

Entry	Explanation
Blank	The system halts only when severe (terminal) errors are found.
D	The system halts for both warning errors and severe errors. The operator can continue the job after a halt occurs for a warning error.

An object program is produced if no severe (terminal) errors are present in the source statements. This object program is written in the library and stays there until deleted by the programmer. Every object program written in the library must be assigned a unique program name in columns 75-80 of the control specifications sheet. (If no name is assigned, RPG II assigns the name RPGOBJ.)

COLUMN 11 (LISTING OPTIONS)

Entry	Explanation
Blank	The object program is produced (if no severe errors are found). The program listing is printed.
B	The object program is produced (if no severe errors are found). The program listing is not printed.
P	The object program is produced (if no severe errors are found). A partial program listing is printed, which includes the source program, information on indicator usage, and diagnostics.

Column 11 provides for listing options at the time your source program is compiled. If any severe errors are found during compilation, the listing is completed (provided a listing is to be printed) and the system halts.

The blank entry is the usual case, producing an object program (if no severe errors are found) and a source program listing. The RPG II listing consists of the source program listing, table and array information, indicator usage information, the relative location of fields and their attributes, unreferenced field names, diagnostics, and a main storage usage map. The main storage usage map lists the identification, the start address, and the size of each uniquely identifiable segment of code in the object program, and defines the amount of main storage required for execution.

The B entry means that no program listing is printed; however, an object program is produced. Use this entry if you want to produce an object program for which you already have a listing.

The P entry means that a partial listing is printed, which includes the source program, information concerning indicator usage, and diagnostics. Use this entry if you do not need a complete listing of the program. Excluded from this printout are table/array information, field information, and main storage usage map.

COLUMNS 12-14 (SIZE TO EXECUTE)

Column 12

Entry	Explanation
Blank, 0	No additional 256-byte increments are needed.
Q	One additional 256-byte increment is needed.
H	Two additional 256-byte increments are needed (512 bytes).
T	Three additional 256-byte increments are needed (768 bytes).

Use column 12 to specify additional 256-byte increments of storage. These increments allow an extra 1/4K, 1/2K, or 3/4K of storage to be available in addition to the storage specified in columns 13-14.

Columns 13-14

Entry	Explanation
Blank	The main storage available for object program execution defaults to system size minus the SCP resident nucleus size (2K).
01-system size minus 2K	The main storage available for program execution.

Use columns 13-14 to specify some multiple of 1K bytes of storage (K = 1,024 bytes).

Example

The following chart shows examples of the possible entries that can be made in columns 12-14 and the amount of storage that is made available for that entry:

Entry	Available Bytes
004	4,096
Q04	4,352 (4,096 + 256)
H04	4,608 (4,096 + 512)
T04	4,864 (4,096 + 768)
005	5,120

Subtract the amount of main storage occupied by the supervisor (2K) from the total main storage of the system used for execution to determine the maximum amount of storage available for execution.

COLUMN 15 (DEBUG)

Entry	Explanation
Blank	DEBUG operation is not performed.
1	DEBUG operation is performed.

Use column 15 to indicate whether or not the DEBUG operation is performed. To perform a DEBUG operation:

- A 1 must appear in column 15 when the source program is compiled.
- The DEBUG operation code must appear in calculation specifications.

See *Operation Codes, Debug Operation* under *Supplementary Information*.

COLUMNS 16-18

Columns 16-18 are not used. Leave them blank.

COLUMNS 19-20 (DATE OPTION)

Column 19 (Date Format)

Entry	Explanation
Blank	Default to month/day/year if column 21 is blank. Default to day/month/year if column 21 contains a D, I, or J.
M	Month/Day/Year.
D	Day/Month/Year.
Y	Year/Month/Day.

Use column 19 to specify the date format for UDATE. The date format should be in the same format as the system date.

Note: The input for UDATE must be in the format expected as output. For example, if D is specified in column 21, the input must be dd/mm/yy.

Column 20 (Date Edit)

Entry	Explanation
Any of the 256 characters including blank	Characters to be used in the edit pattern for the date. The & entry forces blank to be used as the separator. If blank is entered, (1) / (slash) is assumed when column 21 contains a blank or D and column 19 is blank, and (2) . (period) is assumed when column 21 contains I or J and column 19 is blank. If column 19 contains M, / is assumed. If column 19 contains D or Y, . is assumed.

Use column 20 to specify the type of edited output that will appear for the Y edit code.

COLUMN 21 (INVERTED PRINT)

Entry	Explanation
Blank	Decimal periods are used for numeric literals and editing. UDATE format is mmddyy if column 19 is blank. If columns 19 and 20 are blank, / is used for the Y edit code.
I	Decimal commas are used for numeric literals and editing. UDATE format is ddmmyy if column 19 is blank. If columns 19 and 20 are blank, period is used for the Y edit code.
J	This is the same as I except zero is written to the left of the decimal comma when the field contains a fraction. Nondecimal edited fields print with a zero in the low-order (units) position.
D	This is the same as blank – except the UDATE format is ddmmyy if column 19 is blank.

Use column 21 to specify the constants to be used with RPG II edit codes. Decimal period means that numbers are edited with a period before the fraction (183.55) and commas denoting thousands, etc (1,435). Decimal comma means that numbers are edited with a comma before the fraction (183,55) and periods denoting thousands, etc (1.435).

For information on how the entries in column 21 are used to format numeric data, see *Column 38 (Edit Codes)* under *Output Specifications*.

COLUMNS 22-25

Columns 22-25 are not used. Leave them blank.

COLUMN 26 (ALTERNATE COLLATING SEQUENCE)

Entry	Explanation
Blank	Normal collating sequence is used.
S	Alternate collating sequence is used.

Use column 26 only if you are altering the normal collating sequence. The following text contains a description of the specifications required to alter the normal collating sequence.

Alternate Collating Sequence

Every alphabetic, numeric, or special character holds a special position in relation to all other characters. This special order is known as the collating sequence. System/32 uses a collating sequence based on the way characters are represented in the machine (Figure 6).

You may change this collating sequence if you wish. If you want characters to appear in a sequence other than the one used by System/32 or if you want two or more characters to have the same position in the sequence (this means they are considered equal), you must describe an alternate collating sequence. The alternate collating sequence table is printed with the compiled program. (Figure 2 shows the order of the RPG II source program including an alternate collating sequence table.)

Note: An alternate collating sequence applies to:

- Matching fields and sequence checking
- Alphameric compare operations (COMP)

Defining Alternate Collating Sequence

To define an alternate collating sequence, you must first indicate that a sequence other than the normal one is to be used. Do this by entering an S in column 26 of the control specifications sheet.

A table also must be entered which lists the changes you want to make in the normal collating sequence. This is a special table and requires no file description or extension entries. The following entries are needed for each table record entered:

Collating Sequence	Character	Hexadecimal Equivalent
1	Blank	40
2	¢	4A
3	.	4B
4	<	4C
5	(	4D
6	+	4E
7	|	4F
8	&	50
9	↑	5A
10	$	5B
11	*	5C
12	)	5D
13	;	5E
14	¬	5F
15	- (minus)	60
16	/	61
17	,	6B
18	%	6C
19	_ (underscore)	6D
20	>	6E
21	?	6F
22	:	7A
23	#	7B
24	@	7C
25	'	7D
26	=	7E
27	"	7F
28	A	C1
29	B	C2
30	C	C3
31	D	C4
32	E	C5

Collating Sequence	Character	Hexadecimal Equivalent
33	F	C6
34	G	C7
35	H	C8
36	I	C9
37		D0
38	J	D1
39	K	D2
40	L	D3
41	M	D4
42	N	D5
43	O	D6
44	P	D7
45	Q	D8
46	R	D9
47	S	E2
48	T	E3
49	U	E4
50	V	E5
51	W	E6
52	X	E7
53	Y	E8
54	Z	E9
55	0	F0
56	1	F1
57	2	F2
58	3	F3
59	4	F4
60	5	F5
61	6	F6
62	7	F7
63	8	F8
64	9	F9

Figure 6. Normal Collating Sequence and Hexadecimal Equivalents of Characters

Positions 1-6: Enter ALTSEQ to indicate that you are altering the normal sequence.

Positions 7-8: Leave these positions blank.

Positions 9-10: Enter the hexadecimal equivalent of the character being taken out of sequence. The table in Figure 6 lists characters and their hexadecimal equivalents.

Positions 11-12: Enter the hexadecimal equivalent of the character that is replacing the character taken out of sequence.

Positions 13-16, 17-20, 21-24, etc: These positions are used the same way positions 9-12 are used. The first two positions give the character to be replaced by the character specified in the next two positions. There may be as many 4-position entries as can be contained in the record. Additional records may be used with the above format. The first blank position terminates the record. An **b in positions 1-3 ends the table.

The alternate sequence table records must be preceded by a record with **b in positions 1-3. The remaining positions of the record may be used for comments. The alternate collating sequence records must follow the RPG II specifications and file translation records, if used. Figure 2 shows the arrangement of records in an RPG II source program.

Translation Table and Alternate Collating Sequence Coding Sheet

The translation table and alternate collating sequence sheet (Figure 7) can be used for coding an alternate collating sequence. It helps you to determine the entries needed for the alternate collating sequence table input records.

Causing Characters To Be Considered Equal

If you want one character to be considered the same as another character, the characters must hold the same position in the collating sequence. For example, you may want a blank to be considered as a zero. Therefore, you need to define an alternate collating sequence in which the blank is the same as the zero because it holds the same position in the sequence. The alternate collating sequence input record for this example looks like this:

Record Position	Entry
1-6	ALTSEQ
7-8	Blanks
9-12	40F0 (blank takes the zero's position)

Now whenever a blank is read and used in a compare, it is considered as a zero. Thus, if you were comparing numbers to 0036 to find an equal condition, 0036 and bb36 (where b = blank) both compare to 0036.

Altering the Normal Collating Sequence

You can alter the normal collating sequence in a number of ways. For example, you can insert a character between two existing characters, you can take a character out of the sequence, or you can change characters (put A where Z is and Z where A is). Regardless of how you alter the sequence, you must specify every character that is to be changed by the alteration. For example, if you want the dollar sign ($) to be positioned in the collating sequence between A and B, the normal sequence changed as follows:

Normal Sequence	Altered Sequence
A	A
B	$
C	B
D	C
E	D
F	E
G	F
H	G
I	H
	I

Notice on the translation table and alternate collating sequence coding sheet that there are many characters between I and J, R and S, and Z and 0. These characters can be represented in the computer and on records by a certain code. However, they have no printable graphic symbol. Due to this particular arrangement of graphics

IBM

International Business Machines Corporation

GX21-9096-1
Printed in U.S.A.

TRANSLATION TABLE AND ALTERNATE COLLATING SEQUENCE CODING SHEET

Code	Graphic	Entry	Replaced By/Takes Place Of
00000000		00	
00000001		01	
00000010		02	
00000011		03	
00000100		04	
00000101		05	
00000110		06	
00000111		07	
00001000		08	
00001001		09	
00001010		0A	
00001011		0B	
00001100		0C	
00001101		0D	
00001110		0E	
00001111		0F	
00010000		10	
00010001		11	
00010010		12	
00010011		13	
00010100		14	
00010101		15	
00010110		16	
00010111		17	
00011000		18	
00011001		19	
00011010		1A	
00011011		1B	
00011100		1C	
00011101		1D	
00011110		1E	
00011111		1F	
00100000		20	
00100001		21	
00100010		22	
00100011		23	
00100100		24	
00100101		25	
00100110		26	
00100111		27	
00101000		28	
00101001		29	
00101010		2A	
00101011		2B	
00101100		2C	
00101101		2D	
00101110		2E	
00101111		2F	
00110000		30	
00110001		31	
00110010		32	

Code	Graphic	Entry	Replaced By/Takes Place Of
00110011		33	
00110100		34	
00110101		35	
00110110		36	
00110111		37	
00111000		38	
00111001		39	
00111010		3A	
00111011		3B	
00111100		3C	
00111101		3D	
00111110		3E	
00111111		3F	
01000000	Blank	40	
01000001		41	
01000010		42	
01000011		43	
01000100		44	
01000101		45	
01000110		46	
01000111		47	
01001000		48	
01001001		49	
01001010	¢	4A	
01001011	.	4B	
01001100	<	4C	
01001101	(	4D	
01001110	+	4E	
01001111	|	4F	
01010000	&	50	
01010001		51	
01010010		52	
01010011		53	
01010100		54	
01010101		55	
01010110		56	
01010111		57	
01011000		58	
01011001		59	
01011010	!	5A	
01011011	$	5B	
01011100	*	5C	
01011101	)	5D	
01011110	;	5E	
01011111	¬	5F	
01100000	-	60	
01100001	/	61	
01100010		62	
01100011		63	
01100100		64	
01100101		65	

Code	Graphic	Entry	Replaced By/Takes Place Of
01100110		66	
01100111		67	
01101000		68	
01101001		69	
01101010		6A	
01101011	,	6B	
01101100	%	6C	
01101101	—	6D	
01101110	>	6E	
01101111	?	6F	
01110000		70	
01110001		71	
01110010		72	
01110011		73	
01110100		74	
01110101		75	
01110110		76	
01110111		77	
01111000		78	
01111001		79	
01111010	:	7A	
01111011	#	7B	
01111100	@	7C	
01111101	'	7D	
01111110	=	7E	
01111111	"	7F	
10000000		80	
10000001		81	
10000010		82	
10000011		83	
10000100		84	
10000101		85	
10000110		86	
10000111		87	
10001000		88	
10001001		89	
10001010		8A	
10001011		8B	
10001100		8C	
10001101		8D	
10001110		8E	
10001111		8F	
10010000		90	
10010001		91	
10010010		92	
10010011		93	
10010100		94	
10010101		95	
10010110		96	
10010111		97	
10011000		98	

Code	Graphic	Entry	Replaced By/Takes Place Of
10011001		99	
10011010		9A	
10011011		9B	
10011100		9C	
10011101		9D	
10011110		9E	
10011111		9F	
10100000		A0	
10100001		A1	
10100010		A2	
10100011		A3	
10100100		A4	
10100101		A5	
10100110		A6	
10100111		A7	
10101000		A8	
10101001		A9	
10101010		AA	
10101011		AB	
10101100		AC	
10101101		AD	
10101110		AE	
10101111		AF	
10110000		B0	
10110001		B1	
10110010		B2	
10110011		B3	
10110100		B4	
10110101		B5	
10110110		B6	
10110111		B7	
10111000		B8	
10111001		B9	
10111010		BA	
10111011		BB	
10111100		BC	
10111101		BD	
10111110		BE	
10111111		BF	
11000000		C0	
11000001	A	C1	
11000010	B	C2	
11000011	C	C3	
11000100	D	C4	
11000101	E	C5	
11000110	F	C6	
11000111	G	C7	
11001000	H	C8	
11001001	I	C9	
11001010		CA	
11001011		CB	

Code	Graphic	Entry	Replaced By/Takes Place Of
11001100		CC	
11001101		CD	
11001110		CE	
11001111		CF	
11010000	}	D0	
11010001	J	D1	
11010010	K	D2	
11010011	L	D3	
11010100	M	D4	
11010101	N	D5	
11010110	O	D6	
11010111	P	D7	
11011000	Q	D8	
11011001	R	D9	
11011010		DA	
11011011		DB	
11011100		DC	
11011101		DD	
11011110		DE	
11011111		DF	
11100000		E0	
11100001		E1	
11100010	S	E2	
11100011	T	E3	
11100100	U	E4	
11100101	V	E5	
11100110	W	E6	
11100111	X	E7	
11101000	Y	E8	
11101001	Z	E9	
11101010		EA	
11101011		EB	
11101100		EC	
11101101		ED	
11101110		EE	
11101111		EF	
11110000	0	F0	
11110001	1	F1	
11110010	2	F2	
11110011	3	F3	
11110100	4	F4	
11110101	5	F5	
11110110	6	F6	
11110111	7	F7	
11111000	8	F8	
11111001	9	F9	
11111010		FA	
11111011		FB	
11111100		FC	
11111101		FD	
11111110		FE	
11111111		FF	

Figure 7. Translation Table and Alternate Collating Sequence Coding Sheet

or nongraphics in the collating sequence, a character, when inserted between A and B, changes only the position of graphics B-I. All other graphics are not affected. B-I all move down one position causing the I to take the place of the nongraphic represented by hexadecimal CA. This does not matter, however, since the original character CA cannot be printed anyway. See Figure 8 for the entries on the translation table and alternate collating sequence coding sheet.

International Business Machines Corporation

GX21-9096-
Printed in U.

TRANSLATION TABLE AND ALTERNATE COLLATING SEQUENCE CODING SHEET

Code	Graphic	Entry	Replaced By/Takes Place Of
00110011		33	
00110100		34	
00110101		35	
00110110		36	
00110111		37	
00111000		38	
00111001		39	
00111010		3A	
00111011		3B	
00111100		3C	
00111101		3D	
00111110		3E	
00111111		3F	
01000000	Blank	40	
01000001		41	
01000010		42	
01000011		43	
01000100		44	
01000101		45	
01000110		46	
01000111		47	
01001000		48	
01001001		49	
01001010	¢	4A	
01001011	.	4B	
01001100	<	4C	
01001101	(	4D	
01001110	+	4E	
01001111	|	4F	
01010000	&	50	
01010001		51	
01010010		52	
01010011		53	
01010100		54	
01010101		55	
01010110		56	
01010111		57	
01011000		58	
01011001		59	
01011010	!	5A	
01011011	$	5B	C2(B)
01011100	*	5C	
01011101	)	5D	
01011110	;	5E	
01011111	¬	5F	
01100000	-	60	
01100001	/	61	
01100010		62	
01100011		63	
01100100		64	
01100101		65	

$ takes B's position.

Code	Graphic	Entry	Replaced By/Takes Place Of
01100110		66	
01100111		67	
01101000		68	
01101001		69	
01101010		6A	
01101011	,	6B	
01101100	%	6C	
01101101	–	6D	
01101110	>	6E	
01101111	?	6F	
01110000		70	
01110001		71	
01110010		72	
01110011		73	
01110100		74	
01110101		75	
01110110		76	
01110111		77	
01111000		78	
01111001		79	
01111010	:	7A	
01111011	#	7B	
01111100	@	7C	
01111101	'	7D	
01111110	=	7E	
01111111	"	7F	
10000000		80	
10000001		81	
10000010		82	
10000011		83	
10000100		84	
10000101		85	
10000110		86	
10000111		87	
10001000		88	
10001001		89	
10001010		8A	
10001011		8B	
10001100		8C	
10001101		8D	
10001110		8E	
10001111		8F	
10010000		90	
10010001		91	
10010010		92	
10010011		93	
10010100		94	
10010101		95	
10010110		96	
10010111		97	
10011000		98	

Code	Graphic	Entry	Replaced By/Takes Place Of
10011001		99	
10011010		9A	
10011011		9B	
10011100		9C	
10011101		9D	
10011110		9E	
10011111		9F	
10100000		A0	
10100001		A1	
10100010		A2	
10100011		A3	
10100100		A4	
10100101		A5	
10100110		A6	
10100111		A7	
10101000		A8	
10101001		A9	
10101010		AA	
10101011		AB	
10101100		AC	
10101101		AD	
10101110		AE	
10101111		AF	
10110000		B0	
10110001		B1	
10110010		B2	
10110011		B3	
10110100		B4	
10110101		B5	
10110110		B6	
10110111		B7	
10111000		B8	
10111001		B9	
10111010		BA	
10111011		BB	
10111100		BC	
10111101		BD	
10111110		BE	
10111111		BF	
11000000		C0	
11000001	A	C1	
11000010	B	C2	C3(C)
11000011	C	C3	C4(D)
11000100	D	C4	C5(E)
11000101	E	C5	C6(F)
11000110	F	C6	C7(G)
11000111	G	C7	C8(H)
11001000	H	C8	C9(I)
11001001	I	C9	CA
11001010		CA	
11001011		CB	

B takes C's position.

(no printable character)

Code	Graphic	Entry	Replaced By/Takes Place Of
11001100		CC	
11001101		CD	
11001110		CE	
11001111		CF	
11010000	}	D0	
11010001	J	D1	
11010010	K	D2	
11010011	L	D3	
11010100	M	D4	
11010101	N	D5	
11010110	O	D6	
11010111	P	D7	
11011000	Q	D8	
11011001	R	D9	
11011010		DA	
11011011		DB	
11011100		DC	
11011101		DD	
11011110		DE	
11011111		DF	
11100000		E0	
11100001		E1	
11100010	S	E2	
11100011	T	E3	
11100100	U	E4	
11100101	V	E5	
11100110	W	E6	
11100111	X	E7	
11101000	Y	E8	
11101001	Z	E9	
11101010		EA	
11101011		EB	
11101100		EC	
11101101		ED	
11101110		EE	
11101111		EF	
11110000	0	F0	
11110001	1	F1	
11110010	2	F2	
11110011	3	F3	
11110100	4	F4	
11110101	5	F5	
11110110	6	F6	
11110111	7	F7	
11111000	8	F8	
11111010		FA	
11111011		FB	
11111100		FC	
11111101		FD	
		FE	
		FF	

C takes D's position.

Figure 8. Altering the Collating Sequence

The alternate sequence input record is as follows:

Record Position	Entry
1-6	ALTSEQ
7-8	Blanks
9-12	5BC2 ($ takes B's position)
13-16	C2C3 (B takes C's position)
17-20	C3C4 (C takes D's position)
21-24	C4C5 (D takes E's position)
25-28	C5C6 (E takes F's position)
29-32	C6C7 (F takes G's position)
33-36	C7C8 (G takes H's position)
37-40	C8C9 (H takes I's position)
41-44	C9CA (I is given a new position held by no other printable character)

COLUMNS 27-36

Columns 27-36 are not used. Leave them blank.

COLUMN 37 (INQUIRY)

Entry	Explanation
Blank or I	Program cannot be interrupted (does not recognize an inquiry request).
B	Program can be interrupted (recognizes an inquiry request).

The system allows certain programs to be interrupted while they are executing. A request for interruption is called an inquiry request (made by pressing the INQ key on the keyboard). Programs are usually interrupted to permit another program to execute. Control is then given back to the first program. The OCL or command statements for the program must be entered from the keyboard.

A blank or I entry in column 37 indicates that the program cannot be interrupted (does not recognize an inquiry request). An entry of B in column 37 indicates that the program can be interrupted (recognizes an inquiry request). The program to be loaded following an inquiry request can have an I, B, or blank in column 37. However, if the program being loaded is a B-type program, it cannot be interrupted by another inquiry request.

A B-type program recognizes an inquiry request in the following manner:

1. The inquiry key is pressed and the 1 option is selected in response to the corresponding halt. This indicates to the RPG program that an inquiry request is pending.

2. When the program recognizes an inquiry request (at the start or the end of the RPG cycle), a rollout routine moves the interrupted program from main storage to disk.

3. The program for which the interrupt was requested must be loaded by entering the required OCL or command statement from the keyboard. This interrupting program cannot be interrupted by an inquiry request.

4. After the interrupting program is executed, the interrupted program is brought back into main storage by a rollin routine. The interrupted program begins execution at the point of interruption and terminates in a normal manner.

CAUTION
If you are using inquiry, *do not* change any files that were being used by the interrupted (rolled-out) program. System/32 system control programming does not always check for duplicate file labels in the inquiry and interrupted programs. For example, program X is interrupted while it is loading file A. File A is then deleted using inquiry. A return to program X will cause unpredictable results.

For more information on inquiry, including restrictions on the use of system utilities in inquiry mode, see *Inquiry Option* in *IBM System/32 System Control Programming Reference Manual,* GC21-7593.

Note: An inquiry request can also be made by using IBM-written subroutine SUBR95 instead of inquiry. For information on this method, see *Supplementary Information,* Part 3, *IBM-Written Subroutines.*

COLUMNS 38-40

Columns 38-40 are not used. Leave them blank.

COLUMN 41 (1P FORMS POSITION)

Entry	Explanation
Blank	First 1P line is printed only once.
1	First 1P line can be printed repeatedly.

Use column 41 only when the first output line conditioned by the first page (1P) indicator is written to a printer file.

When forms are first inserted in the printer, they may not be in perfect alignment. Sometimes several lines must be printed to determine the correct positioning of the form. You may not want to print several lines of your report before you get the forms positioned correctly. In this case, you have the option of repeatedly printing the first line conditioned by the first page (1P) indicator by selecting option 1 and pressing the ENTER key. Each time the 1P line is printed, the program halts so you can reposition the forms if needed. When the forms are positioned correctly, the operator can continue the program by selecting option 0 and pressing the ENTER key. The page counter is not incremented until after the forms have been positioned correctly.

COLUMN 42

Column 42 is not used. Leave it blank.

COLUMN 43 (FILE TRANSLATION)

Entry	Explanation
Blank	No file translation is needed.
F	Input, output, update, or combined files are to be translated.

Use column 43 only when information contained in an input, output, update, or combined file is in a form not usable by your program. When file translation is specified for an update, both the input and output portion of the file is translated.

An F in column 43 indicates either or both of the following: (1) the character code used in the input data must be translated into a form that can be used by your program, or (2) the output data must be in a character code different from that used by your program.

The specifications for forming a file translation table are described in the following text.

File Translation

At some time you may want to use data that is in a character code different from the character code used by System/32. RPG II allows you to translate a different character code used as input into the System/32 code. You can also have the System/32 code translated for output into a different character code. This capability is called file translation.

Specify file translation by entering file translation table records. This is a special table and requires no file description or extension specifications. The file translation records must immediately follow the RPG II specifications in the source program (Figure 2). The file translation table is printed with the compiled program.

You can specify file translation for input, output, update, and combined files. For input files, a different character code is translated into the System/32 code. For output files, the System/32 code is translated into a different character code. For update and combined files, the input data in a different code is translated into the System/32 code, then translated back to the different code for output.

In the following text, any character represented by the System/32 code is called as an internal character; any character represented by a different code is called as an external character.

Specifications for File Translation

You must first indicate that there are files to be translated. Do this by entering an F in column 43 of the RPG II control specifications sheet. Use table input records to specify how the translation is to be done. The following entries are needed for each file translation table input record used.

Positions 1-6: Enter *FILES to indicate that all input, output, update, and combined files are to undergo translation (both the input and output portions of the update and combined files are translated). Then complete your file translation input record by making the entries listed in the following text, beginning with positions 9-10. All files are translated according to the table specified beginning in position 9.

If only certain files are to be translated, they must be named individually in positions 1-8. (The *FILES entry is not made in positions 1-6.)

Positions 1-8: Enter the filename of the input, output, combined, or update file to be translated (both the input and output portions of update files and combined are translated). Then use the specifications listed, beginning with positions 9-10.

Positions 9-10: Enter the hexadecimal equivalent of the external character. This is the character in a different character code to be translated from input data or for output data.

Positions 11-12: Enter the hexadecimal equivalent of the internal character. This is the character in the System/32 code which represents internally the external input or output character.

Positions 13-16, 17-20, and 21-24, etc: These groups of positions are used the same way as positions 9-12 are used. For instance, columns 13-14 contain the hexadecimal equivalent of the external character, and columns 15-16 contain the hexadecimal equivalent of the related internal character.

All table records for one file must be kept together. The file translation table input records must be preceded by one record with **b (b means blank) in positions 1-3. The remaining positions of this record can be used for comments.

Example

Assume that while working for a department store, you must process sales slips for all items sold. Each sales slip contains a printed record of the actual, or wholesale, cost of its associated item along with a retail price.

Obviously, wholesale cost must remain confidential, so the store uses individual letters of a code name in place of numbers comprising wholesale costs.

A typical code name generally consists of a combination of letters that can be easily remembered by the store's personnel. The only restriction, however, is that the code name must contain 10 different letters, one for each of the numbers zero through nine.

Using the code name BUCKINGHAM to represent numbers one through nine and zero, the letter B represents the number 1; letter U represents number 2, etc. Letter M represents zero. Individual letters are combined to represent each item's wholesale cost. Thus a wholesale cost of BBU.CC translates as $112.33.

In the following chart, hexadecimal equivalents of each letter in the word BUCKINGHAM are listed along with the hexadecimal equivalents of numbers one through nine and zero.

Letter in Code Name (External Character)	Hexadecimal Equivalent	Number	Hexadecimal Equivalent (Internal Character)
B	C2	1	F1
U	E4	2	F2
C	C3	3	F3
K	D2	4	F4
I	C9	5	F5
N	D5	6	F6
G	C7	7	F7
H	C8	8	F8
A	C1	9	F9
M	D4	0	F0

Hexadecimal equivalents are merely a different way of representing the 8-bit code that the computer examines to recognize individual characters in your language.

See Figure 9. Note that if letters BBU were read and never translated, hexadecimal equivalents C2, C2, and E4 are used by System/32. As a result, it is impossible to perform an arithmetic operation involving the wholesale cost, BBU. Therefore, with the aid of file translation, the computer replaces the letters BBU with numbers.

International Business Machines Corporation — Form X21-9096 Printed in U.S.A.

TRANSLATION TABLE AND ALTERNATE COLLATING SEQUENCE CODING SHEET

Entry	Replaced By/Takes Place Of	Code	Graphic	Entry	Replaced By/Takes Place Of	Code	Graphic	Entry	Replaced By/Takes Place Of	Code	Graphic	Entry	Replaced By/Takes Place Of
33		01100110		66		10011001		99		11001100		CC	
34		01100111		67		10011010		9A		11001101		CD	
35		01101000		68		10011011		9B		11001110		CE	
36		01101001		69		10011100		9C		11001111		CF	
37		01101010		6A		10011101		9D		11010000	}	D0	
38		01101011	,	6B		10011110		9E		11010001	J	D1	
39		01101100	%	6C		10011111		9F		11010010	K	D2	
3A		01101101	—	6D		10100000		A0		11010011	L	D3	
3B		01101110	>	6E		10100001		A1		11010100	M	D4	
3C		01101111	?	6F		10100010		A2		11010101	N	D5	
3D		01110000		70		10100011		A3		11010110	O	D6	
3E		01110001		71		10100100		A4		11010111	P	D7	
3F		01110010		72		10100101		A5		11011000	Q	D8	
40		01110011		73		10100110		A6		11011001	R	D9	
41		01110100		74		10100111		A7		11011010		DA	
42		01110101		75		10101000		A8		11011011		DB	
43		01110110		76		10101001		A9		11011100		DC	
44		01110111		77		10101010		AA		11011101		DD	
45		01111000		78		10101011		AB		11011110		DE	
46		01111001		79		10101100		AC		11011111		DF	
47		01111010	:	7A		10101101		AD		11100000		E0	
48		01111011	#	7B		10101110		AE		11100001		E1	
49		01111100	@	7C		10101111		AF		11100010	S	E2	
4A		01111101	'	7D		10110000		B0		11100011	T	E3	
4B		01111110	=	7E		10110001		B1		11100100	U	E4	
4C		01111111	"	7F		10110010		B2		11100101	V	E5	
4D		10000000		80		10110011		B3		11100110	W	E6	
4E		10000001		81		10110100		B4		11100111	X	E7	
4F		10000010		82		10110101		B5		11101000	Y	E8	
50		10000011		83		10110110		B6		11101001	Z	E9	
51		10000100		84		10110111		B7		11101010		EA	
52		10000101		85		10111000		B8		11101011		EB	
53		10000110		86		10111001		B9		11101100		EC	
54		10000111		87		10111010		BA		11101101		ED	
55		10001000		88		10111011		BB		11101110		EE	
56		10001001		89		10111100		BC		11101111		EF	
57		10001010		8A		10111101		BD		11110000	0	F0	
58		10001011		8B		10111110		BE		11110001	1	F1	
59		10001100		8C		10111111		BF		11110010	2	F2	
5A		10001101		8D		11000000		C0		11110011	3	F3	
5B		10001110		8E		11000001	A	C1		11110100	4	F4	
5C		10001111		8F		11000010	B	C2		11110101	5	F5	
5D		10010000		90		11000011	C	C3		11110110	6	F6	
5E		10010001		91		11000100	D	C4		11110111	7	F7	
5F		10010010		92		11000101	E	C5		11111000	8	F8	
60		10010011		93		11000110	F	C6		11111001	9	F9	
61		10010100		94		11000111	G	C7		11111010		FA	
62		10010101		95		11001000	H	C8		11111011		FB	
63		10010110		96		11001001	I	C9		11111100		FC	
64		10010111		97		11001010		CA		11111101		FD	
65		10011000		98		11001011		CB		11111110		FE	
										11111111		FF	

E4, which, if translated, represents the number 2, is the letter U in the code used by System/32.

C2, which, if translated, represents the number 1, is the letter B in the code used by System/32.

Figure 9. Differences in Character Codes

The file translation specifications for letters in the word BUCKINGHAM are:

Record Position	Entry
1-6	*FILES
7-8	Blank
9-12	C2F1
13-16	E4F2
17-20	C3F3
21-24	D2F4
25-28	C9F5
29-32	D5F6
33-36	C7F7
37-40	C8F8
41-44	C1F9
45-48	D4F0

Only the letters of the previous example are specified for translation. All other characters are handled in the normal manner. Remember that these letters are translated for all fields in the file, not just the code name, BUCKINGHAM.

Translation Table and Alternate Collating Sequence Coding Sheet

This coding sheet is helpful for determining the correct entries you wish to make in the file translation table input record. Figure 10 shows the entries made on the sheet for the previous example.

COLUMN 44

Column 44 is not used. Leave it blank.

COLUMN 45 (NONPRINT CHARACTERS)

Entry	Explanation
Blank	Program halts if an unprintable character was in the last line printed.
1	Program does not halt for unprintable characters.

Use column 45 to bypass machine halts for unprintable characters. This column applies only to the printer.

All characters are known to the system by a numeric code. If a numeric code is formed that is not in your system (not in your character set) and that character is to be printed, the machine halts after printing the line. The unprintable characters are replaced with blanks.

To bypass this halt, enter a one (1) in column 45. An unprintable character is printed as a blank and no halt occurs. Note, however, that the option could make your output meaningless (for example, when printing a packed key field or a nonprintable field built by calculation specifications).

COLUMNS 46-47

Columns 46-47 are not used. Leave them blank.

COLUMN 48 (SHARED I/O)

Entry	Explanation
1	All disk files share a single input/output area.
Blank	All disk files use a separate input/output area.

Normally, an RPG II program uses one input/output area for each file. An entry in column 48 allows all disk files to use one input/output area. By specifying a shared input/output area, you can reduce the amount of main storage needed to process a program. This is particularly important if a program is so large that it cannot run in the main storage available. However, the use of a shared input/output area can increase the time required to process your program. Therefore, before you indicate that all disk files are to share one input/output area, be sure that the program would otherwise exceed the capacity of the system.

Note: Additional input/output areas (entry in column 32 of the file description specifications sheet) cannot be specified for disk files using a shared input/output area.

Columns 49-74

Columns 49-74 are not used. Leave them blank.

COLUMNS 75-80 (PROGRAM IDENTIFICATION)

See *Common Entries*.

International Business Machines Corporation

GX21-9096-1
Printed in U.S.A.

TRANSLATION TABLE AND ALTERNATE COLLATING SEQUENCE CODING SHEET

ıde	Graphic	Entry	Replaced By/Takes Place Of
ı110011		33	
ı110100		34	
ı110101		35	
ı110110		36	
ı110111		37	
ı111000		38	
ı111001		39	
ı111010		3A	
ı111011		3B	
ı111100		3C	
ı111101		3D	
ı111110		3E	
ı111111		3F	
000000	Blank	40	
000001		41	
000010		42	
000011		43	
000100		44	
000101		45	
000110		46	
000111		47	
001000		48	
001001		49	
001010	¢	4A	
001011	.	4B	
001100	<	4C	
001101	(	4D	
001110	+	4E	
001111	|	4F	
010000	&	50	
010001		51	
010010		52	
010011		53	
010100		54	
010101		55	
010110		56	
010111		57	
011000		58	
011001		59	
011010	!	5A	
011011	$	5B	
011100	*	5C	
011101	)	5D	
011110	;	5E	
011111	¬	5F	
100000	-	60	
100001	/	61	
100010		62	
100011		63	
100100		64	
100101		65	

Code	Graphic	Entry	Replaced By/Takes Place Of
01100110		66	
01100111		67	
01101000		68	
01101001		69	
01101010		6A	
01101011	,	6B	
01101100	%	6C	
01101101	_	6D	
01101110	>	6E	
01101111	?	6F	
01110000		70	
01110001		71	
01110010		72	
01110011		73	
01110100		74	
01110101		75	
01110110		76	
01110111		77	
01111000		78	
01111001		79	
01111010	:	7A	
01111011	#	7B	
01111100	@	7C	
01111101	'	7D	
01111110	=	7E	
01111111	"	7F	
10000000		80	
10000001		81	
10000010		82	
10000011		83	
10000100		84	
10000101		85	
10000110		86	
10000111		87	
10001000		88	
10001001		89	
10001010		8A	
10001011		8B	
10001100		8C	
10001101		8D	
10001110		8E	
10001111		8F	
10010000		90	
10010001		91	
10010010		92	
10010011		93	
10010100		94	
10010101		95	
10010110		96	
10010111		97	
10011000		98	

Code	Graphic	Entry	Replaced By/Takes Place Of
10011001		99	
10011010		9A	
10011011		9B	
10011100		9C	
10011101		9D	
10011110		9E	
10011111		9F	
10100000		A0	
10100001		A1	
10100010		A2	
10100011		A3	
10100100		A4	
10100101		A5	
10100110		A6	
10100111		A7	
10101000		A8	
10101001		A9	
10101010		AA	
10101011		AB	
10101100		AC	
10101101		AD	
10101110		AE	
10101111		AF	
10110000		B0	
10110001		B1	
10110010		B2	
10110011		B3	
10110100		B4	
10110101		B5	
10110110		B6	
10110111		B7	
10111000		B8	
10111001		B9	
10111010		BA	
10111011		BB	
10111100		BC	
10111101		BD	
10111110		BE	
10111111		BF	
11000000		C0	
11000001	A	C1	F9
11000010	B	C2	F1
11000011	C	C3	F3
11000100	D	C4	
11000101	E	C5	
11000110	F	C6	
11000111	G	C7	F7
11001000	H	C8	F8
11001001	I	C9	F5
11001010		CA	
11001011		CB	

Code	Graphic	Entry	Replaced By/Takes Place Of
11001100		CC	
11001101		CD	
11001110		CE	
11001111		CF	
11010000	}	D0	
11010001	J	D1	
11010010	K	D2	F4
11010011	L	D3	
11010100	M	D4	F0
11010101	N	D5	F6
11010110	O	D6	
11010111	P	D7	
11011000	Q	D8	
11011001	R	D9	
11011010		DA	
11011011		DB	
11011100		DC	
11011101		DD	
11011110		DE	
11011111		DF	
11100000		E0	
11100001		E1	
11100010	S	E2	
11100011	T	E3	
11100100	U	E4	F2
11100101	V	E5	
11100110	W	E6	
11100111	X	E7	
11101000	Y	E8	
11101001	Z	E9	
11101010		EA	
11101011		EB	
11101100		EC	
11101101		ED	
11101110		EE	
11101111		EF	
11110000	0	F0	
11110001	1	F1	
11110010	2	F2	
11110011	3	F3	
11110100	4	F4	
11110101	5	F5	
11110110	6	F6	
11110111	7	F7	
11111000	8	F8	
11111001	9	F9	
11111010		FA	
11111011		FB	
11111100		FC	
11111101		FD	
11111110		FE	
11111111		FF	

This is the hexadecimal equivalent of the character to be translated.

This is the hexadecimal equivalent of the System/32 character that will be substituted for the character that is to be translated.

Figure 10. Specifications for File Translation Input Records

Chapter 3. File Description Specifications

File description specifications are required for every file used by a program. Write these specifications on the control and file description specifications sheet (Figure 11). Only one line is needed to describe a file. A maximum of 20 file description records are allowed per program. (The maximum number of file description records may be less if BSCA files are specified. See *IBM System/32 RPG II Telecommunications Programming Reference Manual,* SC21-7597.)

A series of charts at the end of this chapter show all possible files that can be defined on the file description specifications sheet (Figures 28 through 36). The charts are arranged by device, showing the basic entries for all possible disk, keyboard, console, printer, and display screen files.

COLUMNS 1-2 (PAGE)

See *Common Entries*.

COLUMNS 3-5 (LINE)

See *Common Entries*.

COLUMN 6 (FORM TYPE)

An F must appear in column 6.

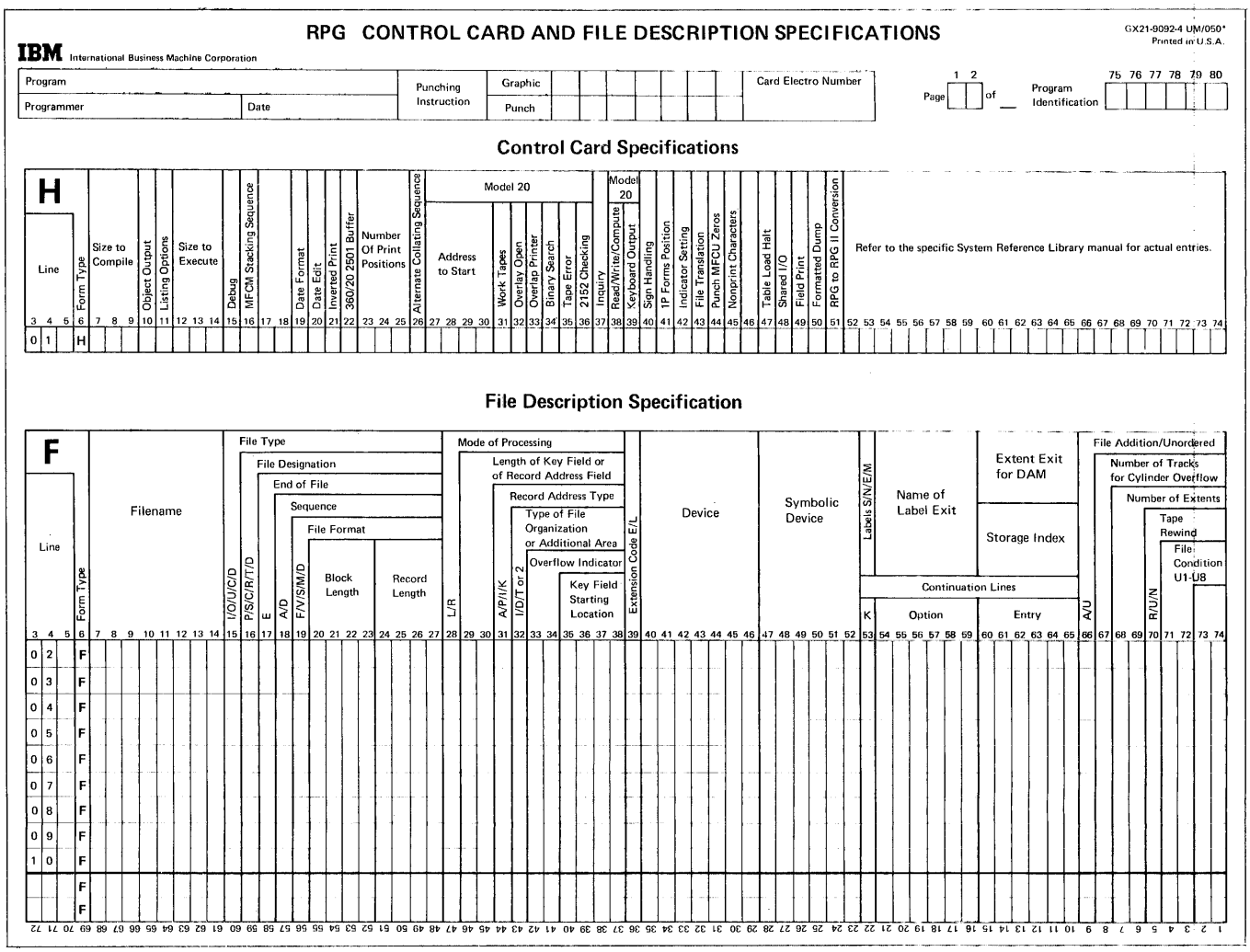


Figure 11. Control and File Description Specifications Sheet

COLUMNS 7-14 (FILENAME)

Use columns 7-14 to assign a unique filename to every file used in your program. Every file must be named with the following exceptions:

- Compile-time tables and arrays do not require a filename.
- If multiple tables or arrays are read in at preexecution-time from the same device, multiple filenames are required.

Note: All tables and arrays must be defined by the extension specifications.

The filename can be from 1-8 characters long, and must begin in column 7. The first character must be an alphabetic character. The remaining characters can be any combination of alphabetic and numeric characters (special characters are not allowed). Blanks cannot appear between characters in the filename.

COLUMN 15 (FILE TYPE)

Entry	Explanation
I	Input file
O	Output file
U	Update file
C	Combined file

Input Files

Input files contain records that a program uses as a source of data. When input files are described in a program, the program indicates that records are to be read from the file. All input files must be further described by input specifications with the following exceptions:

- Preexecution-time tables and arrays and record address files are described by extension specifications. There is, however, a method of loading arrays using extension and input specifications. (See *Supplementary Information, Arrays* for complete information.)
- Input files using the device name KEYBORD are further described by calculation specifications when the KEY operation is used.

Output Files

Output files contain records written or printed by a program. All output files, except table output files, must be further described by output specifications. Output table files are further described by extension specifications.

Update Files

Update files are disk files from which a program reads a record, updates fields in the record, and writes the record back in the location from which it was read. Update files must be further described by input and output specifications; only the fields to be updated must be described by input and output specifications. A chained file or a demand file can be updated at detail time, total time, or at exception output time. All other disk files can be updated only at detail or exception time, during the same program cycle in which the record is read.

Combined Files

A combined file is both an input file and an output file. A combined file can be assigned to the SPECIAL device only. A program reads records from a combined file and includes output data on the records in the file. The result is one file that contains both input and output data. Combined files must be further described by both input and output specifications. Output to a combined file can occur only once per cycle.

COLUMN 16 (FILE DESIGNATION)

Entry	Explanation
P	Primary file
S	Secondary file
C	Chained file
R	Record address file
T	Table file (execution-time tables or arrays)
D	Demand file

Use column 16 to further identify the use of input and update files. Leave the column blank for display screen files and all output files except chained output files (direct load).

Primary Files

A primary file is the main file from which a program reads records. In multifile processing, the primary file is used to control the order in which records are selected for processing. (See *Supplementary Information, Multifile Processing* for more information on record selection in primary files.)

A primary file can be an input, combined, or update file. In programs that read records from only one file, that file is the primary file. Every program must have one, and only one, primary file.

Note: If the keyboard is specified as a primary input file, no other files in the program can be specified as primary or secondary files. If the keyboard is specified as a primary input file, you must also provide exit for your program (that is, provide for the setting on of the LR indicator).

Secondary Files

Secondary files apply to programs that do multifile processing. All files involved in multifile processing, except the primary file, are secondary files. A secondary file can be an input, update, or combined file. Secondary files are processed in the order in which they are written in the file description specifications.

Note that table, chained, record address, and demand files are not involved in record selection in multifile processing. (See *Supplementary Information, Multifile Processing* for more information on primary and secondary files.)

Chained Files

A chained file is a disk file that uses the CHAIN operation code to do one of the following:

- Read records randomly
- Load a direct file

A chained file can be an input, output, or update file. See *Column 28 (Mode of Processing), Random by Relative Record Number or Key* in this section for a discussion of random processing. See *Supplementary Information, Operation Codes, CHAIN* for information about CHAIN operation code.

Record Address Files

A record address file is an input file that indicates to your program (1) which records are to be read from a disk file, and (2) the order in which the records are to be read from the disk file. You cannot use more than one record address file in a program. All record address files must be further defined by extension specifications. Record address files contain record-key limits or a disk address.

Record address files that contain record-key limits can be disk files or can be entered through the console. Files that contain limits are used with indexed files only. See *Column 28 (Mode of Processing), Sequential Within Limits* in this section for a complete discussion of this topic.

Record address files that contain disk addresses can only be disk files. Record address files on disk that contain disk addresses are called ADDROUT (address output) files. They are produced by the sort program and can be used with any type of disk file. See *Column 28 (Mode of Processing), Random by ADDROUT File* in this section for a complete discussion of this topic.

Table or Array Files

A table or array file is an input file that contains table or array entries. Tables or arrays can be entered from DISK. The entries can be read into the program during the compilation or execution of the program. Only pre-execution-time table or array files are described by file description specifications. However, tables and arrays must be described by extension specifications.

Entries read during compilation become a permanent part of the program. Both compile and preexecution-time tables or arrays can be changed at execution time. Compile-time tables or arrays, however, can be permanently altered only by recompiling the program. Preexecution-time tables can be altered any time the program is executed by changing the table input file.

Table files are not involved in record selection and processing. They are only a means of supplying entries for tables used by the program. When preexecution-time table or array files are read during the execution of the program, the program reads all the entries from the table or array files before it begins record processing. All table and array files must be further defined by extension specifications.

Demand Files

Demand files can be input, update, or combined files. The READ operation code must be used in the calculation specifications to read any demand files except those entered from files assigned to the KEYBORD. (The KEY operation code must be used in calculation specifications to read from KEYBORD demand files.) See *Supplementary Information, Operation Codes, READ* for a complete discussion of processing demand files.

COLUMN 17 (END OF FILE)

Entry	Explanation
E	All records from the file must be processed before the program can end.
Blank	The program can end whether or not all records from the file are processed. If column 17 is blank for all files, all records from every file must be processed before the program can end.

Column 17 applies to programs that perform multifile processing. Use it to indicate whether the program can end before all records from the file are processed.

This column applies only to input, update, and combined files used as primary, secondary, or record address files. The devices associated with column 17 are disk files and console files. (An E entry is not valid for files processed as record address files.) End of file for data files entered from the console is identified by pressing the CMD key and the / key.

A program that performs multifile processing could reach the end of one file before reaching the end of the others. It needs, therefore, some indication of whether it is to continue reading records from the other files or end the program. An entry in column 17 provides that indication.

If the records from all files must be processed, column 17 must be blank or contain E's for all files.

Note: An entry cannot be made in column 17 for a keyboard file. To terminate the program with a primary keyboard file, the LR indicator must be set on by calculation specifications.

COLUMN 18 (SEQUENCE)

Entry	Explanation
A	Sequence checking is to be done. Records in the file are in ascending order.
D	Sequence checking is to be done. Records in the file are in descending order.
Blank	No sequence checking is to be done.

Use column 18 to indicate whether or not the program is to check the sequence of records. Column 18 applies to input, update, or combined files used as primary or secondary files. The devices associated with column 18 are disk files (except those processed randomly), and console files. Use columns 61-62 on the input specifications sheet to identify the record fields containing the sequence information.

Sequence checking is required when match fields are used in the records from the file. When a record from a matching input file is found to be out of sequence, the program halts and the operator has three options:

- Bypass the record out of sequence and read the next record from the same file.
- Bypass the record out of sequence, turn on the LR indicator, and perform all end of job and final total procedures.
- Cancel the entire program.

COLUMN 19 (FILE FORMAT)

Entry	Explanation
F	Fixed-length records

An F in column 19 indicates that all records in the file are of the same length. If this column is blank, F is assumed.

COLUMNS 20-23 (BLOCK LENGTH)

Entry	Explanation
1-4096	- Block length for disk or console files. - Multiple of disk record length. - Length of largest field keyed for keyboard files. - Length of largest output record for display screen files.
Blank	The block length for this file is the same as the record length.

Columns 20-23 have a different use depending on the device named for the file. The block length entry must end in column 23, and leading zeros can be omitted. Columns 20-23 can be left blank for any file (Figure 12).

When CONSOLE is used as the device name, the block length must be at least two greater than the record length specified in columns 24-27.

Block Length for Disk Records

Block length must be a number, either record length, or a multiple of record length. The maximum block length is 4096.

Block length does not affect the way records are written on the disk. Its function is to specify the amount of main storage to use for the input/output area.

Note: If record length is used in these columns for disk files, RPG II assigns an efficient block length.

Device	Cols 20-23 Block Length	Cols 24-27 Record Length	Maximum Record Length
Disk	Record length or a multiple of record length	Record length	4096
Console	Record length +2 or (multiple of record length) +2	Record length	160
Keyboard	Length of largest field to be keyed	Length of largest field to be keyed	40 - alphameric 15 - numeric
Printer (132-position)	Record length	Record length	132
Display screen	Length of longest output record	Length of longest output record	40
SPECIAL	Record length or a multiple of record length	Record length	4096
BSCA	Record length or a multiple of record length	Record length	4096

● **Figure 12. Block Length and Record Length Entries**

COLUMNS 24-27 (RECORD LENGTH)

Entry	Explanation
1-4096	- Record length for disk files or console (IDE) files. - Length of largest field keyed for keyboard files. - Length of largest output record for display screen files - Length of largest output record for printer files.

Columns 24-27 have a different use depending on the device named for the file. An entry must be made for all files. Entries in these columns must end in column 27, and leading zeros can be omitted (Figure 12).

All records in one file must be the same length. (For update files, the length of the record after the record is updated must be the same as it was before the record was updated.) The maximum length allowed depends upon the device assigned to the file. The record length can be shorter than the maximum length for the device but no longer.

The record length for keyboard files should be the length of the largest field to be keyed and should not exceed 40 alphabetic characters or 15 numeric characters (the largest field length in columns 49-51 of the calculation specifications sheet when the KEY operation is used).

COLUMN 28 (MODE OF PROCESSING)

Entry	Explanation
L	Sequential within limits.
R	- Random by relative record number. - Random by key. - Random by ADDROUT file. - Direct file load (random load).
Blank	- Sequential by key. - Consecutive.

Use column 28 to indicate the method by which reocrds are to be read from the file, or to indicate that a direct file load (random load) is to take place.

For disk files specified as primary, secondary, or chained, the possible methods depend upon the organizations of the files (Figure 13). For the other types of files, consecutive processing is the only possible method.

Column 31 is used to further identify the access method for the program. See *Column 31 (Record Address Type)* in this section.

Consecutive

The consecutive method applies only to sequential and direct files. During consecutive processing, records are read in the order they appear in the file. The contents of spaces left for missing records in direct files are read as though the records were there. (When a direct file is loaded, such spaces are filled with blanks.)

The program reads records from the file until either the end of that file is reached or the program ends due to the end-of-file condition of another file. See *Column 17 (End Of File)* in this section for more information about the second condition.

By ADDROUT File

An ADDROUT (address output) file is a record address disk file produced by the sort program. It contains addresses of records in a disk file. (Each address is a 3-byte binary number.) You can use ADDROUT files to process input or update files that are designated as primary or secondary files.

When an RPG II program uses an ADDROUT file, it reads a disk address from the ADDROUT file. The program then locates and reads records at that address in the original disk file. Records are read in this manner until either the end of the ADDROUT file is reached or the program ends due to the end-of-file condition of another file (see *Examples, Example 1*).

ADDROUT files must be further described by extension specifications. Both the ADDROUT file and the file to be processed by the ADDROUT file must be described by file description specifications. See *Column 17 (End Of File)* in this section for more information about end of file.

Sequential By Key

The sequential by key method of processing applies only to indexed disk files that are used as primary files, secondary files, or demand files.

Records are read in ascending key sequence (the order in which the record keys are arranged in the index portion of the file). The program reads records until all records in the file are processed or the program ends due to the end-of-file condition of another file. See *Column 17 (End Of File)* for more information about the second condition.

Primary, Secondary, or Demand Files

Organization	Possible Methods
Sequential	- Consecutively - By ADDROUT file
Direct	- Consecutively - By ADDROUT file
Indexed	- Consecutively - By ADDROUT file - Sequentially by key - Sequentially within limits

Chained Files

Organization	Possible Methods
Sequential	Randomly by relative record number
Direct	Randomly by relative record number
Indexed	Randomly by key or by relative record number

● **Figure 13. Possible Record Retrieval Methods for Disk Files**

Sequential Within Limits

The sequential within limits method of processing can be executed by using either: (1) a record address file containing limit records, or (2) the SETLL operation code during calculation specifications.

The sequential within limits method applies only to indexed disk files used as primary files, secondary files, or demand files. A limits record consists of the lowest record key and the highest record key of the records in the indexed disk file which are to be read. Limits records are contained in a record address file. The record address file can be on disk or entered by the keyboard.

To process sequentially within limits, the program reads:

1. A limits record from the record address file.
2. Records with keys greater than or equal to the low record key and less than or equal to the high record key.

The program repeats these two steps until either the end of the record address file is reached or the program ends due to the end-of-file condition of another file. See *Column 17 (End Of File)* in this section for more information about end of file.

The format of records in a record address file containing limits must conform to these rules:

- Only one set of limits is allowed per record in the record address file. The length of a record in a record address file, therefore, must be twice the length of the record key.

- The low record key must begin in position 1 of the record. The high record key must immediately follow the low record key. A record key can be from 1-29 characters in length.

- The low record key and the high record key must have the same length and each key must have the same length as the key field length specified in columns 29-30. Therefore, leading zeros may be necessary in specifying numeric record keys.

- An alphameric record key can contain blanks.

Files containing limits and files being processed by limits may have keys in different formats. For example, one file may have packed keys and the other unpacked keys. During execution time, the format of the key from the file containing limits is changed to the format of the file being processed by limits. The format of the keys on each file must be indicated by an A or a P in column 31. Also, the unpacked key length must be twice the packed length, minus one or two. See *Packed Decimal Format (P)* for more information concerning this calculation.

The same set of limits can appear in more than one record address record. Data records, therefore, can be processed as many times as you wish. The two records keys in a limits record can be equal. Only one data record is read in this case.

Note: Double buffering (column 32) should not be specified for the record address file.

The SETLL operation code method of limits processing applies to any indexed disk file used as demand files (D in column 16 and L in column 28 of the file description specifications sheet). You cannot, however, process an indexed demand file with SETLL if you are using a record address file to set the limits of the file.

The maximum number of files which can be processed using SETLL is limited by the number of demand files permitted in an RPG II program (a maximum of 15 demand and/or chain files is allowed per program). See *Examples, Example 2* for an example of SETLL. For more information on how to set limits using the SETLL operation code, see *Operation Codes*.

When the end-of-file indicator is turned on, another SETLL can be issued and processing of the file may continue.

Random By Relative Record Number or Key

Random processing by relative record number or by key applies to chained files only. Either method requires use of the CHAIN operation code. The records of a file to be read or written must be processed by the CHAIN operation code. The records are read or written only when the CHAIN statements that identify them are executed.

For sequential and direct files, relative record numbers must be used to identify the records (see *Examples, Example 3*). Relative record numbers identify the positions of the records relative to the beginning of the file. For example, the relative record numbers of the first, fifth, and seventh records in a file are 1, 5, and 7 respectively.

For indexed files, record keys must be used to identify the records (see *Examples, Example 4*). A record key is the information from the key field of a record. The information is used in the index portion of the file to identify the record.

Records are read during the calculation phase of the program. Therefore, they can be executed during detail or total calculations. Note then, that fields of records read from chained update files can be read and altered during total calculations and the records can be updated (written back on the file with alterations) during total output; the same also applies to detail calculations and detail output (see *Examples, Example 5*).

Examples

Example 1

Figure 14 shows processing a sequential disk file by an ADDROUT file. The record address file, ADRTFILE, defined as an ADDROUT disk file, consists of 3-byte binary relative record numbers which correspond to locations of records on the input disk file, MASTER. As each record is read from ADRTFILE, the indicated record from MASTER is located and read. For each record read from MASTER (indicator 01 is on), a detail line is printed on the printer output file, PRINTER.

Since end of file (E in column 17 of the file description specifications sheet) is specified for the ADDROUT file, processing continues until all records in ADRTFILE have been read.

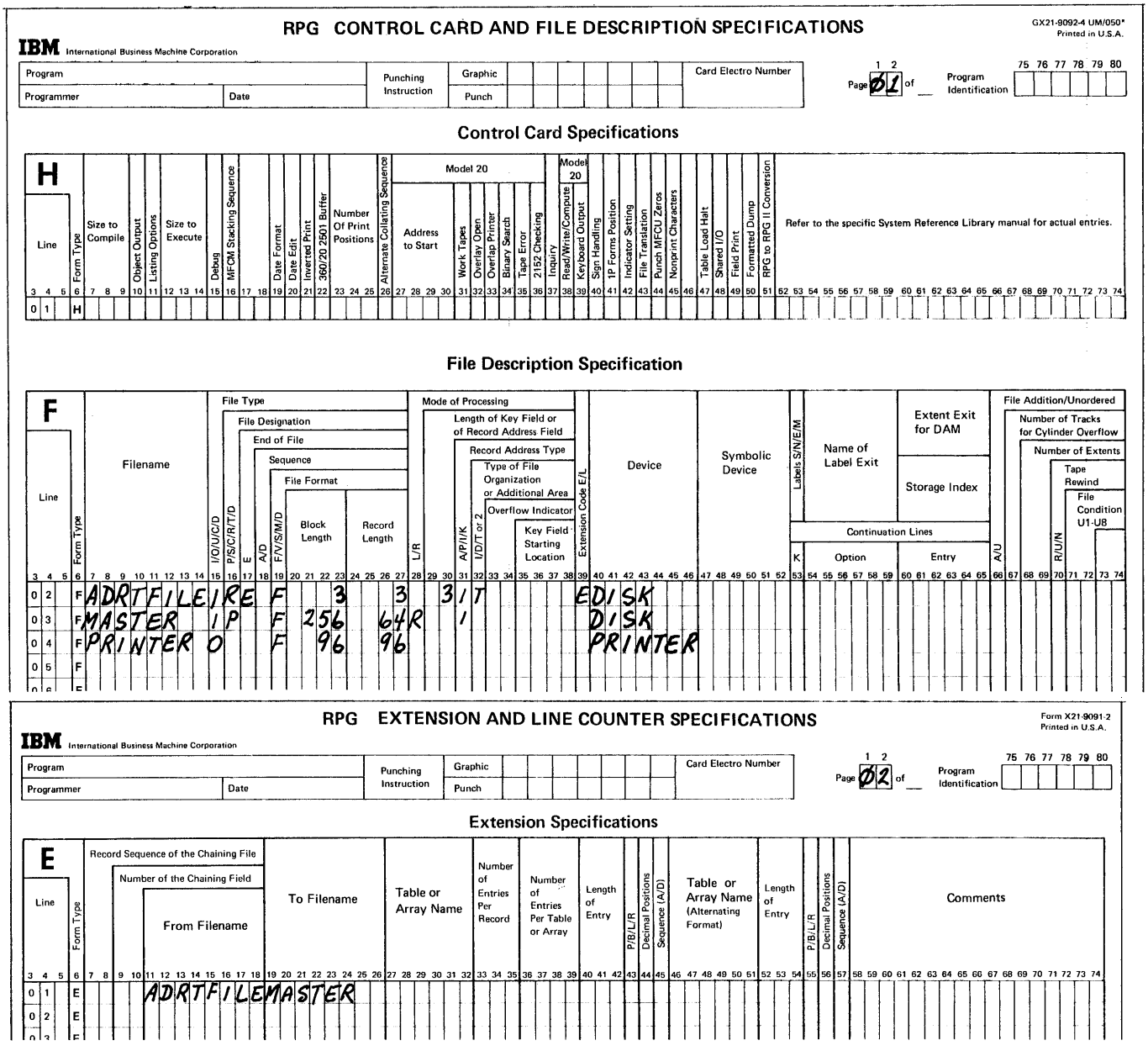
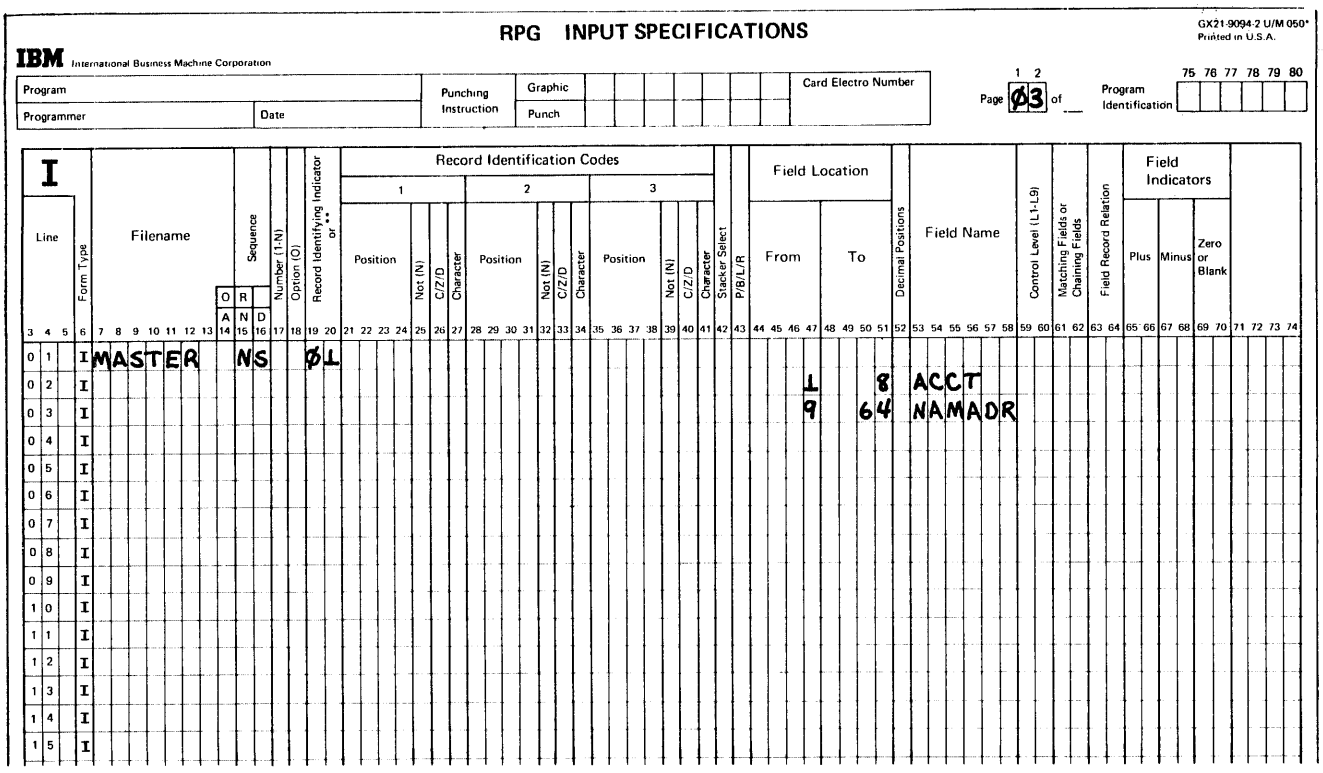


Figure 14 (Part 1 of 2). Processing a Sequential Disk File with an ADDROUT File



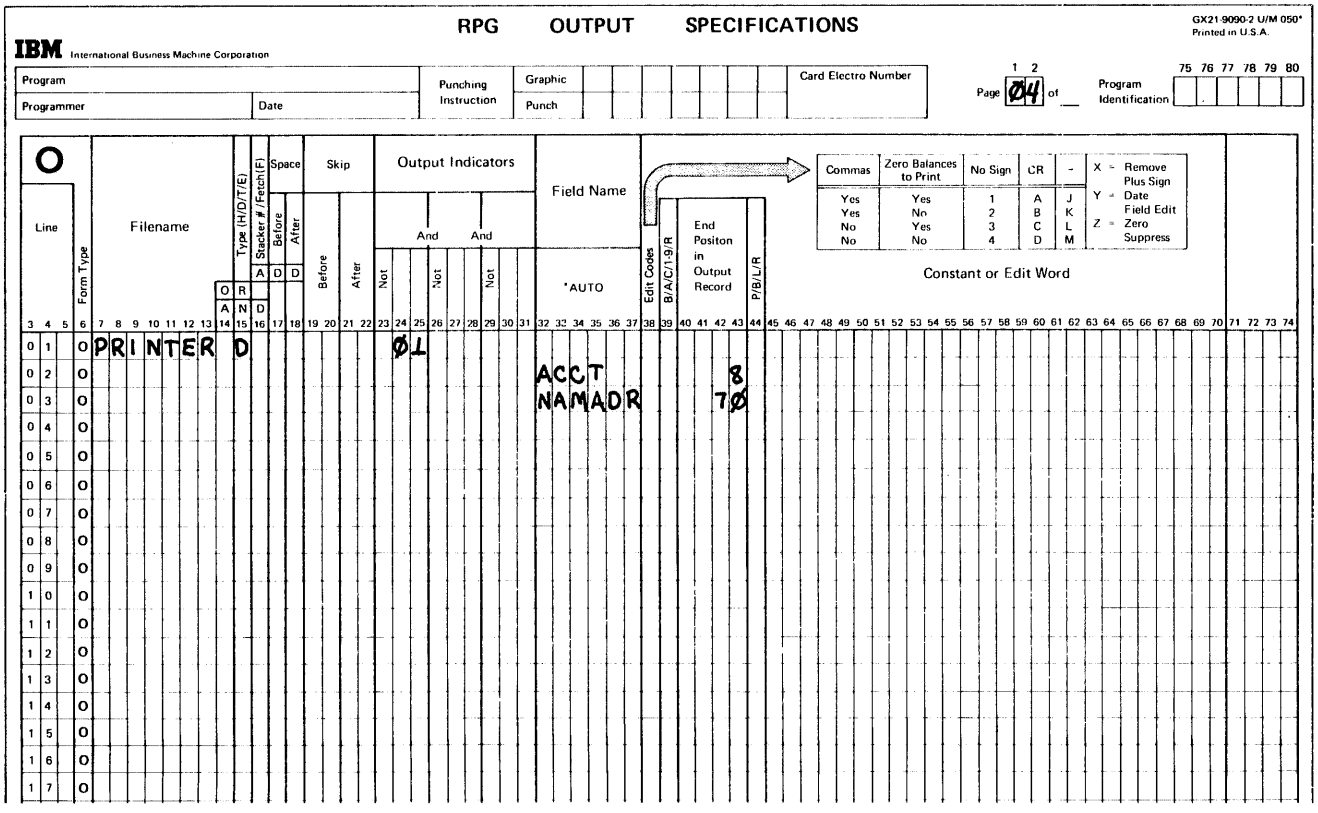


Figure 14 (Part 2 of 2). Processing a Sequential Disk File with an ADDROUT File

Example 2

In Figure 15, the input disk file, MASTER, described as an indexed file to be processed by record keys is to be processed within the limits contained on the record address file, LIMITS. The LIMITS file, which is further described on the extension specifications sheet, is entered from the console.

Each set of limits read from LIMITS will consist of the low and high account numbers to be processed. Since the account number key field (ACCT) is eight positions long, each set of limits will include two 8-position keys.

As MASTER is processed within each set of limits, the corresponding records are written out on the printer output file, PRINTER. Processing is complete when all sets of limits have been processed.

File SMASTER is processed by the SETLL operation code. It is characterized by having no extension specifications, and its filename appears in factor 2 of the SETLL operation code. In this example the first record read from file SMASTER would be the one whose key is equal to or the next higher than the literal 'AAAAAAAA'. Records are read sequentially to end of file unless the cycle is interrupted by additional SETLL operations.

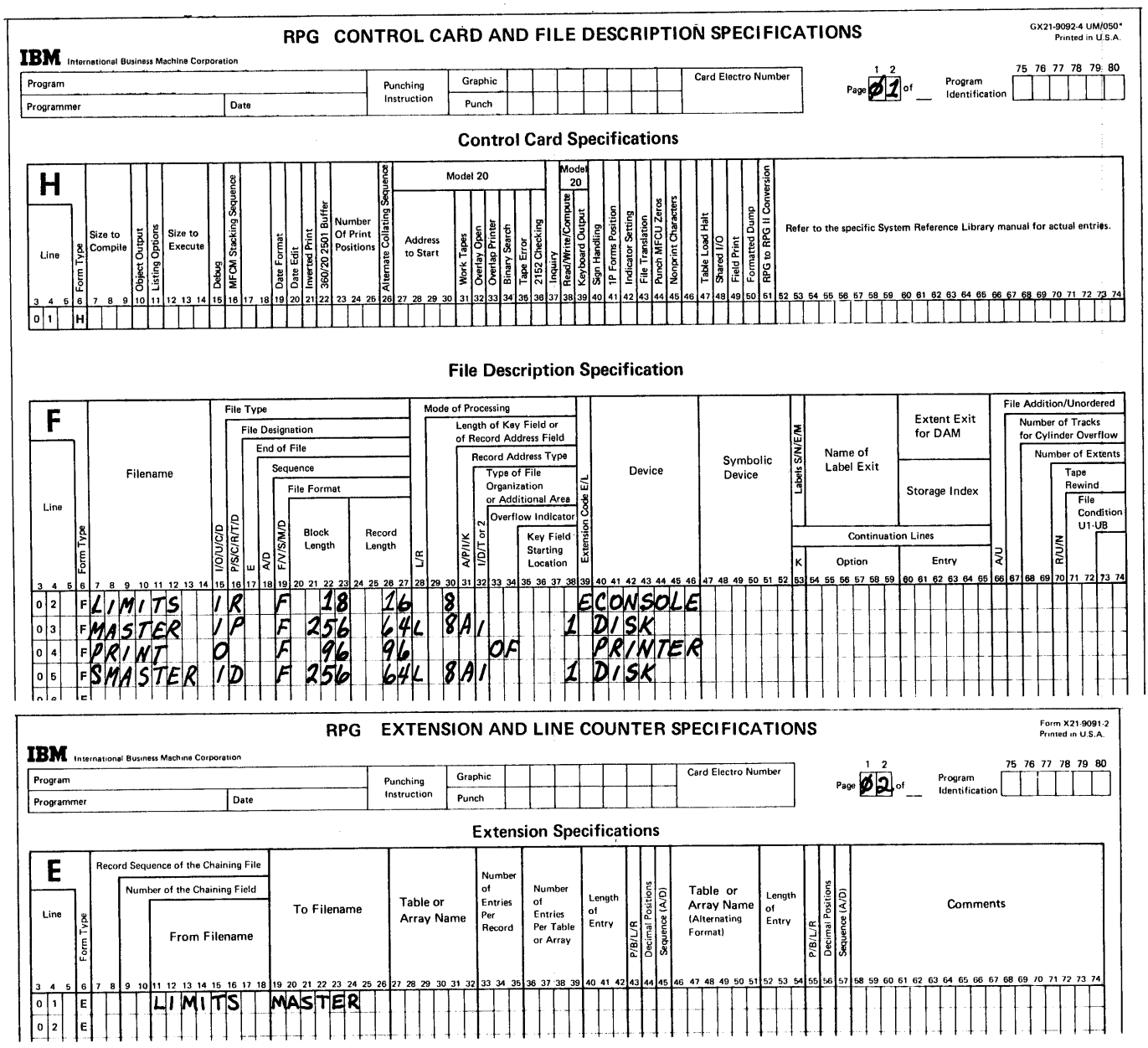
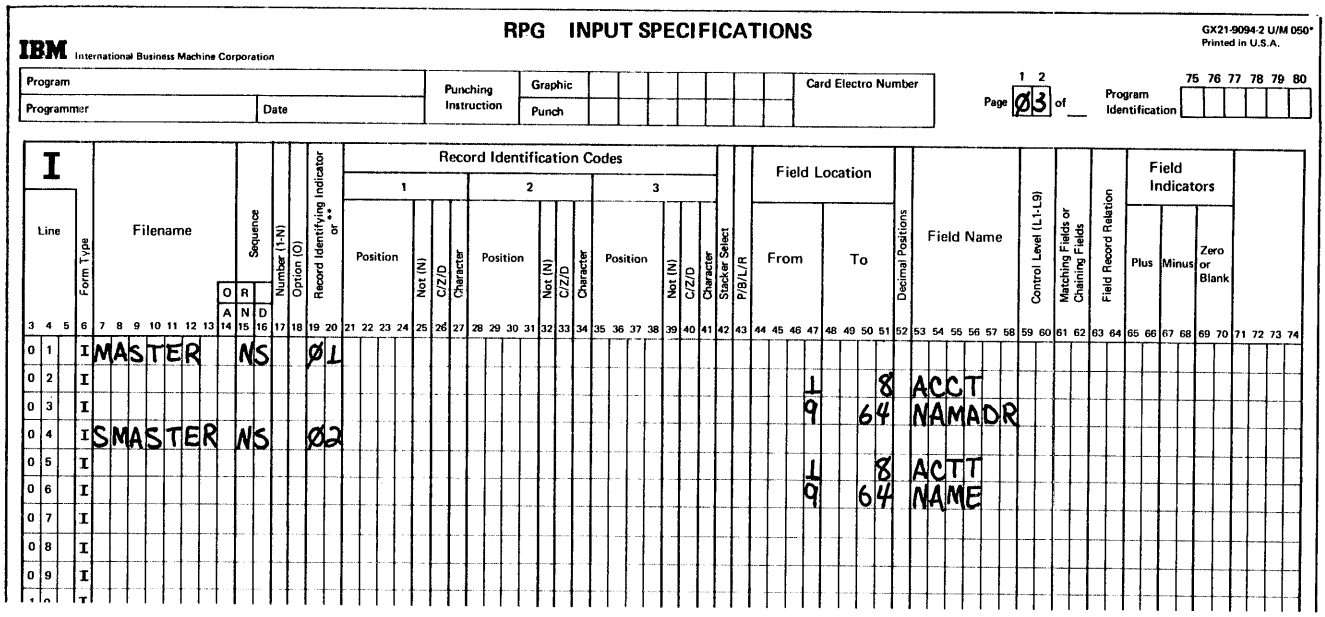
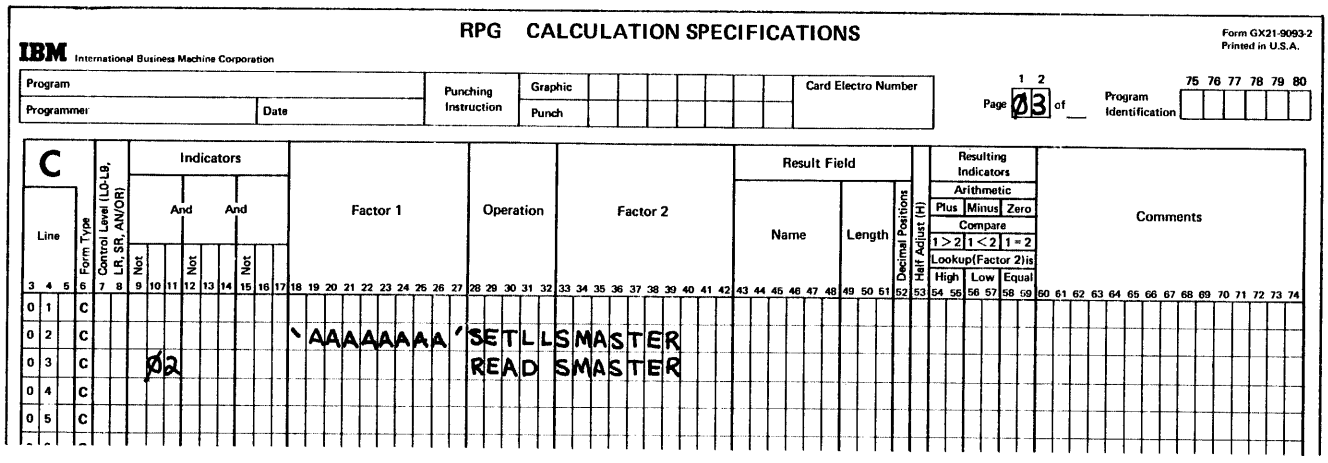


Figure 15 (Part 1 of 2). Processing an Indexed File Sequentially Within Limits





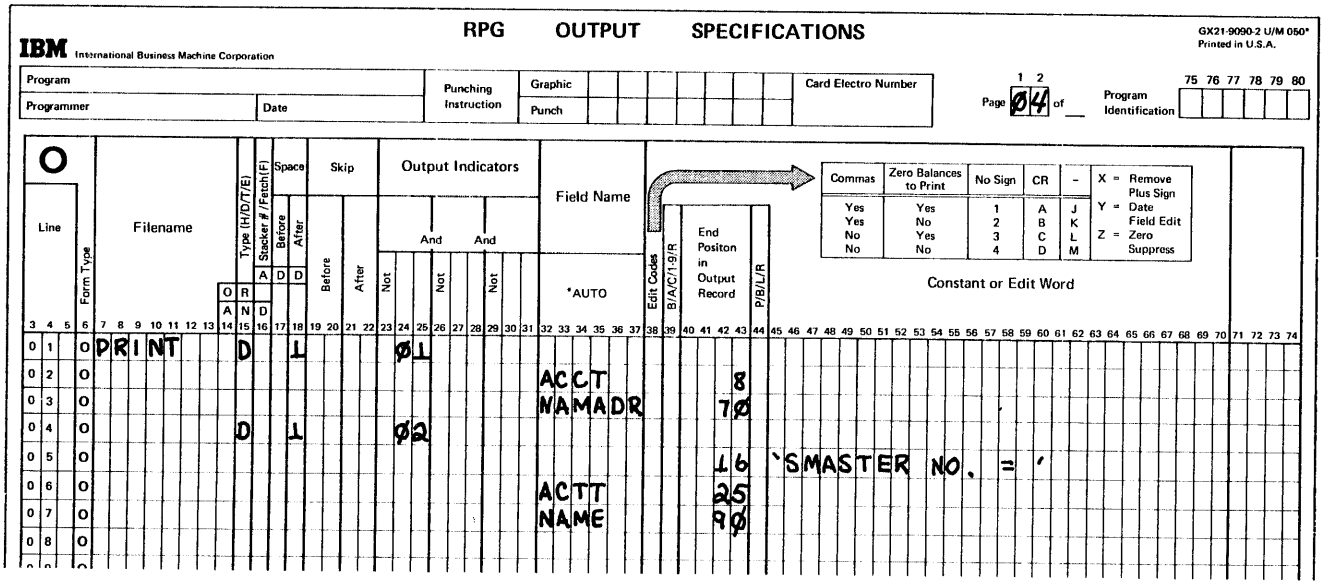


Figure 15 (Part 2 of 2). Processing an Indexed File Sequentially Within Limits

Example 3

In Figure 16, the direct update file, MASTER, is to be processed randomly by relative record numbers. The account number (ACCT) from the primary DISK file, CHANGE, is used as the relative record number.

As each record is read from CHANGE, the MASTER record corresponding to the account number is read during calculation time by the CHAIN operation code. At detail output time, the data in the NEW field replaces the original data in the NAMADR field and the updated MASTER record is output to its original relative record location on the disk file.

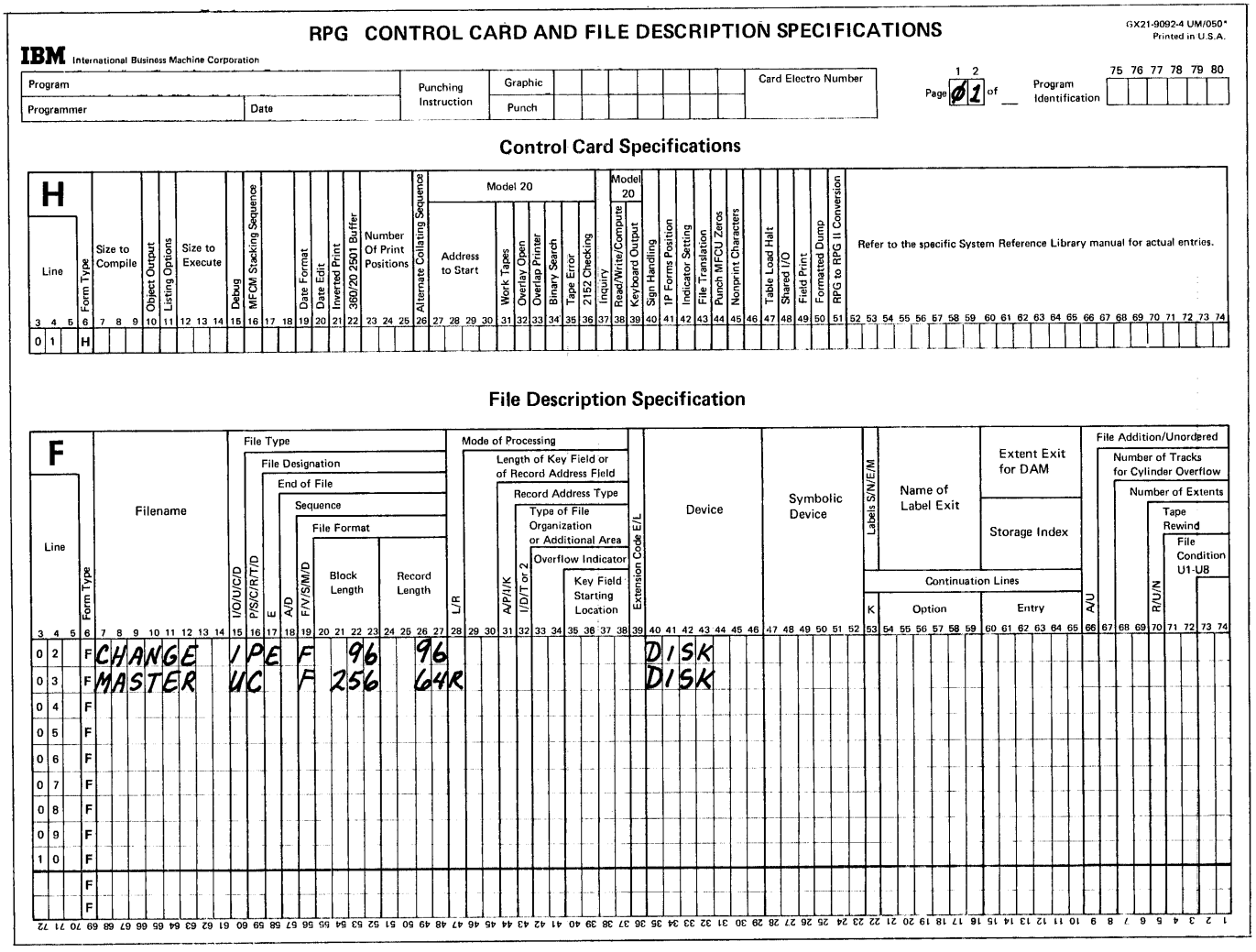
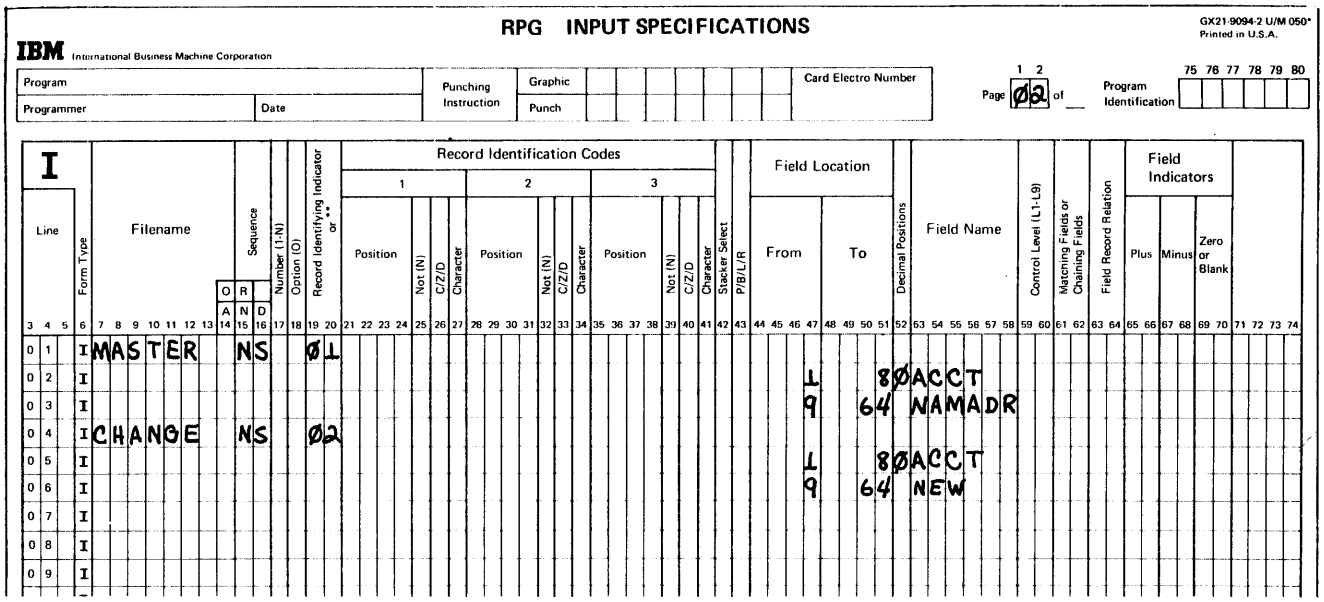
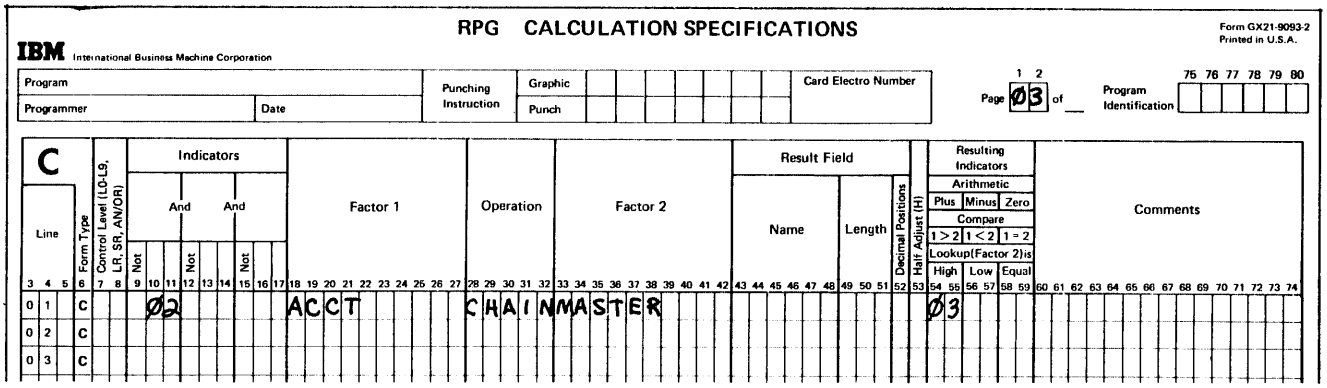


Figure 16 (Part 1 of 2). Random Processing of a Direct File by Relative Record Number





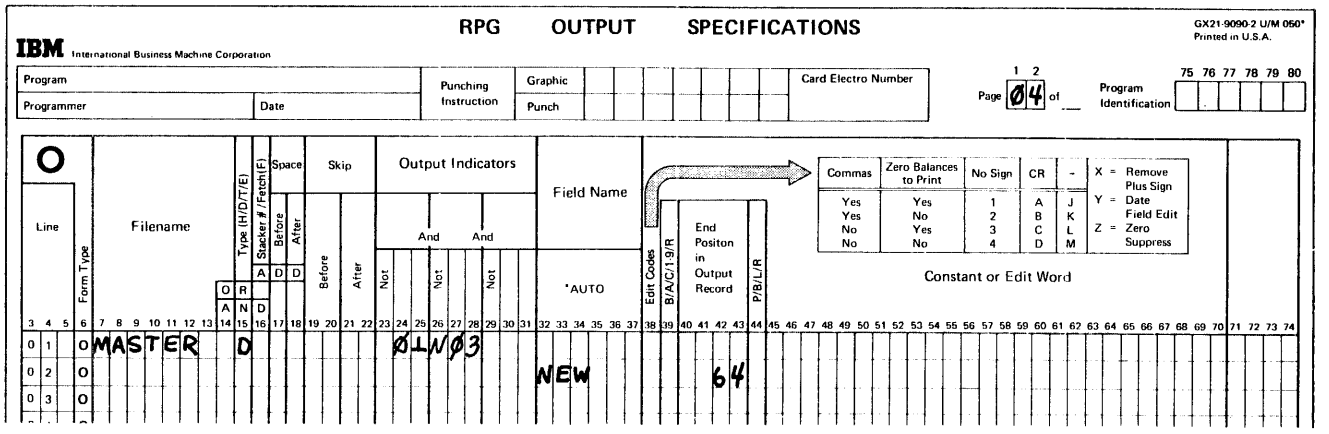


Figure 16 (Part 2 of 2). Random Processing of a Direct File by Relative Record Number

Example 4

Figure 17 shows random processing by key of an indexed file. MASTER, a chained update file, is described on the file description specifications sheet as an indexed file to be processed by keys. As each record is read from the input disk file, CHANGE, the account number (ACCT) is used as the key to chain to the corresponding record in MASTER at calculation time. At detail output time, the data in the NEW field of CHANGE replaces the original data in the NAMADR field. The updated MASTER record is then written on its original disk location. See *Column 32 (File Organization or Additional I/O Area)* in this chapter for a description of indexed file organization.

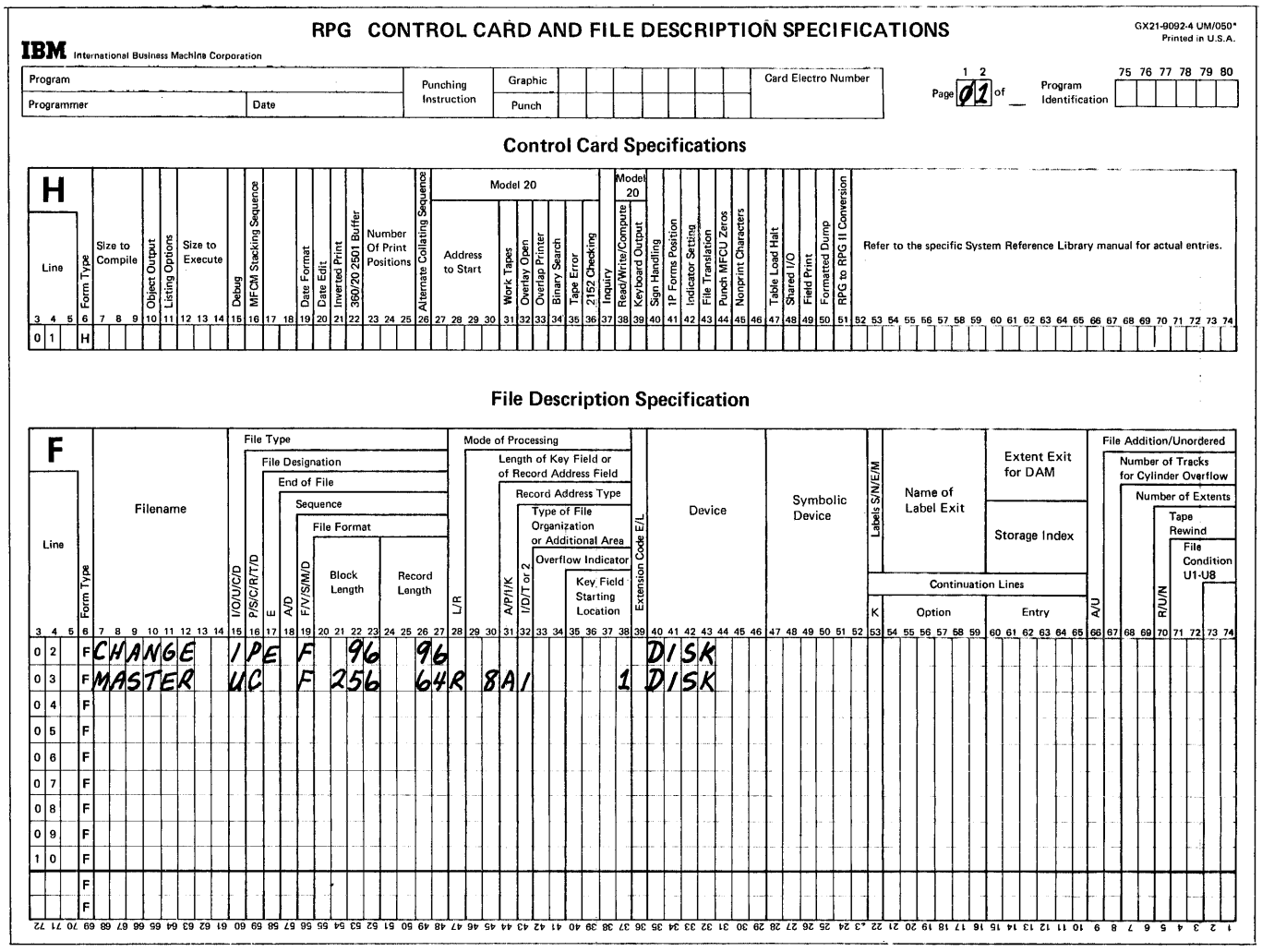
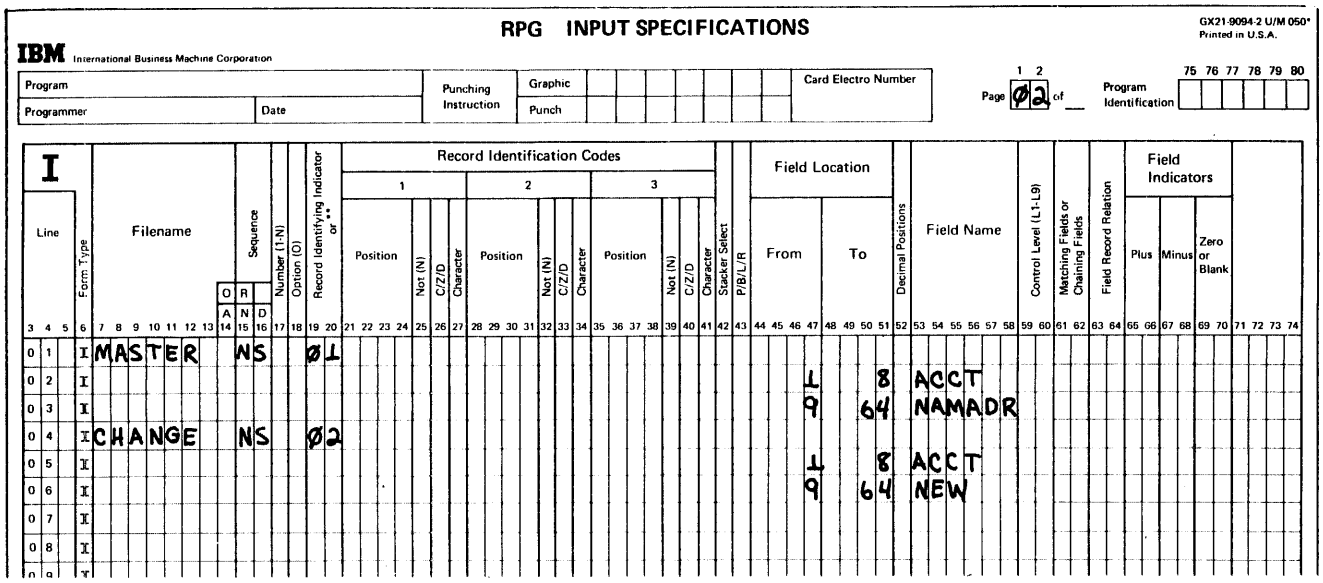
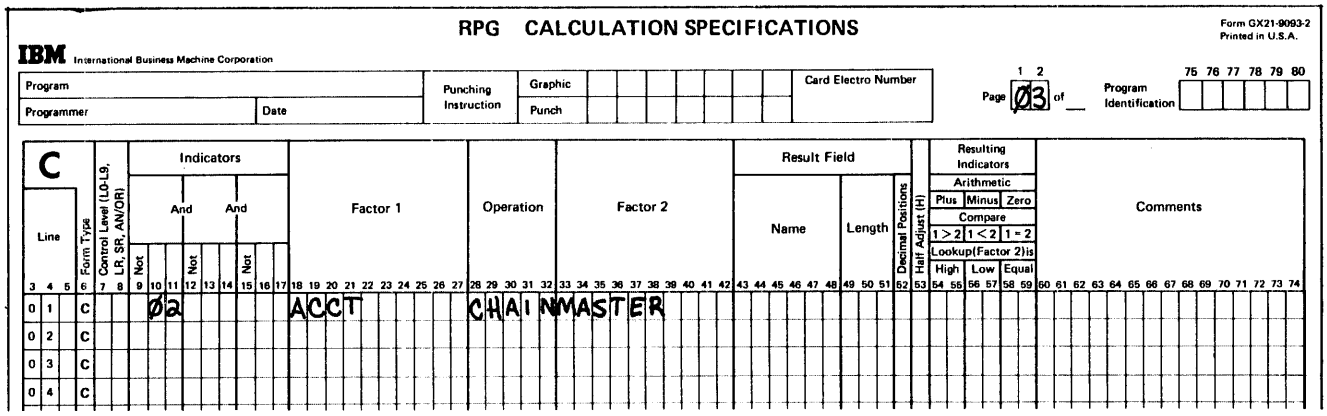


Figure 17 (Part 1 of 2). Random Processing of an Indexed File by Key





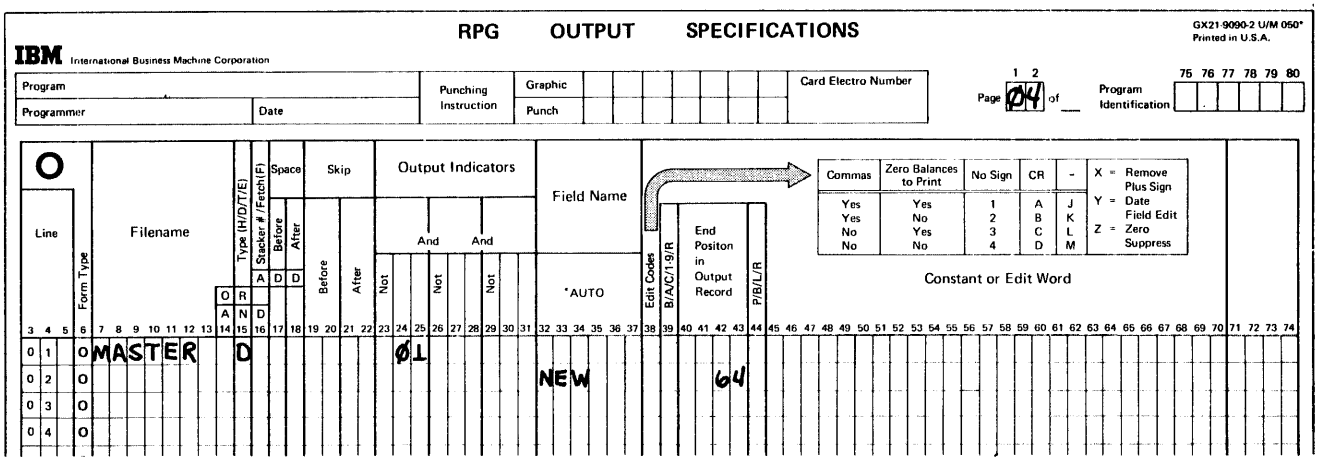


Figure 17 (Part 2 of 2). Random Processing of an Indexed File by Key

Example 5

Figure 18 shows the updating of an indexed file. The indexed file, MASTER, is described as a chained update file to be processed by keys. The key field in MASTER is ITEMNO, in positions 1-10. The index will be sorted into ascending sequence when processing is complete.

As each record is read from TRANS, the input transaction file, the ITEMNO field is used as the key to chain to MASTER during calculations. If the character 2 is in position 64 of the transaction record, the quantity in ADJUST is added to the ONHAND field of MASTER. If the character 3 is in position 64, ADJUST is subtracted from ONHAND. If the character 1 appears in location 64 of the MASTER record, the updated ONHAND field is written out on its original location in the MASTER record at detail output time.

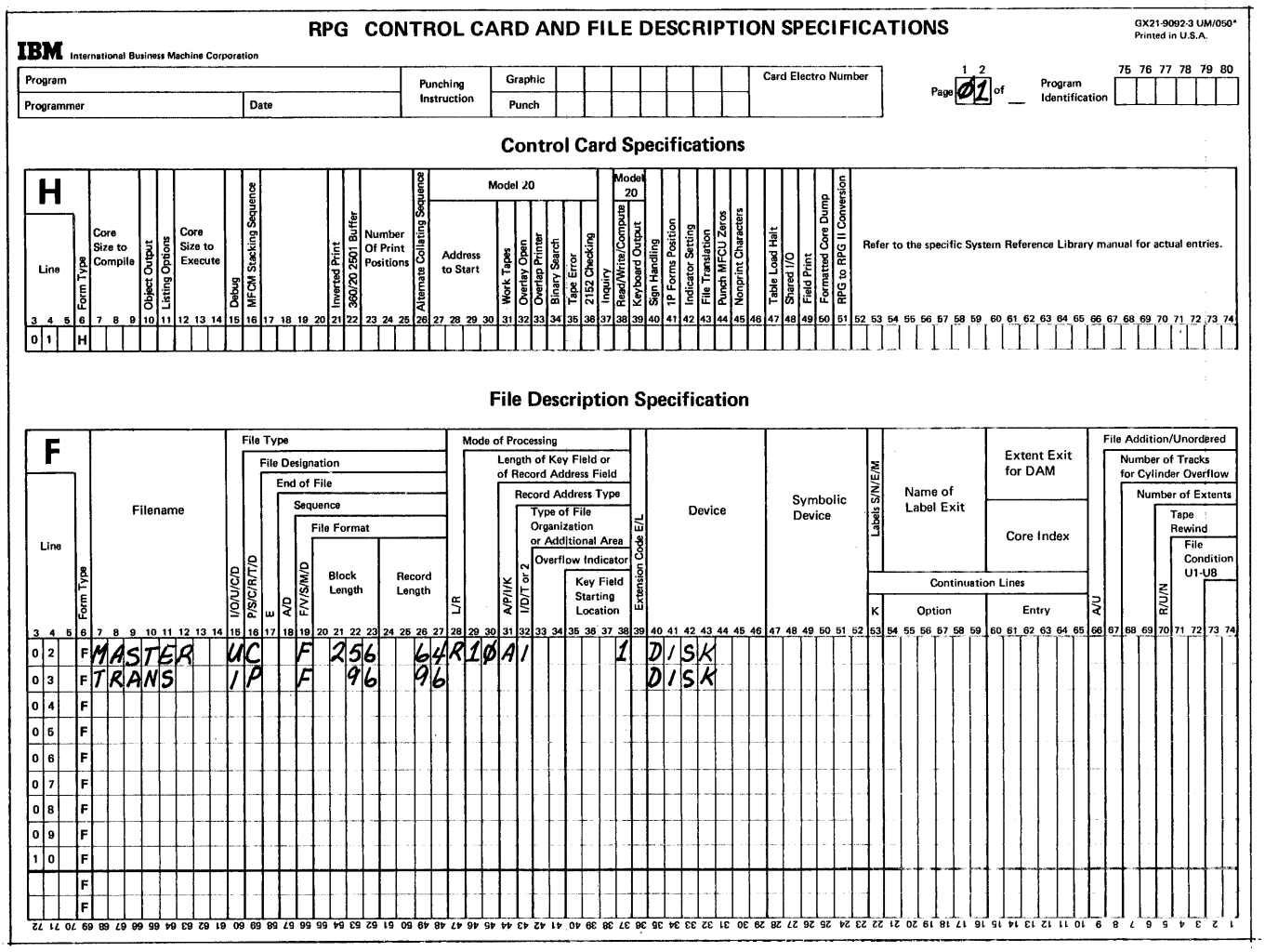
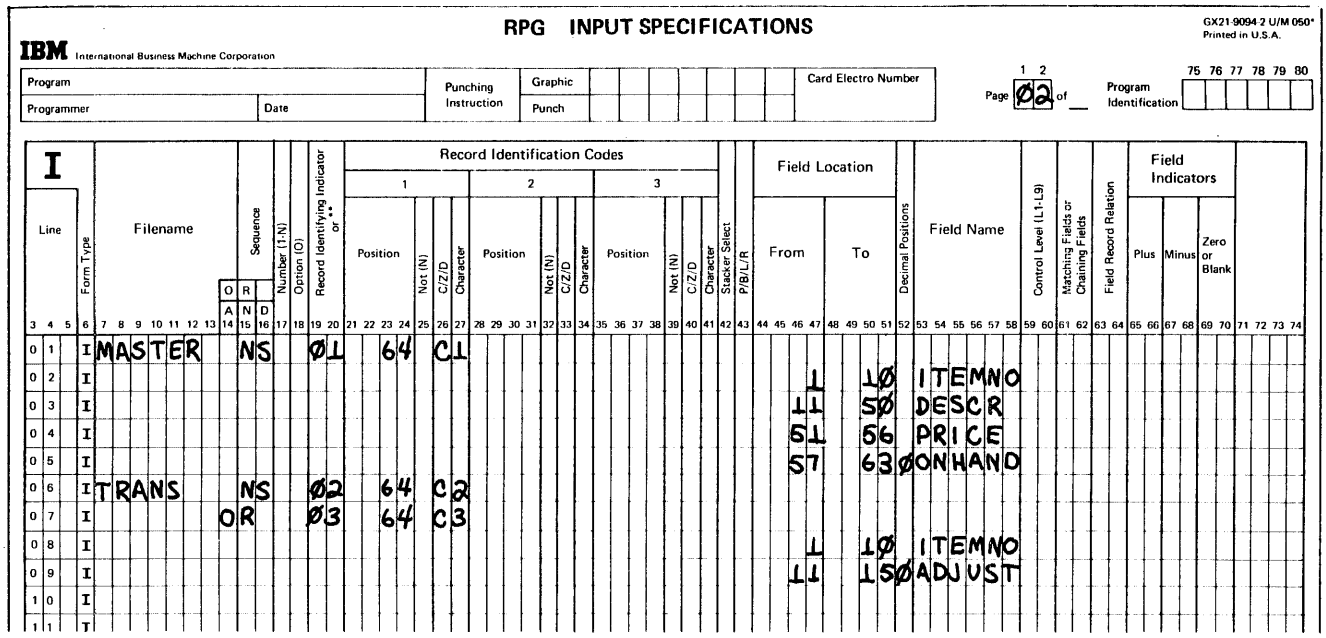


Figure 18 (Part 1 of 2). Updating an Indexed File



RPG INPUT SPECIFICATIONS

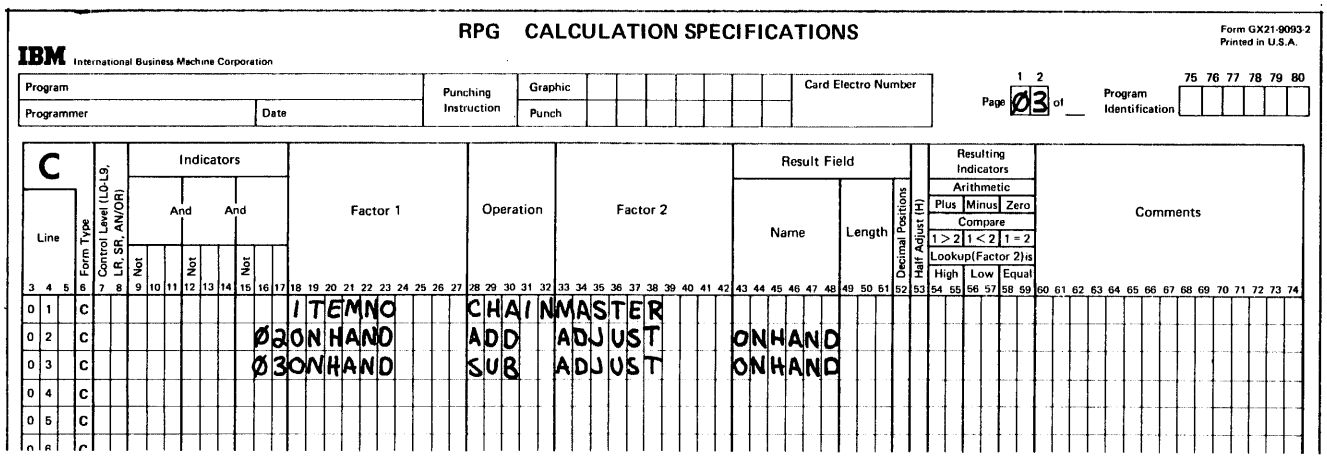
GX21-9094-2 U/M 050* Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 02 of __ Program Identification

Line	Form Type	Filename	Sequence (O / AN / R / D)	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	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
01	I	MASTER	NS			01	64		C	1																				
02	I																				1	10		ITEMNO						
03	I																				11	50		DESCR						
04	I																				51	56		PRICE						
05	I																				57	63	0	ONHAND						
06	I	TRANS	NS			02	64		C	2																				
07	I		OR			03	64		C	3																				
08	I																				1	10		ITEMNO						
09	I																				11	15	0	ADJUST						
10	I																													
11	I																													



RPG CALCULATION SPECIFICATIONS

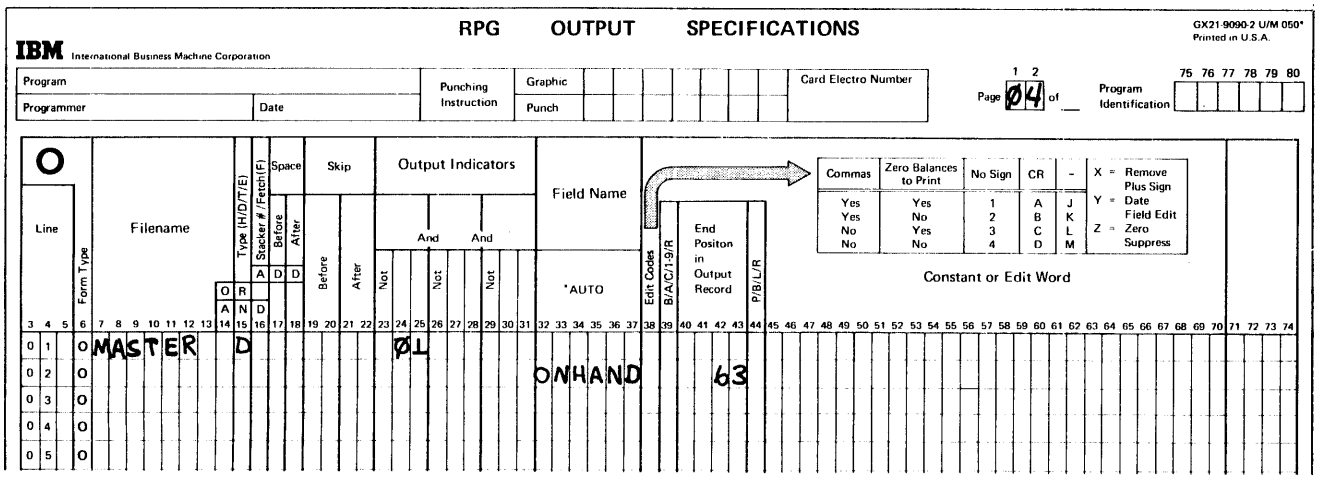
Form GX21-9093-2 Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 03 of __ Program Identification

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And Not	Indicators: And Not	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus	Arithmetic Minus	Arithmetic Zero	Compare 1>2	Compare 1<2	Compare 1=2	Lookup(Factor 2)is High	Lookup(Factor 2)is Low	Lookup(Factor 2)is Equal	Comments
01	C				ITEMNO	CHAIN	MASTER														
02	C			02	ONHAND	ADD	ADJUST	ONHAND													
03	C			03	ONHAND	SUB	ADJUST	ONHAND													
04	C																				
05	C																				
06	C																				



RPG OUTPUT SPECIFICATIONS

GX21-9090-2 U/M 050* Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 04 of __ Program Identification

Line	Form Type	Filename	Type (H/D/T/E) (O / AN / R / D)	Stacker # /Fetch (F)	Space: Before	Space: After	Skip: Before	Skip: After	Output Indicators: And Not	Output Indicators: And Not	Output Indicators: And Not	Field Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R
01	O	MASTER	D						01							
02	O											ONHAND			63	
03	O															
04	O															
05	O															

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

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

Constant or Edit Word

Figure 18 (Part 2 of 2). Updating an Indexed File

COLUMNS 29-30 (LENGTH OF KEY FIELD OR RECORD ADDRESS FIELD)

Entry	Explanation
1-29	Length of record key or disk address

Columns 29-30 apply only to indexed disk files and record address files. Use it to indicate:

- The length in bytes of the record keys in indexed files and record address files.
- The length of the disk addresses in ADDROUT files.

All of the key fields in the records in an indexed file must be the same length. The maximum length is 29 bytes (8 bytes for record keys in packed format). All of the disk addresses contained in an ADDROUT file are three characters long.

COLUMN 31 (RECORD ADDRESS TYPE)

Entry	Explanation
A	Record keys in unpacked format are used in processing or loading indexed files and record address files.
I	The file is being processed by using disk addresses from the ADDROUT file, or the file is an ADDROUT file consisting of disk addresses.
P	Record keys in packed format are used in processing or loading indexed files and record address files.
Blank	- Relative record numbers are used in processing sequential and direct files. - A sequential or direct file is being loaded. - Records are read consecutively. - Keys in the record address file are in the same format as keys in the indexed files.

Column 31 applies to disk files specified as input, update, or chained output files. It indicates how records in the file are identified (Figure 13.1). Together, columns 28 and 31 indicate:

- The method by which records are read from the file
- A direct file load

For ADDROUT files, column 31 must contain an I, indicating that disk addresses are used in processing.

Note: When building a file with packed keys (P in column 31), you must specify the key field as packed in your output specifications.

COLUMN 32 (FILE ORGANIZATION OR ADDITIONAL I/O AREA)

Entry	Explanation
I	Indexed file.
T	ADDROUT file.
1-9	Sequential file or direct file. Use two input/output areas for the file.
Blank	Sequential file or direct file. Use one input/output area for the file.

Use column 32 to (1) identify the organization of all files except ADDROUT files, (2) identify ADDROUT files, and (3) indicate whether one or two input/output areas are to be used for sequential files or direct files.

File Organization

File organization is the arrangement of records in a file. The three types are indexed, direct, and sequential. Files organized in these ways are called indexed files, direct files, and sequential files, respectively.

Indexed Files

An indexed file is a disk file in which the location of records is recorded in a separate portion of the file called an index. The index and its associated file occupy adjacent positions on the disk. The index contains the record key and disk address of every record (Figure 19).

A record key is the information from the key field of a record. The record key can be used to identify the records of an indexed file. Record keys are always required in an indexed file. Indexed files can be loaded with the keys in ascending sequence or keys in nonascending sequence. After a file is loaded in nonascending key sequence, the keys in the index are placed in ascending sequence. See *Column 66 (File Addition)* for a definition of the unordered load function.

Primary and Secondary Files

Method	Column 28	Column 31
Consecutive	Blank	Blank
By ADDROUT	R	I
Sequential by key	Blank	A or P
Sequential within limits	L	N

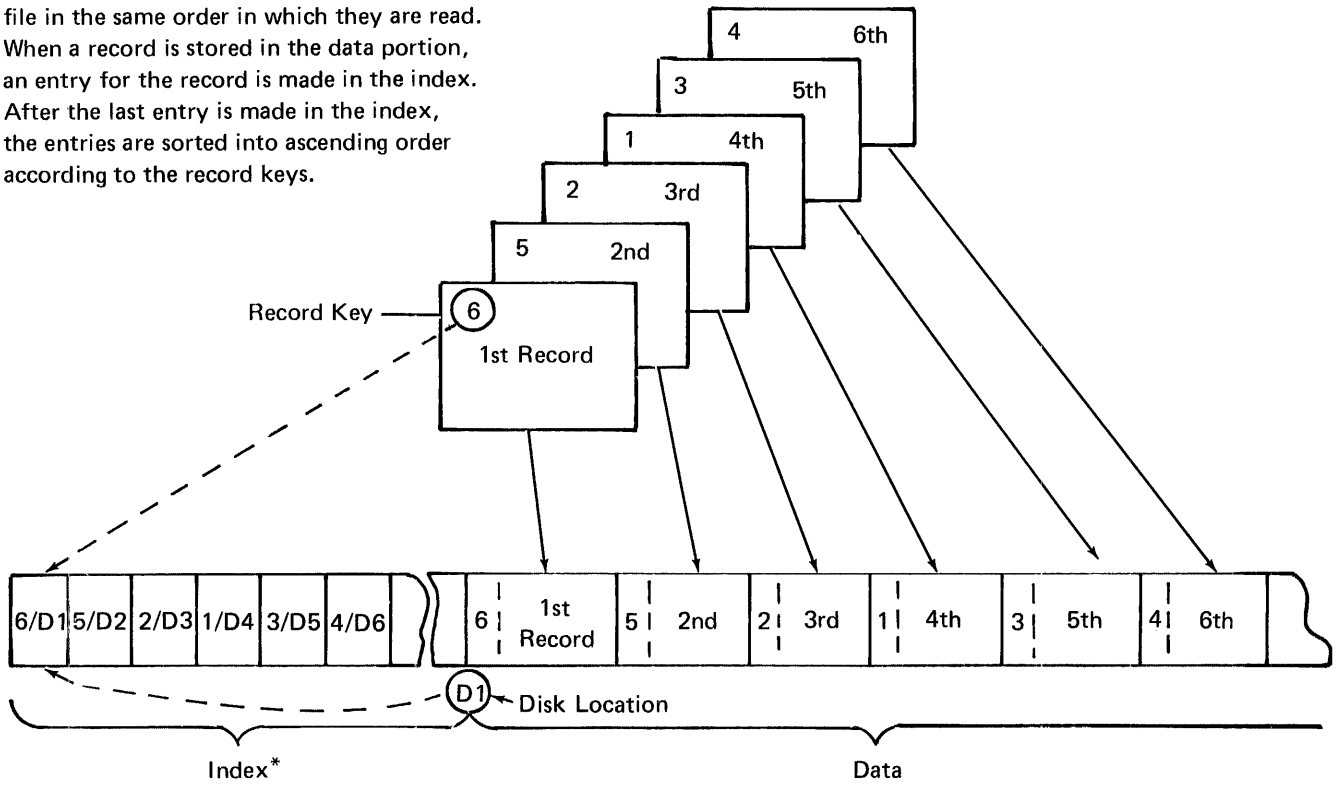
Chained Files

Random by relative record number	R	Blank
Random by key	R	A
Direct file load (random load)	R	Blank[1]

[1] A direct file load requires an O in column 15 and a C in column 16.

● **Figure 13.1. Specifications Identifying Methods for Retrieving Records**

Records are stored in the data portion of the file in the same order in which they are read. When a record is stored in the data portion, an entry for the record is made in the index. After the last entry is made in the index, the entries are sorted into ascending order according to the record keys.



* Entries are of the form record-key/disk-location (D1 = 1st disk location, D2 = 2nd disk location, and so on)

The order of the records in the data portion remains unchanged when the entries in the index are sorted.

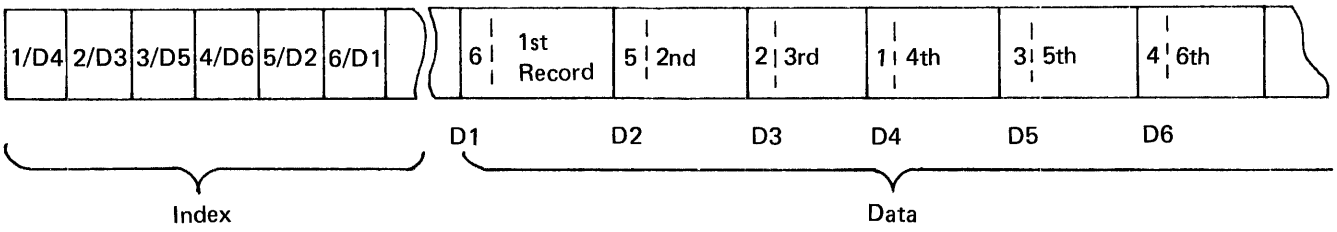
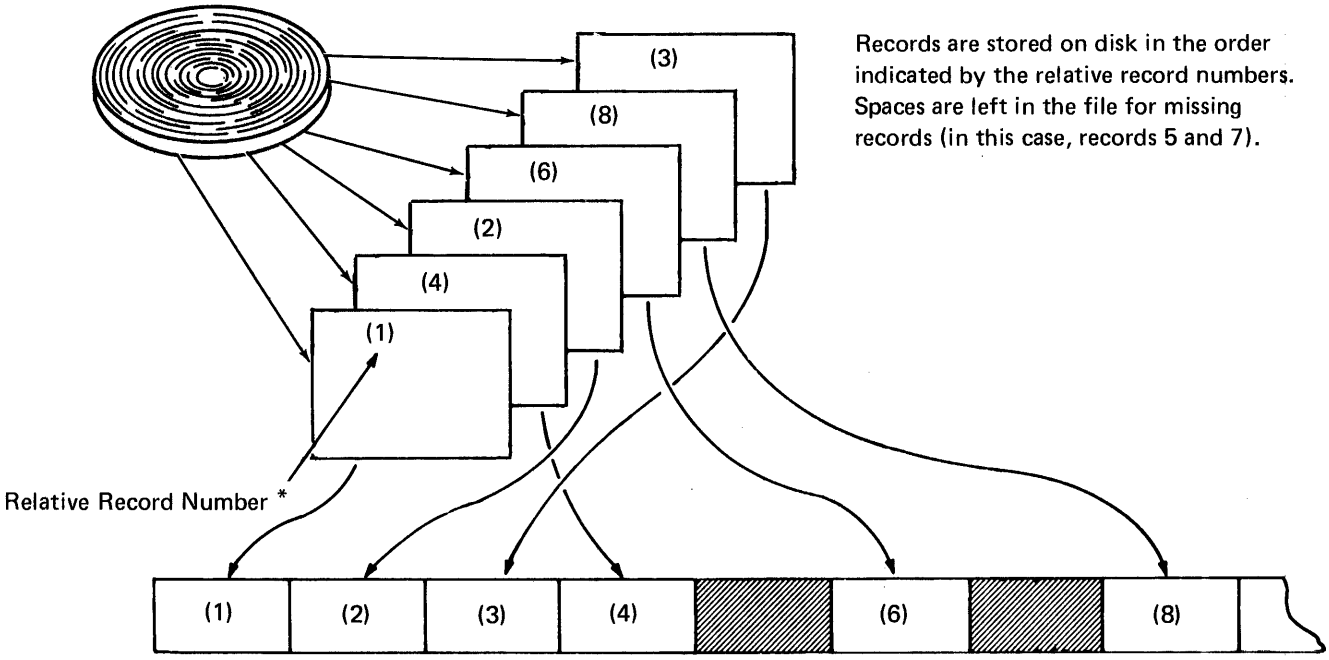


Figure 19. Indexed File Organization

Direct Files

Direct files are disk files in which records are assigned specific record positions. Regardless of the order in which the records are put in the file, they always occupy a specific position (a specific disk address). Relative record numbers identify the relative position of a record within the file.

Before a direct file is loaded, the entire disk area for the direct file is cleared to blanks. Spaces are reserved in a direct file for records not available at the time the file is loaded (Figure 20).



* The programmer usually derives relative record numbers from information in the records.

Figure 20. Direct File Organization

Sequential Files

Sequential files are files in which the order of the records is determined by the order in which the records are put in the file. For example, the tenth record put in the file occupies the tenth record position (Figure 21). Files other than disk files are always sequential files. Disk files can be sequential, direct, or indexed files.

Additional Input/Output Area

Normally the program uses one input/output area for each file. A second area, however, can be used for direct or sequential files specified as input or output files in column 15.

Additional input/output areas cannot be used for table files, demand files or for disk files using a shared I/O area. The use of two I/O areas increases the efficiency of the program. However, it also increases the size of the program. Therefore, before you indicate that two areas are to be used for a file, be sure that the increase in size will not make your program exceed the capacity of your system.

Additional I/O devices cannot be specified for disk files with a shared input/output area (column 48 of the control specifications sheet). If both additional I/O and shared input/output areas are specified, the additional I/O area specification is ignored and a warning message is given.

ADDROUT Files

When describing an ADDROUT file, you must place a T in column 32. The ADDROUT file must be a disk file. See *Column 28 (Mode of Processing)* for a description and example of ADDROUT processing.

COLUMNS 33-34 (OVERFLOW INDICATORS)

Entry	Explanation
OA-OG, OV	An overflow indicator is used to condition records in the file. The indicator specified is the one used.
Blank	No overflow indicator is used.

Columns 33-34 apply to output files assigned to the printer. Use these columns to indicate that you are using an overflow indicator to condition records being printed in the printer output file.

Any overflow indicators used in a program must be unique for each file. The use of overflow indicators is described under *Overflow Indicators* in the *Supplementary Information* section. Note that only one overflow indicator can be assigned to a file. Do not assign overflow indicators to a console file.

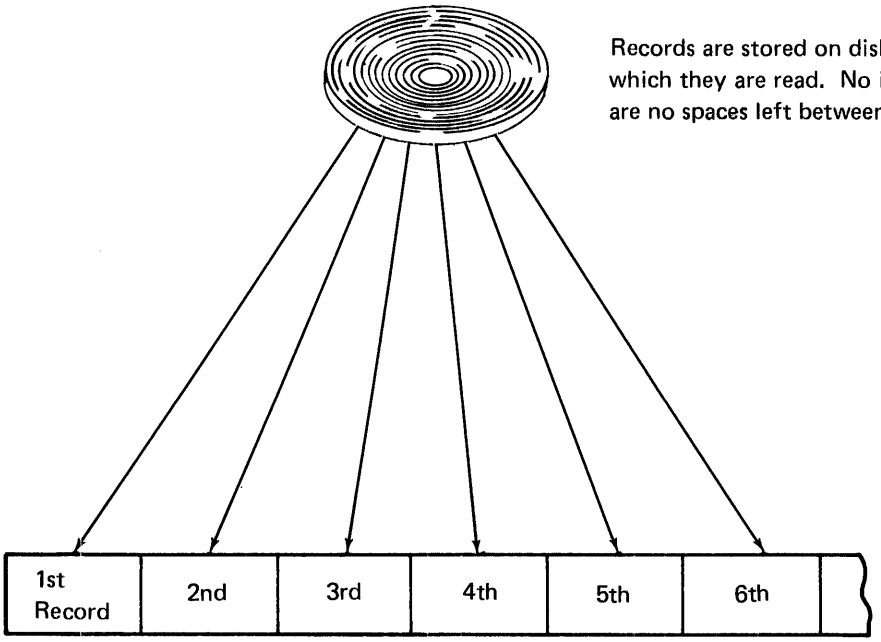


Figure 21. Sequential File Organization

COLUMNS 35-38 (KEY FIELD STARTING LOCATION)

Entry	Explanation
1-4096	Record position in which the key field begins

Columns 35-38 apply to indexed disk files only. An entry must be made in these columns for an indexed disk file. This entry identifies the record position in which the key field begins. The key field of a record is the field that contains the information that identifies the record. The information is used in the index portion of the file. The key field must be in the same location in all of the records in the file. The number you place in these columns must end in column 38. Leading zeros can be omitted.

COLUMN 39 (EXTENSION CODE)

Entry	Explanation
E	Extension specifications further describe the file.
L	Line counter specifications further describe the file.

Column 39 applies only to (1) table and array files that are to be read during program execution, (2) record address files, and (3) output files assigned to the printer. Use column 39 to indicate whether the file is further described on the extension specifications sheet or the line counter specifications sheet. Describe output files that are assigned to the printer on the line counter specifications sheet. Table, array, and record address files must be described on the extension specifications sheet.

COLUMNS 40-46 (DEVICE)

Entry	Explanation
DISK	Disk
KEYBORD	Keyboard
PRINTER	132-position printer
BSCA	Binary synchronous communications adapter
CONSOLE	Interactive data file
CRT	Display screen
SPECIAL	Used for a device not supported directly by RPG II

Use columns 40-46 to identify the input/output device used for the file. All entries must begin in column 40. The device used depends upon the form of the records (Figure 22). Figure 23 shows the columns that can be used for the devices named.

For information on RPG II telecommunications (BSCA), see *IBM System/32 RPG II Telecommunications Programming Reference Manual,* SC21-7597.

CRT (Display Screen)

The display screen can be used as an output device for normal and exception output. (See *Output Specifications, Column 15* in this part for more information on exception output.) Any alphameric character can be written on the display screen. As many as 40 characters can be written across the width of the screen, a maximum of six such lines can appear on the screen at one time.

Data moves onto the screen from bottom to top: after the bottom (sixth) line is written on the screen, if space 1 after is coded for the sixth line, the top line moves off. Data is written on the display screen at the normal output times (total and detail) or at calculation time for exception output.

The display screen is designed to display messages and instructions to the operator and to display operator responses. It should not be used interchangeably with the printer as a major output device because of the speed with which data moves on and off the screen.

Output operations such as spacing and skipping can be specified with some restrictions. You can specify spacing before and after (0-3 entry in columns 17-18 of the output specifications sheet). You can specify a skip before to line 01 only (01 entry in columns 19-20 of the output specifications sheet). Make this specification whenever you want to erase data from the display screen. If a skip before to any line other than 01 is specified, the system assumes the entry to be 01 and the screen is erased. You cannot specify a skip after (columns 21-22 of the output specifications sheet) for display screen files. When a file is printed on the display screen over a previous line, the previous line is erased. Edit codes, edit words, and output indicators can be specified for display screen files.

File	Form	Possible Devices
Primary Input Files	Keyed in by operator	KEYBORD
Primary or Secondary Input Files	Disk	DISK
	Keyed in by operator	CONSOLE
	Special device	SPECIAL
	TP lines	BSCA
Record Address Files Containing Record-Key Limits	Disk	DISK
	Keyed in by operator	CONSOLE
Record Address Files Containing Disk Addresses (ADDROUT File)	Disk	DISK
Demand Files	Disk	DISK
	Keyed in by operator	KEYBORD
	Keyed in by operator	CONSOLE
Demand Files	Special device	SPECIAL
	TP lines	BSCA
Table Files	Disk	DISK
Chained Input Files	Disk	DISK
Update Files (Primary Secondary, or Chained)	Disk	DISK
	Special device	SPECIAL
Combined Files (Primary or Secondary)	Special device	SPECIAL
Output Files	Disk	DISK
	Printed lines	CRT
	Printed pages	PRINTER
	Special device	SPECIAL
	TP lines	BSCA

● **Figure 22. Device Assignment**

Console

Use CONSOLE as the device name in one of two ways: (1) for a record address file, (2) for an input file for interactive data entry. If CONSOLE is used for a record address file, it must be further defined by extension specifications.

Keyboard

The entries CONSOLE and KEYBORD refer to the same physical unit which includes both the keyboard and the display screen. Use CONSOLE when you want to enter data records in an interactive mode; use KEYBORD when you want to use the KEY or SET operation codes.

Note: If KEYBORD is specified for the primary input file, you must provide a means of exit from the program (that is, you must provide for the setting on of the LR indicator).

Printer

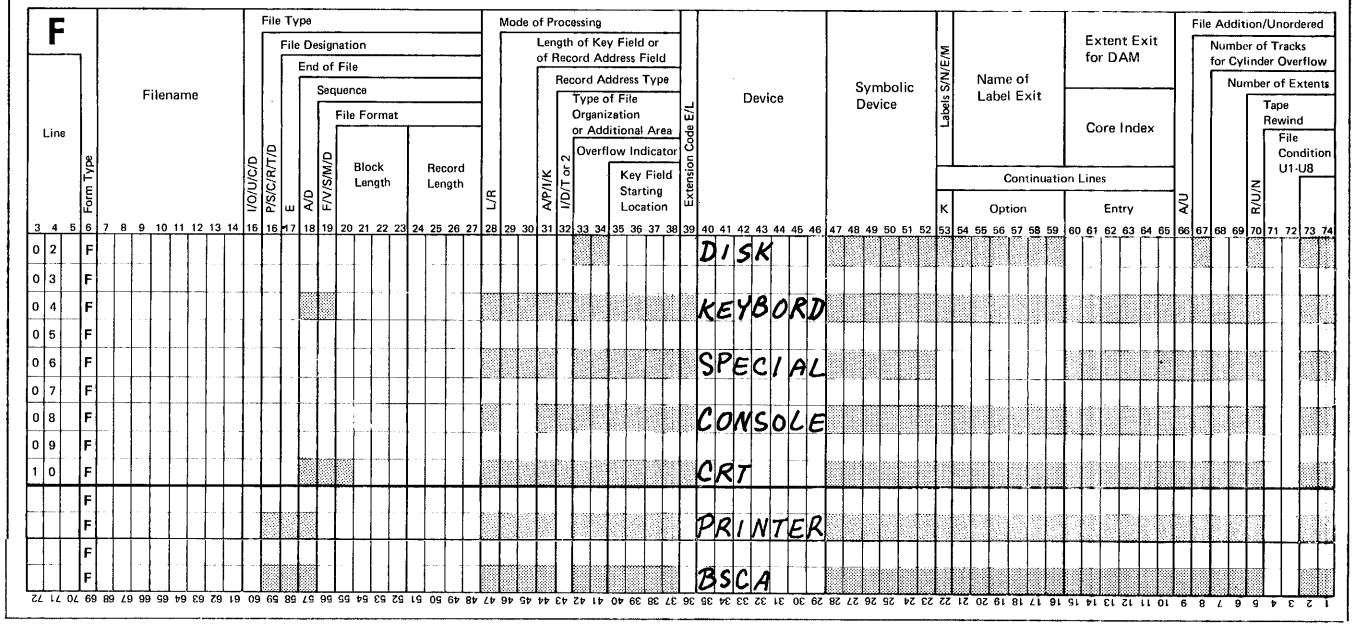
The print unit allows you to produce a separate output file in each program on a 132-position printer. Only one printer file is allowed per program.

SPECIAL Device Support

You can process files using devices not supported by RPG II. To do this, you must indicate that the file is handled by a special device (SPECIAL in columns 40-46 of the file description specifications sheet).

You must also supply a subroutine to perform the I/O operations required to transfer data between the special unit and main storage (subroutine name in columns 54-59 of the file description specifications sheet). Control cannot be transferred from one user assembler subroutine to another user assembler subroutine.

File Description Specification



Shaded columns must be blank

● **Figure 23. Columns That Apply to Device Named**

File Description Specifications for SPECIAL Device

The following file description specifications apply to files assigned to SPECIAL device.

Column	Entry
7-14	Valid RPG II filename.
15	I, O, U, or C.
16	P, S, D, or blank.
17	E or blank.
18	A, D, or blank.
19	F.
20-23	Block length.
24-27	Record length.
28-31	Must be blank.
32	1-9 or blank.
33-39	Must be blank.
40-46	SPECIAL.
47-53	Must be blank.
54-59	Name of the user-written subroutine which performs the input/output operations. The subroutine name must be in the form SUBRxx, where x is any alphabetic character or in the form SRyzzz where y is any of the following 15 characters: B, C, D, F, G, H, I, L, M, O, P, R, S, T, or U; and Z is any of the following 16 characters: A, B, C, D, F, G, H, I, L, M, O, P, R, S, T, or U.
60-70	Must be blank.
71-72	U1-U8 or blank.
73-74	Must be blank.

The following can be used with SPECIAL files:

- FORCE operation code.
- READ operation code.
- File translation.

The following cannot be used with SPECIAL files:

- CHAIN operation code.
- Spacing and skipping.
- *PRINT.

SPECIAL files can only be processed consecutively. See Figure 24 for possible file description entries for SPECIAL files.

COLUMNS 47-52

Columns 47-52 are not used. Leave them blank.

COLUMN 53 (CONTINUATION LINES–K)

Entry	Explanation
K	Continuation record

Continuation records provide additional information about the special file being defined. You can specify one continuation record for each special file. When specifying a continuation record, columns 54-59 (continuation line option) must be coded. Figure 24 shows an example of the coding necessary on the file description specifications sheet for a continuation line.

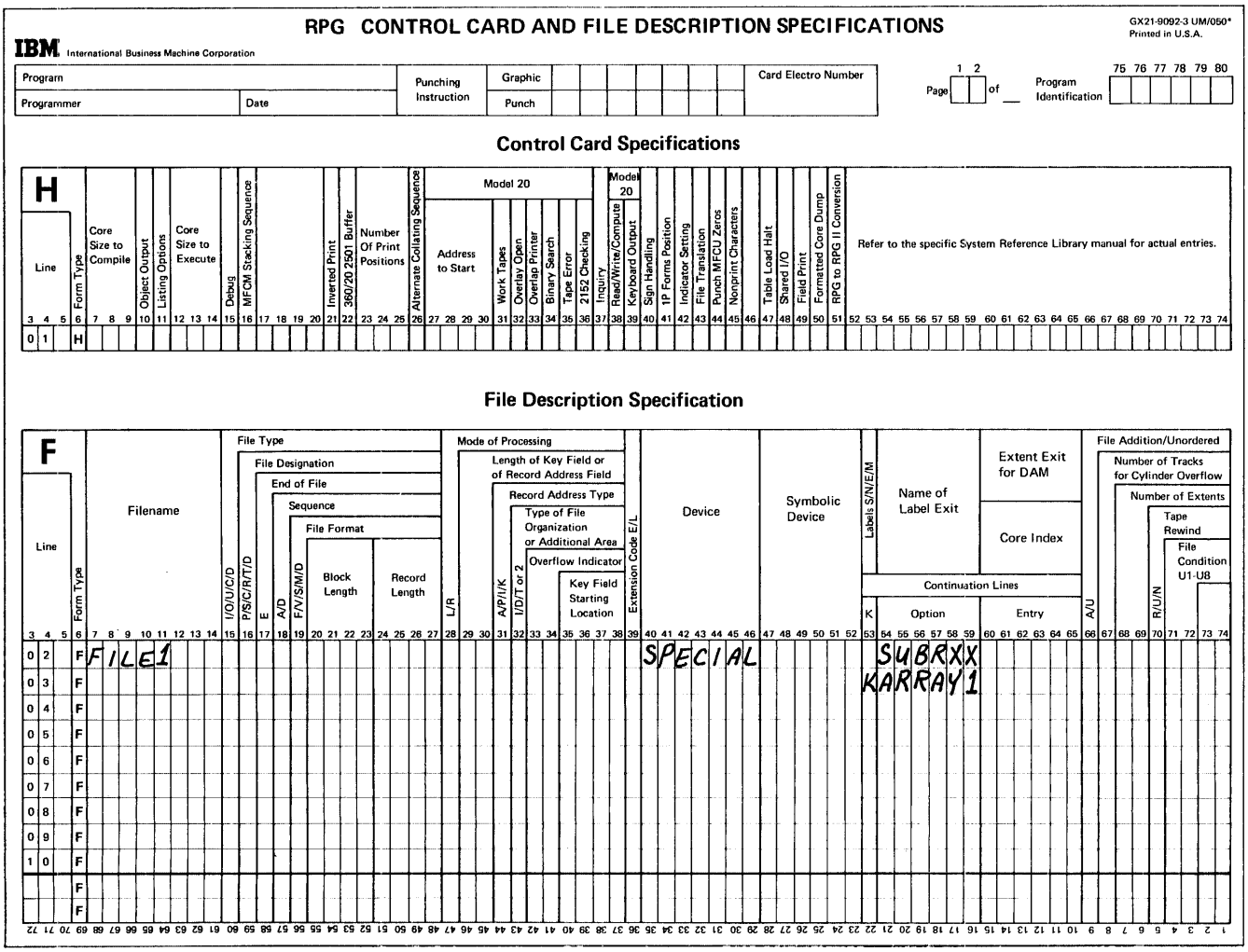


Figure 24. Continuation Record

COLUMNS 54-59

Name of Label Exit

Entry	Explanation
SUBRxx	Name of the user-written subroutine which performs the I/O operation for a SPECIAL device (x = any alphabetic character).
SRyzzz	Name of the IBM-written subroutine (6-character name in library is #$yzzz), which performs the I/O operation for a device supported by SPECIAL (y = any of the following 15 characters: B, C, D, F, G, H, I, L, M, O, P, R, S, T, or U; z = any of the following 16 characters: A, B, C, D, F, G, H, I, L, M, O, P, R, S, T, or U).
Blank	No SPECIAL device is used.

Note: Subroutines of the type SRyzzz can be overlaid. Modifications within the subroutine code may or may not be present the next time the subroutine is used.

Columns 54-59 must contain an entry for each data file assigned to a SPECIAL device. Use these columns to specify the subroutine which performs the input/output operations for a file assigned to a SPECIAL device. The subroutine name entered in columns 54-59 can be from four to six characters long. The first four characters must be SUBR; the remaining characters can be any alphabetic character.

Continuation Line Option

Entry	Explanation
Table/ array name	Name of table/array used by the user-written subroutine. The array name cannot be ASCII or BUFOFF.

COLUMNS 60-65 (STORAGE INDEX)

Entry	Explanation
6-9999	Number of bytes reserved for the storage index.
Blank	No storage index is kept in storage.

Columns 60-65 apply only to indexed files processed randomly using the CHAIN operation code. Storage index cannot be specified with shared I/O. Entries must be right-justified. Leading zeros are not required. You can specify up to 9999 bytes for the storage index. This provides faster retrieval of records.

The storage index is a table containing entries for sectors in the index portion of a disk data file. Each entry contains a sector address and the lowest key field associated with the next sector of the file index. The file index is that portion of a disk data file containing the position of the records in the data file. Each sector of the file index contains entries consisting of a key and a 3-byte disk address for each record in the file. The last sector of the file index contains all FFs to indicate the end of the index. Figure 25, part B shows the layout of the index for the indexed file INDEXT. Figure 25, part A shows the layout of the first sector of the file index. Figure 25, part C shows the most efficient storage index for the file INDEXT. Notice that the storage index contains one entry for each sector of file index minus one. Each entry of storage index contains the address of a sector of file index plus the lowest key from the next file index sector.

Use of the storage index significantly reduces the amount of time needed to process an indexed file. It enables the system to go more directly to the specific record you want by searching only a small portion of the file index. Without the storage index, all file index entries which precede the record you want must be searched. Using the storage index shown in part C of Figure 25, the record with field 125 can be found in the following manner:

1. Search the storage index until the first key field higher than record 125 is located. In this instance the key is 151; therefore, field 125 has file index sector 5 associated with it.
2. Search sector 5 in the file index until key 125 is located.
3. Chain directly to the associated data record.

In columns 60-65, specify the number of storage positions (bytes) you want reserved for the storage index. Using the amount of main storage you specify, the system builds the most efficient storage index it can. The storage index is built immediately before your RPG II program is executed.

For efficient processing, the storage index you specify should be large enough to contain one entry (key length + 2) for the number of file index sectors minus one. The number of bytes required for the most efficient storage index is equal to the (key length + 2) multiplied by the (number of file index sectors - 1). For example, if file INDEXT has seven sectors of file index space and a key length of 4, the most effeicient storage index is 36 bytes, (4 + 2) times (7 - 1). If you specify more storage than required for efficient processing (results of the preceding calculation), the excess storage is not used.

To determine the number of file index sectors (for use in calculating storage index size), use the following calculation:

1. Find the number of records in the file and the number of records available by referring to the system catalog.

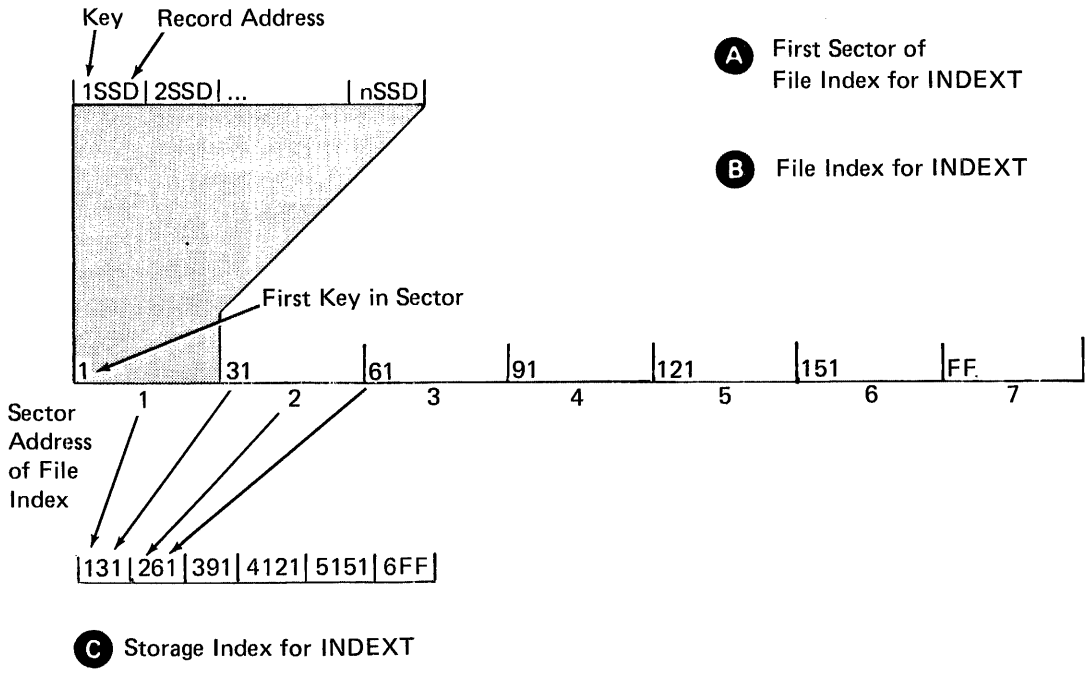
2. Add the number of records in the file and the number of records available to determine the total records in the file.

3. Divide the number of bytes in a sector (256) by the (record key length + 3) to determine the number of record index entries per sector. If the quotient is not a whole number, round it *down* to the nearest whole number.

4. Divide the total number of records in the file (step 2) by the number of record index entries per sector (step 3) to determine the number of sectors of index in the file. If this quotient is not a whole number, round it *up* to the next whole number.

5. Add one for the final file index sector of FFs to the number resulting from the calculation in step 4.

If the storage space specified in columns 60-65 is not large enough to contain an entry for every sector of file index, the system divides the given area into as many entries as there is room for with each entry pointing to a larger sector of the file index. As the number of entries in the storage index becomes fewer, the amount of processing time increases. If the storage index specified is not large enough for two entries, enough storage for two entries is allocated.



● **Figure 25. Calculating Storage Index**

COLUMN 66 (FILE ADDITION)

Entry	Explanation
A	New records are added to the file.
U	Records are to be loaded for an indexed file in unordered sequence (nonascending sequence).

Column 66 applies to consecutive and indexed disk files. This column indicates:

- The program is to add new records to the file (*Example 1*).
- The program is to load records in an unordered sequence (*Example 2*).

Adding Records to a File (A)

Records added to a consecutive file are added at the end of the file.

Records added to an indexed file are added at the end of the file and entries for the new records are made in the index. The index is then reorganized so that the record keys (including the new ones) are in ascending order. File addition (column 66) cannot be specified for indexed files from which records are read using the sequential within limits method. Records added to an indexed file should be in ascending sequence to achieve better performance.

After a file has been loaded on disk, it may be necessary to add records to the file. Records can be added at detail, total, or exception time during the program cycle. When chained files are specified with add, the records to be added may:

- Contain keys or record numbers that are above the highest presently in the file. In this case, the records constitute an extension of the file, or
- Contain keys or record numbers that are either lower than the lowest presently in the file, or fall between those already in the file.

If records are to be added to an indexed file sequentially:

- The key of the record to be added must be lower than the key retrieved and higher than the preceding key, or
- The file must be at end of file.

To add a record to any indexed file processed randomly, the program searches the index to the file to determine if the record is on the file; if it is, a halt occurs. Otherwise, the record is added.

To add a record to a sequentially processed file, the program determines if the key of the record to be added is lower than the key currently in process and higher than the preceding record. If these requirements are not met, a halt occurs; otherwise, the record is added.

Note: Adding records to a file also requires a corresponding ADD entry in columns 16-18 of the output specifications sheet.

Loading Records in an Unordered Sequence (U)

Unordered load (U in column 66) is specified when an indexed file is to be built from records in an unordered sequence. After records are loaded and an index is built in the unordered sequence, the index is sorted into ascending sequence. In the following chart, combinations of entries in column 15 and column 66 show the functions that can be performed for indexed files (I in column 32).

Column 15	Column 66	Function
O	Blank	Load records in ascending key sequence to an indexed file.
O	U	Load records in unordered key sequence to an indexed file.
O	A	Add records to an existing indexed or sequential file.
I	Blank	Read records of an indexed file without adding new records or updating records.
I	A	Read records of an indexed file and add new records to the file that are not presently there. No updating is performed.
U	Blank	Update records of an indexed file without adding new records.
U	A	Update records of an indexed file and add new records to the file.

Examples

Example 1

Figure 26 shows how records can be added to an indexed disk file. The new records are contained in another disk file, DISKIN. The file INDEXED is the existing disk file to which new records are added. A printer file, PRINT, provides a report showing all the records in DISKIN with an indication of which records are added to INDEXED and which records are not added.

On the file description specifications sheet, an A must appear in column 66 for the file INDEXED. On the output specifications sheet, ADD must appear in columns 16-18 for the new record to be added.

As defined on the input specifications sheet, all the records in DISKIN should have an A in position 120. The code identifies a record to be added to the disk file, and this record type is assigned indicator 01. On the output specifications sheet, notice that when 01 is on, the data from DISKIN is written on the disk file INDEXED and is also printed on the file PRINT to keep a visual report of new records.

There may be records in DISKIN that do not belong in that file, or some records may have an error. These records are identified on the input specifications sheet as not having the character A in position 120. These records turn on indicator 02, and are not to be added to the disk file INDEXED. However, these records are printed on the file PRINT for a visual report, but they must be identified in the printed report as records that were not added to the disk file INDEXED. On the output specifications sheet, the constant RECORD NOT ADDED is printed only on indicator 02, indicating a record that was not added to the disk file. In this manner, a report of all records in DISKIN is printed and the records not added to INDEXED are identified by the constant RECORD NOT ADDED.

File Description Specification

Line	Form Type	Filename	File Type (I/O/U/C/D)	File Designation (P/S/C/R/T/D)	End of File (E)	Sequence (A/D)	File Format (F/V/S/M/D)	Block Length	Record Length	Mode of Processing (L/R)	Length of Key Field or of Record Address Field	Record Address Type (A/P/I/K)	Type of File Organization or Additional Area (I/D/T or 2)	Overflow Indicator	Key Field Starting Location	Extension Code (E/L)	Device	Symbolic Device	Labels (S/N/E/M)	Name of Label Exit	Extent Exit for DAM / Storage Index	Continuation Lines (K)	Continuation Lines (Option)	Continuation Lines (Entry)	File Addition (A/U)	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind (R/U/N)	File Condition U1-U8
02	F	DISKIN	I	P	E	F	F	120	120								DISK												
03	F	INDEXED	O			F	F	256	64			9	A	I		2	DISK								A				
04	F	PRINT	O			F	F	132	132					OA			LPRINTER												
05	F																												
06	F																												
07	F																												
08	F																												
09	F																												
10	F																												
	F																												
	F																												

Line Counter Specifications

Line	Form Type	Filename	1 Line Number	1 FL or Channel Number	2 Line Number	2 OL or Channel Number	3 Line Number	3 Channel Number	4 Line Number	4 Channel Number	5 Line Number	5 Channel Number	6 Line Number	6 Channel Number	7 Line Number	7 Channel Number	8 Line Number	8 Channel Number	9 Line Number	9 Channel Number	10 Line Number	10 Channel Number	11 Line Number	11 Channel Number	12 Line Number	12 Channel Number
11	L	PRINT	66	FL	60	OL																				
12	L																									
	L																									

RPG INPUT SPECIFICATIONS

IBM International Business Machine Corporation

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

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

Programmer | Date

Page ___ of ___ | Program Identification

Line	Form Type	Filename	Sequence (OR)	Sequence (AND)	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1 Position	1 Not (N)	1 C/Z/D	1 Character	2 Position	2 Not (N)	2 C/Z/D	2 Character	3 Position	3 Not (N)	3 C/Z/D	3 Character	Stacker Select	P/B/L/R	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
01	I	DISKIN		AA			01	120		CA																				
02	I		OR				02	120	NC	A																				
03	I																					2	10		KEY					
04	I																					12	20		FIELD1					
05	I																					21	30		FIELD2					
06	I																					31	64		MORE					
07	I																													
08	I																													
09	I																													
10	I																													

Figure 26 (Part 1 of 2). File Addition

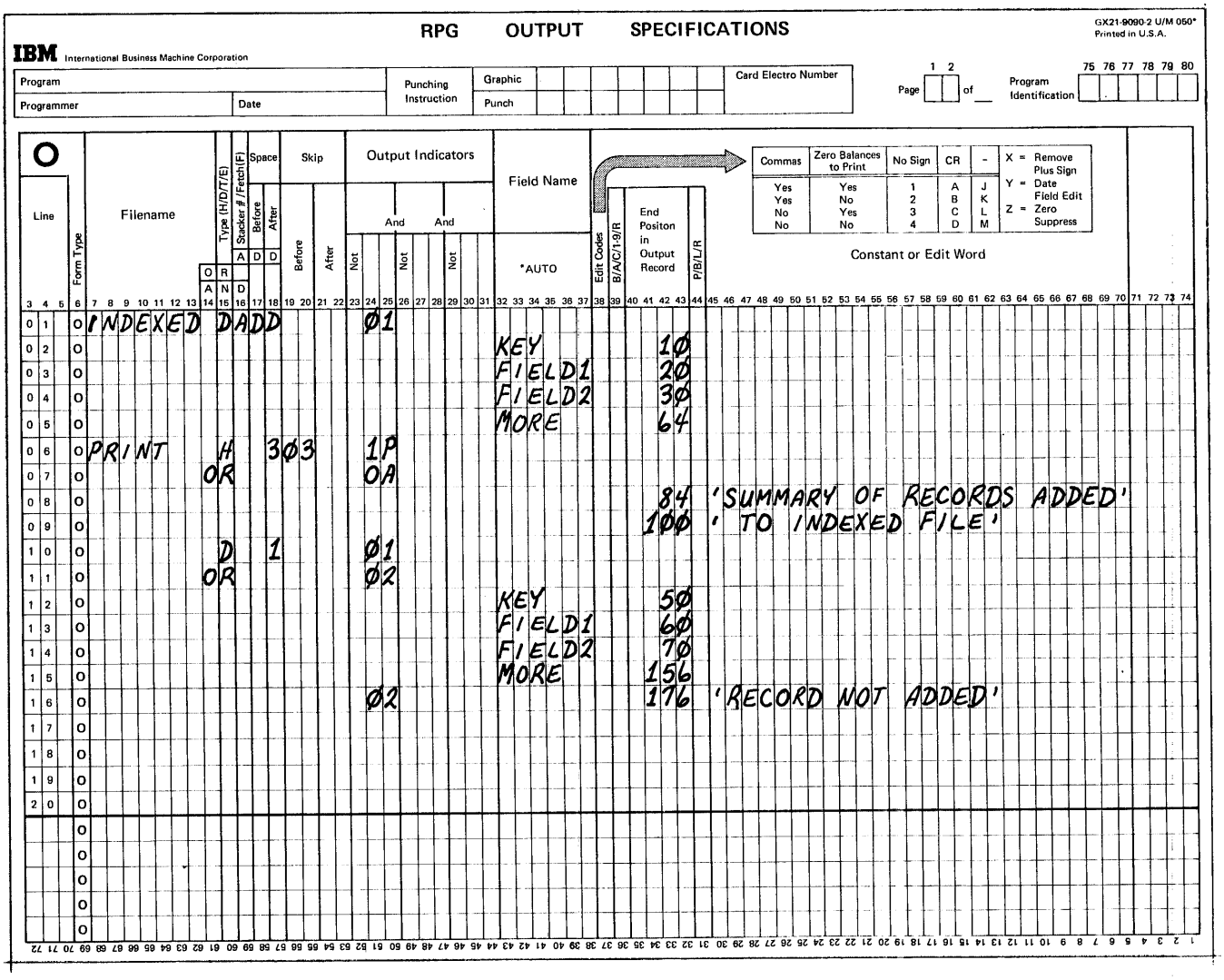


Figure 26 (Part 2 of 2). File Addition

Example 2

Figure 27 shows the coding for unordered loading of an indexed disk file from an unsequenced input disk file. The output file, MASTER, is described as an indexed file to be loaded and processed by record keys. The U in column 66 of the file description specifications sheet indicates that an unordered load is to be done. The input file, INPUT, is described by the input specifications as being unsequenced.

The keys from which the index is to be built appear as the first eight positions of the output record. As the disk file is loaded, the key is extracted from the record and an index entry is built including the location of the record on the disk. After the entire file is loaded and an index entry is constructed for each record, the index entries are sorted into ascending sequence.

COLUMN 67

Column 67 is not used. Leave it blank.

COLUMNS 68-69 (NUMBER OF EXTENTS)

Columns 68-69 are ignored. Leave them blank. For information on how to specify multivolume files using OCL statements, see *IBM System/32 System Control Programming Reference Manual*, GC21-7593.

COLUMN 70

Column 70 is not used. Leave it blank.

COLUMNS 71-72 (FILE CONDITION)

Entry	Explanation
U1-U8	The file is conditioned by the specified external indicator.
Blank	The file is not conditioned by an external indicator.

Columns 71-72 apply to input (excluding table input files), update, and output files. These columns indicate whether the file is conditioned by an external indicator. A file conditioned by an external indicator is used only when the indicator is on. When the indicator is off, the file is treated as though the end of the file is reached. (No records can be read from or written in the file.) See Part 2, *RPG II Programmer's Guide, Indicators, External Indicators* for more information.

COLUMNS 73-74

Columns 73-74 are not used. Leave them blank.

COLUMNS 75-80 (PROGRAM IDENTIFICATION)

See *Common Entries*.

FILE DESCRIPTION CHARTS

The file description charts (Figures 28-36) are for:

- Disk files and presented by disk file organization and processing method.
- Keyboard, console, and printer files.
- The entries in the chart must be made for the processing method and type of file described on that line.
- The shaded columns must be blank for the file described on that line.
- The other columns may be required or optional, but cannot be indicated on the chart because the entries represent information that changes from program to program.

Example

If you are updating an indexed disk file using the CHAIN operation code, see Figure 28 and refer to indexed disk files, random processing by CHAIN operation code. Then choose the chained update file with or without record addition.

In this example, the following columns are required but may change from one program to another: filename, record length, length of key field, and key field starting location. Optional entries are: line, end of file, sequence, block length, cylinder index in main storage, and file condition.

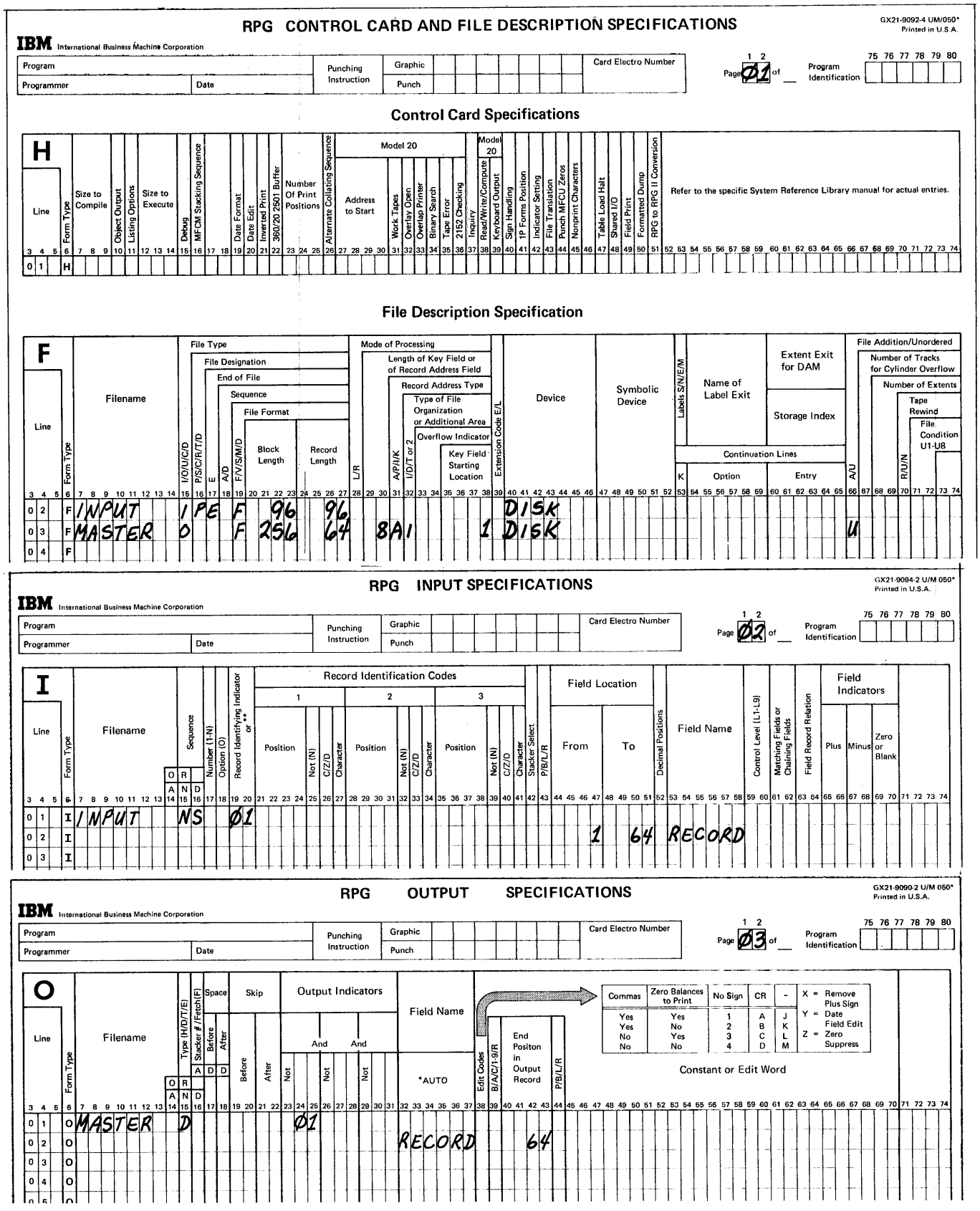


Figure 27. Unordered Load of an Indexed File

Figure 28. Processing Methods for Indexed Disk Files

Type of Processing		I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Core Index	K	Option	Entry	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	R/U/N	File Condition U1-U8
Sequential[1]	by Key, no ADD	I	P			F				A	I				DISK														
	by Key, no ADD	I	S			F				A	I				DISK														
	by Key, no ADD	I	D			F				A	I				DISK														
	by Key, with ADD	I	P			F				A	I				DISK									A					
	by Key, with ADD	I	S			F				A	I				DISK									A					
	by Key, with ADD	I	D			F				A	I				DISK									A					
	by Key, no ADD	U	P			F				A	I				DISK														
	by Key, no ADD	U	S			F				A	I				DISK														
	by Key, no ADD	U	D			F				A	I				DISK														
	by Key, with ADD	U	P			F				A	I				DISK									A					
	by Key, with ADD	U	S			F				A	I				DISK									A					
	by Key, with ADD	U	D			F				A	I				DISK									A					
	by Limits	I	P			F			L	A	I				DISK														
	by Limits	I	S			F			L	A	I				DISK														
	by Limits	I	D			F			L	A	I				DISK														
	by Limits	U	P			F			L	A	I				DISK														
	by Limits	U	S			F			L	A	I				DISK														
	by Limits	U	D			F			L	A	I				DISK														
Random	by CHAIN, no ADD	I	C			F			R	A	I				DISK														
	by CHAIN, with ADD	I	C			F			R	A	I				DISK									A					
	by CHAIN, no ADD	U	C			F			R	A	I				DISK														
	by CHAIN, with ADD	U	C			F			R	A	I				DISK									A					
	by ADDROUT	I	P			F			R	I	I				DISK														
	by ADDROUT	I	S			F			R	I	I				DISK														
	by ADDROUT	U	P			F			R	I	I				DISK														
	by ADDROUT	U	S			F			R	I	I				DISK														
Load	Unordered	O				F				A	I				DISK									U					
	Ordered	O				F				A	I				DISK														
Add Records Only	ADD only	O				F				A	I				DISK									A					

[1] Sequential processing by key or limits must use the file index, which is always arranged in ascending sequence. When an indexed file is processed record by record from beginning to end, the file is processed through the index using the sequential by key method.

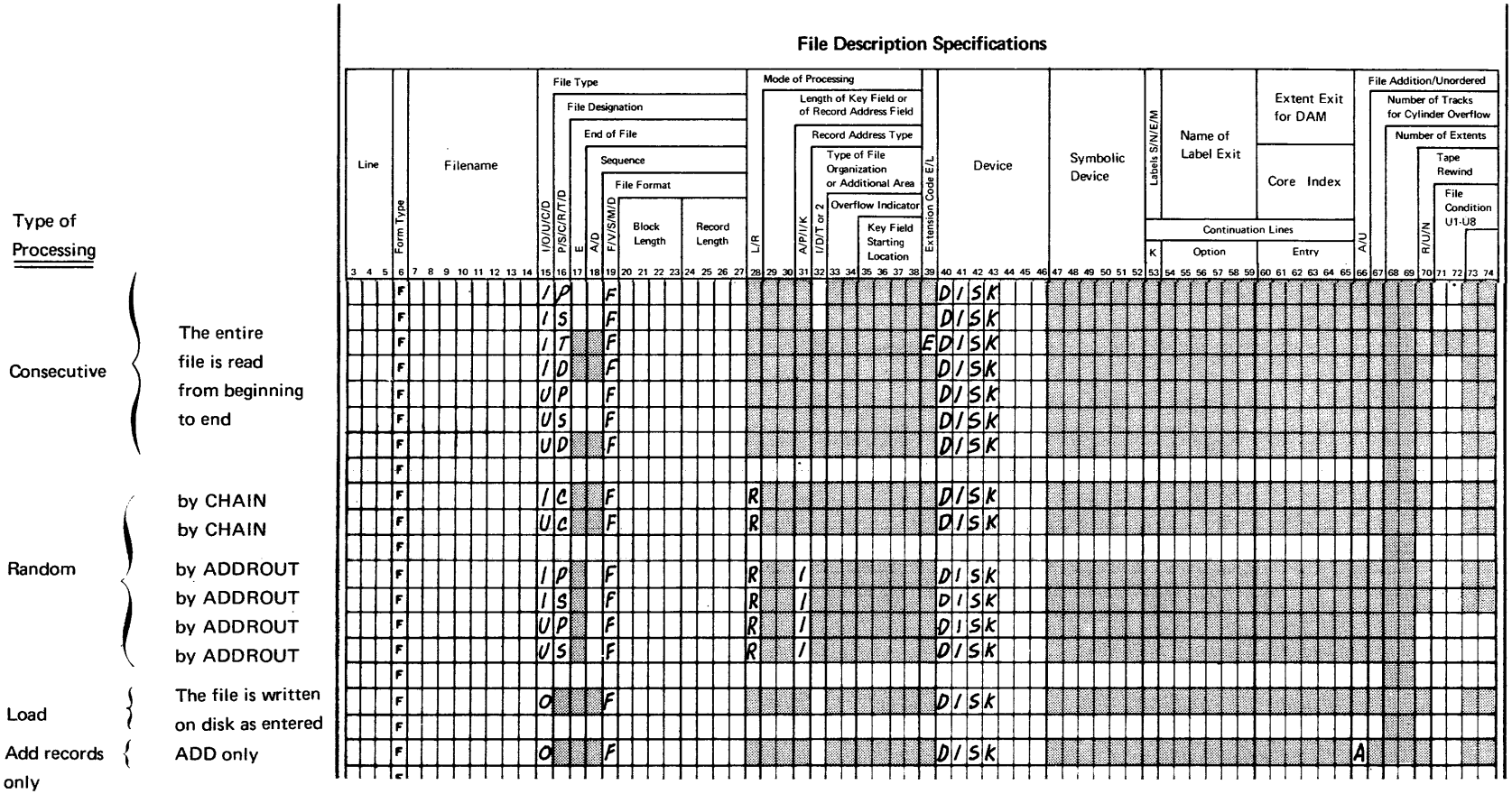
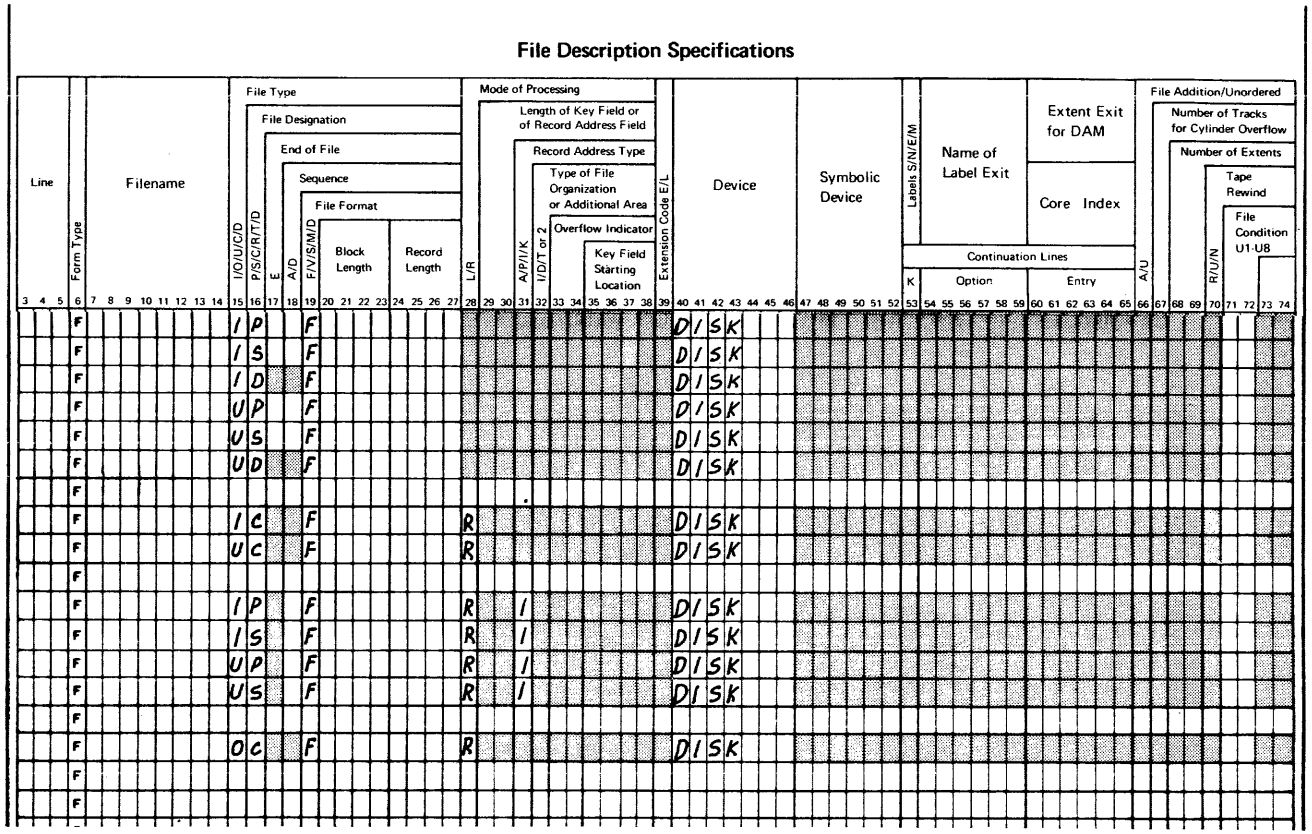


Figure 29. Processing Methods for Sequential Disk Files

Figure 30. Processing Methods for Direct Disk Files

Type of Processing [1]

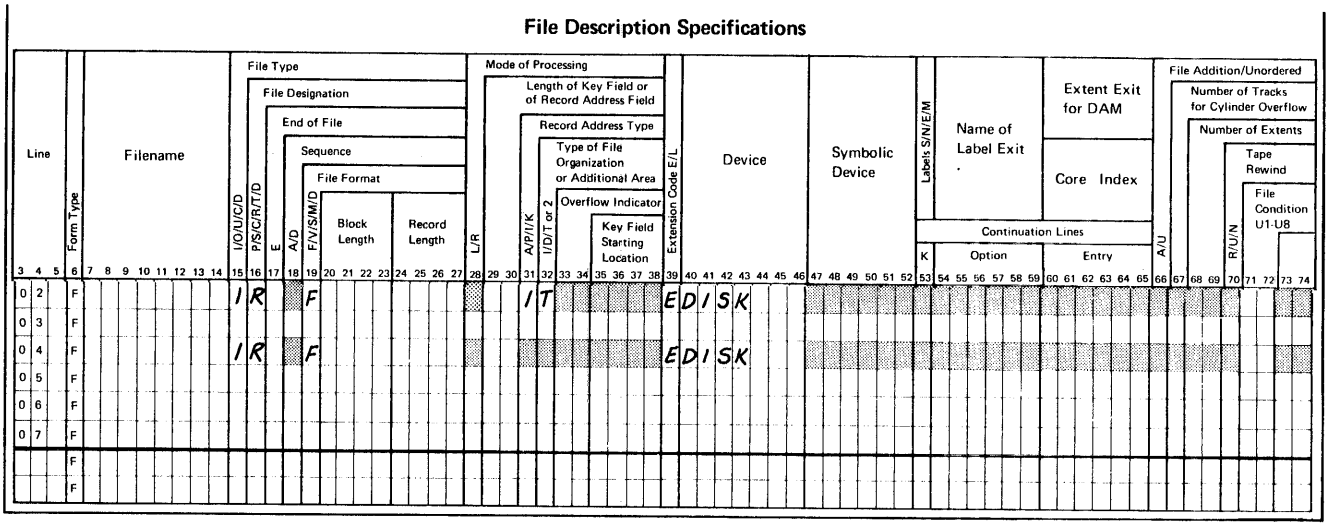
- **Consecutive** — The entire file is read from beginning to end.
- **Random** — by CHAIN; by CHAIN; by ADDROUT; by ADDROUT; by ADDROUT; by ADDROUT
- **Load** — Disk addresses are developed for each record entered.



[1] Records are inserted or changed in a direct file by defining the file as an update file processed consecutively, or an update file processed randomly by the CHAIN operation code.

Record Address Files[1]

Containing: 1. Disk Addresses (ADDROUT file)
2. Record Key Limits

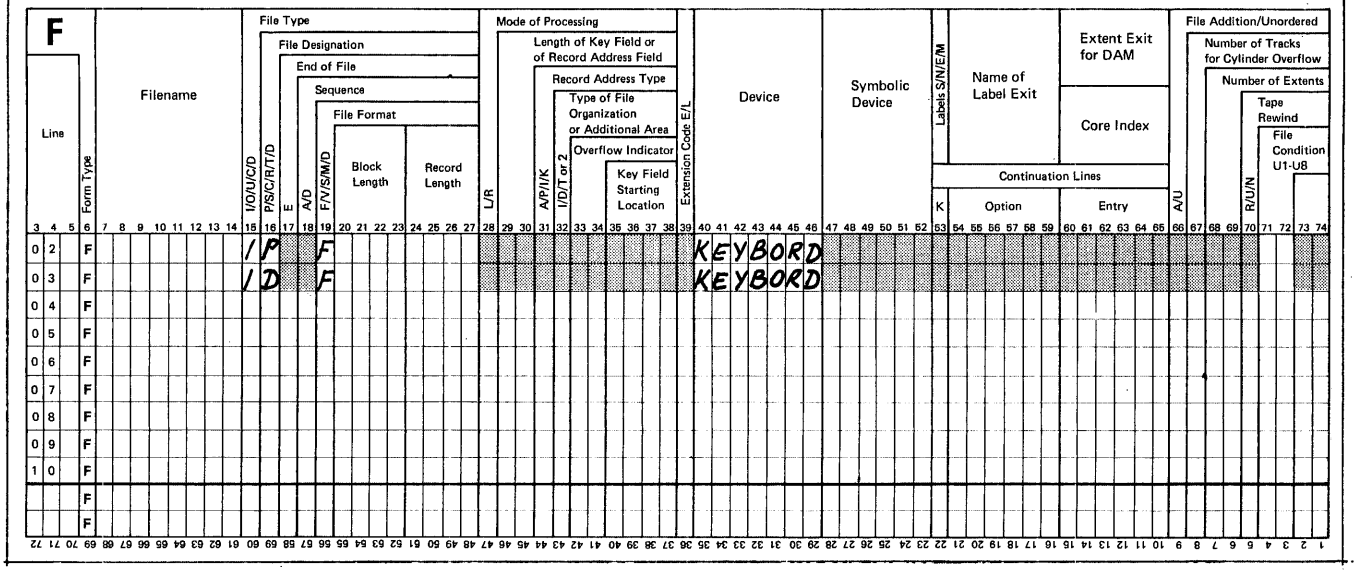


[1]Record address files containing disk addresses may be associated with indexed, sequential, or direct disk files.

Record address files containing record key limits may only be associated with indexed disk files, but may be a disk or console file. (See charts for console files.)

Figure 31. Record Address Files Located on Disk

File Description Specification



- If the keyboard is specified as a *primary input* file, no other *input* files in the program can be specified as *primary* or *secondary* files.

- Input data entered from the KEYBORD device must be defined in calculation specifications for a KEY operation.

Figure 32. Keyboard (KEYBORD) Files

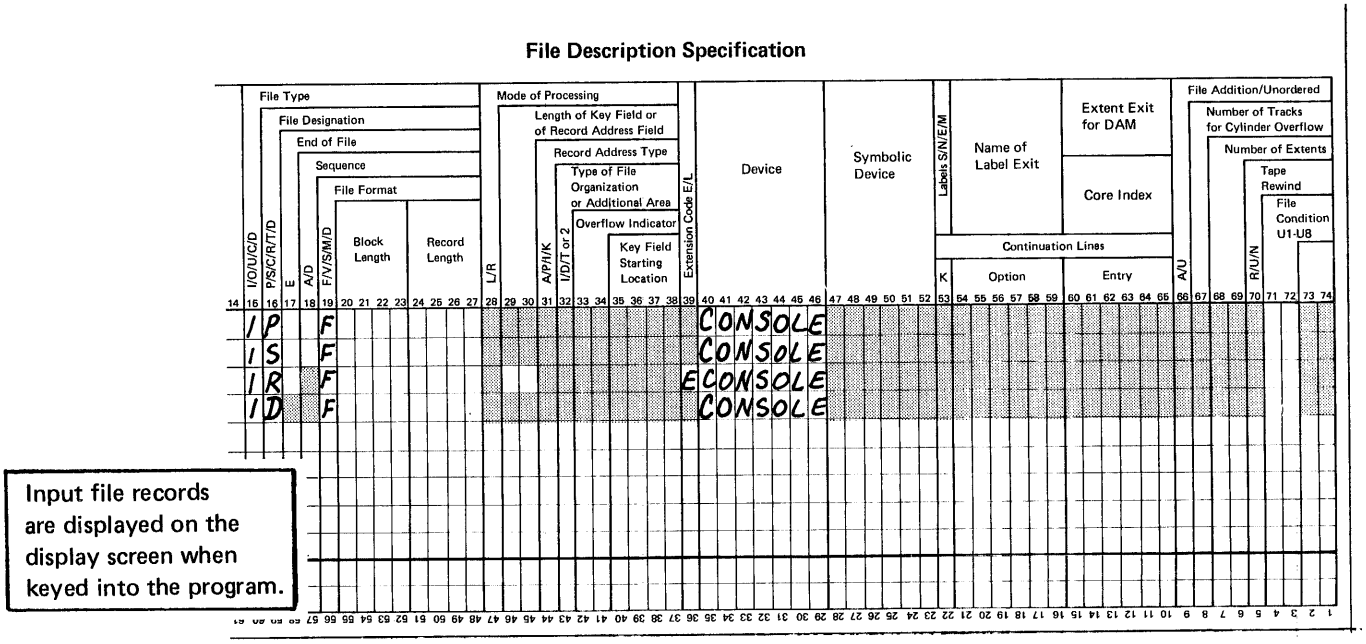


Figure 33. Console Files (Interactive Data Entry)

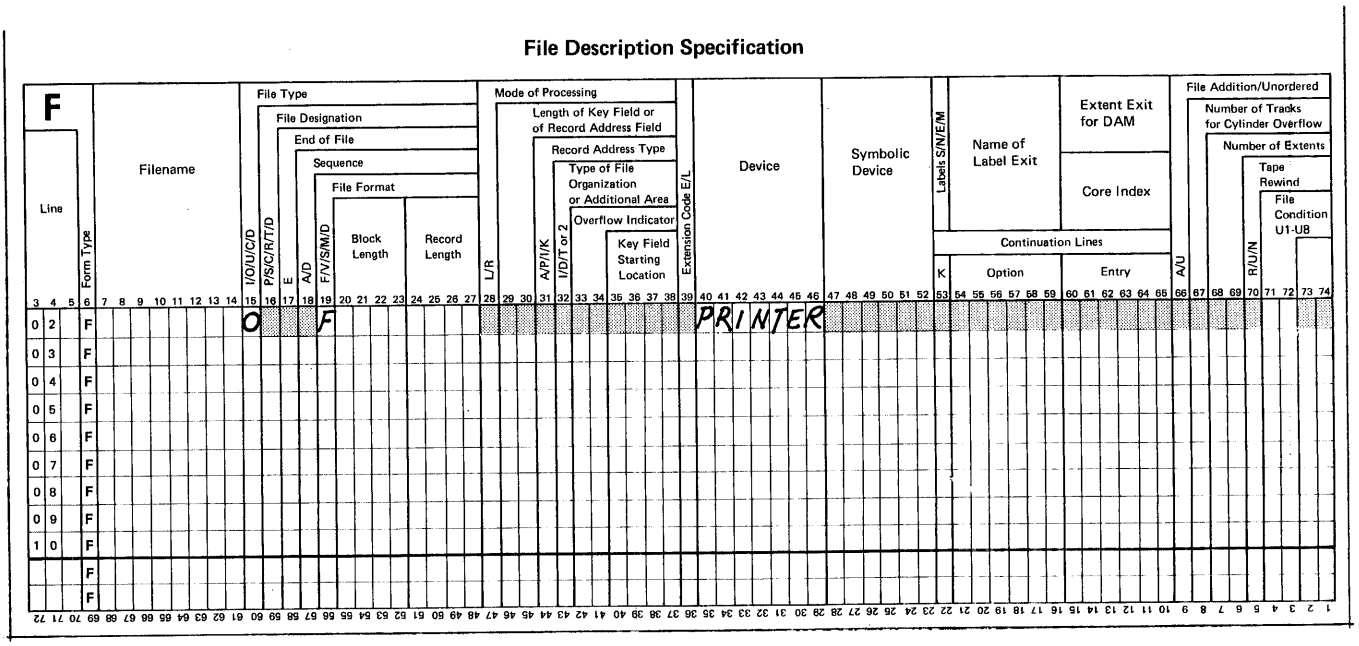


Figure 34. Printer Files

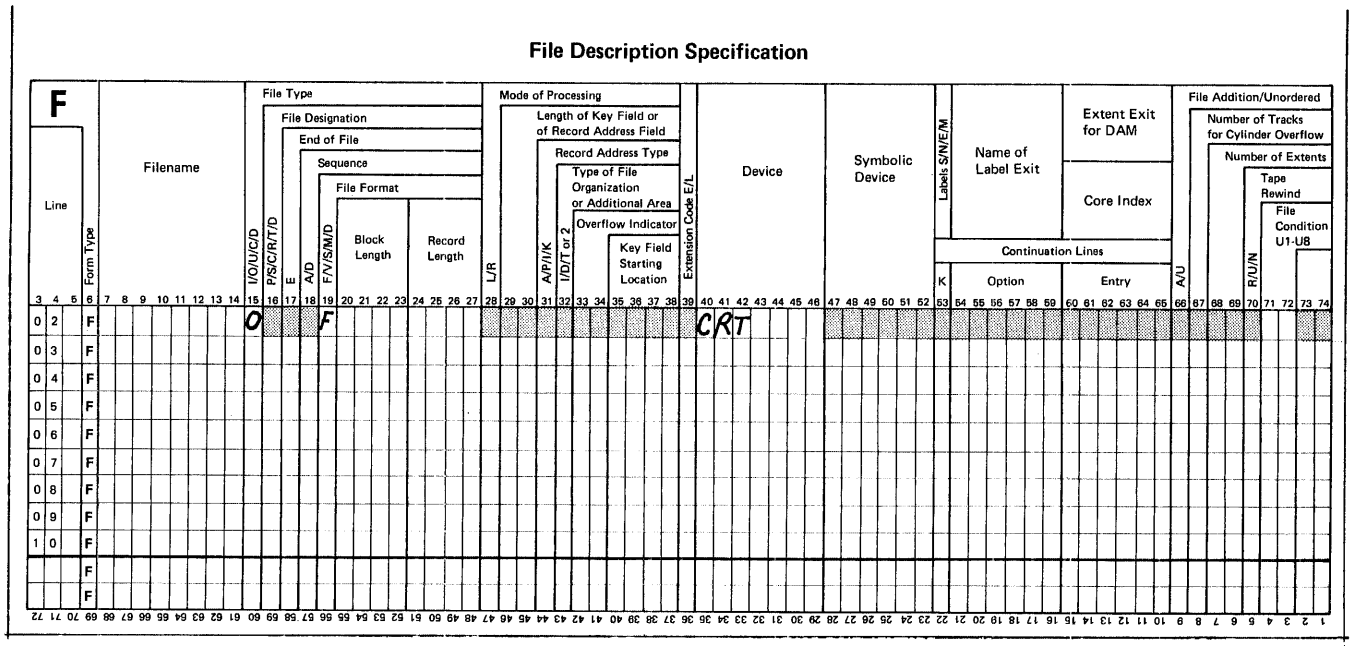


Figure 35. Display Screen Files

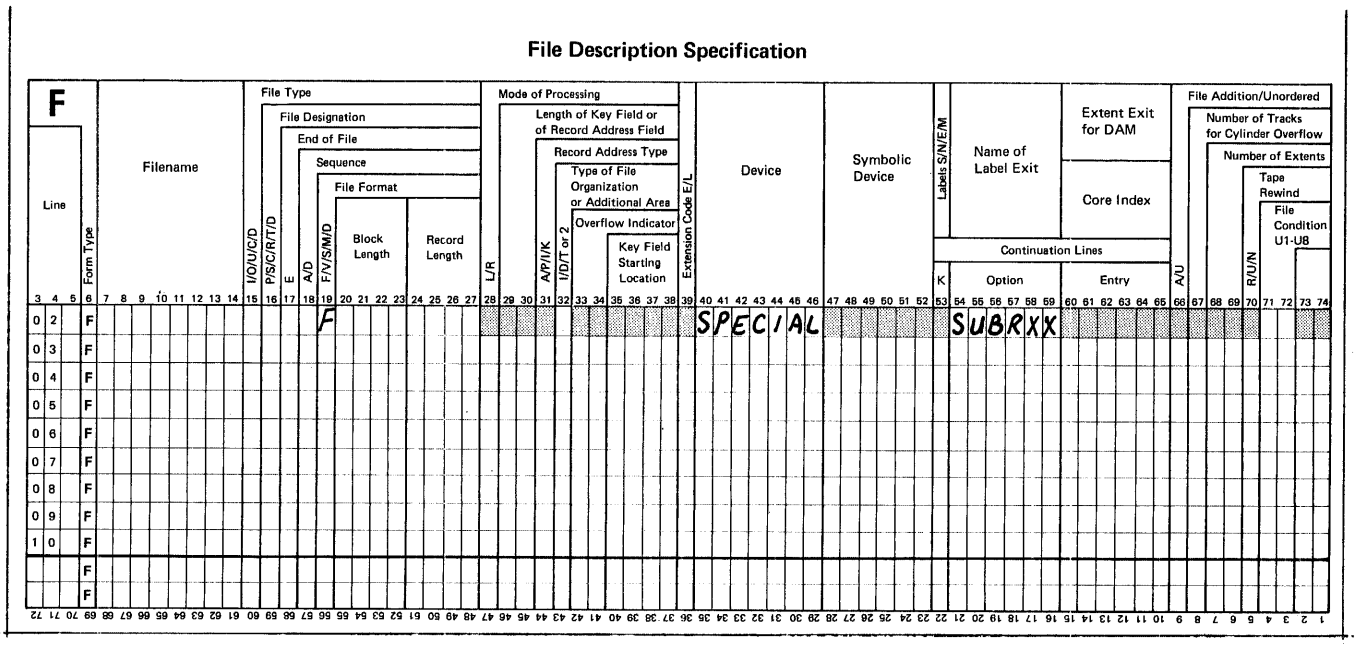


Figure 36. SPECIAL Files

Extension specifications are needed to describe all record address files, tables, and arrays used in your job. Enter these specifications on the extension and line counter specifications sheet (Figure 37).

Record address files require entries on the extension specifications sheet in columns 11-26. Preexecution-time tables and arrays are described in columns 11-45. Compile-time tables and arrays are described in columns 19-45. If an alternating table or array is to be specified with another table or array, it is described in columns 46-57 of the same line as the first table or array. A maximum of 63 tables or arrays can be used in a program. Only 60 of these can be compile-time tables or arrays.

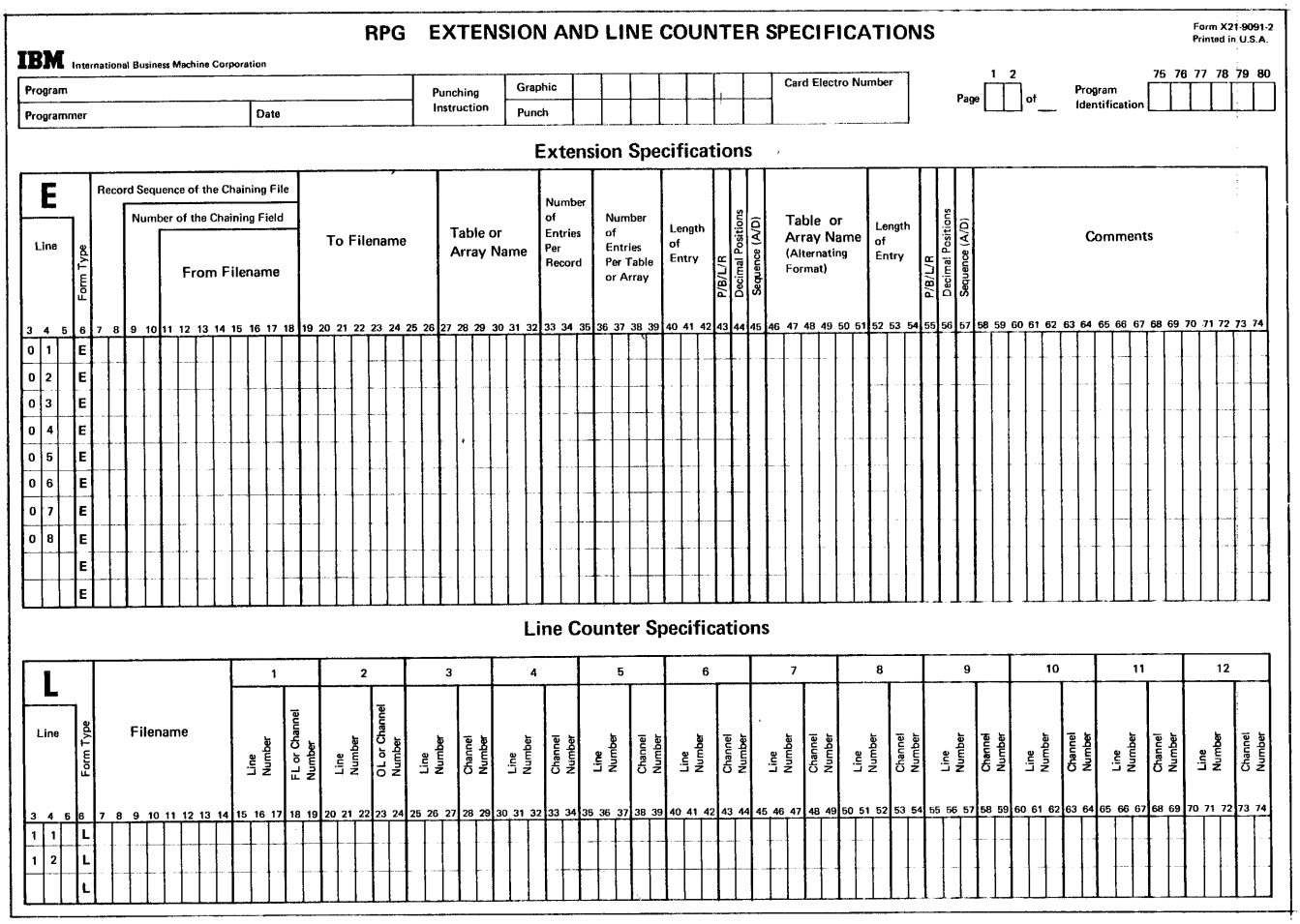


Figure 37. Extension and Line Counter Specifications Sheet

Figure 42 at the end of this chapter shows possible extension specifications. See Part 2, *RPG II Programmer's Guide, Tables and Arrays* for a complete discussion of arrays.

COLUMNS 1-2 (PAGE)

See *Common Entries*.

COLUMNS 3-5 (LINE)

See *Common Entries*.

COLUMN 6 (FORM TYPE)

An E must appear in column 6.

COLUMNS 7-10

Columns 7-10 are not used. Leave them blank.

COLUMNS 11-18 (FROM FILENAME)

Entry	Explanation
Record Address Filename	The name of the record address file.
Table or Array Filename	Table or array loaded at preexecution-time.
Blank	- Table or array loaded at compilation time if there is an entry in columns 33-35. - Array loaded at execution time (via input and/or calculations specifications) if there is no entry in columns 33-35.

Use columns 11-18 to name a table file, array file, or record-address file. Filenames must begin in column 11. The record address filename must always be entered in these columns and in the file description specifications. Leave columns 11-18 blank for compile-time tables or arrays or for arrays loaded via input and/or calculation specifications.

These columns must contain the table or array filename of every preexecution-time table or array used in your program. When the table or array is loaded at compilation time, it is compiled along with the source program and included in the object program. Thus, a table file is not needed in addition to the object program every time the program is run. Only those tables and arrays which do not change often should be compiled with the program.

When tables or arrays are being compiled with the program, table records must always follow the RPG II source program. A record with **b (b = blank) in positions 1-3 is needed to separate the RPG II source program from the table or array records. Tables or arrays must be separated from each other by records with **b in positions 1-3 (Figure 2). Because **b in positions 1-3 indicates the start of a table or array, you must not specify **b in positions 1-3 of your table input records.

Short tables (tables that are not full) can be compiled with the program, but a warning is issued. See *Columns 36-39* for a discussion of short tables.

COLUMNS 19-26 (TO FILENAME)

Entry	Explanation
Name of an input or update file	The file processed via the record address file named in columns 11-18.
Name of an output file	The output file to which a table or array is to be written.

Columns 19-26 define the relationship between a file named in these columns and a file named in columns 11-18. Filenames must begin in column 19.

If a record address file is named in columns 11-18, the name of the input or update file that contains the data records to be processed must be entered in columns 19-26. Do not enter the record address filename in these columns.

If you want a table or array to be written, use columns 19-26 to enter the filename of the output file you use to do this. This output file must be named previously in the file description specifications.

A table or array can be written on only one output device, leave columns 19-26 blank if you do not want the table or array written.

If a table or array is assigned to an output file, it is automatically written at the end of the execution after all other records are written. The table or array is written in same format in which it was entered.

Since there is no program control over the output format when an entry is made in columns 19-26, those cases where formatting is required should be provided for in the program through the output specifications or by using exception lines to write one item at a time (see Part 2, *RPG II Programmer's Guide, Operation Codes, Exception*). Tables or arrays should be written only after all records are processed (last record indicator is on).

COLUMNS 27-32 (TABLE OR ARRAY NAME)

Entry	Explanation
Table or Array Name	Name of each table or array used in the program

Use columns 27-32 to name your table or array. No two tables or arrays may have the same name. The rules for forming table and array names are discussed in the following text.

Table Name

Every table used in your program must have a name. The entire table name can be from 3-6 characters long, and must begin with the letters TAB. After the letters TAB, 1-3 alphabetic or numeric characters may be used (no special characters allowed). Blanks cannot appear between characters in the table name. Any name in columns 27-32 which does not begin with TAB is considered an array name. The table name is used throughout the program. However, different results can be obtained depending upon how the table name is used. When the table name is used in factor 2 or the result field (on the calculation specifications sheet) with a LOKUP operation, the name refers to the entire table. When the table name is used with any other operation code, the name refers to the table item last selected from the table by a LOKUP operation (see Part 2, *RPG II Programmer's Guide, Operation Codes, Look Up and Tables and Arrays*).

Table files are processed in the same order as specified by the extension specifications. Therefore, if you have more than one table file, the files are to be loaded in the same order as they appear on the sheet.

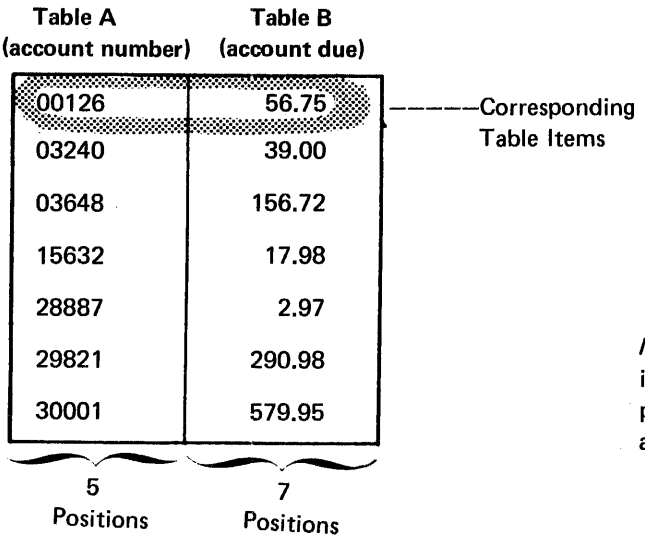
If two tables are in alternating format in one table file, the table whose item appears first must be named in columns 27-32. The second table is named in columns 46-51 (Figure 38).

Array Name

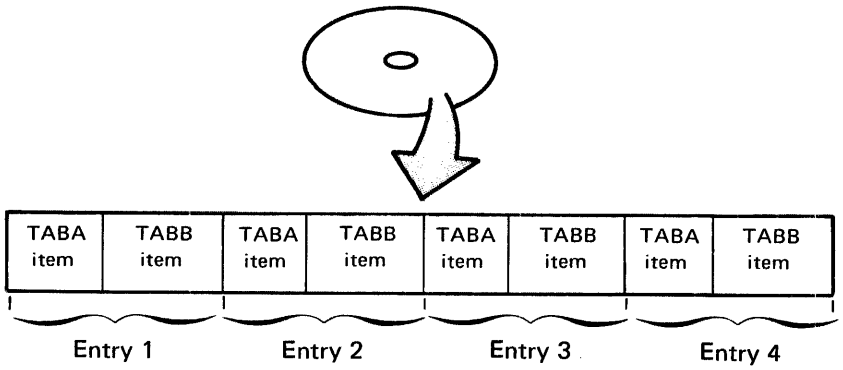
Every array used in your program must be given a name. An array name cannot begin with the letters TAB. This array name is used throughout the program. See Part 2, *RPG II Programmer's Guide, Tables and Arrays* for more information on forming array names.

Example

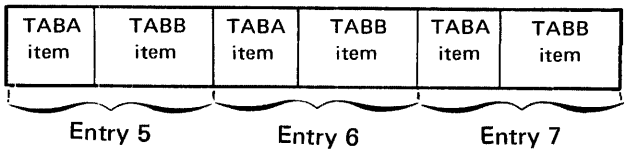
Figure 38, insert A, shows two tables (TABA and TABB) described in alternating format. An item for TABA appears first. Thus, in insert B, TABA is named in columns 27-32 of the extension specifications sheet; TABB is named in columns 46-51.



Note: The decimal points shown in Table B are only for illustration purposes. Decimal points are not a part of table or array input data.



Ⓐ



The corresponding items from the tables are entered in the machine in alternating format. Corresponding items from the two tables are considered as one entry.

Figure 38 (Part 1 of 2). Related Tables

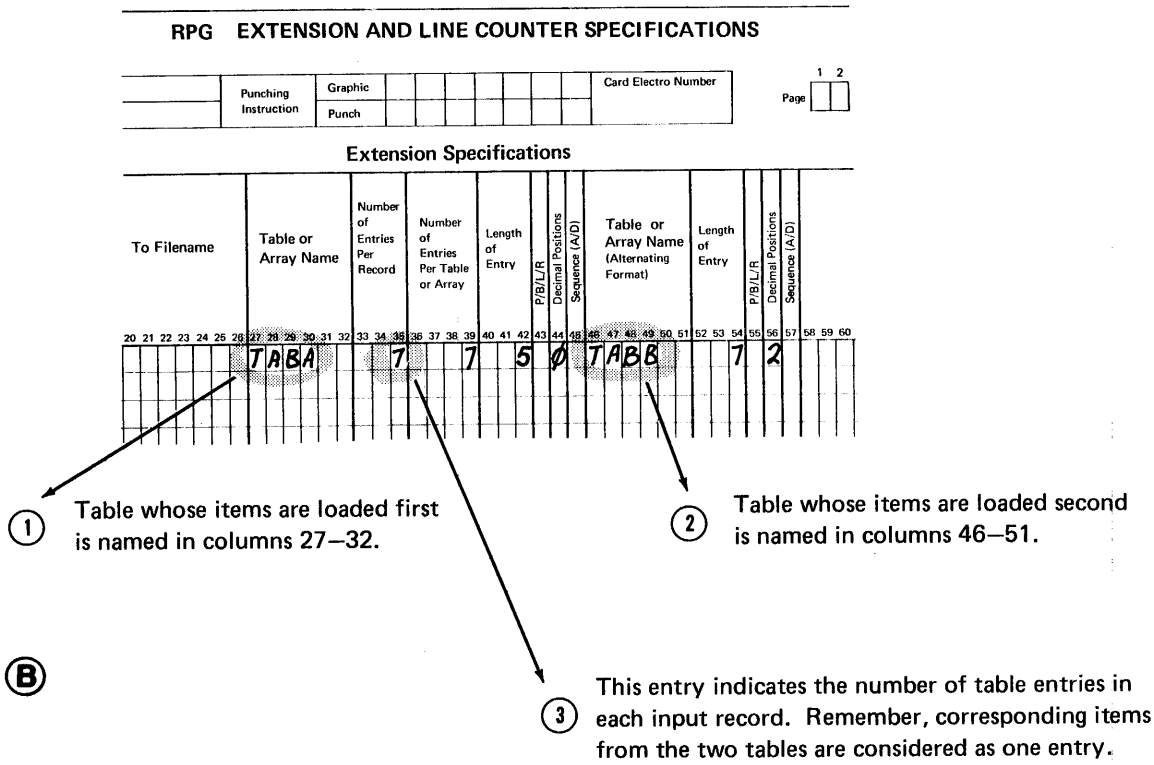


Figure 38 (Part 2 of 2). Related Tables

COLUMNS 33-35 (NUMBER OF ENTRIES PER RECORD)

Entry	Explanation
1-999	Number of table or array entries found in each table or array input record.

Indicate in columns 33-35 the exact number of table or array entries in each table or array input record. Every table or array input record except the last must contain the same number of entries as indicated in columns 33-35. The last record may contain fewer entries than indicated, but never more.

When two tables or arrays are described in one file, each table or array input record must contain the corresponding items from each table or array written in alternating format. These table items are considered one entry. The number entered must end in column 35. Corresponding items from tables or arrays in alternating format must be on the same record. Comments may be entered on table input records in positions following table entries.

When loading an array, the following must be considered:

- To load a preexecution-time array, a filename must be entered in columns 11-18, and an entry must be made in columns 33-35.
- To load an array at compile time, columns 11-18 must be blank, but an entry must be made in columns 33-35.
- To load an execution time array (via the input and/or calculations specifications), columns 11-18 and columns 33-35 must be blank.

Example

Figure 38, insert B, shows table entries for the two tables, A and B, entered in alternating format. A1 and B1, the corresponding items in tables A and B, are considered one entry. Even though there are 14 table items listed, there are only seven table entries.

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

Entry	Explanation
1-4096	Maximum number of table or array entries

Use columns 36-39 to indicate the maximum number of table items which can be contained in the table named in columns 27-32, or the maximum number of array items which can be contained in the array named in columns 27-32. This number may apply to one table or to two tables in alternating format. Any number entered in these columns must end in column 39.

Since the number of items for two tables or arrays entered in alternating format must be the same, the entry in these columns also gives the number of items in a second table or array (columns 46-51).

If your table or array is full, this entry gives the exact number of items in it. However, if the table or array is not full, the entry gives the number of items that can be put into it (Figure 39). A table or array that is not full is known as a short table or array.

If a table or array is to be compiled, it should be full. However, if it is not full (a short table or array), the table or array is compiled with the program and a warning is issued. The short table or array is completed in storage by filling it with blanks or zeros (for alphameric or numeric tables or arrays, respectively). A preexecution-time table or array need not be full.

Note: The decimal points shown in these tables are only for illustration purposes. Decimal points are not part of table input data.

TABPRT (Part Number)	TABAMT (Price)
001	127.62
002	198.32
003	.27
004	.01
005	1.98
009	3.79
010	5.67
014	2.33
026	14.67
045	29.33
096	29.34
097	.05
098	.09
099	1.19
100	2.22
101	126.73
110	596.74
115	393.75
126	697.75
137	1.92

If this data is entered into the machine, TABPRT and TABAMT will be full (20 entries fill the table).

TABPRT (Part Number)	TABAMT (Price)
001	127.62
002	198.32
003	.27
004	.01
005	1.98

If this data is entered into the machine, TABPRT and TABAMT will not be full.

EXTENSION AND LINE COUNTER SPECIFICATIONS

Punching Instruction: Graphic / Punch — Card Electro Number — Page 1 2

Extension Specifications

Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)
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
TABPRT	12	20	3		Ø		TABAMT	5		2	

This entry indicates that TABPRT and TABAMT may both have a maximum of 20 entries.

Figure 39. Table Entries (Number Per Table)

COLUMNS 40-42 (LENGTH OF ENTRY)

Entry	Explanation
1-15	Length of a numeric entry
1-256	Length of an alphabetic entry

Use columns 40-42 to give the length of each entry in the table or array named in columns 27-32. The number entered must end in column 42. For numeric tables or arrays in packed decimal format, enter the unpacked decimal length in columns 40-42. For numeric tables or arrays in binary format, enter the number of digits required in storage for the binary field. For a 2-position binary field, the entry in columns 40-42 is 4; for a 4-position binary field, the entry is 9.

All table or array items must have the same number of characters. It is almost impossible, however, for every item to be the same length. Therefore, add zeros or blanks to make them the same length. Add leading zeros for numeric items. Add blanks after alphameric items (see *Examples, Example 1*).

If two tables or arrays are entered in alternating format, the specification in columns 40-42 applies to the table or array whose item appears first in the record (see *Examples, Example 2*).

The maximum length of a numeric table or array item is 15 characters. The maximum length of an alphameric table or array item is 256 characters for disk records and up to 158 characters for console records.

See *Arrays or Tables* under Part 2, *RPG II Programmer's Guide* for more information.

Examples

Example 1

Figure 40 shows a table, called TABMO, which lists months of the year. The name SEPTEMBER, having nine characters, is the longest entry. Because the lengths of the entries must be the same, blanks are added to the remaining names to make each of them nine characters long.

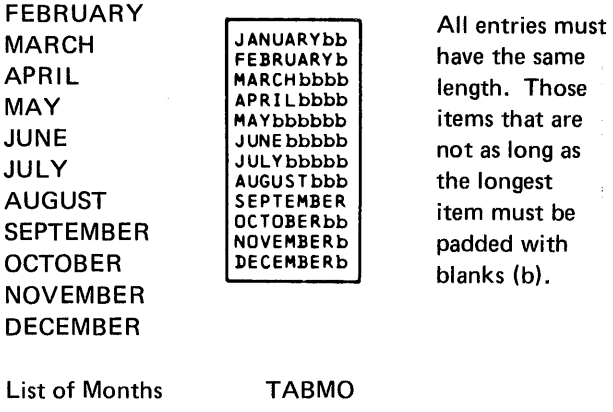


Figure 40. Length of Table Entries

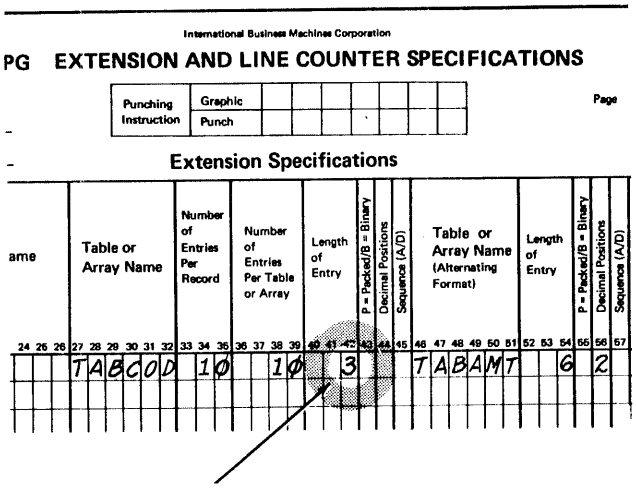
Example 2

Figure 41 shows entries for two tables entered in alternating format, TABCOD and TABAMT. Each item in TABCOD is three characters long; each item in TABAMT is six characters long. Since TABCOD is entered in the machine first, its length, 3 is specified in columns 40-42. The length of items in TABAMT is in columns 52-54.

TABCOD (Code)	TABAMT (Amount)
021	217.43
022	93.06
023	8.14
040	2166.58
041	39.23
060	1741.78
117	83.33
118	5.12
143	72.03
352	253.96
3 Positions	6 Positions

Note: The decimal points shown in these tables are only for illustration purposes. Decimal points are not a part of table input data.

Ⓐ



Ⓑ The length of the table item which is entered in the machine first must appear in columns 40-42.

Figure 41. Length of Corresponding Table Items

COLUMN 43 (PACKED OR BINARY FIELD)

Entry	Explanation
Blank	Data for table or array is in unpacked decimal format or is alphameric.
P	Data for table or array is in packed decimal format on disk.
B	Data for table or array is in binary format on disk.

Use column 43 to indicate that a numeric field in a preexecution-time table or array file is in packed or binary format. Leave column 43 blank if the field is unpacked. See *Column 43* under *Input Specifications* for more information on packed or binary formats.

COLUMN 44 (DECIMAL POSITIONS)

Entry	Explanation
Blank	Alphameric table or array.
0-9	Number of positions to the right of the decimal in numeric table or array items.

Column 44 must always have an entry for a numeric table or array. If the items in a table or array have no decimal positions, enter a 0.

If two tables or arrays are entered in alternating format, the specification in this column applies to the table or array containing the item which appears first on the record.

COLUMN 45 (SEQUENCE)

Entry	Explanation
Blank	No particular order
A	Ascending order
D	Descending order

Use column 45 to describe the sequence (either ascending or descending) of the data in a table or array file.

When an entry is made in column 45, the table or array is checked for the specified sequence. If a compile-time table or array is out of sequence, a severe error occurs. The program halts after compilation. If a preexecution-time table or array is out of sequence, an error occurs and the program halts immediately. The program can be restarted from the point where it halted if you do not want to correct the out-of-sequence condition; however, if you do correct the out-of-sequence condition, program execution must be restarted from the beginning.

Ascending order means that the table or array items are entered starting with the lowest data item (according to the collating sequence) and proceeding to the highest. Descending order means that the table or array items are entered starting with the highest data item and proceeding to the lowest.

If two tables or arrays are entered in alternating format, the entry in column 45 applies to the table or array containing the item which appears first on the record.

When you are searching a table or array for an item (LOKUP) and want to know if the item is high or low compared with the search word, your table or array must be in either ascending or descending order. See Part 2, *RPG II Programmer's Guide, Operation Codes, LOKUP* for more information. When a specific sequence has been specified, RPG II checks the data in the table or array to see if it really is in that sequence.

An execution-time array (built-in input and/or calculation specifications) is not sequence checked. However, an A or D entry must be specified if a high or low look up operation is performed.

COLUMNS 46-57

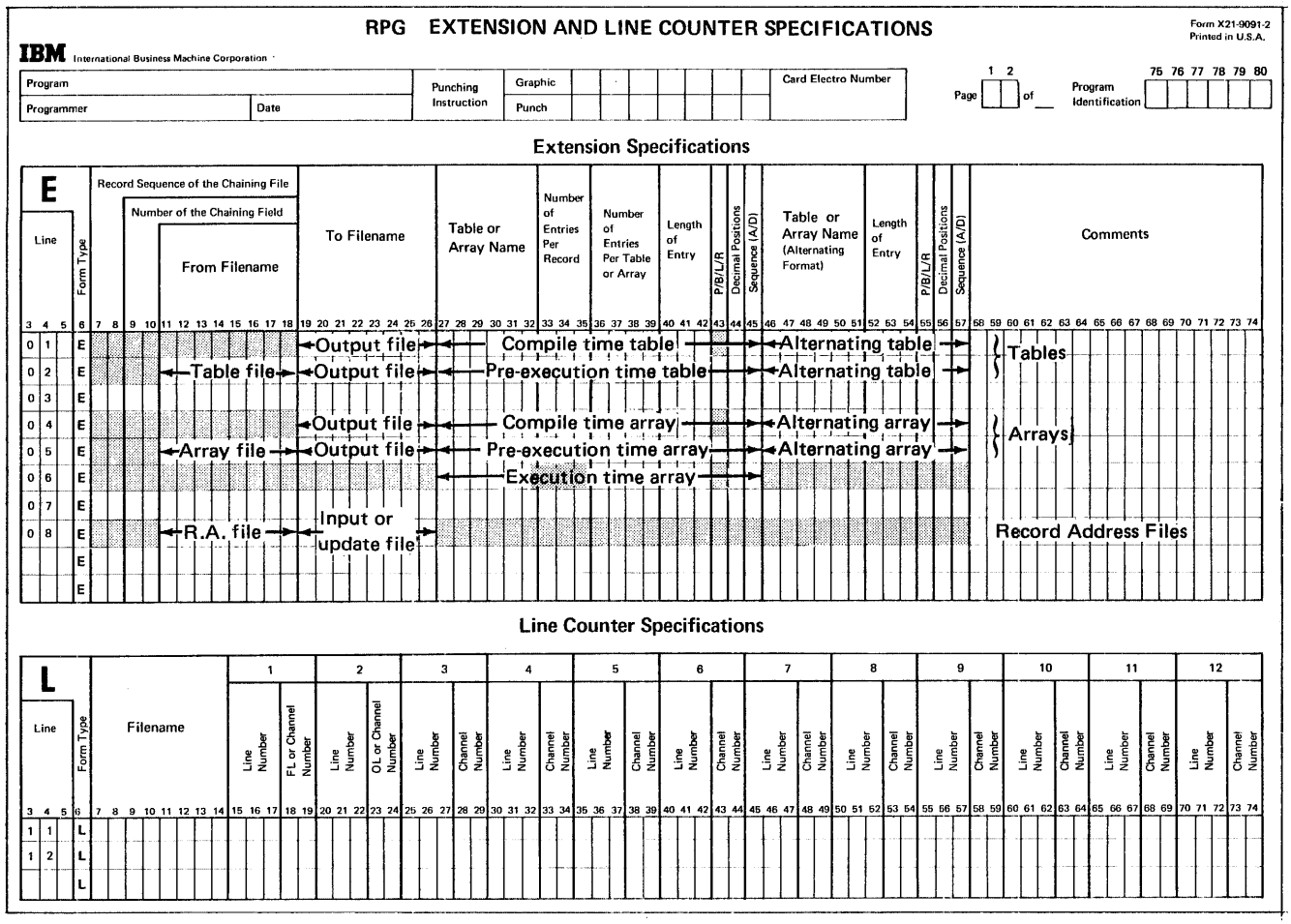
Use columns 46-57 only when describing a second table or array which is entered in an alternating format with another table or array. Usually, the second table or array corresponds with the table or array named in columns 27-32. All fields in this section have the same significance and require the same entries as the fields with corresponding titles in columns 27-45. See the previous discussion on those columns for information about correct specifications. Leave these columns blank for a single table or array.

COLUMNS 58-74 (COMMENTS)

Enter any information you wish in columns 58-74. The comments you use should help you understand what you are doing in each specification line. Comments are not instructions to the RPG II program; they serve only as a means of documenting your program.

COLUMNS 75-80 (PROGRAM IDENTIFICATION)

See *Common Entries*.



- The shaded columns must be blank for the file named.
- For tables and all arrays except execution time arrays, columns 19-26 and columns 46-57 are optional.
- Execution arrays are loaded via input and/or calculation specifications.
- For record address files, columns 11-26 must have entries.

Figure 42. Possible File Entries for Extension Specifications

Line counter specifications should be used for each printer file in your program. Line counter specifications indicate at what line overflow occurs and the length of the form used in the printer. Both of these entries must be specified on the line counter specifications sheet (Figure 43). If no line counter specifications exist, the forms length used is either:

- The forms length specified on the // FORMS entry, or
- The forms length specified by the SET command statement (see *IBM System/32 System Control Programming Reference Manual*, GC21-7593), or
- The forms length specified at system generation time.

In either case, the overflow line is assumed to be six lines less than the specified forms length.

COLUMNS 1-2 (PAGE)

See *Common Entries*.

COLUMNS 3-5 (LINE)

See *Common Entries*.

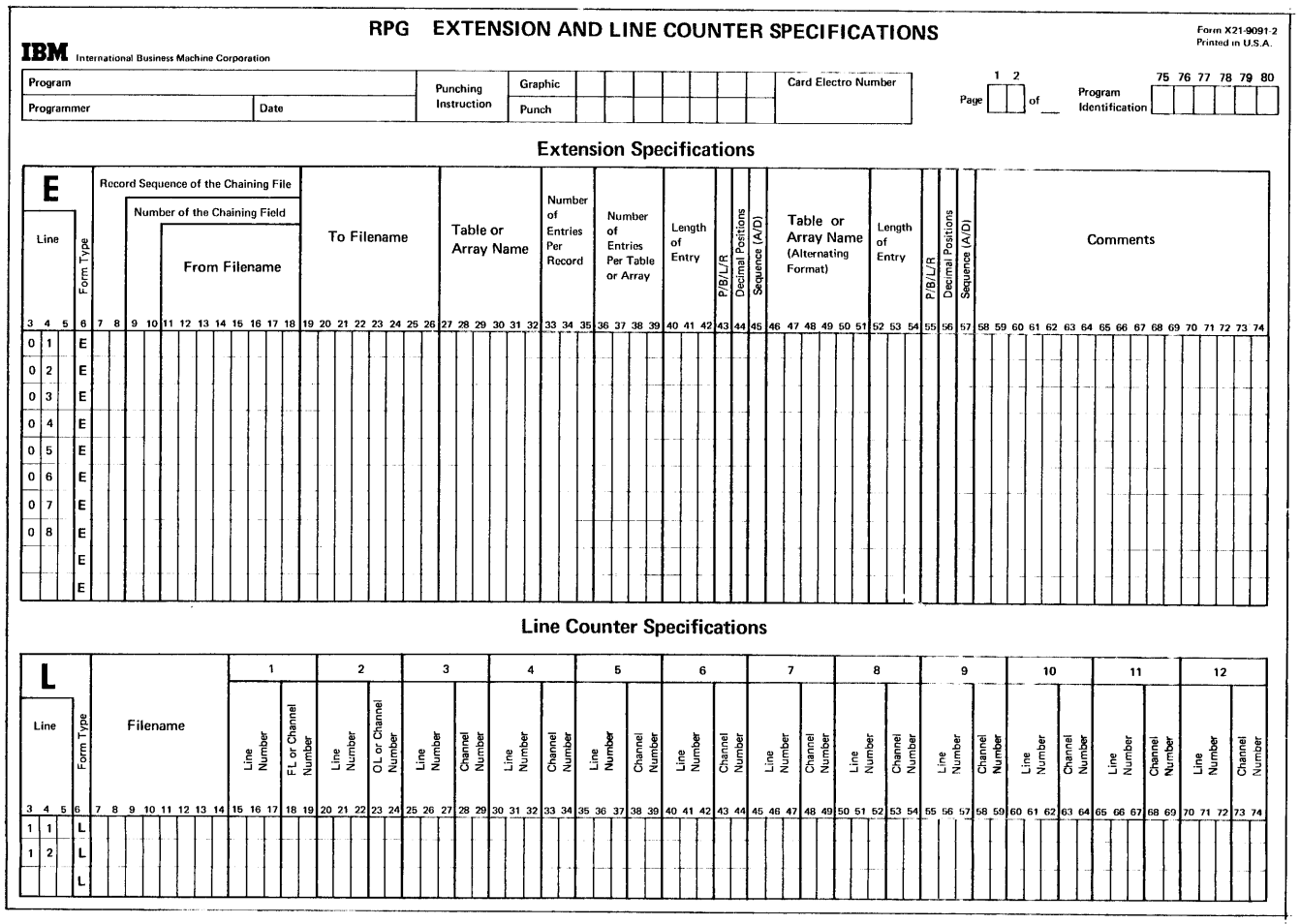


Figure 43. Extension and Line Counter Specifications Sheet

COLUMN 6 (FORM TYPE)

An L must appear in column 6.

COLUMNS 7-14 (FILENAME)

Use columns 7-14 to identify the output file to be printed on the printer. The filename must begin in column 7. Any filename entered in these columns must be a filename previously defined by file description specifications. The output device assigned to the file by file description specifications must be the printer.

COLUMNS 15-17 (LINE NUMBER–NUMBER OF LINES PER PAGE)

Entry	Explanation
1-84	Number of printing lines available is from 1-84.

Columns 15-17 specify the exact number of lines available on the form or page to be used. The entry must end in column 17. Leading zeros are not necessary.

COLUMNS 18-19 (FORM LENGTH)

Entry	Explanation
FL	Form length

Columns 18-19 must contain the entry FL. This entry indicates that the preceding entry (columns 15-17) is the form length.

COLUMNS 20-22 (LINE NUMBER–OVERFLOW LINE)

Entry	Explanation
1-84	A line number from 1-84 is the overflow line.

Columns 20-22 specify the line number that is the overflow line. The entry must end in column 22. Leading zeros may be omitted. When the line which you have specified as the overflow line is printed, the overflow indicator turns on. When the overflow indicator is on and fetch overflow is not specified, the following occurs before forms advance to the next page:

1. Detail lines are printed (if this part of the program cycle has not already been completed).
2. Total lines are printed (if conditions are met).
3. Total lines conditioned by the overflow indicator are printed.

Because all these lines are printed on the page after the overflow line, specify the overflow line high enough on the page to allow all these lines to print. You know the data to be printed after the overflow line is reached. Thus, you can judge what line should be the overflow line on this basis. See *Overflow Indicators* under Part 2, *RPG II Programmer's Guide* for more information.

COLUMNS 23-24 (OVERFLOW LINE)

Entry	Explanation
OL	Overflow line

Columns 23-24 must contain the entry OL. This indicates that the preceding entry (columns 20-22) is the overflow line.

COLUMNS 25-74

Columns 25-74 are not used. Leave them blank.

COLUMNS 75-80 (PROGRAM IDENTIFICATION)

See *Common Entries.*

Input specifications describe the data files, records, and fields of the records to be used by your program. All input files apply to the input specifications except files with the device name KEYBORD. KEYBORD files are described by calculation specifications when the KEY operation is used.

The input specifications are divided into two categories:

- File and record type identification specifications (columns 7-42) describe the input record and its relationship to other records in the file.
- Field description specifications (columns 43-74) describe the fields in the records.

The specifications are written on the input specifications sheet (Figure 44). The field description entries must start one line lower than file and record type identification entries.

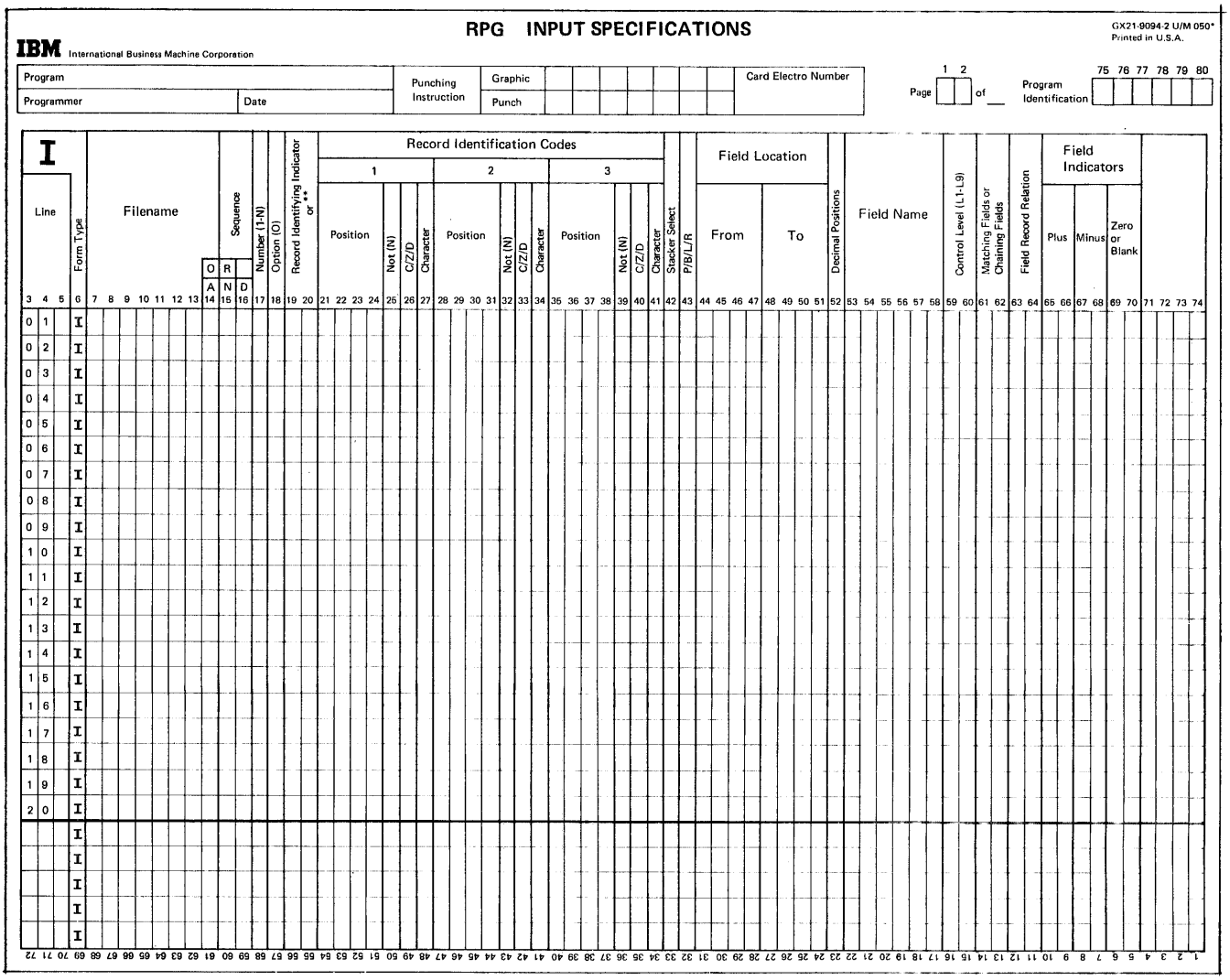


Figure 44. Input Specifications Sheet

COLUMNS 1-2 (PAGE)

See *Common Entries*.

COLUMNS 3-5 (LINE)

See *Common Entries.*

COLUMN 6 (FORM TYPE)

An I must appear in column 6.

COLUMNS 7-14 (FILENAME)

Columns 7-14 identify the input, combined, or update file you are describing. The filename must begin in column 7. Use the same filename given by the file description specifications. The filename must appear on the first line that contains information concerning the records in that file.

COLUMNS 14-16

Columns 14-16 may contain the characters AND, and columns 14-15 may contain the characters OR. These AND/OR lines may be used to indicate a relationship between record identifying indicators or record types. For further information, see *AND Relationship* and *OR Relationship* for columns 21-41 and columns 53-58 in this chapter.

COLUMNS 15-16 (SEQUENCE)

Entry	Explanation
Any two alphabetic characters	Do not check for special sequence.
01-99	Check for special sequence.

Enter a numeric entry in columns 15-16 to assign a special sequence to different record types in a file. If different types of records do not need to be in any special order, use two alphabetic characters. Within one file, all record types having alphabetic entries in columns 15-16 must be specified before those types with numeric entries. Columns 15-16 must contain an alphabetic entry for chained files.

Alphabetic Characters

If you do not want to check for a special sequence of record types, enter any two alphabetic characters in columns 15-16 (see *Examples, Example 1).* Alphabetic characters must be used for chained files and look ahead records.

Numeric Characters (01-99)

Use columns 15-16 to assign sequence numbers to different types of records within a file. Your job may require that one record type (identified by a record identification code) must appear before another record type within a sequenced group. For instance, you may want a name record before an address record. You must provide a record identification code for each type of record and then number the record types in the order that they should appear. The program checks this order as the records are read. The first record type must have the lowest sequence number (01), the next record type should be given a higher number, etc (see *Examples, Example 2*).

Numeric sequence numbers only ensure that all records of record type 01 precede all records of record type 02, etc, in any sequenced group. The sequence numbers do not ensure that records within a record type are in any certain order. Numeric sequence numbers have no relationship with control levels, nor do they provide for sequence checking of data in fields of a record (see *Examples, Example 3*).

Note: Numeric sequence is not allowed on demand files.

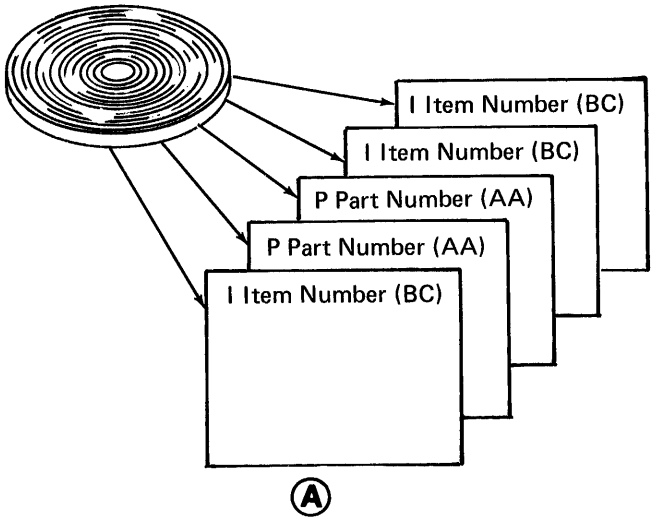
Gaps in sequence numbers are allowed, but the numbers used must be kept in ascending order. The first sequence number must be 01.

A record type out of sequence causes the program to stop. Restart the program by selecting the appropriate option and pressing ENTER. The record that causes the halt is bypassed and the next record is read from the same file.

Records in an OR or AND line cannot have a sequence entry in these columns. The entry in these columns from the previous line also applies to the entry in the OR or AND line. See *Columns 53-58 (Field Name)* for information on OR relationships.

Examples

Example 1: Figure 45, insert A, shows a file having two types of records (part number and item number) which may appear in any order. Since they are not to be checked



RPG INPUT SPI

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic
Programmer	Date		Punch

I

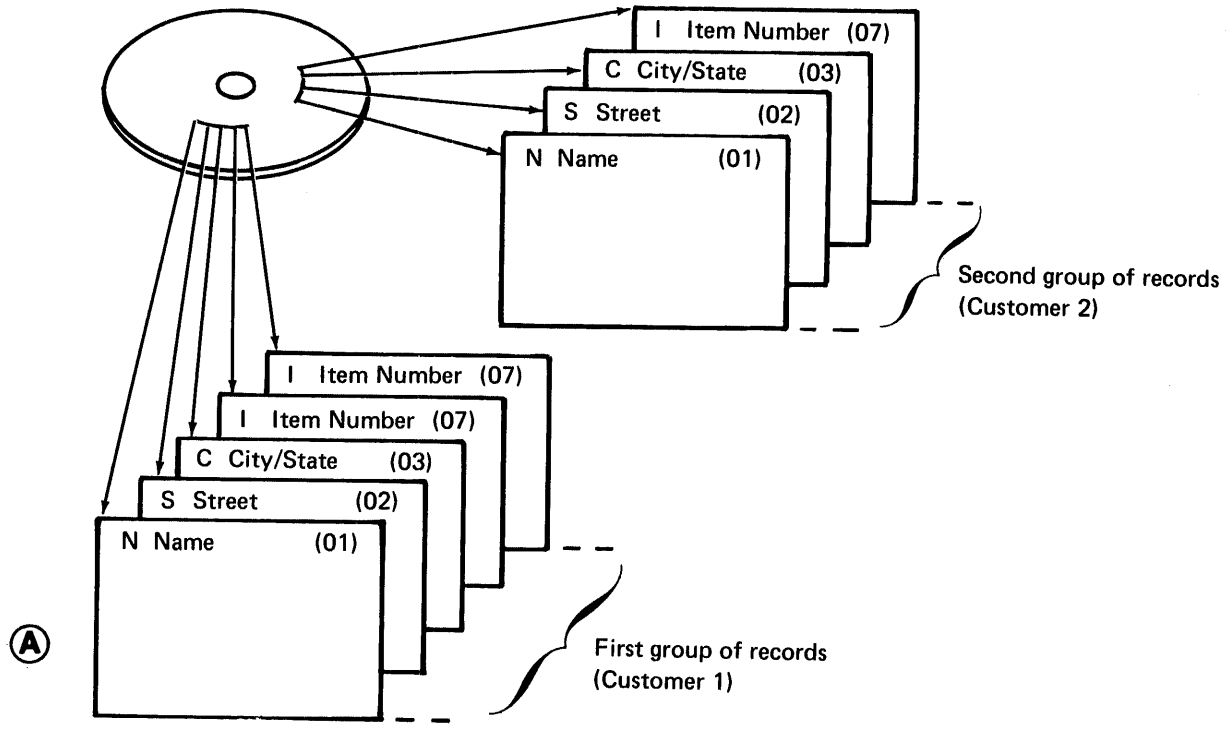
Record Identification Codes

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	1 Position	1 Not (N)	1 C/Z/D	1 Character	2 Position	2 Not (N)	2 C/Z/D	2 Character	3 Position
3 4 5	6	7 8 9 10 11 12 13	O A 14 / R N 15	D 16	17	18	19 20	21 22 23 24	25	26	27	28 29 30 31	32	33	34	35 36 37 38
0 1	I	RECORDA	AA				Ø1	1	C	P						
0 2	I															
0 3	I															
0 4	I															
0 5	I															
0 6	I															
0 7	I		BC				Ø2	1	C	1						
0 8	I															
0 9	I															

B

Figure 45. Unsequenced Record Type in a File

for sequencing, they are assigned two alphabetic characters (AA and BC, respectively) instead of numbers. See Figure 45, insert B, for the coding of this example.



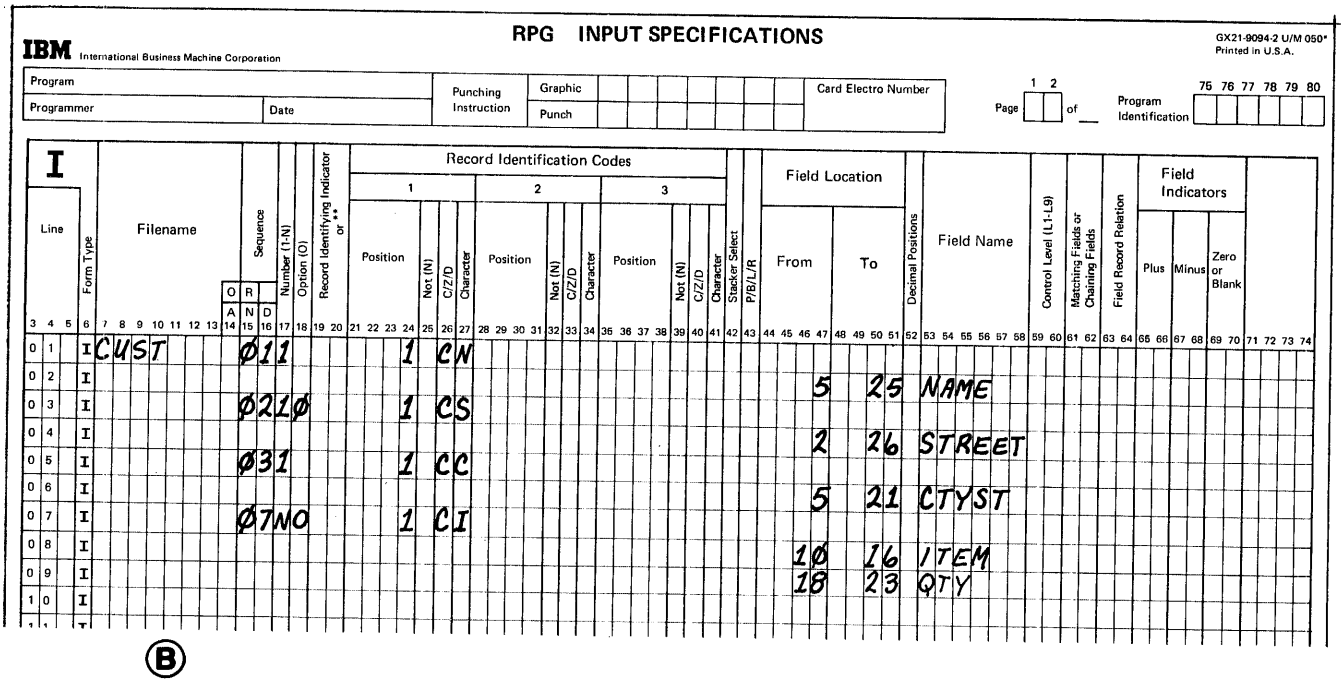


Figure 46. Sequence Checking of Record Types

Example 2: Figure 46, insert A, shows the order of four different types of records within a file. The records are arranged in groups according to a customer name control field. The name record is first in each group and is assigned sequence number 01. Street record is next and is assigned 02. City/state record is 03. (Remember gaps are allowed.) Item number record is 07. More than one item number record may be present. See Figure 46, insert B, for the coding of this example.

Example 3: Figure 47 shows three groups of four different record types. Each group is in proper sequence according to the assigned sequence numbers (01, 02, 03, and 07). Notice, however, that the city/state record for customer #3 is in the group for customer #2 and vice versa. The sequence entry which you specify in columns 15-16 does not catch this mistake since the sequence entry does not cause the data on the record to be checked. See Figure 46, insert B, for the coding of this example.

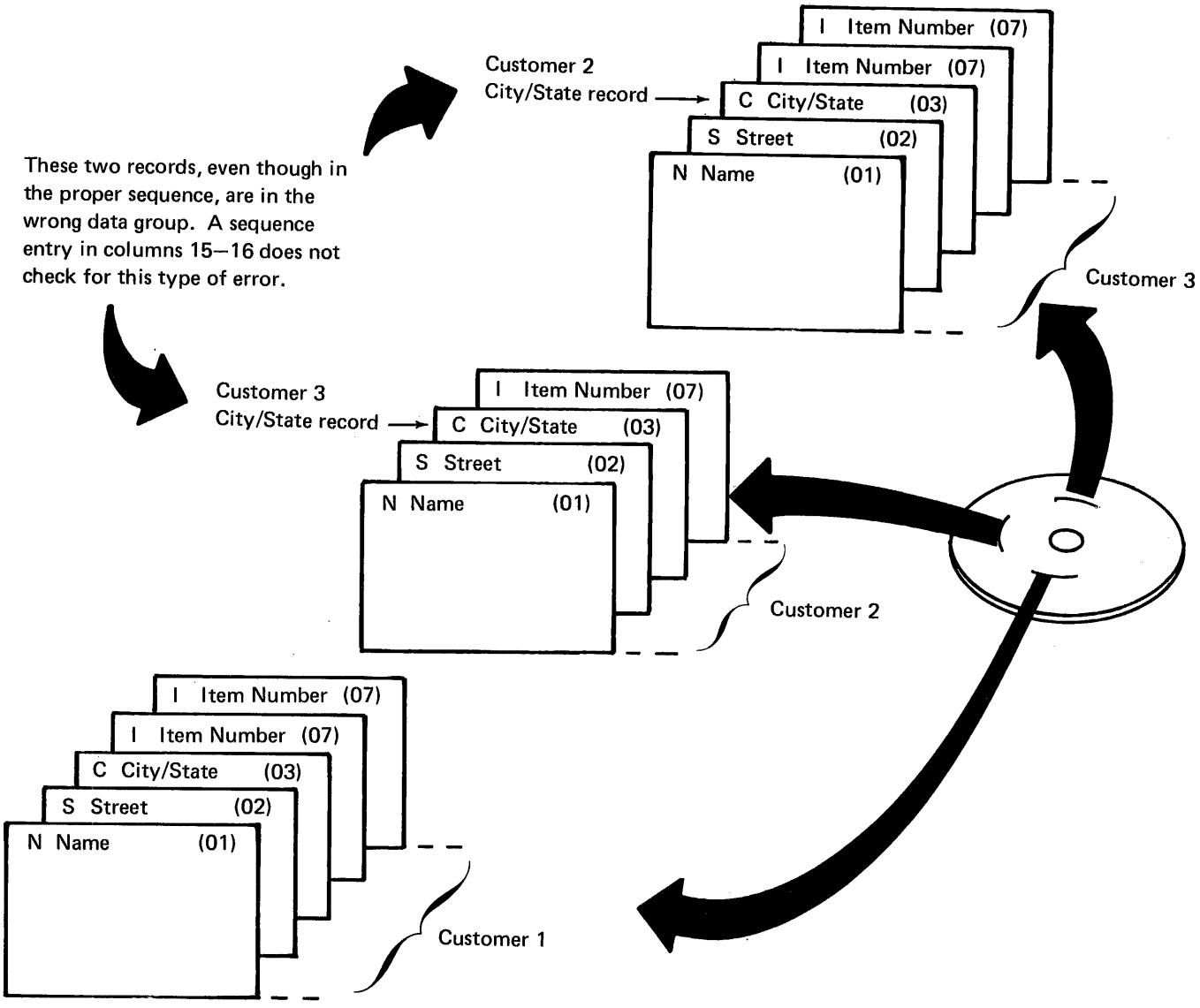


Figure 47. Correct Record Sequence (Incorrect Data Within Groups)

COLUMN 17 (NUMBER)

Entry	Explanation
Blank	Record types are not being sequence checked (columns 15-16 have alphabetic entries).
1	Only one record of this type can be present in the sequenced group.
N	One or more records of this type may be present in the sequenced group.

Use column 17 only if sequence checking is to be done (columns 15-16 contain a numeric entry). Often, when sequence checking, you have more than one record of a particular type within the sequenced group (Figure 48). You must indicate by an N in column 17 that more than one record of one type may be found in the sequence group.

OR lines (columns 14-15 have the letters OR) and AND lines (columns 14-16 have the letters AND) should not have an entry in this column. It is assumed that the number of records of this type to be found in the sequenced group is the same as the number entered in column 17 of the previous line. See *Columns 53-58 (Field Name)* for more information on OR lines.

Example

Figure 48 shows a sequence record file in which there is more than one record per type in a group. The record type called item number appears three times.

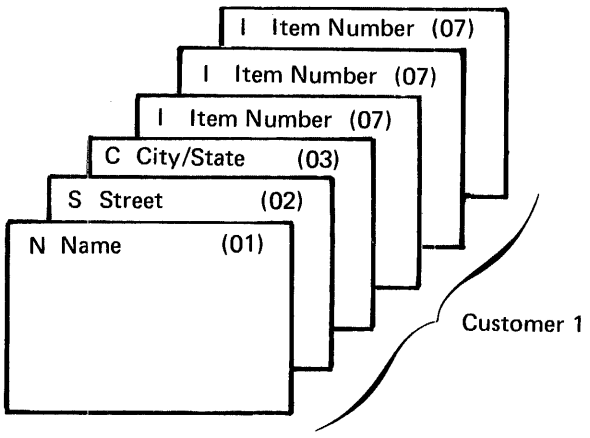


Figure 48. Sequenced Record File (More Than One Record Per Type in a Group)

Refer to Figure 46, insert B, for the coding of this example. There is no reason for a name, street, or city/state record to appear more than once in a group. A one (1) is entered in column 17 to indicate that these record types appear only once in each group. However, since one person may have purchased more than one item, there may be two or more item number records per group; an N is entered in column 17 for this field.

COLUMN 18 (OPTION)

Entry	Explanation
Blank	Record type must be present if sequence checking is specified.
O	Option. Record type may or may not be present if sequence checking is specified.

Column 18 is used only when record types are being sequence checked (columns 15-16 contain a numeric entry). A blank entry specifies that a record of this record type must be present in each sequenced group.

The O entry specifies that a record of this record type may or may not be present in each sequenced group (Figure 49). If all record types are optional, no sequence error is found.

OR and AND lines should not have an entry in this column. The entry in this column on the previous line also applies to this record in the OR or AND relationship. See *Columns 53-58 (Field Name)* for more information on OR lines.

Example

Figure 49 shows a sequenced file in which record type is optional. For instance, the street or item number records may not be included. Since it is not always necessary to have a street address, this record is optional. Suppose this job required a list of all items purchased during one month by the individual named in the name record. It is possible that a person might not buy anything during the month. In this case, there would be no item record; therefore, the item record is also optional. See Figure 46, insert B, for a coding example.

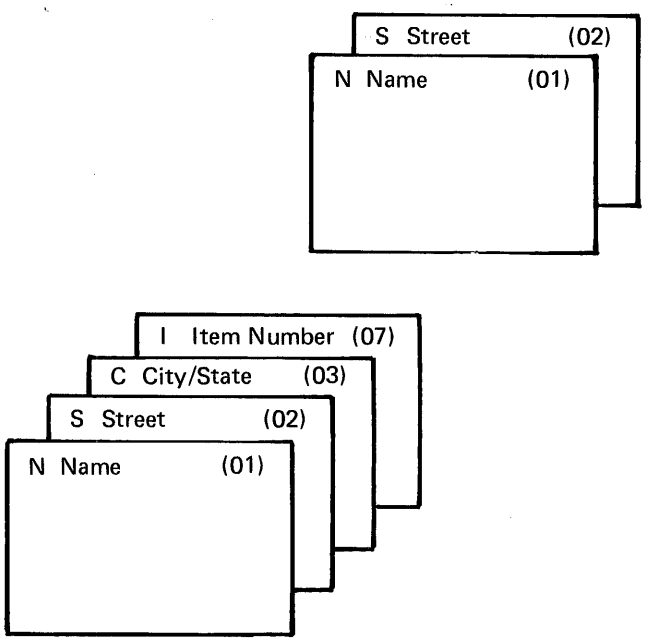


Figure 49. Sequenced Record File (Optional Record Types)

COLUMNS 19-20 (RECORD IDENTIFYING INDICATOR,**)

Columns 19-20 have two purposes:

- To specify record identifying indicators
- To indicate a look ahead field

Entry	Explanation
01-10	Record identifying indicator for console files.
01-99	Record identifying indicator.
L1-L9	Control level indicator used for a record identifying indicator when a record type rather than a control field signals the start of a new control group.
LR	Last record indicator.
H1-H9	Halt indicator used for a record identifying indicator when checking for a record type that causes an error condition.
**	Look ahead field.

Record Identifying Indicators

Use columns 19-20 to assign an indicator to each record type. When you have different types of records within a file, you often want to do different operations for each record type. Therefore, you must have some way of knowing which type of record was just read. To do this, you assign different record identifying indicators to each record type. Whenever a record type is selected to be processed next, its corresponding identifying indicator is turned on. (All other record identifying indicators are off at this time unless chained files or demand files are being processed; in this case, several indicators may be on at the same time.) This indicator signals throughout the rest of the program cycle which record type was just selected.

When a control level indicator used as a record identifying indicator is turned on to indicate the type of record read, only that one control level indicator is turned on. All lower control level indicators remain unchanged.

Because the record identifying indicator is on for the rest of the program cycle, you may use it to condition calculation operations (see *Columns 9-17* under *Calculation Specifications*) and output operations (see *Columns 23-31* under *Output Specifications*).

Record identifying indicators do not have to be assigned in any order.

You may assign the same indicator to two or more different record types provided you want the same operation performed on these types. Do this by using the OR relationship. See *Columns 21-41 (Record Identification Codes)*.

No record identifying indicator can be specified in the AND line of an AND relationship. Record identifying indicators for OR lines can be specified for every record type in the OR relationship that requires special processing. See *Columns 21-41 (Record Identification Codes)* for information on AND lines. See *Columns 53-58 (Field Name)* for information on OR lines.

If RPG II telecommunications specifications are used, a description of the permanent error indicator can be found in *IBM System/32 RPG II Telecommunications Programming Reference Manual,* SC21-7597.

Look Ahead Fields

Use asterisks in columns 19-20 to indicate that fields named in columns 53-58 in the following specifications are look ahead fields. A look ahead field allows you to look at information in a field on the next record that is available for processing in any input or update file. Because of this capability you are able to use the information from the

look ahead field to determine what operation should be done next.

A look ahead field allows you to:

- Determine when you are processing the last record of a control group.
- Extend the RPG II matching record capability.

See *Look Ahead* under Part 2, *RPG II Programmer's Guide* for information on when and how to use look ahead fields.

Note: Look ahead fields are not valid with console files.

COLUMNS 21-41 (RECORD IDENTIFICATION CODES)

Use columns 21-41 to describe the information that identifies a record type.

Note: Only columns 21-34 are valid for console files. See Part 2, Chapter 9 for console coding information.

When you have many record types in one file, you often want to perform different operations for each type. Therefore, you must identify each type by giving each a special code consisting of a combination of characters in certain positions in the record. This code must be described in columns 21-41 so that when a record is read the record type can be determined by these specifications.

When more than one record type is used in a file, only one record type is selected for processing in each cycle. The record identifying indicator for that record type is turned on at the time of selection. When all records are to be processed alike regardless of their type, or if there is only one type, leave columns 21-41 blank.

Seven columns are set aside for the description of one character in the record identification code. Each specification line contains three sets of seven columns: columns 21-27, 28-34, and 35-41. Each set consists of four fields: Position, Not, C/Z/D, and Character. Coding is the same for all three sets.

Note: Any record that is read by the system and is not described by a record identification code in columns 21-41 causes the program to halt. You may continue, however, by selecting the appropriate option and pressing ENTER. The record that causes the halt is not processed and the next record in that file is read.

Position

Entry	Explanation
Blank	No record identification code is needed.
1-4096	Record position of one character in the record identification code.

Use columns 21-24, 28-31, and 35-38 to give the location in the record of every character in the identification code. Entries in these columns must end in columns 24, 31, and 38 respectively.

Not (N)

Entry	Explanation
Blank	Character is present in the specified record position.
N	Character is not present in the specified record position (not valid for console IDE files, see Part 2, Chapter 9).

Use columns 25, 32, and 39 to indicate that a certain character should not be present in the specified position.

C/Z/D

Entry	Explanation
C	Entire character
Z	Zone portion of character
D	Digit portion of character

Use columns 26, 33, and 40 to indicate what portion of a character is used as part of the record identifying code. Only the zone portion, only the digit portion, or both portions (the whole character) may be used (see *Examples, Example 3 and Example 4*). When establishing record identifying codes, remember that many characters have either the same zone or the same digit portion. For a list of characters that have identical zone or digit portions, see *Character, Character Grouping by Zone or Digit* in the following text.

Note: Must be C for console files, see Part 2, Chapter 9.

Character

Use any alphabetic character, special character, or digit in columns 27, 34, and 41 to identify the character that was used in the record to serve as the code or part of the code.

Character Grouping by Zone or Digit: When selecting characters for record identification purposes on a digit or zone only basis, you must understand that all characters having the same zone or digit are selected by the system as meeting record identification requirements. When a character is read into the system, it is converted into an 8-bit code. It is the 8-bit code that is tested to see if the character meets the requirements of the record identifying character in the input specifications. Figure 50 lists the character grouping for zone or digit only in the character/zone/digit columns (26, 33, or 40) and character columns (27, 34, or 41) of the input specifications.

As an example, a digit only entry in C/Z/D and an A in character cause all records having a / (slash), A, J, or 1 in the specified column to be selected. Using the same letter A but now selecting records on a zone only basis, & and A-I meet the requirements and are selected.

AND Relationship

A maximum of three identifying characters can be described in one specification line. Thus, if the identification code consists of more than three characters, an AND line must be used. This means that the first three identifying characters are described in the first line. The additional identifying characters are described in as many following lines as are needed. Write the word AND in columns 14-16 to indicate an AND line (see *Examples, Example 1*).

A maximum of 20 AND lines can be used to describe the record identifying code for a record sequence if no OR lines are used. The record must contain all the characters indicated as its record identification code before the record identifying indicator turns on. AND lines are not allowed on console files which are interactive data entry files.

OR Relationship

A particular record type may be identified by two different codes. If this is the case, OR lines must be used to indicate that either one of the codes may be present to identify the record. A maximum of 20 OR lines can appear for each record sequence if no AND lines are used. Write the word OR in columns 14-15 to indicate an OR line (see *Examples, Example 2*).

Note: If AND lines and OR lines are combined, the total number of AND and OR lines for one record sequence cannot exceed 20.

Character Grouping by Zone (Z)	
¢ . < (+ |	& A B C D E F G H I
¦ $ *) ; ¬	– (minus) } J K L M N O P Q R
/ (comma) % (underscore) > ?	S T U V W X Y Z
: # @ ' (apostrophe) = ≠	blank 0 1 2 3 4 5 6 7 8 9

Character Grouping by Digit (D)	
blank & – (minus)] 0	H Q Y 8
/ A J 1	I R Z 9
B K S 2	¢ ¦ :
C L T 3	. $, #
D M U 4	< * % @
E N V 5	( ) _ (underscore) ' (apostrophe)
F O W 6	+ ; > =
G P X 7	| ¬ ? ≠

Figure 50. Characters Interpreted as Having the Same Zone or Digit

Examples

Example 1: Figure 51, insert A, shows a record identification code consisting of five characters. The first character is located in position 1, the other four positions 93, 94, 95, and 96. Since only three identifying characters can be described on one line, the word AND must be used on the next line to indicate that the last two characters of the code are part of the preceding record identification entries.

Example 2: Figure 51, insert B, shows the use of an OR line to describe record type identification codes. The record assigned resulting indicator 12 can be identified by two different codes. The record can be identified by a code consisting of a 5 in position 1 and a 6 in position 2 or a code consisting of a 6 in position 1.

Example 3: In Figure 51, insert A, the entry in column 32 indicates that the digit 9 must not be present in position 93 for record type 12.

Example 4: Figure 51, insert A, shows that only the zone portion of the character T located in position 94 is part of the identifying code. In position 96 only the digit portion of the character E is part of the code.

COLUMN 42

Column 42 is not used. Leave it blank.

Note: Column 43 begins the field description entries (columns 43-74) which must begin one line below the file and record identification entries (columns 7-42) for each file.

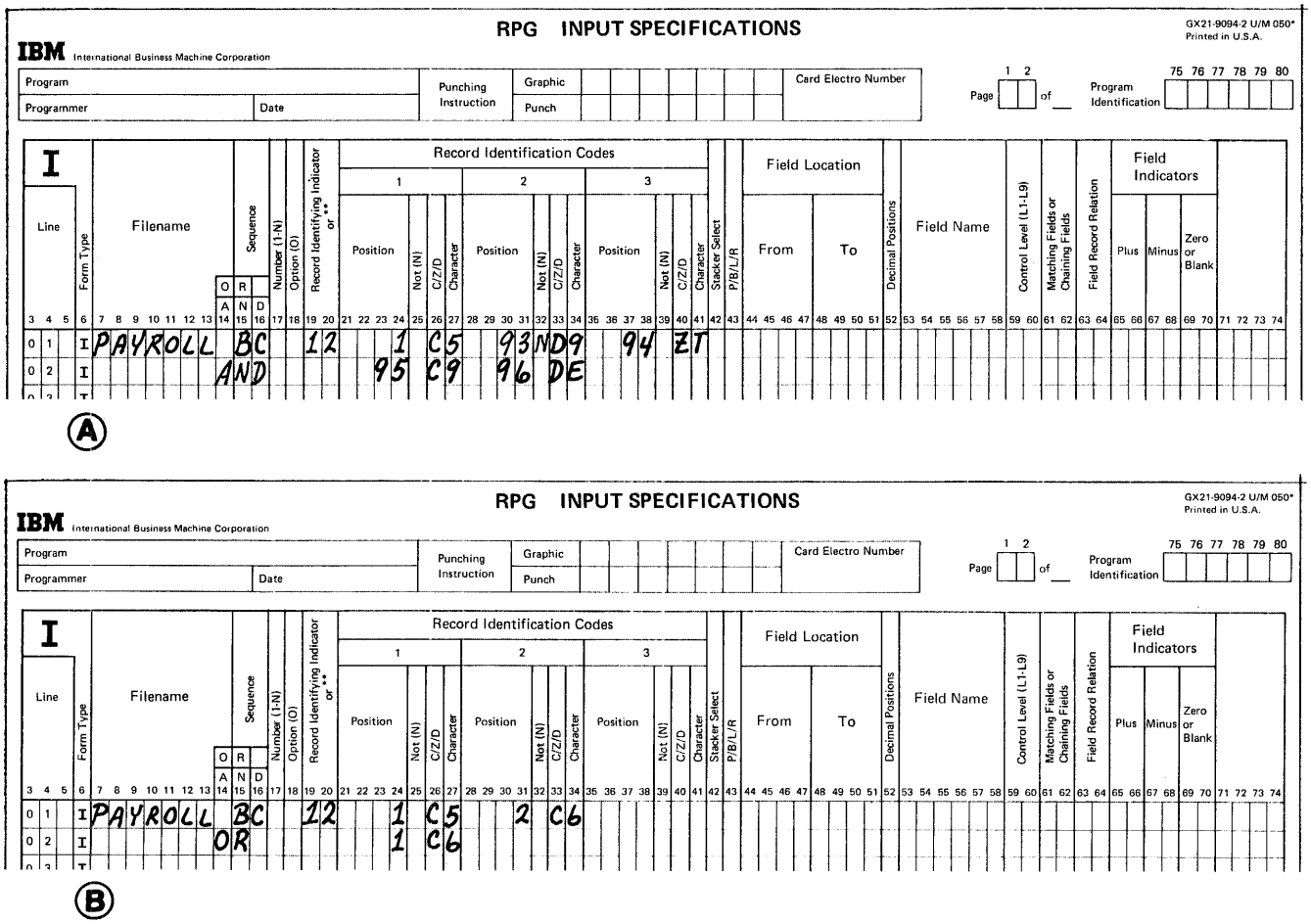


Figure 51. Record Identification Codes

COLUMN 43 (PACKED OR BINARY FIELD)

Entry	Explanation
Blank	Field is in unpacked decimal format or is alphameric. (Must be blank for console files.)
P	Field is in packed decimal format on the disk.
B	Field is in binary format on the disk.

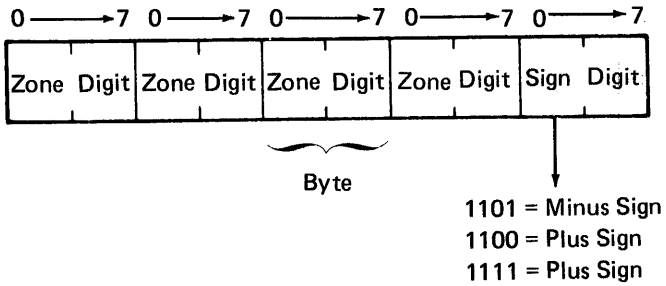
Column 43 indicates that a numeric field is in packed decimal or binary format. Packed decimal or binary fields can only be read from or written on the disk. Numeric data fields in packed decimal or binary format are converted by the system to the unpacked decimal format before they are processed. This conversion ignores decimal points.

Column 43 must contain a P if the input field named in columns 53-58 is in packed decimal format. Column 43 must contain a B if the input field named in columns 53-58 is in binary format.

Any array which was read in packed or binary format should have an entry in column 43 of the input specifications sheet. In this case, the from and to columns on the input specifications sheet should define the positions the array occupies in the record in the packed or binary format. The unpacked decimal length of each array element is defined by extension specifications.

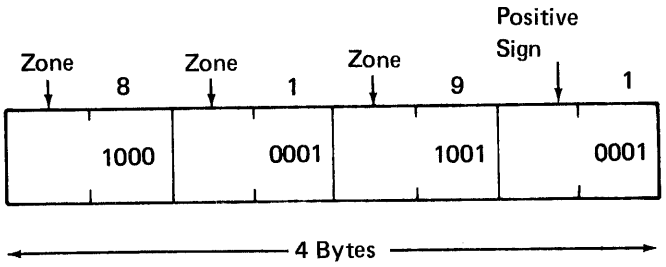
Unpacked Decimal Format (Blank)

Unpacked decimal format means that each byte of storage, whether on disk or in the computer, can contain one character. That character can be a decimal number or an alphabetic or special character. In the unpacked decimal format, each byte of storage is divided into a 4-bit zone portion and a 4-bit digit portion. The unpacked decimal format looks like this:



Note: RPG II does not perform data verification on numeric data. The value of the digit portion of a character is assumed to be the numeric value of that character.

The zone portion of the low-order byte indicates whether the decimal number is positive or negative. In unpacked decimal format, the zone portion is included for each digit in a decimal number; however, only the low-order zone portion serves as the sign. The decimal number 8,191 looks like this in unpacked decimal format:

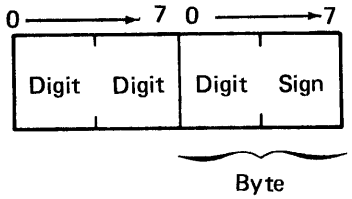


Once data is read into the computer, it must be represented in the unpacked decimal format before it can be processed. Thus, it is perfectly correct to store data on disk and read it into the computer in the unpacked decimal format. This eliminates the need to convert the input field since it is already in the required format. However, for more efficient use of disk storage space, you may store numeric data (decimal numbers) on disk in either the packed decimal or the binary format. Only numeric fields can be represented in the packed decimal or binary format.

Packed Decimal Format (P)

Packed decimal format means that a byte of disk storage (except for the low-order byte) can contain two decimal numbers. Since many of the fields in your disk files are made up of decimal numbers, you can conserve disk space by storing these fields in the packed decimal format. This format allows you to get almost twice as much data into a byte as you can using the unpacked decimal format.

In the packed decimal format, each byte of disk storage, except the low-order byte, is divided into two 4-bit digit portions. The rightmost portion of the low-order byte contains the sign (plus or minus) for that field. The packed decimal format looks like this:



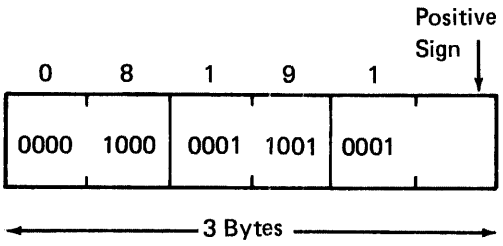
The sign portion of the low-order byte indicates whether the numeric value represented in the digit portions is positive or negative. In the packed decimal format, the sign is included for each decimal number; the zone portion is not given for each digit in the number. Compare how the decimal number 8,191 is represented in packed decimal format with its unpacked representation shown before (Figure 52).

Since data must be represented in unpacked decimal format once it is inside the computer, you must give the RPG II program an indication when input fields are in another format. A P-entry in column 43 indicates that the input field is in the packed decimal format and that the system must convert this field to the required unpacked format.

Packed fields can be up to eight bytes long. The following chart shows the packed equivalents for unpacked fields up to 15 bytes long:

Unpacked Length in Bytes	Packed Length in Bytes
15 14	8
13 12	7
11 10	6
9 8	5
7 6	4
5 4	3
3 2	2
1	1

Packed Decimal Format:



Unpacked Decimal Format:

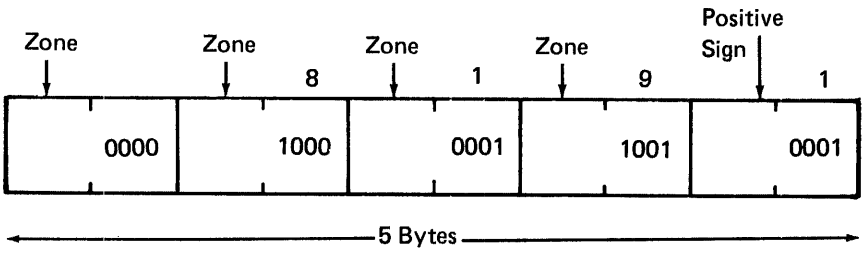
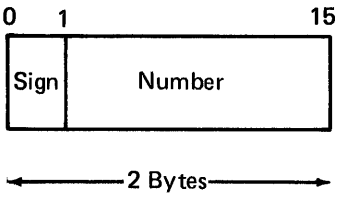


Figure 52. Packed Decimal and Unpacked Decimal Representation of 8,191

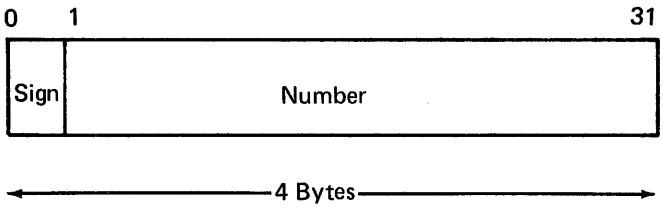
Binary Format (B)

Binary format means that two bytes of disk storage can contain a 4-digit number, and that four bytes of disk storage can contain a 9-digit number. The binary format allows you to save even more disk storage space than you can save using the packed decimal format. In the binary format, each field on disk must be either two or four bytes long.

Each 2-byte binary field consists of a 1-bit sign followed by a 15-bit numeric value. In binary format, a decimal number as high as 9,999 requires only two bytes of disk storage. For each 2-byte binary field stored on disk, the system automatically sets aside four bytes of storage to accommodate the field when it is unpacked. A 2-byte field in binary format looks like this:



Each 4-byte binary field consists of a 1-bit sign followed by a 31-bit numeric value. In binary format, a decimal number as high as 999,999,999 requires only four bytes of disk storage. For each 4-byte binary field stored on disk, the system automatically sets aside nine bytes of storage to accommodate the field when it is unpacked. A 4-byte field in binary format looks like this:



In each case, the sign portion of the high-order byte indicates whether the numeric value is positive (sign bit off) or negative (sign bit on). Notice that, in the binary format, the zone position of the decimal number is not given. Compare how the decimal number 8,191 is represented in binary format with packed and unpacked representation (Figure 53).

Since data must be represented in unpacked decimal format once it is inside the computer, you must give the RPG II program an indication when input fields are in another format. A B-entry in column 43 indicates that the input field is in the binary format and that the system must convert this field to the required unpacked format.

Note: Although packed and binary fields require less disk storage space, the conversion routines needed to handle such data increase the object program size.

COLUMNS 44-51 (FIELD LOCATION)

Entry	Explanation
1-4096	Beginning of a field (from) or end of a field (to)

Use columns 44-51 to describe the location on the record of each field containing input data named in columns 53-58. Enter the number of the record position in which the field begins in columns 44-47. Enter the number of the record position in which the field ends in columns 48-51.

A single-position field is defined by putting the same number in both from (columns 44-47) and to (columns 48-51). If a field of more than one position is defined, the number entered in columns 44-47 must be smaller than the number entered in columns 48-51.

The maximum field length for an unpacked numeric field is 15 positions (eight if the field is packed, four if it is binary). The maximum field length for an alphameric field is 256 characters.

Entries in these columns must end in columns 47 and 51. Leading zeros may be omitted.

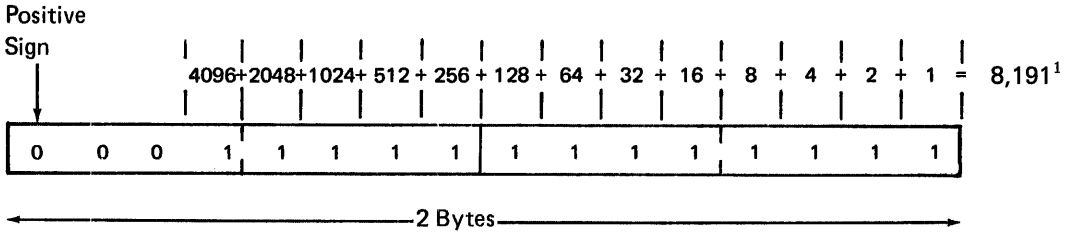
Note: See Part 2, Chapter 9 for console file considerations.

COLUMN 52 (DECIMAL POSITIONS)

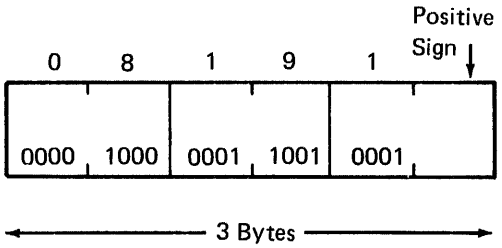
Entry	Explanation
Blank	Alphameric field
0-9	Number of decimal positions in numeric field

Use column 52 to indicate the number of positions to the right of the decimal in any numeric field named in columns 53-58. Column 52 must always have an entry when the field named in columns 53-58 is numeric. If you want to define a field as numeric with no decimal position, enter a 0. If a field is to be used in arithmetic operations or is

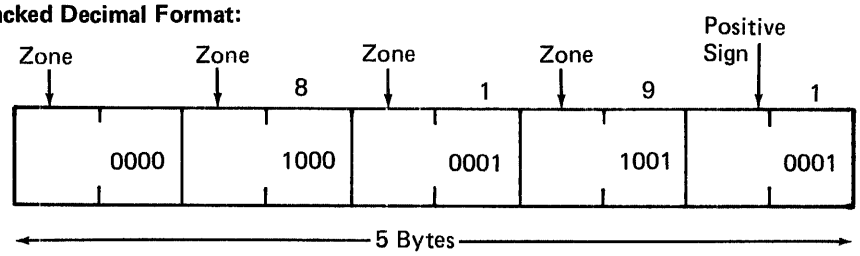
Binary Format:



Packed Decimal Format:



Unpacked Decimal Format:



[1]The numeric value of a positive binary number is obtained by adding the value of the bits that are on (1). The sign bit is not included. The numeric value of a negative binary number is obtained by adding the values of the bits that are off (0), plus one. The sign bit is not included.

Figure 53. Binary, Packed, and Unpacked Representation of 8,191

to be edited, it must be numeric. If the number of decimal positions specified for a field exceeds the length of that field, the number of decimal positions is assumed equal to the length of the field.

COLUMNS 53-58 (FIELD NAME)

Entry	Explanation
1-6 alphameric characters	Field name, array name, or array element
PAGE, PAGE 1, PAGE 2	Special words

Use columns 53-58 to name a field, array, or array element found on your input records. If you are referencing an array, additional entries may be needed in these columns (See *Arrays* under Part 2, *RPG II Programmer's Guide*). Use this name throughout the program whenever you refer to this field. You must indicate the names of the fields for all types of records. However, you should name only the fields that you use. For example, if you want to use only the first 10 positions of a record that is 96 positions long, define only positions 1-10 on the input specifications sheet.

Field Names

A field name can be from 1-6 characters long and must begin in column 53. The first character must be an alphabetic character. The remaining characters can be any combination of alphabetic and numeric characters (special characters are not allowed). Blanks cannot appear between characters in the name.

All fields in one type of record should have different names. If two or more fields on the same record type have the same name, only the field described last is used. However, fields from different record types can have the same name if the fields are the same length and contain the same type of data. This applies even if the fields are found in different locations in each record type. Duplicate field names should not be used if matching fields are specified in your program.

Numeric fields can have a maximum length of 15 characters. Alphameric fields can have a maximum length of 256 characters (40 for console files).

Fields that are used in arithmetic operations (see *Operation Codes* under Part 2, *RPG II Programmer's Guide*) or fields that are edited or zero suppressed (see *Column 38 and Columns 45-70* under *Output Specifications*) must be defined as numeric. This means that column 52 must have a decimal position entry.

A separate line is used for each field description.

Field Names in OR Relationship

Even though two or more record types contain identical fields, you must describe each field. This may require duplicate coding. To eliminate duplicate coding of identical fields from different record types, use the OR relationship. A maximum of 20 OR lines can be used for each record sequence group.

An OR relationship means that the fields named can be found in either one of the record types. You can use OR lines when:

- Two or more record types have the same fields in the same positions (Figure 54).
- Two or more record types have some fields which are identical and some fields which differ in location, length, or type of data. See *Columns 63-64* for sample coding of such record types.

Write the word OR in columns 14 and 15 to indicate an OR line (Figure 54). If there are several AND or OR lines, field description lines start after the last record identification line.

Example: Figure 54 shows how the use of OR lines can save duplicate coding. The two different record types (one identified by a 5 in column 1, the other by a 6 in column 1) both have identical fields which must be described. Figure 54, insert A, shows one way of doing this. Figure 54, insert B, shows the use of OR lines to do the same thing with less coding. The coding in Figure 54, insert B, says that all four fields can be found on either the record type identified by the 5 in column 1 or the record type with a 6 in column 1.

Special Words (PAGE, PAGE1, PAGE2)

If your printed report has several pages, you may want to number the pages. The special word PAGE allows you to indicate that page numbering is to be done. When you use a PAGE entry on the output specifications sheet, page numbering automatically starts with 1 (see *Columns 32-37* under *Output Specifications*).

If you want to start at a page number other than 1, you can enter that page number in a field of an input record and name that field PAGE in columns 53-58. The number entered in the PAGE field of the input record should be one number less than the starting page number. If your numbering starts with 24, enter a 23 in the PAGE field. The PAGE field can be of any length, but must have zero decimal positions (Figure 55). Any entry you make in the PAGE field should be right-justified, such as 0023.

Page numbering can be restarted during a program run by entering a number in a PAGE field of any input record. The PAGE field can be defined and used in calculations like any other field.

The three possible PAGE entries (PAGE, PAGE1, and PAGE2) are provided for numbering different page types in the output file.

RPG INPUT SPECIFICATIONS

IBM International Business Machine Corporation

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

Program | Programmer | Date | Punching Instruction | Graphic | Punch | Card Electro Number | Page 1 2 of __ | Program Identification 75 76 77 78 79 80

I

Line	Form Type	Filename	Sequence (O A, R N, D 14 15 16)	Number (1-N)	Option (O)	Record Identifying Indicator or ** 19 20	Record Identification Codes 1: Position 21 22 23 24	1: Not (N) 25	1: C/Z/D 26	1: Character 27	2: Position 28 29 30 31	2: Not (N) 32	2: C/Z/D 33	2: Character 34	3: Position 35 36 37 38	3: Not (N) 39	3: C/Z/D 40	3: Character 41	Stacker Select 42	P/B/L/R 43	Field Location From 44 45 46 47	Field Location To 48 49 50 51	Decimal Positions 52	Field Name 53 54 55 56 57 58	Control Level (L1-L9) 59 60	Matching Fields or Chaining Fields 61 62	Field Record Relation 63 64	Field Indicators Plus 65 66	Field Indicators Minus 67 68	Field Indicators Zero or Blank 69 70	71 72 73 74
01	I	SALES	AA			14	1		C5																							
02	I																				5	8	DEPT								
03	I																				9	14	EMPNO								
04	I																				46	50	ITEM								
05	I																				66	70	COST								
06	I		BB			15	1		C6																							
07	I																				5	8	DEPT								
08	I																				9	14	EMPNO								
09	I																				46	50	ITEM								
10	I																				66	70	COST								
11	I																															

(A)

RPG INPUT SPECIFICATIONS

IBM International Business Machine Corporation

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

Program | Programmer | Date | Punching Instruction | Graphic | Punch | Card Electro Number | Page 1 2 of __ | Program Identification 75 76 77 78 79 80

I

Line	Form Type	Filename	Sequence (O A, R N, D 14 15 16)	Number (1-N)	Option (O)	Record Identifying Indicator or ** 19 20	Record Identification Codes 1: Position 21 22 23 24	1: Not (N) 25	1: C/Z/D 26	1: Character 27	2: Position 28 29 30 31	2: Not (N) 32	2: C/Z/D 33	2: Character 34	3: Position 35 36 37 38	3: Not (N) 39	3: C/Z/D 40	3: Character 41	Stacker Select 42	P/B/L/R 43	Field Location From 44 45 46 47	Field Location To 48 49 50 51	Decimal Positions 52	Field Name 53 54 55 56 57 58	Control Level (L1-L9) 59 60	Matching Fields or Chaining Fields 61 62	Field Record Relation 63 64	Field Indicators Plus 65 66	Field Indicators Minus 67 68	Field Indicators Zero or Blank 69 70	71 72 73 74
01	I	SALES	AA			14	1		C5																							
02	I		OR			15	1		C6																							
03	I																				5	8	DEPT								
04	I																				9	14	EMPNO								
05	I																				46	50	ITEM								
06	I																				66	70	COST								
07	I																															
08	I																															

(B)

Figure 54. Record Types with Identical Fields

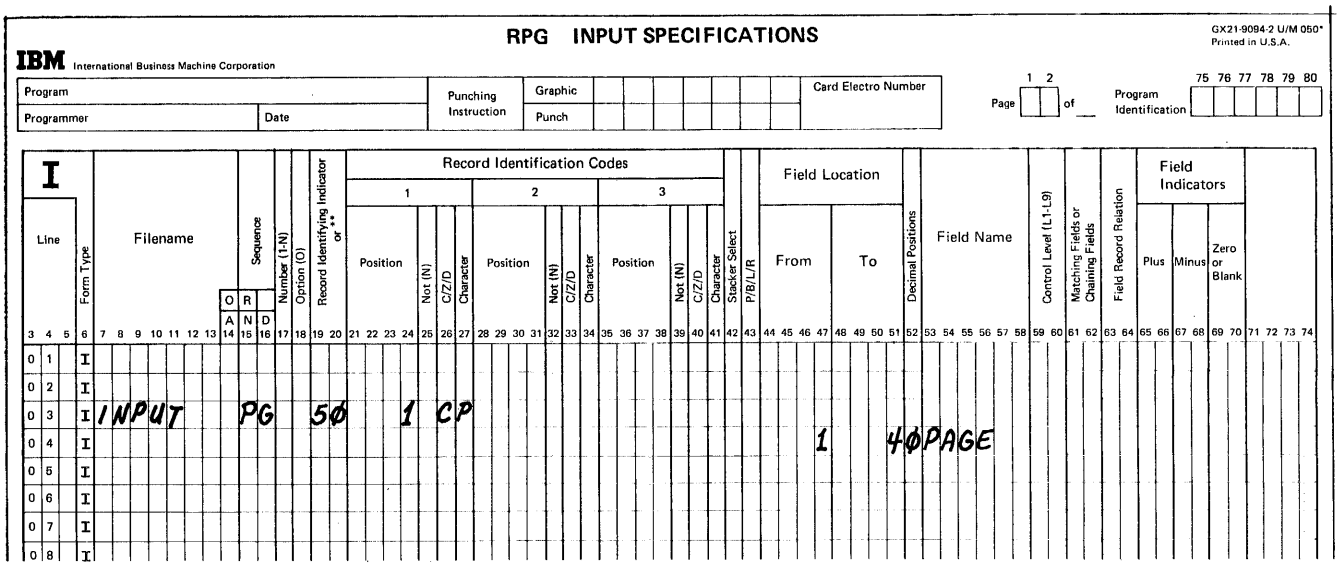


Figure 55. Page Record Description

COLUMNS 59-60 (CONTROL LEVEL)

Entry	Explanation
L1-L9	Any control level indicator

Use columns 59-60 to assign control level indicators to input fields. Control level indicators cannot be associated with a chained file or a demand file. Control level indicators specify when operations are to be performed. You can assign a control level indicator to any field; this field is then known as a control field. It is checked for a change in information. When information in the control field changes, a control break occurs. All records having the same information in the control field are known as a control group.

Whenever a record containing a control field is read, the data in the control field is compared with data in the same control field from the previous record. When a control break occurs, the control level indicator turns on. Operations conditioned by the control level indicator are then performed (see *Columns 7-8* under *Calculation Specifications* or *Columns 23-31* under *Output Specifications*).

There are nine different control levels (L1-L9). When a certain control level indicator turns on, all control level indicators lower than it also turn on. For example, if control level indicator 3 turns on, control level indicators 1 and 2 also turn on.

The indicators are ranked in order of importance. The larger numbers rank higher than smaller numbers. L4 has a higher rank than L1. The importance of a control field in relation to other fields determines how you assign indicators. For example, the type of data which demands a subtotal has a lower control level indicator than data which needs a final total. A field containing department numbers is given a higher control level indicator than a field containing employee numbers (see *Examples, Example 1*).

Control level indicator L0, since it is always on, cannot be assigned to a control field. Nevertheless, you can use it to condition operations (see *Columns 7-8* under *Calculation Specifications*).

Normally, control level indicators are used to:

- Condition certain total calculations to be performed when the information in the control field changes.
- Condition certain total output operations to be done after totals are accumulated for one control group.
- Condition certain detail calculation or output operations to be done on the record that causes a change in a control field (first record of a new control group).

Assigning Control Level Indicators

The following considerations apply to assigning control level indicators:

- If the same control level indicator is used in different record types or in different files, the control fields associated with that control level indicator must be the same length and same type (alphabetic or numeric). See *Examples, Example 2.*

- In the same record type, record positions in control fields assigned different control level indicators may overlap (Figure 56). However, the total number of positions assigned as control fields must not be greater than 144. In Figure 56, for example, a total of 15 positions has been assigned to control levels.

- Field names are ignored in control level operations. Therefore, fields from different record types which have been assigned the same control level indicator may have the same name.

- Control levels need not be written in any sequence. An L2 entry can appear before L1. Also, there may be gaps in the control levels assigned.

- When numeric control fields with decimal positions are compared to see if a control break has occurred, they are always treated as if they have no decimal positions. For instance, 3.46 is considered equal to 346.

- If a field is specified as numeric, only the digit portion determines if a control break has occurred. This means that a field is always considered to be positive. A -5 is considered equal to +5.

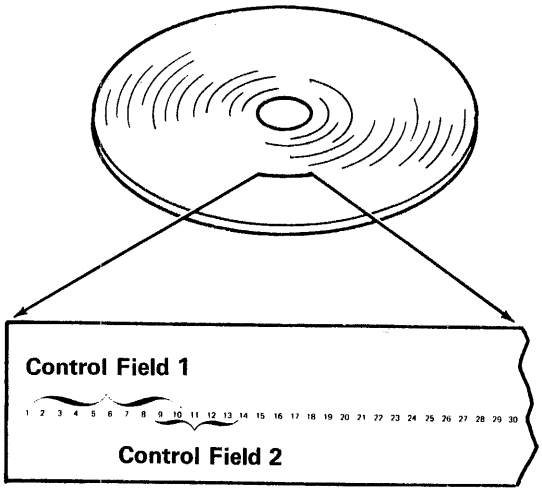


Figure 56. Overlapping Control Fields in a Disk Record

- All control fields given the same control level indicator are considered numeric if any one of those control fields is described as numeric (column 52 has an entry). This means that when numeric control fields are compared to see if the information has changed, only the digit portion of each character is compared.

- Control fields are initialized to hexadecimal zeros.

- A control break may occur after the first record containing a control field is read. The control fields in this record are compared to an area in storage which contains hexadecimal zeros. Since fields from two different records are not being compared, total calculations and total output operations are bypassed for this cycle. A control break does occur then, but it is not considered to be a true control break.

- If different record types in a file do not have the same number of control fields, unwanted control breaks can occur. See *Examples, Example 3* for a method of avoiding unwanted control breaks.

- A control field cannot be specified as binary (B in column 43). However, it can be specified as packed numeric (P in column 43).

Split Control Fields

If a control field is made up of more than one field of a record, it is then known as a split control field. A split control field is created when the same indicator is assigned to two or more connected or unconnected fields on the same record type.

All fields in one record that have the same control level indicators are combined by the program in the order specified by the input specifications and treated as one control field (see *Examples, Example 4*). Some special rules for split control fields are:

- For one control level indicator, a field can be split in some record types and not in others if the field names are different. However, the length of the field, whether split or not, must be the same in all record types.

- The length of the portions of a split control field can vary for different record types if the field names are different. However, the total length of the portions must always be the same.

- No other specification lines can come between lines which describe split control fields.
- If one section of a split control field is numeric, the whole field is considered numeric.
- A numeric split control field can have more than 15 characters if ~~any~~ one portion of the split field ~~does~~ not exceeds 15 characters and the sum of all control fields is not greater than 144 characters.
- A split control field cannot be made up of a packed decimal field and an unpacked decimal field. Both portions of the control field must be packed or both must be unpacked.

Note: Additional rules applying to control level indicators when used with indicators in the field record relation columns are discussed in *Columns 63-64 (Field Record Relation)*.

Examples

Example 1: Figure 57 shows the assignment of three indicators. The names of the control fields (DIVSON, DEPT, EMPLNO) give an indication of their relative importance. The division (DIVSON) is the most important group. It is given the highest control level indicator used (L3). The department (DEPT) ranks below the corporation; L2 is assigned to it. The employee field has the lowest control level indicator (L1) assigned.

Example 2: Figure 57 shows that the same control level indicators can be used for different record types. Notice, however, that the control fields having the same indicators are the same length. EMPLNO, in both cases, is six columns in length, DEPT is four, and DIVSON is one.

RPG INPUT SPECIFICATIONS

IBM International Business Machine Corporation

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

Program		Punching Instruction	Graphic							Card Electro Number
Programmer	Date		Punch							

Page ___ of ___ Program Identification

Line	Form Type	Filename	Sequence (OR AND)	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	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
01	I	EMPLREP	AB			10		1		CA																				
02	I																				5	10		EMPLNO	L1					
03	I																				11	30		NAME						
04	I																				80	80		DIVSON	L3					
05	I																				33	33		SHIFT						
06	I																				2	5		DEPT	L2					
07	I	.	CD			20		1	N	CA																				
08	I																				5	10		EMPLNO	L1					
09	I																				11	14		DEPT	L2					
10	I																				60	70		HRSWKD						
11	I																				80	80		DIVSON	L3					
12	I																													

Figure 57. Control Level Indicators (Two Record Types)

Example 3: Different record types normally contain the same number of control fields. However, some applications require a different number of control fields in some records. This is shown in Figure 58, insert A. The salesman records contain only the L2 control field. The item records contain both L1 and L2 control fields. With normal RPG II coding, an unwanted control break is created by the first item record following the salesman record. This is recognized by an L1 control break immediately following the salesman record and results in an asterisk being printed on the line below the salesman record (Figure 58, insert B).

A	(L2) Salesman Number	Salesman Name
1	2 3	4 16

Salesman Record

B	(L2) Salesman Number	(L1) Item Number	Amount
1	2 3	4 6	7 9

Item Record

Ⓐ

```
01    DICK LOVE                 Unwanted
                          *     Control
          100          3        Break
          100          2
                       5  *
          101          4
                       4  *
                       9  **

02    CAL WINBUSH               Unwanted
                          *     Control
          100          6        Break
          100          2
                       8  *
          101          3
                       3  *
                      11  **

                      20
```

Output Showing Unwanted Control Level Break

```
01    DICK LOVE

          100          3
          100          2
                       5  *
          101          4
                       4  *
                       9  **

02    CAL WINBUSH

          100          6
          100          2
                       8  *
          101          3
                       3  *
                      11  **

                      20
```

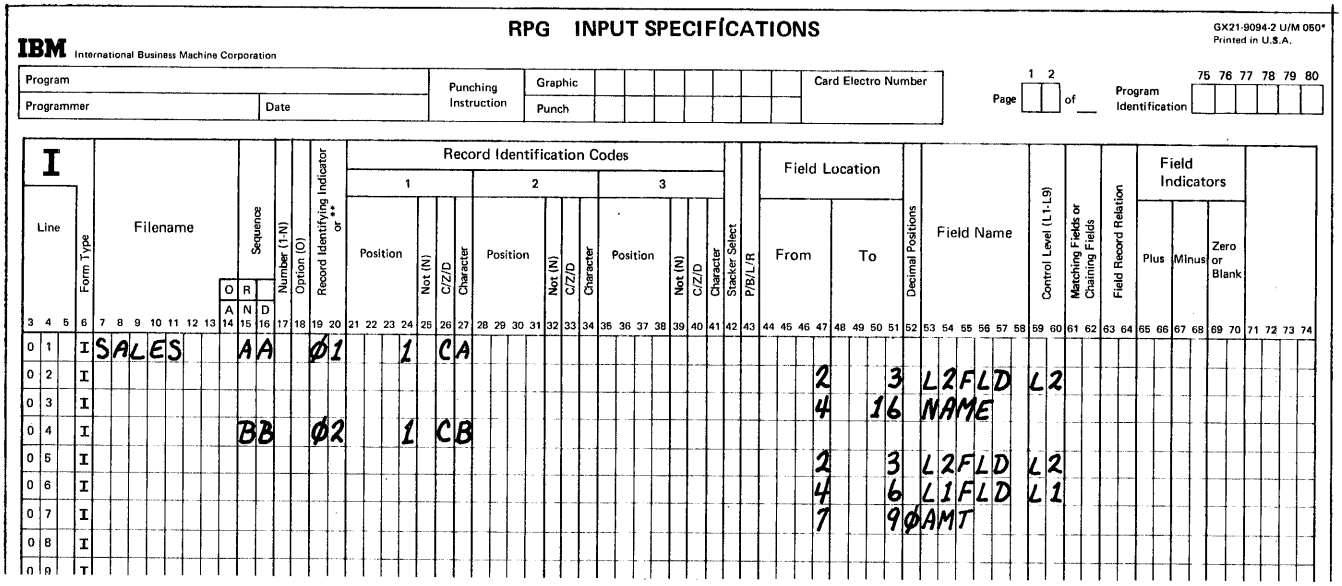
Corrected Output

Ⓑ

Figure 58 (Part 1 of 3). Unwanted Control Breaks

Figure 58, inserts C and D, contains excerpts from a program that processes the input shown in Figure 58, insert A, and prevents the unwanted control break from occurring. The corrected output produced is shown in Figure 58, insert B.

Line 01 of the calculation specifications sheet sets on indicator 11 when the salesman record is read. When the next item record causes an L1 control break, no total output is printed because indicator 11 is on (line 07 of output specifications sheet). Detail calculations are then processed for the item record and line 02 of the calculation specifications sheet sets indicator 11 off. This allows the normal L1 control break to occur.

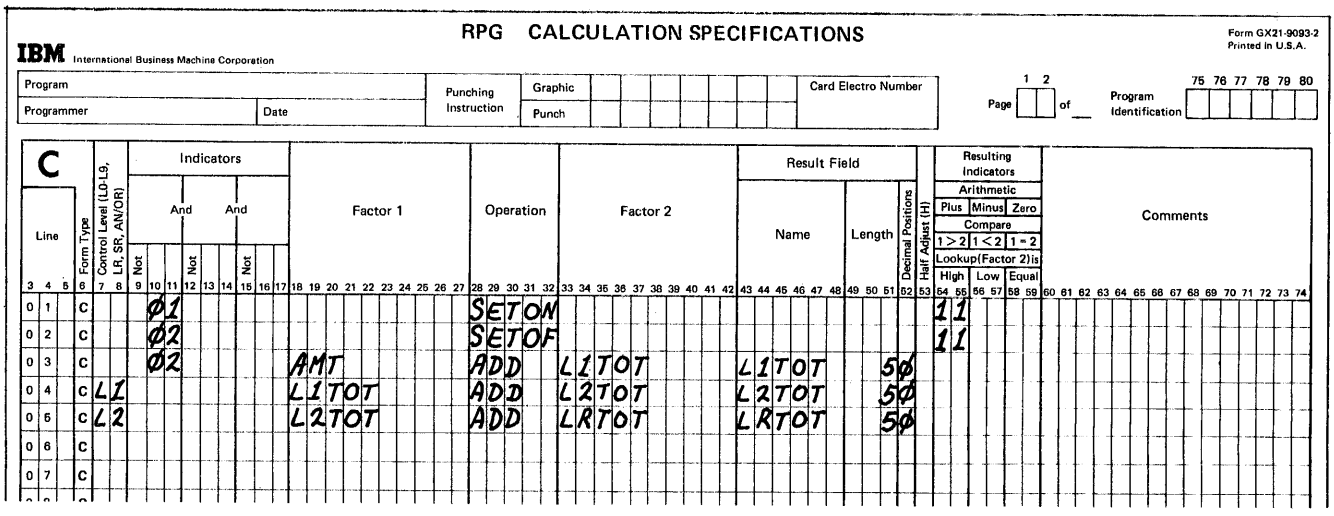


RPG INPUT SPECIFICATIONS

IBM International Business Machine Corporation

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

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or *	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	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
01	I	SALES	AA			01	1		CA																					
02	I																				2	3	L2FLD	L2						
03	I																				4	16	NAME							
04	I		BB			02	1		CB																					
05	I																				2	3	L2FLD	L2						
06	I																				4	6	L1FLD	L1						
07	I																				7	90	AMT							
08	I																													
09	I																													



RPG CALCULATION SPECIFICATIONS

IBM International Business Machine Corporation

Form GX21-9093-2 Printed in U.S.A.

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And Not	Indicators: And Not	Indicators: Not	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Result Field: Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus	Resulting Indicators: Arithmetic Minus	Resulting Indicators: Arithmetic Zero	Resulting Indicators: Compare 1>2	Resulting Indicators: Compare 1<2	Resulting Indicators: Compare 1=2	Resulting Indicators: Lookup(Factor 2)is High	Resulting Indicators: Lookup(Factor 2)is Low	Resulting Indicators: Lookup(Factor 2)is Equal	Comments
01	C		01				SETON						11									
02	C		02				SETOF						11									
03	C		02			AMT	ADD	L1TOT	L1TOT	50												
04	C	L1				L1TOT	ADD	L2TOT	L2TOT	50												
05	C	L2				L2TOT	ADD	LRTOT	LRTOT	50												
06	C																					
07	C																					
08	C																					

Figure 58 (Part 2 of 3). Unwanted Control Breaks

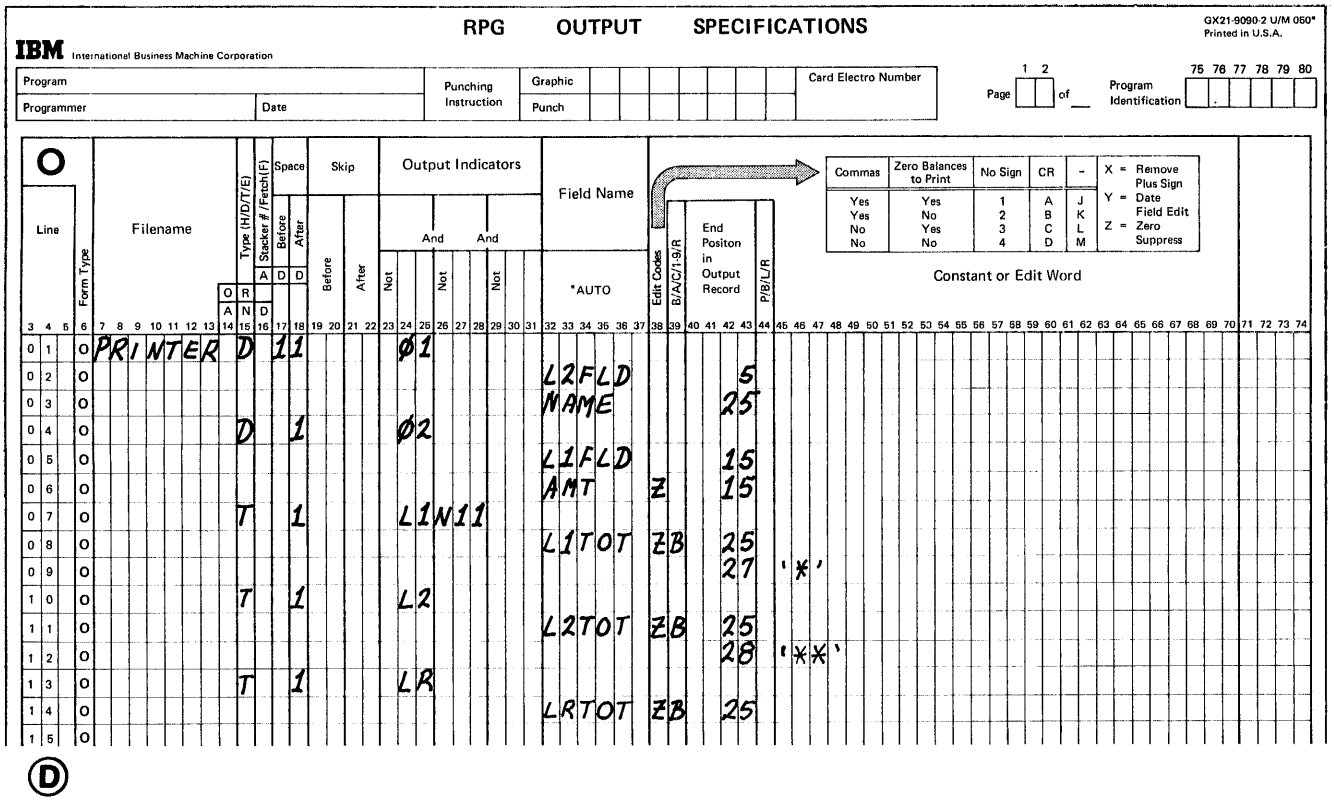


Figure 58 (Part 3 of 3). Unwanted Control Breaks

Example 4: Figure 59 shows a split control field made up of three portions. The control level indicator (L4) which is used for all three portions indicates that they are all to be treated as one control field. The field can be pictured as follows:

CUSNO	ACCTNO	REGNO
1 6 7 9	8 6 5 3 9 7	1 1 1

The first field assigned the same control level indicator begins the control field; the last ends it.

COLUMNS 61-62 (MATCHING FIELDS)

Entry	Explanation
M1-M9	Any matching level

Use columns 61-62 to specify matching fields and sequence checking. Matching fields and sequence checking cannot be specified for chained files or demand files.

An entry in columns 61-62 indicates:

- Matching fields and sequence checking when you have two or more input or update files with match fields.
- Sequence checking only when you have just one input or update file.

Matching Fields

Specify matching fields when you want to compare records from two or more input or update files to determine when records match. Records can be matched by matching one field, many fields, or entire records. You can indicate as many as nine matching fields (M1-M9). Whenever the contents of the match fields from records of the primary file are the same as the contents of the match fields from a secondary file, the matching record (MR) indicator turns on. M1-M9 are used only to identify fields by which records are matched. The values M1-M9 are not indicators but do cause MR to turn on when a match occurs. You

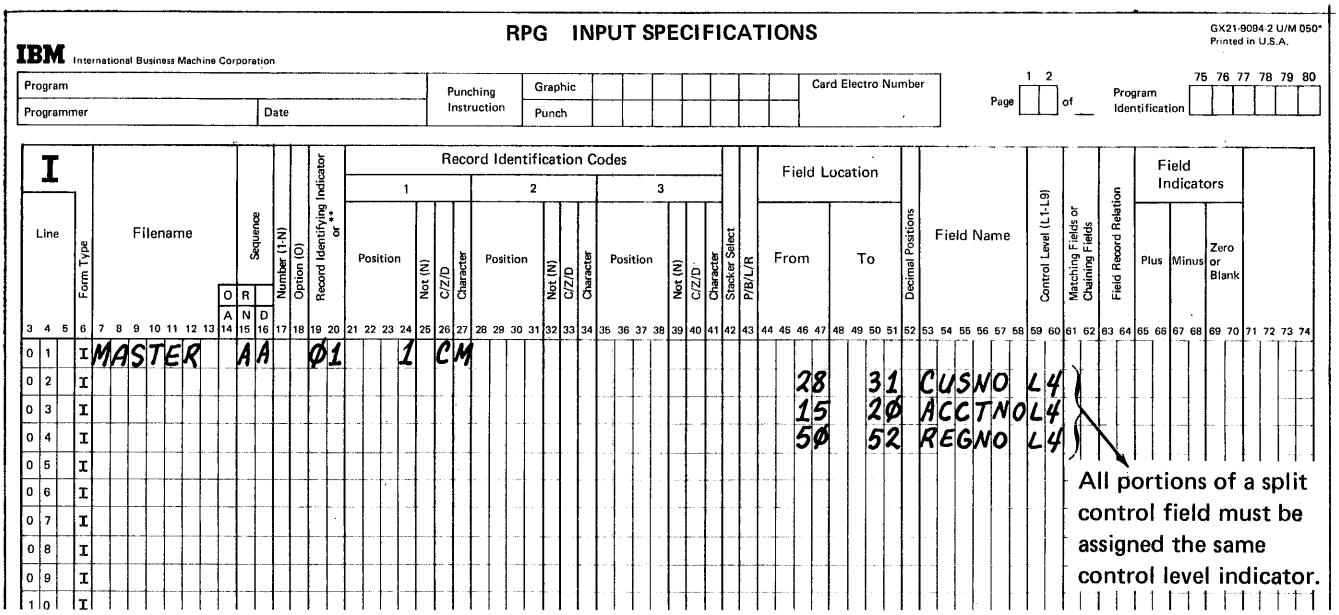


Figure 59. Split Control Fields

can then use the MR indicator to condition those operations which are to be done only when records match (see *Columns 9-17* under *Calculation Specifications* and *Columns 23-31* under *Output Specifications*).

Assigning Matching Field Values (M1-M9)

The following considerations apply to assigning matching field values:

- Sequence checking is automatically done for all record types with matching field specifications. The contents of the fields to which M1-M9 are assigned are checked for correct sequence. An error in sequence stops the program. The record which caused the halt is not processed. When the machine is restarted, the next record from the same file is read. Thus, all matching fields must be in the same order, either all ascending or all descending. See *Column 18 (Sequence)* under *File Description Specifications.*

- Not all files used in the job must have matching fields. Not all record types within one file must have matching fields either. However, at least one record type from two files must have matching fields if files are ever to be matched.

- The same number of matching fields must be specified for all record types which are used in matching. The same matching record values must also be used for all types (see *Examples, Example 1*).

- All match fields given the same matching record value (M1-M9) must be the same length and type (alphameric or numeric).

- Record positions of different matching fields can overlap, but the total length of all fields must not exceed 144 characters.

- If more than one matching field is specified for a record type, all the fields are combined and treated as one continuous matching field (see *Examples, Example 2*). They are combined according to descending sequence (M9-M1) of matching record values.

- Matching fields cannot be split. This means that the same matching field value cannot be used twice for one type of record.

- Matching fields can be either alphameric or numeric. However, all matching fields given the same matching record value (M1-M9) are considered numeric if any one of those matching fields is described as numeric.

- When numeric fields having decimal positions are matched, they are treated as if they had no decimal position. For instance, 3.46 is considered equal to 346.

- Only the digit portions of numeric match fields are compared. Even though a field is negative, it is considered to be positive since the sign of the numeric field is ignored. Thus, a -5 will match with a +5.

- Whenever more than one matching record value is used, all match fields must match before the MR indicator turns on. For example, if matching fields M1, M2, M3 are specified, all three fields from one record must match all three fields from the other record. A match on only the M1 and M2 fields will not turn on the MR indicator (see *Examples, Example 1*).

- Field names are ignored in matching record operations. Therefore, fields from different record types that are assigned the same match level can have the same name.

- If you have defined an alternate collating sequence for your program, alphameric fields are matched according to the sequence you have specified.

- A matching field cannot be specified as binary (B in column 43). However, a matching field can be specified as packed (P in column 43).

Note: Additional rules applying to matching records when used with entries in the field record relation columns are discussed in *Columns 63-64*.

Processing Matching Records

Matching records for two or more files are processed in the following manner:

- Whenever a record from the primary file matches a record from the secondary file, the primary file is processed first. Then the matching secondary file record is processed. (Remember, the record identifying indicator which identifies the record type just selected is on at the time the record is processed. This indicator is often used to control the type of processing that takes place.)

- Whenever records from ascending files do not match, the record having the lowest match field content is processed first (Figure 60). Whenever records from descending files do not match, the record having the highest match field content is processed first.

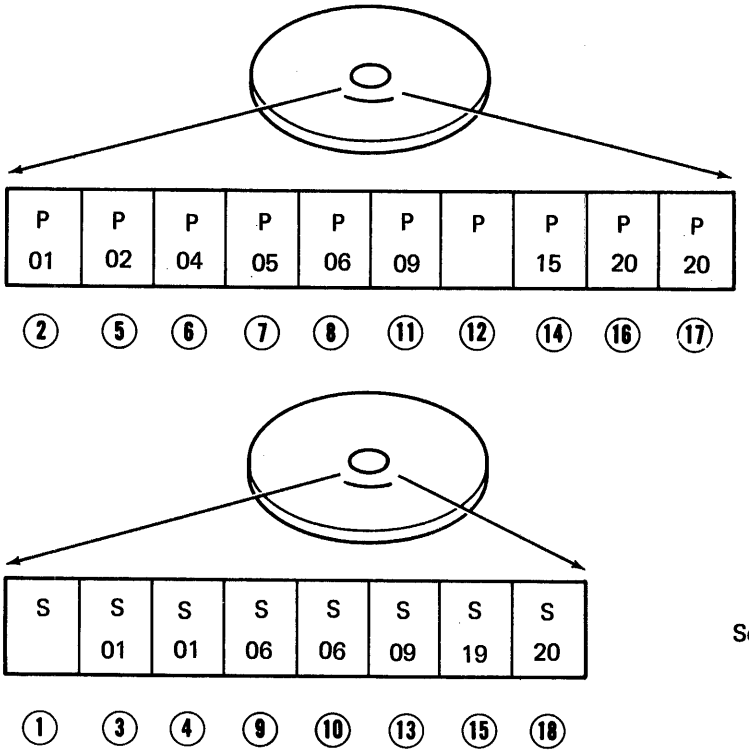
- A record type which has no matching field specification is processed immediately after the record it follows. The MR indicator is off. If this record type is first in the file, it is processed first even if it is not in the primary file (Figure 60).

- The matching of records makes it possible to enter data from primary records into their matching secondary records since the primary record is processed before the matching secondary record. However, the transfer of data from secondary records into matching primary records can only be done through look ahead fields (see *Look Ahead* under Part 2, *RPG II Programmer's Guide*).

For additional information on matching records from more than two files, see Chapter 3, *Multifile Processing* under Part 2, *RPG II Programmer's Guide*.

Sequence Checking

Make an entry in columns 61-62 when you want to sequence check records within one input or update file. This entry causes sequence checking of the data in the fields to which M1-M9 are assigned. See *Columns 15-15 (Sequence)* for sequence checking of record types. You can use as many as nine fields (M1-M9) to sequence check. The sequence (ascending or descending) of your record file must be specified by the file description specifications. See *Column 18 (Sequence)* under *File Description Specifications.* An entry in columns 61-62 indicates that the records are to be checked to see if they really are in the sequence specified (see *Examples, Example 3*).



The circled numbers represent the order of record selection from the two files. The following notes clarify the reasons for the selection of various records.

(1) This record is selected first because it has no match field.

(2) When a match occurs, the primary record is always selected first.

(5) When records have no match, the record with the lowest match field is selected, regardless of file.

(12) A record with no match field is processed immediately after the record it follows, regardless of file or match condition.

Figure 60. Processing Order According to Matching Records

Examples

Example 1: Figure 61 shows three record types that are used in matching records. All record types have three matching fields specified, and all use the same values (M1, M2, M3) to indicate which fields must match. The MR indicator turns on only if all three match fields in either of the record types from the MASTER file are the same as all three fields from the record in the WEEKLY file.

Example 2: Figure 61 indicates three matching fields on one record. These three are combined and treated as one matching field organized as follows:

EMPLNO	DEPT	DIVSON
M1	M2	M3

The order in which the fields are specified by the input specifications does not affect the organization of the match fields in the computer.

RPG INPUT SPECIFICATIONS

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

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic	Card Electro Number
Programmer	Date		Punch	

Page ___ of ___ Program Identification

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	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
01	I	MASTER	AA			01	95		CS												1	5		EMPLNO		M1				
02	I																				10	25		PAYRAT						
03	I																				36	40		DIVSON		M3				
04	I																				31	35		DEPT		M2				
05	I																													
06	I		BB			02	95		CT																					
07	I																				1	5		EMPLNO		M1				
08	I																				10	15		HRSWKD						
09	I																				20	24		DEPT		M2				
10	I																				25	29		DIVSON		M3				
11	I	WEEKLY	DL			03																								
12	I																				1	5		EMPLNO		M1				
13	I																				6	10		DIVSON		M3				
14	I																				11	15		DEPT		M2				
15	I																				20	25		HRSWKD						
16	I																													
17	I																													
18	I																													
19	I																													
20	I																													

Figure 61. Match Fields

Example 3: An input file called MASTER is to be sequence checked using three fields (Figure 62). Data from two records is shown below:

Data from First Record		Data from Second Record	
DEPT	008	DEPT	003
REGION	051	REGION	025
DIVSON	003	DIVSON	005

In sequence checking, all fields are treated as one continuous field. Thus, the matching fields look like:

	M3	M2	M1
Record 1	003	051	008
Record 2	005	025	003

The matching field from record 1 is compared with the matching field from record 2. If the file is specified to be in ascending sequence, the records are in order since 005025003 is higher than 003051008. However, if the file is specified as having a descending sequence, record 2 is out of order.

COLUMNS 63-64 (FIELD RECORD RELATION)

Entry	Explanation
01-99	Record identifying indicator assigned to a record type.
L1-L9	Control level indicator previously used.
MR	Matching record indicator.
U1-U8	External indicator previously set.
H1-H9	Halt indicator previously used.

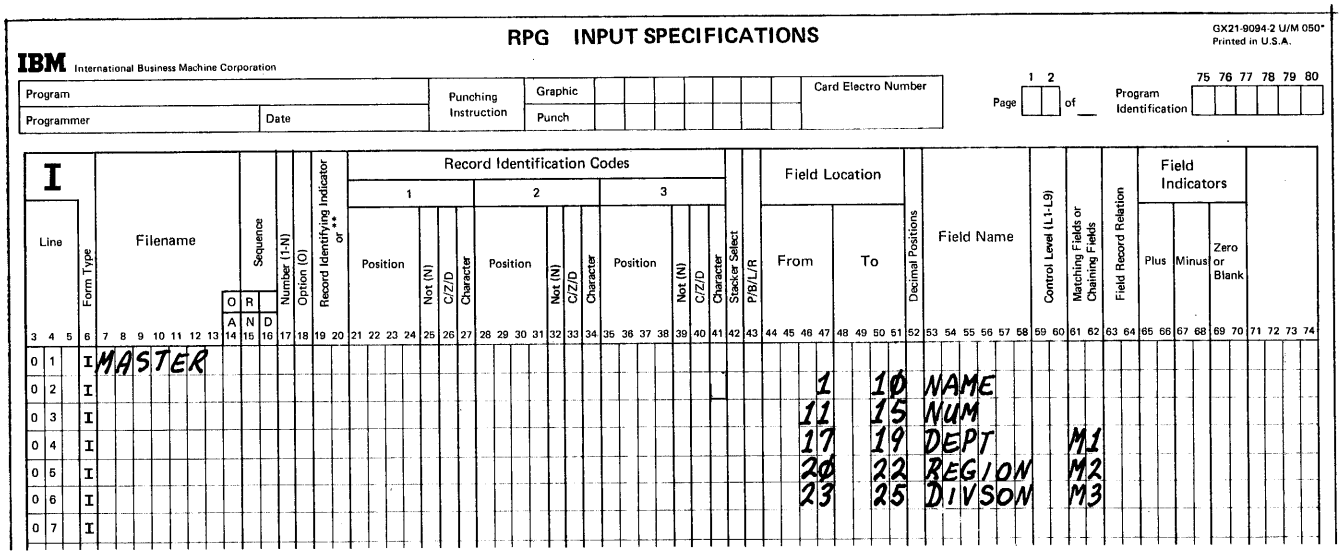


Figure 62. Match Fields (Sequence Checking Within a File)

Columns 63-64 have several uses which are discussed after these general rules:

- All fields, including matching or control fields, that have no field record relation specification should come before those that do.

- All fields related to one record type (that is, having the same field record relation entry) should be entered as a group in specification lines following one another for more efficient use of storage. These fields can, however, be entered in any order.

- All portions of a split control field must be assigned the same field record relation indicator and must be entered as a group in specification lines following one another (see *Examples, Example 3*). For more information on split control fields, see *Columns 59-60 (Control Level)*.

- When used with match or control fields, the field record relation indicator must match a record identifying indicator for this file, and the match or control fields must be grouped according to the field record relation indicator.

- When any match value (M1-M9) is specified without field record relation, all match values used must be specified once without field record relation. If all match fields are not common to all records, a dummy match field should be used.

Note: Columns 63-64 must be blank for console files, see Part 2, Chapter 9.

Record Identifying Indicators (01-99)

Columns 63-64 are commonly used when several record types are specified in an OR relationship. Fields which have no field record relation indicator are associated with all the record types in the OR relationship. This is fine when all record types have the same fields. But if the record types in the OR relationship have some fields that are the same and some that are not the same, you do not want to associate every field with all records. Therefore, you must have some way of relating a field to a certain record. To do this, place in columns 63-64 the record identifying indicator found in columns 19-20 of the record type on which the field is found or specify an indicator (01-99) which was defined in your program and condition movement from the input area to the storage area (see *Examples, Example 1*).

Control fields (indicated by entries in columns 59-60) and matching fields (indicated by entries in columns 61-62) can also be related to a particular record type in an OR relationship by a field record relation entry. Control fields or matching fields that are not related to any particular record type in the OR relationship by the field record relation indicator are used with all record types in the OR relationship.

When two control fields have the same control level indicator or two matching fields have the same matching level entry, it is possible to assign a field record relation indicator to just one of the control fields or to just one of the matching fields. In this case, only the specification having the field record relation indicator is used when that indicator is on. If none of the field record relation indicators are on for that control field or matching field, the specification without a field record relation indicator is used. Control fields and matching fields cannot have an L1-L9 or MR entry in columns 63-64.

When any match value (M1-M9) is specified without field record relation, all match values used must be specified once without field record relation. If all match fields are not common to all records, a dummy match field should be used (see *Examples, Example 2*).

Control Level (L1-L9) and Matching Record (MR) Indicators

Another situation for which you can use these columns is when you want to accept and use data from a particular field only when a certain condition (such as matching records or a control break) occurs. You indicate the conditions under which you accept data from a field by indicator L1-L9 or MR. MR cannot be used with demand files. Data from the field named in columns 53-58 is accepted only when the indicator is on.

External Indicators (U1-U8)

You can also use these columns to condition a specification by an external indicator (U1-U8). The external indicator, which you set prior to processing, controls whether or not the specification is done. When the indicator is on, the specification is done; when it is off, the specification is not done.

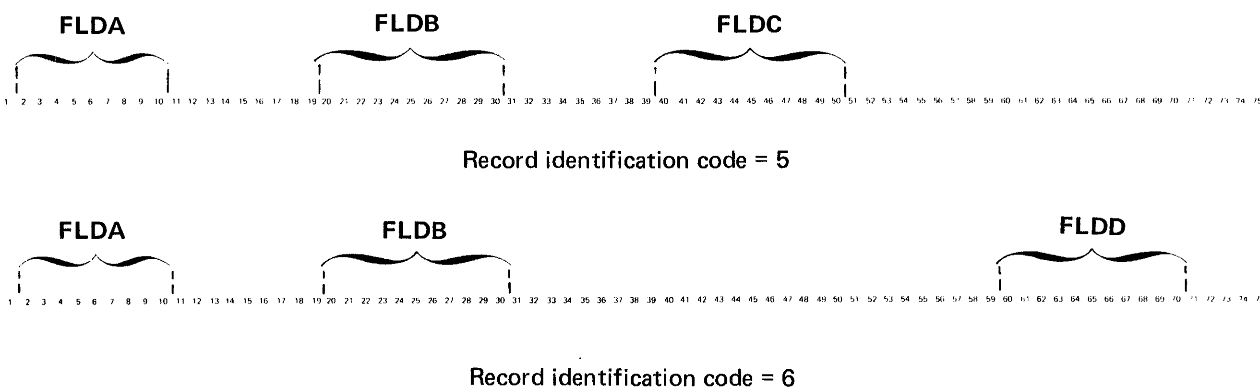
External indicators are primarily used when file conditioning is done by an entry in columns 71-72 of the file description specifications sheet. However, they can also be used to condition when a specification should or should not be done even though file conditioning is not specified (see *Indicators, External Indicators* under Part 2, *RPG II Programmer's Guide*).

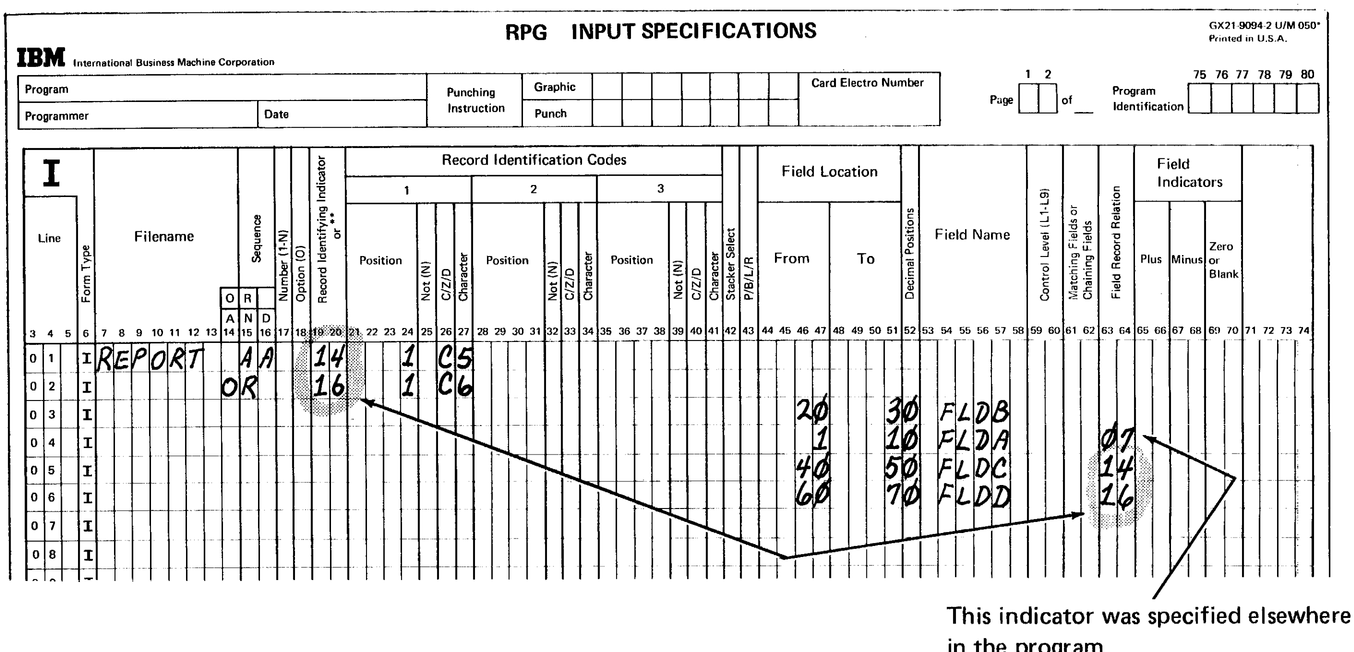
Halt Indicators (H1-H9)

A halt indicator is used to relate a field to a record that is in an OR relationship and also has a halt indicator specified in columns 19-20.

Examples

Example 1: Figure 63 shows how record identifying indicators are used to relate a field to a record. The file contains two different types of records, one identified by a 5 in position 1 and the other by a 6 in position 1. FLDC is related by record identifying indicator 14 to the record type which is identified by a 5 in position 1. FLDD is related to the record type having a 6 in position 1 by record identifying indicator 16. This means that FLDC is found on only one type of record (that identified by 5 in position 1) and FLDD is found only on the other type. FLDA is conditioned by indicator 07, which was previously defined elsewhere in the program. FLDB is found on both types since they are not related to any one type by a record identifying indicator.

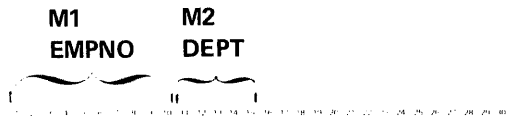




● **Figure 63. Field Record Relation**

Example 2: Figure 64 shows how dummy match fields are used in field record relation. Three different record types are found in the input file (see insert A). All three contain a match field in positions 1-10. Two of them have a second match field. Insert B shows the correct field record relation specifications. Since M1 is found on all record types, it can be specified without an entry in columns 63-64. If one match value (M1-M9) is specified without field record relation entries, all match values must be specified once without field record relation entries. Since the value M1 is specified without field record relationship, an M2 value must also be specified once without field record relationship. The M2 field is not on all record types, thus a dummy M2 field must be specified next. The dummy field can be given any unique name but its specified length must be equal to the length of the true M2 field. The M2 field is then related to the record types on which it is found by field record relation entries (lines 06 and 07).

Record identifying indicator 01



Record identifying indicator 02

Record identifying indicator 03

Ⓐ

RPG INPUT SPECIFICATIONS

IBM International Business Machine Corporation

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

Program | Punching Instruction | Graphic Punch | Card Electro Number | Page 1 2 of __ | Program Identification 75 76 77 78 79 80

Programmer | Date

I

Line	Form Type	Filename	Sequence (O A / R N / D)	Number (1-N)	Option (O)	Record Identifying Indicator or *	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	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
01	I	DISK	AB			01	1		C	1																				
02	I		OR			02	1		C	2																				
03	I		OR			03	1		C	3																				
04	I																				1	10		EMPNO		M1				
05	I																				11	15		DUMMY		M2				
06	I																				11	15		DEPT		M202				
07	I																				21	25		DEPT		M203				
08	I																													
09	I																													

Ⓑ

Figure 64. Dummy Match Fields

Example 3: Split control fields on one record type must have the same record relation entry. Figure 65, insert A, shows several record types with split control fields in each. The record identified by a 1 in position 95 has two split control fields:

FLD1A and FLD1B
FLD2A and FLD2B

The record with a 2 in position 95 has three split control fields:

FLD1A and FLD1B
FLD2A and FLD2B
FLD3A, FLD3B, and FLD3C

The third record type, identified by the 3 in position 95, also has three split control fields:

FLD1A and FLD1B
FLD2A and FLD2B
FLD3D and FLD3E

All portions of the split control field must be assigned the same control level indicator and all must have the same field record relation entry. Figure 65, insert B, shows the field record relation entries required for the three record types.

COLUMNS 65-70 (FIELD INDICATORS)

Entry	Explanation
01-99	Numeric indicator.
H1-H9	Halt indicator (when checking for an error condition in the data).

Columns 65-70 are used to check the condition of numeric fields. The three conditions are:

- Plus (columns 65-66). Any valid indicator entered here is turned on if the numeric field named in columns 53-58 is greater than zero.
- Minus (columns 67-68). Any valid indicator entered here is turned on if the numeric field in columns 53-58 is less than zero.
- Zero or blank (columns 69-70). Any valid indicator entered here is turned on if a numeric field named in columns 53-58 is all zeros or if an alphameric field is all blanks. A numeric field which is all blanks will turn on an indicator specified for zeros. However, if an alphameric field is all zeros, the field will not turn on an indicator specified for all blanks.

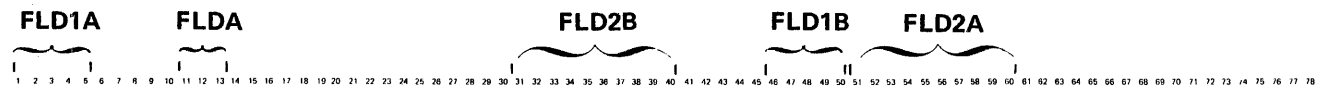
In the input specifications, you specify the indicators that will be used to condition operations. In the calculation specifications and output specifications, you actually use these indicators. When conditioning operations, you must know when the indicators will be off and when they will be on.

Assigning Indicators in Columns 65-70

The following considerations apply to numeric indicators and halt indicators:

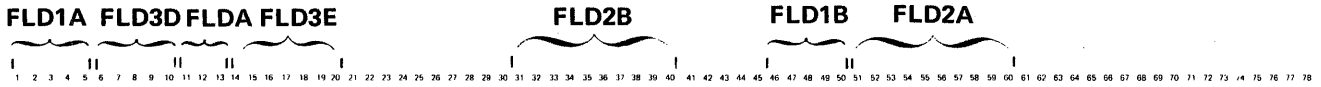
- Indicators for plus or minus are off at the beginning of the program. They are not turned on until the condition (plus or minus) is satisfied by the field being tested on the record just read.
- Columns 65-70 must be blank when using table or array names in input specifications. However, an entry can be made for an array element.
- An indicator assigned to zero or blank is off at the beginning of the program. It remains off until the field being tested is zero or blank.
- One input field can be assigned two or three field indicators. However, only the one which signals the result of the test turns on; the others are turned off.
- If the same field indicator is assigned to fields in different record types, its status is always based on the last record type selected.
- When different field indicators are assigned to fields in different record types, a field indicator turned on remains on until another record of that type is read. Similarly, a field indicator assigned to more than one field within a single record type always reflects the status of the last field defined.

Field indicators assigned in these columns can be SETON or SETOF by calculation specifications.



Record identification code = 1

Record identification code = 2



Record identification code = 3

Ⓐ

RPG INPUT SPECIFICATIONS

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

IBM International Business Machine Corporation

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

I

Line	Form Type	Filename	Sequence (OR / AN / D)	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	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
01	I	DISK	BC			91	95		C1												1	5		FLD1A	L1					
02	I		OR			92	95		C2												46	50		FLD1B	L1					
03	I		OR			93	95		C3												11	13		FLDA	L2					
04	I																				51	60		FLD2A	L3					
05	I																				31	40		FLD2B	L3					
06	I																				71	75		FLD3A	L4		92			
07	I																				26	27		FLD3B	L4		92			
08	I																				41	45		FLD3C	L4		92			
09	I																				61	70		FLDB			92			
10	I																				21	25		FLDC			92			
11	I																				6	10		FLD3D	L4		93			
12	I																				14	20		FLD3E	L4		93			
13	I																													
14	I																													
15	I																													
16	I																													
17	I																													
18	I																													
19	I																													
20	I																													

All portions of a split control field must have the same field record relation entry.

Ⓑ

Figure 65. Field Record Relation (Split Control Fields)

Numeric Indicators (01-99)

Use numeric indicators 01-99 when you want to test a field for a condition of either plus, minus, zero, or blank. The indicator specified turns on if the condition is true; it remains off or turns off if the condition is not true. You usually use these same indicators to control certain calculation or output operations. See *Columns 9-17 (Indicators)* under *Calculation Specifications* or *Columns 23-31 (Output Indicators)* under *Output Specifications*.

Halt Indicators (H1-H9)

Specify any halt indicator (H1-H9) in columns 65-70 when you want to check for an error condition in your data. For example, if a field should not be zero, you can specify a halt indicator to check for that zero condition. If a zero field is found, the halt indicator turns on and the job stops after the record with the zero field has been processed.

Indicators H1-H9 cause the program to halt after the record which caused the indicator to turn on is completely processed (all calculations for that record are complete). The operator can restart the system by responding to the system halt.

COLUMNS 71-74

Columns 71-74 are not used. Leave them blank.

COLUMNS 75-80 (PROGRAM IDENTIFICATION)

See *Common Entries*.

Calculation specifications describe the calculations you want performed on your data and the order in which you want them performed. Each calculation specification can be divided into three parts:

- When the operation is to be performed (columns 7-17). The indicators entered in these columns determine under what conditions the operation specified is to be done.
- What kind of operation is to be performed (columns 18-53). Entries in these fields describe the kind of operation to be done. They also specify the data upon which the operation is to be performed.
- What tests are to be made on the results of the operation (columns 54-59). The indicators entered here signal the result of the operation and can condition other operations.

Write these specifications on the calculation specifications sheet (Figure 66).

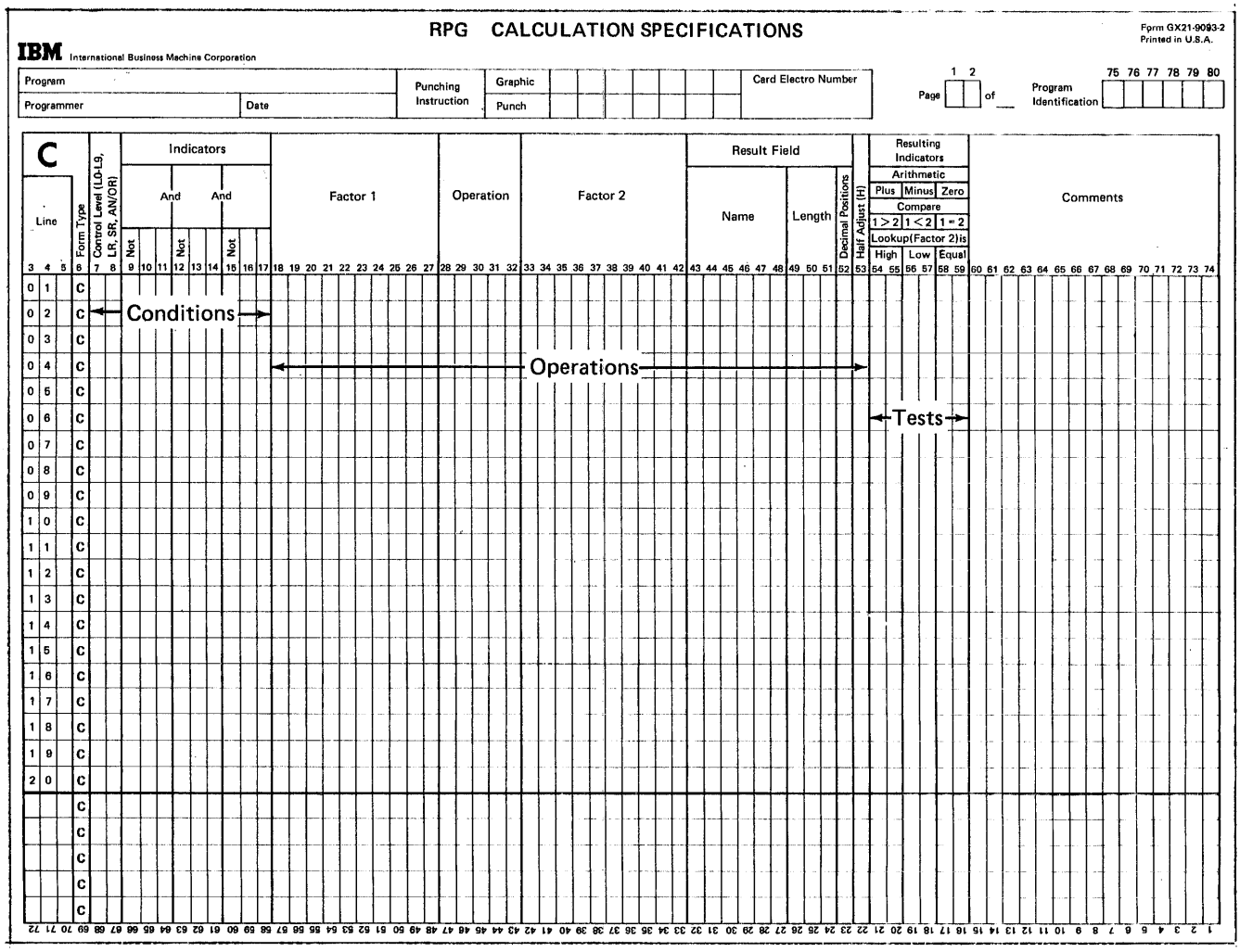


Figure 66. Calculation Specifications Sheet

COLUMNS 1-2 (PAGE)

See *Common Entries*.

COLUMNS 3-5 (LINES)

See *Common Entries*.

COLUMN 6 (FORM TYPE)

A C-entry must appear in column 6.

COLUMNS 7-8 (CONTROL LEVEL)

Entry	Explanation
Blank	Calculation operation to be done at detail calculation time for each program cycle.
L0	Calculation operation to be done at total calculation time for each program cycle.
L1-L9	Calculation operation is done when the appropriate control break occurs at total calculation time.
LR	Calculation operation is done after the last record has been processed.
SR	Calculation operation is part of a subroutine.
AN, OR	Establishes AND and OR relationships between lines of indicators.

Columns 7-8 can be used for four different purposes:

- To perform total calculation operations when the appropriate control break occurs.
- To perform calculation operations which are done only after the last record has been read.
- To indicate that an operation is part of a subroutine.
- To specify that certain lines of indicators are in an AND/OR relationship.

If you leave columns 7-8 blank, the operation specified on the same line is done every time a record is read, provided indicators in columns 9-17 allow it. See *Columns 9-17 (Indicators)*.

Control Level Indicators (L0, L1-L9)

The L0 indicator is on during the entire program. You need never assign this indicator, but you may use it. The indicator is often used when no control fields have been assigned. Remember, when a control break occurs, all operations conditioned by control level indicators are done before those that are not conditioned. If you have no control field but want total calculations to be done and total output records to be written, you can use the L0 indicator to condition those operations (see *Examples, Example 1*).

Use control level indicators L1-L9 to signal when certain operations are to occur. If you specify a control level indicator (L1-L9) in columns 7-8, the operation described on the same specifications line is done only when that indicator is on. Remember that a control level indicator turns on when information in a control field changes. See *Columns 59-60 (Control Level)* under *Input Specifications.*

A control break for a certain level causes all lower control level indicators to turn on. Thus, if you used indicators L3, L2, and L1 in your program and L3 turns on, L1 and L2 also turn on. All operations conditioned by L3, L2, and L1 are done.

There is an exception, however. When a control level indicator used as a record identifying indicator turns on to reflect the type of record read or when a control level indicator turns on by the SETON instruction, only that one control level indicator turns on. All lower level indicators remain unchanged.

Note: In one program cycle, all operations conditioned by control level indicators in columns 7-8 are done at total calculation time. Operations that are conditioned by control level indicators in columns 9-17 are done at detail calculation time immediately following the control break (see *Relationship Between Columns 7-8* and *Columns 9-17*).

Last Record Indicator (LR)

The last record (LR) indicator automatically turns on after the last record is read and processed. You may have certain operations which are to be done only after the last record is read. Condition these operations with the LR indicator. Place operations conditioned by LR after all calculations conditioned by L0-L9 (columns 7-8) or after detail calculations if there are no total calculations. The last record causes the LR indicator and all other control level indicators specified (L1-L9) to turn on.

Subroutine Lines (SR)

An SR entry in columns 7-8 indicates that a line is part of a subroutine (see *Subroutines* under Part 2, *RPG II Programmer's Guide*). Subroutine lines must be specified last.

AND/OR Lines (AN, OR)

Use columns 7-8 to specify that lines of indicators are in an AND/OR relationship. By using the AND/OR relationship, many lines of indicators can be grouped together to condition an operation. A maximum of seven OR lines or seven AND lines or any combination thereof can be used to condition an operation.

The first line of such a group contains blanks in columns 7-8 or an L0-L9, LR, or SR entry if the group of lines is conditioned by a control level indicator or is part of a subroutine. All lines after the first line in the group must have an AN or OR entry in columns 7-8. The last line of the group contains the operation and the necessary operands. All lines in the group prior to the last line must contain blanks in columns 18-59 (see *Examples, Example 2* and *Example 3*).

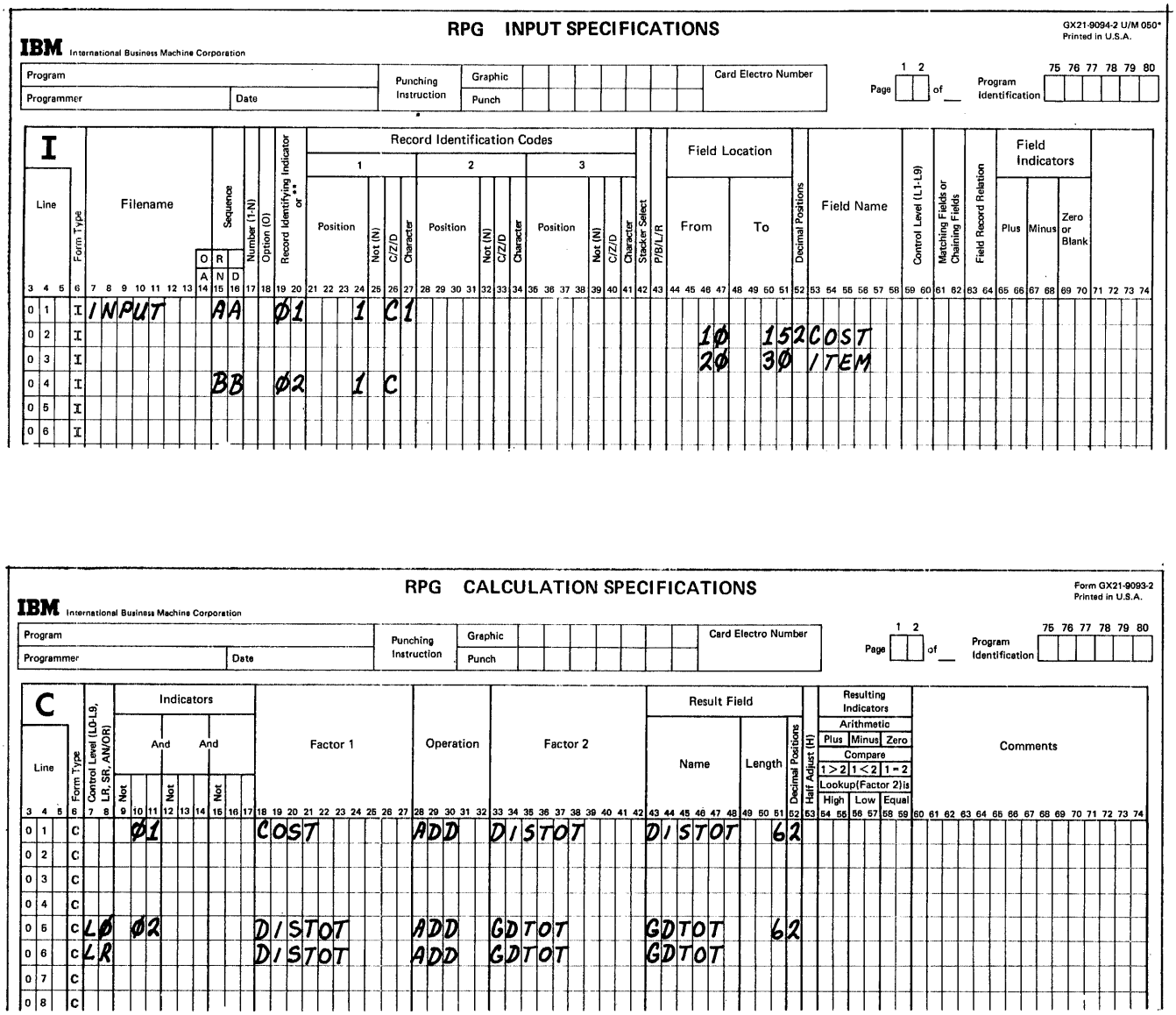
Examples

Example 1: Figure 67 shows the format of the report printed by the job described in Figure 68. The job shows how total operations can be performed even though there is no control field (no L1-L9 indicators). The job requires:

- A list of items sold in each district
- A total of all sales for each district
- A grand total of all sales in all districts

```
J102      4.50
J202      3.75
K450      2.98
B231      9.08

          20.31  *

G10H     92.79
G10K     98.89
A126      4.29

         195.97  *


         216.28  **
```

Figure 67. Format of a Printed Report



RPG INPUT SPECIFICATIONS

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

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 1 2 of __ Program Identification 75 76 77 78 79 80

I

Line	Form Type	Filename	Sequence (OR / AND)	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	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
0 1	I	INPUT	AA			Ø1	1		C	1																				
0 2	I																				10	15	2	COST						
0 3	I																				20	30		ITEM						
0 4	I		BB			Ø2	1		C																					
0 5	I																													
0 6	I																													

RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 1 2 of __ Program Identification 75 76 77 78 79 80

C

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators And Not	And Not	And Not	Factor 1	Operation	Factor 2	Result Field Name	Result Field Length	Result Field Decimal Positions	Half Adjust (H)	Resulting Indicators Arithmetic Plus	Arithmetic Minus	Arithmetic Zero	Compare 1>2	Compare 1<2	Compare 1=2	Lookup(Factor 2)is High	Lookup(Factor 2)is Low	Lookup(Factor 2)is Equal	Comments
0 1	C		Ø1			COST	ADD	DISTOT	DISTOT	6	2											
0 2	C																					
0 3	C																					
0 4	C																					
0 5	C	L0	Ø2			DISTOT	ADD	GDTOT	GDTOT	6	2											
0 6	C	LR				DISTOT	ADD	GDTOT	GDTOT													
0 7	C																					
0 8	C																					

Figure 68 (Part 1 of 2). Use of the L0 Indicator

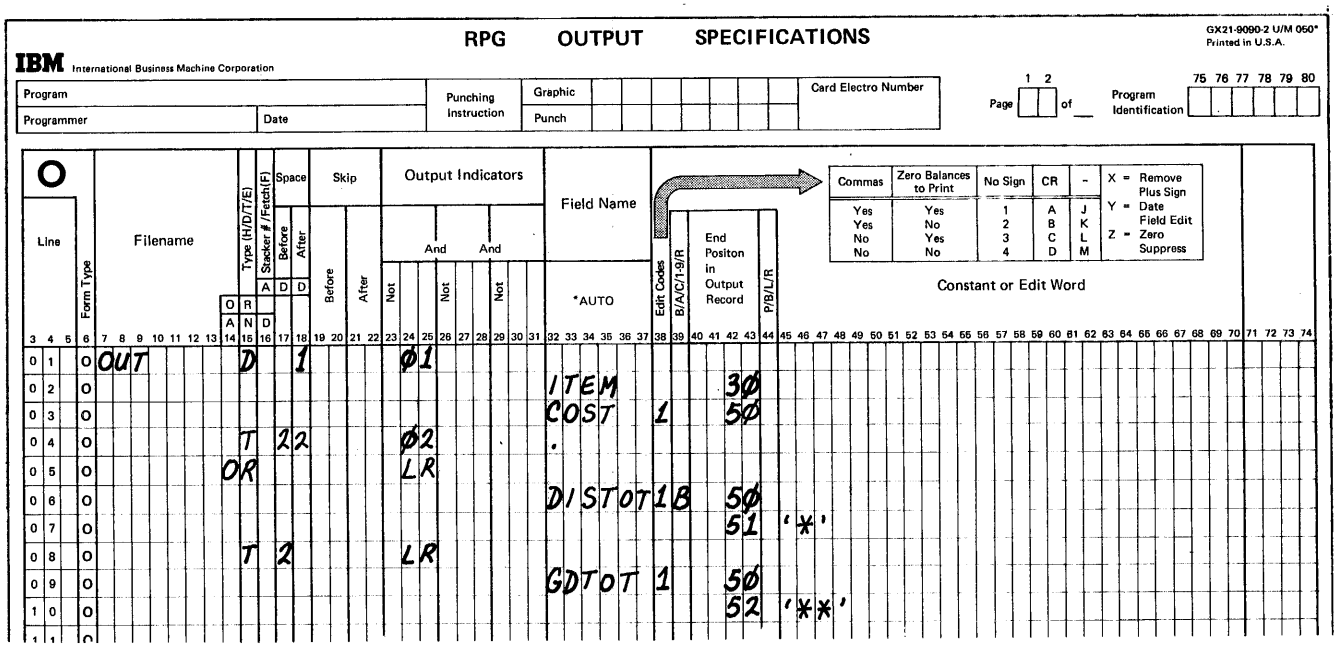


Figure 68 (Part 2 of 2). Use of the L0 Indicator

The input records have ITEM and COST fields and a 1-position record identification field. The records are grouped in ascending sequence by district; that is, the district 1 records as a group are followed by a blank record, and the district 2 records as a group are followed by a blank record (Figure 69).

There is no field that can serve as a control field, since the district number is not on the records. Instead of a control field, the blank record is used to signal a new district. When the blank record is read, indicator 02 turns on. The blank record tells the program that total calculations and total output operations must be done. However, no total operations can be performed unless they are conditioned by some kind of control level indicator.

Even though L0 is on all the time, it must be used in columns 7-8 because some type of control level indicator must be assigned to all total operations.

Example 2: Figure 70, insert A, shows the use of AN and OR entries to group lines of indicators. When indicators 01, 02, 03, and 04 are on, or when indicators 01, 02, 03, and 05 are on, the calculation is performed.

Example 3: Figure 70, insert B, illustrates a case in which three conditions cause the L4 total calculations to be performed: 01 and 02 are on, but not 03; or 01 and 03 are on, but not 02; or 02 and 03 are on, but not 01.

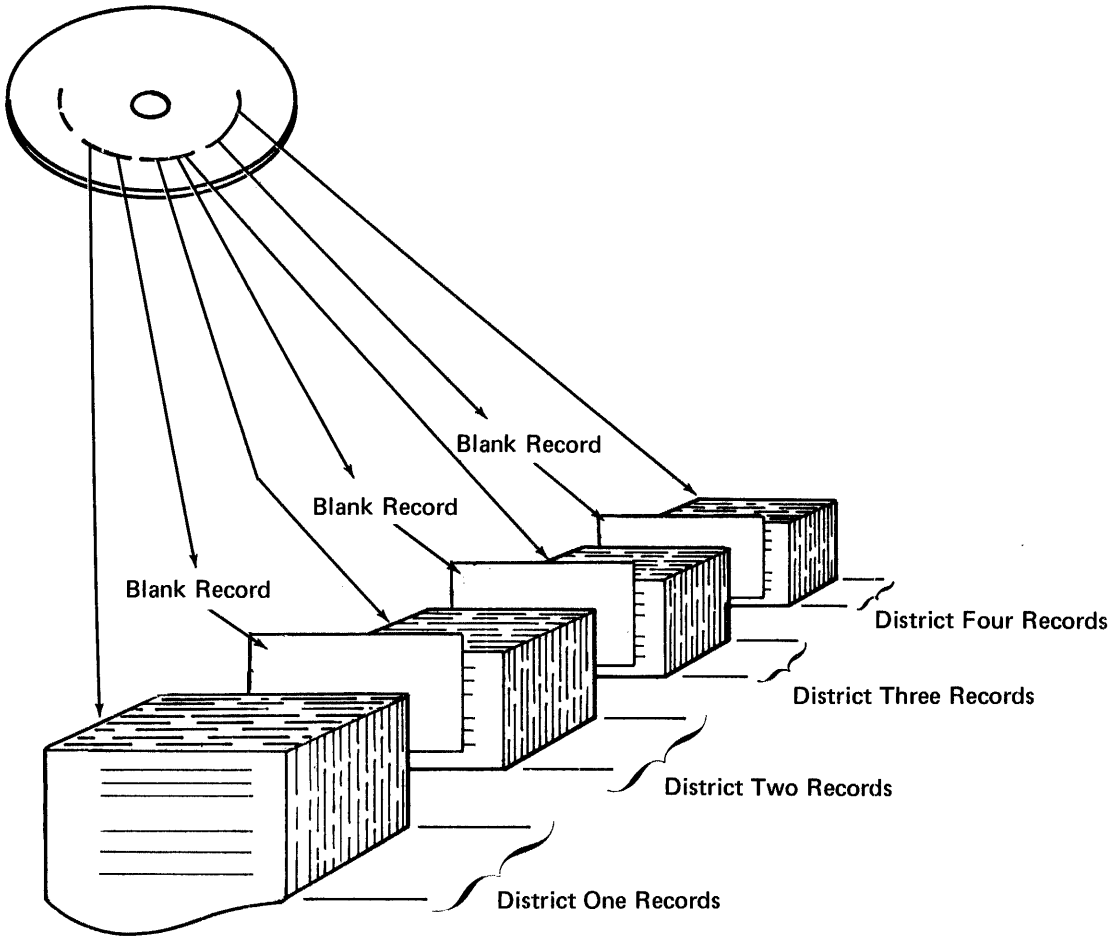


Figure 69. Blank Records Used to Signal Control Breaks

RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 1 2 of __ Program Identification 75 76 77 78 79 80

Line 3 4 5	Form Type 6	Control Level (L0-L9, LR, SR, AN/OR) 7 8	Indicators: And/Not 9 10 11 12 13 14 15 16 17	Factor 1 18-27	Operation 28-32	Factor 2 33-42	Result Field: Name 43-48	Length 49-51	Decimal Positions 52	Half Adjust (H) 53	Resulting Indicators: Arithmetic Plus/Minus/Zero; Compare 1>2/1<2/1=2; Lookup(Factor 2)is High/Low/Equal 54 55 56 57 58 59	Comments 60-74
01	C		Ø1 Ø2 Ø3									
02	C	AN	Ø4									
03	C	OR	Ø1 Ø2 Ø3									
04	C	AN	Ø5	FIELDA	SUB	FIELDB	QTY	4			23	
05	C											
06	C											
07	C											

Ⓐ

RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 1 2 of __ Program Identification 75 76 77 78 79 80

Line 3 4 5	Form Type 6	Control Level (L0-L9, LR, SR, AN/OR) 7 8	Indicators: And/Not 9 10 11 12 13 14 15 16 17	Factor 1 18-27	Operation 28-32	Factor 2 33-42	Result Field: Name 43-48	Length 49-51	Decimal Positions 52	Half Adjust (H) 53	Resulting Indicators: Arithmetic Plus/Minus/Zero; Compare 1>2/1<2/1=2; Lookup(Factor 2)is High/Low/Equal 54 55 56 57 58 59	Comments 60-74
01	C	L4	Ø1 Ø2 N Ø3									
02	C	OR	Ø1 N Ø2 Ø3									
03	C	OR N Ø1	Ø2 Ø3	SUM	ADD	SUMTOT	SUMTOT	82	H			
04	C											
05	C											

Ⓑ

Figure 70. Use of AND/OR Lines for Indicators

COLUMNS 9-17 (INDICATORS)

Entry	Explanation
Blank	Operation is performed on every program cycle.
01-99	Field indicators, record identifying indicators, or resulting indicators used elsewhere in the program.
KA-KN, KP, KQ	Command key indicators assigned elsewhere.
L1-L9	Control level indicators assigned elsewhere.
LR	Last record indicator.
MR	Matching record indicator.
H1-H9	Halt indicators assigned elsewhere.
U1-U8	External indicators previously set.
OA-OG, OV	Overflow indicator previously assigned.

Use columns 9-17 to assign indicators that control the conditions under which an operation is done. You can use from one to three separate fields (10-11, 13-14, and 16-17) on each line, one for each indicator. If the indicator must not be on to condition the operation, place an N before the appropriate indicator (columns 9, 12, 15).

AND/OR Lines (AN, OR)

By using AN or OR entries in columns 7-8, many indicators can be used to condition one operation. A maximum of seven OR lines can be used to condition an operation if no AND lines are used. A maximum of seven AND lines can be used to condition an operation if no OR lines are used. If AND or OR lines are combined, the total number of AND and OR lines used to condition an operation cannot exceed seven.

AND Relationship

All three indicators on one line are in an AND relationship with each other. The indicators on one line or indicators in grouped lines pluse the control level indicator (if used in columns 7-8) must all be exactly as specified before the operation is done (see *Examples, Example 1*).

Field Indicators (01-99)

Use any field indicators specified in columns 65-70 on the input specifications sheet to condition an operation that is to be done only after the status of a field has been checked and has met certain conditions (see *Examples, Example 2*).

Command Keys Indicators (KA-KN, KP, KQ)

Use any command key indicators specified in columns 54-59 of the calculation specifications sheet for a SET or SETOF operation. See *Operation Codes, SET or SETOF* under Part 2, *RPG II Programmer's Guide* for complete information on each operation.

Record Identifying Indicators (01-99)

Use any record identifying indicators specified in columns 19-20 on the input specifications sheet to condition an operation that is to be done only for a certain type of record (see *Examples, Example 3*).

Resulting Indicators (01-99)

Use any resulting indicators specified in columns 54-59 on the calculation specifications sheet to condition operations according to the results of calculation operations. See the example under *Columns 54-59 (Resulting Indicators).*

Control Level Indicators (L1-L9)

Use any control level indicators specified in columns 59-60 on the input specifications sheet or in columns 54-59 on the calculation specifications sheet. If control level indicators are used in these columns but not in columns 7-8, the operation is performed (at detail calculation time) on the first record of a new control group or whenever the indicator is on.

Last Record Indicator (LR)

To condition operations to be performed at end of job, use the last record (LR) indicator in columns 9-17 only if LR is turned on during calculations. All operations to be performed at end of job should be conditioned by LR in columns 7-8.

Matching Record Indicator (MR)

Use the matching record (MR) indicator to condition an operation that is to be done only when matching records are found. See *Columns 61-62 (Matching Fields)* under *Input Specifications* for more information on matching fields.

Halt Indicators (H1-H9)

Use any halt indicators in columns 65-70 on the input specifications sheet or in columns 54-59 on the calculation specifications sheet to prevent the operation from being done when a specified error condition has been found in the input data or on calculations. See *Columns 19-20 (Record Identifying Indicator)* under *Input Specifications.* This is necessary because the record that causes the halt condition is completely processed before your program stops. Thus, if the operation is performed even on an error condition, the results are in error. It is also possible to use a halt indicator to condition an operation that is to be done only when an error occurs.

Note: The system message displayed on a halt can be overridden through the use of the user message member. (See *User Message Member* in Chapter 4, *Operation Codes* of Part 2, *RPG II Programmer's Guide*.)

External Indicators (U1-U8)

Use any external indicator previously specified to condition which operations should be done and which files should be used for a specific job. See *Indicators, U1-U8* under Part 2, *RPG II Programmer's Guide* for more information.

Overflow Indicators (OA-OG, OV)

Use any overflow indicators specified in columns 33-34 on the file description specifications sheet to condition operations that are to be done when the last line to be printed on a page is reached. See *Overflow Indicators* under Part 2, *RPG II Programmer's Guide* for more information.

Relationship Between Columns 7-8 and Columns 9-17

In one program cycle, all operations conditioned by control level indicators in columns 7-8 (total time) are done before operations that are conditioned by control level indicators in columns 9-17 (see *Examples, Example 4*).

When a control level indicator is used in columns 9-17 and columns 7-8 are not used (detail time), the operation conditioned by the indicator is done only on the record that causes control break or any higher level control break.

When a control level indicator (L1-L9) is specified in columns 7-8 (total time) and MR is specified in columns 9-17, MR indicates the matching condition of the previous record and not the one just read that caused the control break. After all operations conditioned by control level indicators (specified in columns 7-8 of the calculation specifications sheet) are done, MR then indicates the matching condition of the record just read.

Examples

Example 1: Figure 71 shows the use of control level indicators to condition calculation operations. The operation in line 02 can be done when the L2 indicator is on provided the other conditions are met. Indicator 10 must be on. The L3 indicator must not be on.

The operation conditioned by both L2 and NL3 is done only when a control level 2 break occurs. These two indicators are used together because this operation is not to be done when a control level 3 break occurs, even though L2 is also on.

Example 2: Figure 72 shows the use of field indicators to condition operations. Assume the job is to find weekly earnings including overtime. The overtime field is checked to see if any overtime was put in. If the employee has worked overtime, the field is positive and indicator 10 turns on. In all cases the weekly regular wage is calculated. However, overtime pay is calculated only if indicator 10 is on (lines 03 and 04).

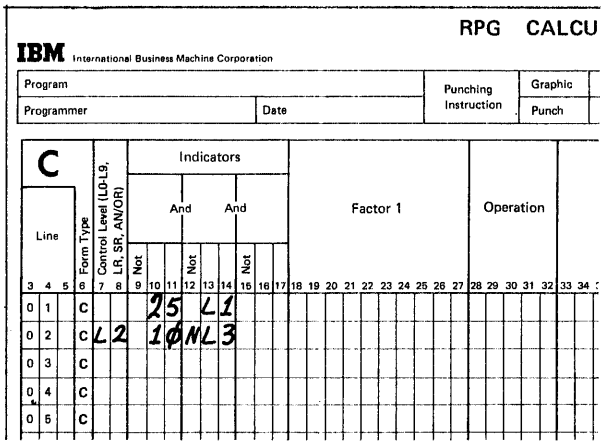
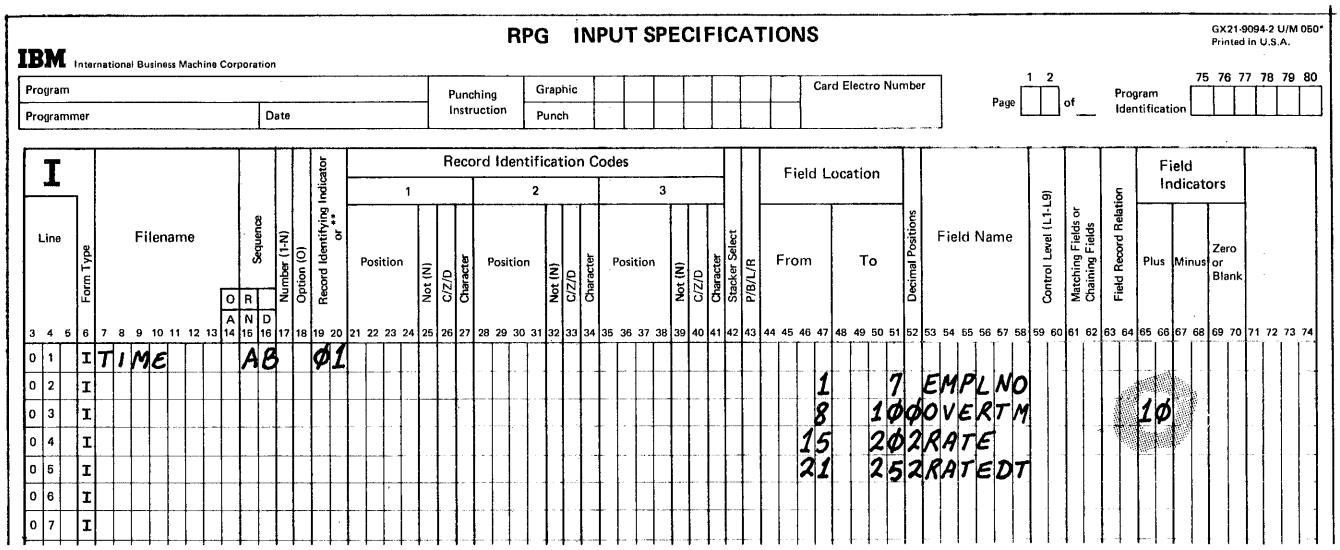


Figure 71. Conditioning Operations (Control Level Indicators)



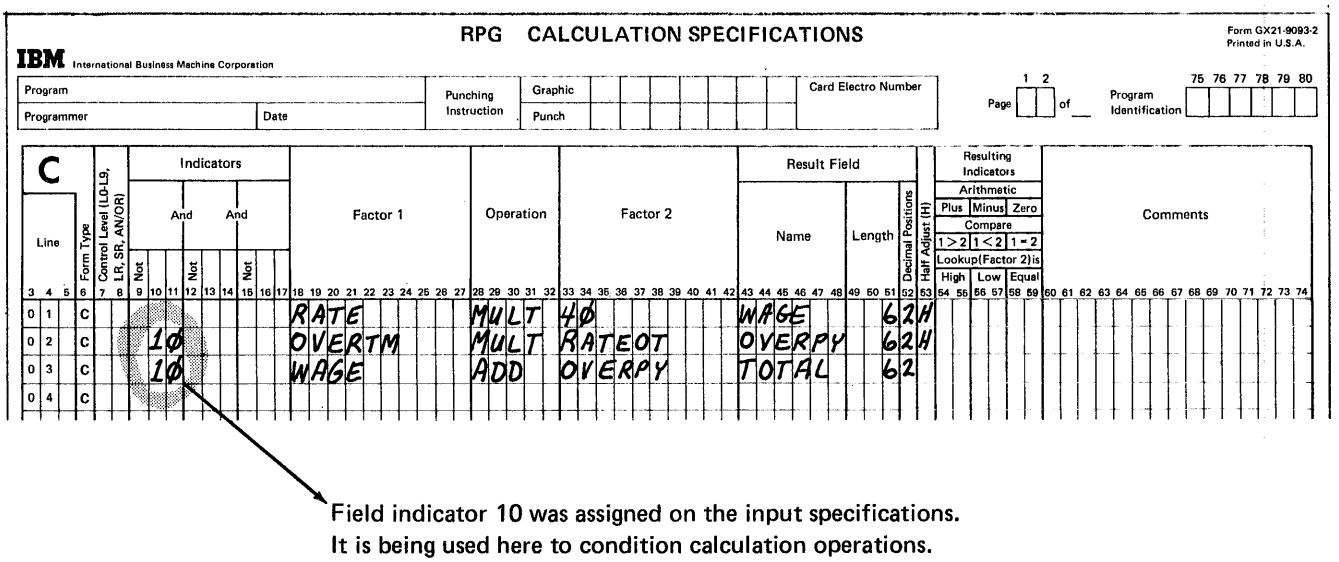


Figure 72. Conditioning Operations (Field Indicators)

Example 3: Figure 73 shows how a record identifying indicator is used to condition an operation. When a record is read that has a T in position 1, the 01 indicator turns on. If this indicator is on, the field named SAVE is added to SUM. When a record having no T in position 1 is read, the 02 indicator is on. The subtract operation, since it is conditioned by 02, is then done instead of the add operation.

Example 4: Line 02 of Figure 71 shows the use of a control level indicator in columns 9-17. Assume that indicator 25 represents a record type and that a control level 2 break occurred when record type 25 was read. L1 and L2 are both on. All operations conditioned by the control level indicators in columns 7-8 are performed before operations conditioned by control level indicators in columns 9-17. Thus, the operation in line 02 occurs before the operation in line 01. The operation in line 01 is done on the first record of the new control group indicated by 25, whereas the operation in line 02 is a total operation done for all records of the previous control group.

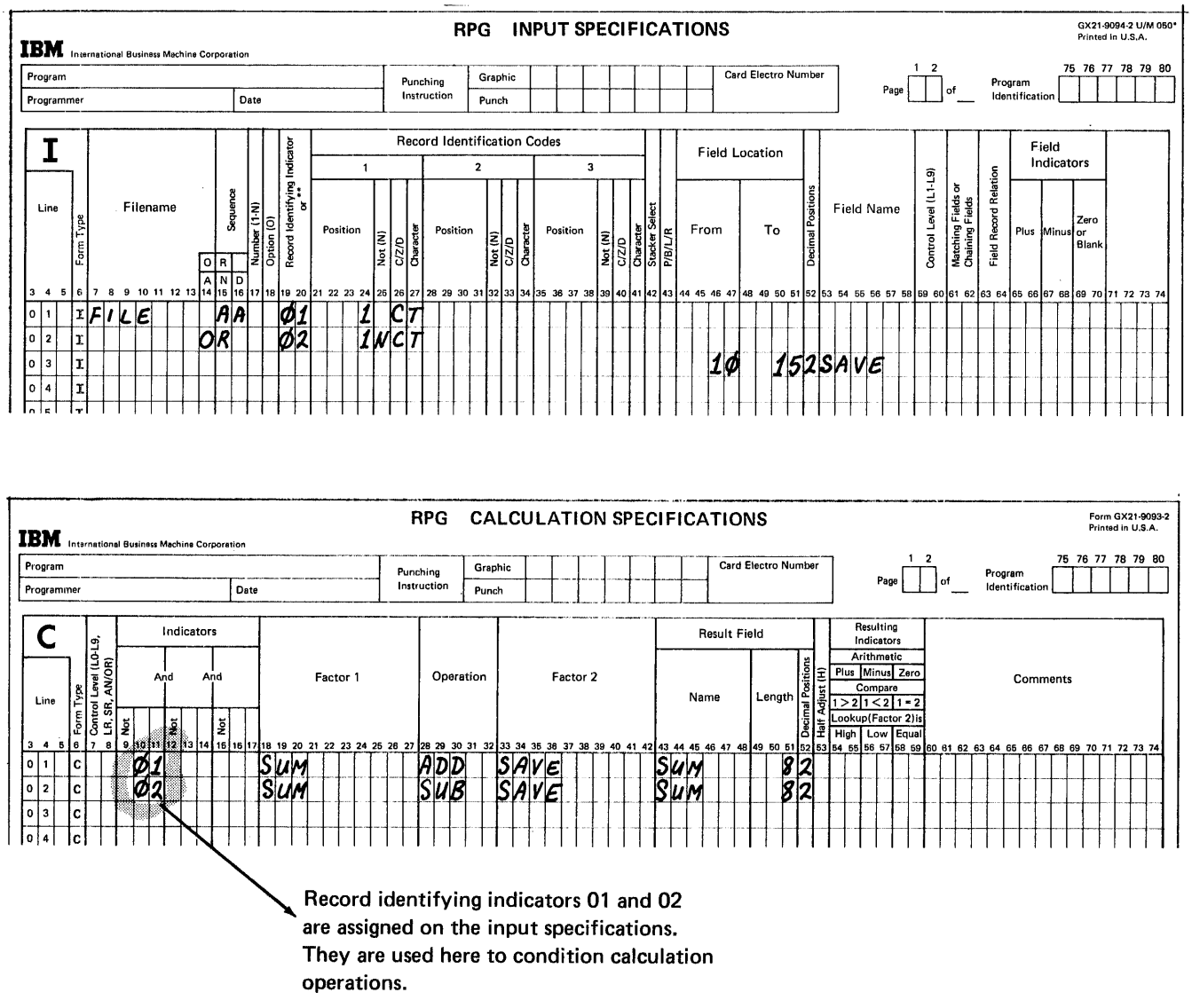


Figure 73. Conditioning Operations (Resulting Indicators)

COLUMNS 18-27 (FACTOR 1) AND COLUMNS 33-42 (FACTOR 2)

Use columns 18-27 and 33-42 to name the fields or to give the actual data (literals) on which an operation is to be performed. See Figure 75 for a summary of the operation codes.

The entries you can use for factor 1 and factor 2 are:

- The name of any field that has been defined.
- Any alphameric or numeric literal.
- Any subroutine, table, array name, or array element.
- Any date field names (UDATE, UMONTH, UDAY, UYEAR)
- The special names PAGE, PAGE1, or PAGE2.

The following entry can be made for factor 1 only:

A label for a TAG, BEGSR, or ENDSR operation.

The following entries can be made for factor 2 only:

- A label for a GOTO or EXSR operation.
- A filename for a SET, CHAIN, DEBUG, READ, or FORCE operation.
- A subroutine name for an EXIT operation.

An entry in factor 1 must begin in column 18; an entry in factor 2 must begin in column 33.

The entries you use depend upon the operation you are describing. Some operations need entries in both sets of columns, some need entries in only one, and some need no entries at all. See *Columns 28-32 (Operation)* for more information on operation codes. If you are naming a subroutine, see *Subroutines* under Part 2, *RPG II Programmer's Guide*.

Literals

A literal is the actual data used in an operation rather than the field name representing that data. A literal can be either alphameric or numeric.

Consider the following rules when using an alphameric literal (Figure 74, insert A):

- Any combination of characters can be used in an alphameric literal. Blanks are also valid.
- The maximum length of an alphameric literal is eight characters.
- Alphameric literals must be enclosed with apostrophes (').
- An apostrophe required as part of a literal is represented by two apostrophes. For example, the literal O'CLOCK is coded as 'O''CLOCK'.
- Alphameric literals cannot be used for arithmetic operations.

Consider the following rules when using a numeric literal (Figure 74, insert B):

- A numeric literal consists of any combination of the digits 0-9. A decimal point or sign can also be included.
- The sign (+ or -), if present, must be the leftmost character. An unsigned literal is treated as a positive number.
- The maximum total length of a numeric literal is 10 characters including signs and decimal points.
- Blanks cannot appear in a numeric literal.
- Numeric literals must not be enclosed with apostrophes (').
- Numeric literals are used in the same way as a numeric field.

RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

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

Programmer | Date

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

C

Line 3 4 5	Form Type 6	Control Level (L0-L9, LR, SR, AN/OR) 7 8	Indicators: And / And — Not 9 10 11 12 13 14 15 16 17	Factor 1 18–27	Operation 28–32	Factor 2 33–42	Result Field: Name 43–48	Length 49–51	Decimal Positions 52	Half Adjust (H) 53	Resulting Indicators: Arithmetic: Plus / Minus / Zero; Compare: 1>2 / 1<2 / 1=2; Lookup(Factor 2)is: High / Low / Equal 54 55 56 57 58 59	Comments 60–74
0 1	C			'512% DT.'								
0 2	C					'FEBRUARY'						
0 3	C			'O''CLOCK'								
0 4	C			''68'								
0 5	C											

Ⓐ

RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

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

Programmer | Date

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

C

Line 3 4 5 | Form Type 6 | Control Level (L0-L9, LR, SR, AN/OR) 7 8 | Indicators: And / And — Not 9 10 11 12 13 14 15 16 17 | Factor 1 18–27 | Operation 28–32 | Factor 2 33–42 | Result Field: Name 43–48 | Length 49–51 | Decimal Positions 52 | Half Adjust (H) 53 | Resulting Indicators: Arithmetic: Plus / Minus / Zero; Compare: 1>2 / 1<2 / 1=2; Lookup(Factor 2)is: High / Low / Equal 54 55 56 57 58 59 | Comments 60–74

0 1	C			125ØØ								
0 2	C			125ØØ.ØØ								
0 3	C					.ØØ1256789						
0 4	C					-.Ø1256789						
0 5	C											

Ⓑ

Figure 74. Alphameric and Numeric Literals

COLUMNS 28-32 (OPERATION)

Use columns 28-32 to specify the kind of operation to be performed using factor 1, factor 2, and/or the result field. The operation code must begin in column 28. A special set of operation codes are defined which you must use to indicate the type of operation desired. Every operation code used requires certain entries on the same specification line. See Figure 75 for a summary of all possible codes and the additional entries required for each code. For further information on the operations that can be performed, see *Operations Code* under Part 2, *RPG II Programmer's Guide.*

The operations are performed in the order specified on the calculation specifications sheet.

Note: All operations conditioned by control level indicators in columns 7-8 must follow those that are not conditioned by control level indicators.

Type of Operation	Function of Operation	Operation Code (Columns 28-32)	Control Level	Indicators	Factor 1	Factor 2	Result Field	Field Length	Decimal Position	Half Adjust	Resulting Indicators
Arithmetic Operations	Add factor 2 to factor 1.	ADD	O	O	R	R	R	O	O	O	O
	Clear result field and add factor 2.	Z-ADD	O	O	B	R	R	O	O	O	O
	Subtract factor 2 from factor 1.	SUB	O	O	R	R	R	O	O	O	O
	Clear result field and subtract factor 2.	Z-SUB	O	O	B	R	R	O	O	O	O
	Multiply factor 1 by factor 2.	MULT	O	O	R	R	R	O	O	O	O
	Divide factor 1 by factor 2	DIV	O	O	R	R	R	O	O	O	O
	Move remainder of preceding division to a result field.	MVR	O	O	B	B	R	O	O	O	O
	Sum elements of an array and put sum in result field.	XFOOT	O	O	B	R	R	O	O	O	O
	Derive the square root of factor 2.	SQRT	O	O	B	R	R	O	O	O	B
Move Operations	Move factor 2 into result field, right-justified.	MOVE	O	O	B	R	R	O	O	B	B
	Move factor 2 into result field, left-justified.	MOVEL	O	O	B	R	R	O	O	B	B
	Move factor 2 into result field, left-justified.	MOVEA	O	O	B	R	R	O	B	B	B
Move Zone Operations	Move zone from low-order position of factor 2 to low-order position of result field.	MLLZO	O	O	B	R	R	O	O	B	B
	Move zone from high-order position of alphameric factor 2 to high-order of alphameric result field.	MHHZO	O	O	B	R	R	O	B	B	B
	Move zone from low-order position of factor 2 to high-order position of alphameric result field.	MLHZO	O	O	B	R	R	O	B	B	B
	Move zone from high-order position of alphameric factor 2 to low-order position of result field.	MHLZO	O	O	B	R	R	O	O	B	B
Compare and Zone Testing Operations	Compare factor 1 to factor 2.	COMP	O	O	R	R	B	B	B	B	R
	Identify the zone in the leftmost position of an alphameric result field.	TESTZ	O	O	B	B	R	O	B	B	R
Bit Operations	Set on specified bits.	BITON	O	O	B	R	R	O	B	B	B
	Set off specified bits.	BITOF	O	O	B	R	R	O	B	B	B
	Test specified bits.	TESTB	O	O	B	R	R	O	B	B	R
Setting Indicators	Set one, two, or three specific indicators on.	SETON	O	O	B	B	B	B	B	B	R
	Set one, two, or three specific indicators off.	SETOF	O	O	B	B	B	B	B	B	R
Branching Within RPG II	Branch to another RPG II calculation specification line.	GOTO	O	O	B	R	B	B	B	B	B
	Identify the name in factor 1 as a destination label to which GOTO may branch.	TAG	O	B	R	B	B	B	B	B	B
Branching to External Subroutines	Branch to user-written assembler subroutine.	EXIT	O	O	B	R	B	B	B	B	B
	Transfer data to user-written assembler subroutine.	RLABL	B	B	B	B	R	O	O	B	B
Look up Operations	Table look up.	LOKUP	O	O	R	R	O	O	O	B	R
	Array look up.	LOKUP	O	O	R	R	B	B	B	B	R

Figure 75 (Part 1 of 2). Operation Codes

Type of Operation	Function of Operation	Operation Code (Columns 28-32)	Control Level	Indicators	Factor 1	Factor 2	Result Field	Field Length	Decimal Position	Half Adjust	Resulting Indicators
Subroutine	Beginning of the subroutine.	BEGSR	[1]	B	R	B	B	B	B	B	B
	End of the subroutine.	ENDSR	[1]	B	O	B	B	B	B	B	B
	Call to execute the subroutine.	EXSR	O	O	B	R	B	B	B	B	B
Program Control of Input and Output	Pause for input data from keyboard.	KEYnn[2]	O	O	O	B	R	O	O	B	O
	Control of console buffer clear and command key specification.	SETnn[2]	O	O	O	O	O	B	B	B	O
	Call for immediate input.	READ	O	O	B	R	B	B	B	B	[3]
	Forcing record to be read on next cycle.	FORCE	B	O	B	R	B	B	B	B	B
	Forcing output printing.	EXCPT	O	O	B	B	B	B	B	B	B
	Sets lower limits for indexed sequential files being processed within limits.	SETLL	O	O	R	R	B	B	B	B	B
	A record is read from a disk file.	CHAIN	O	O	R	R	B	B	B	B	[4]
Debug Function	Aid in finding programming errors.	DEBUG	O	O	O	R	O	B	B	B	B

O — Optional
R — Required
B — Blank

[1] Columns 7-8 must have an SR entry for all subroutine lines.

[2] The nn entries in columns 31-32 are for message indicator (MIC) numbers. If the result field of a SET operation contains the keyword ERASE, factor 2 must contain the name of the CONSOLE file. Otherwise, factor 2 and the result field must be blank.

[3] Columns 58-59 may contain an indicator for this operation; columns 54-57 must be blank.

[4] A resulting indicator should be entered in columns 54-55; columns 56-59 must be blank.

Figure 75 (Part 2 of 2). Operation Codes

COLUMNS 31-32

Entry	Explanation
Blank or 01-99	Message identification code (MIC) of user message member to be displayed for SET or KEY operations unless overridden by a factor 1 entry.

Columns 31-32 must contain entries for all KEY operations and for SET operations in which command key indicators are specified in columns 54-59, unless an entry is made in factor 1. (Entries in these columns are ignored when factor 1 is specified on the same SET and KEY operation.)

The same combination of MICs should not be assigned to more than one KEY or SET operation except when the SET operation immediately precedes a KEY operation conditioned by the same indicators (columns 9-17) and the special SET-KEY combination is desired.

See *Operation Codes, SET and KEY* under Part 2, *RPG II Programmer's Guide* for complete information.

COLUMNS 43-48 (RESULT FIELD)

Entry	Explanation
ERASE	To erase an interactive data file buffer through use of the SET operation code.
Field name, Table name, Array name, or Array element	These entries hold the result of, or are the object of, the operation specified in columns 28-32.
INxx(xx=any RPG II indicator)	The indicator to be transferred to an external subroutine in an RLABL operation.

ERASE

Enter ERASE in columns 43-48 when you want the entire console buffer to be blanked or erased. The filename of the console file must be entered in columns 33-42. This operation indicates to the system that the console buffer should be set to blank just before getting a record at the beginning of the next RPG II cycle.

Since the console buffer is not erased until the beginning of the next RPG II cycle, processing of the current record continues after the ERASE specification is encountered. If the ERASE instruction is being executed because of invalid input data, you should insert code in your program to avoid further calculations and to return to the start of the RPG II cycle. A correct form of the record containing the invalid input data and any records that were entered after that record can then be reentered.

Field Name, Table Name, Array Name, or Array Element

Use columns 43-48 to name the field, table, array, or array element that holds the result of the operation specified in columns 28-32, or that is the field upon which an operation is performed.

You can use the name of a field, table, array, or array element that has already been defined either by the input, extension, or calculation specifications. See *Arrays* under Part 2, *RPG II Programmer's Guide*. Otherwise, you can define a new field by entering a field name that is not already used. Any field you define here is created at the time the program is compiled. The field you name can be either numeric or alphameric. A field used in arithmetic operations (see *Columns 28-32, Operation*) or numeric compare, or a field edited or zero suppressed by output specifications must be numeric.

The result field name must begin with an alphabetic character in column 43 and contain no blanks or special characters.

If you are entering the name of a field that is not defined elsewhere, columns 49-52 should also contain entries.

If you are entering the name of a field that is defined, entries in columns 49-52 are not necessary but, if specified, must agree with the previous definition of that field.

COLUMNS 49-51 (FIELD LENGTH)

Entry	Explanation
1-256	Result field length

Use columns 49-51 to give the length of a result field that is defined elsewhere. If you are naming a new field (one that was not used before), you must consider the form your data is in and the length it will have after the operation is performed.

Whenever the field length is specified for a result field, be careful to make the result field long enough to hold the largest possible result. If the result field is too small, significant digits may be lost. For example, you may want to add field A (eight characters long, four decimal places) to field B (10 characters long, six decimal positions). Fields A and B have four characters to the left of the decimal, but the result field, field C, must allow for more characters to the left of the decimal.

9999.0000	Field A
0001.111111	Field B
10000.111111	Field C (result field)

In this case, field C was defined as 11 characters long with six decimal positions. Some of the numbers to the right of the decimal could be lost without changing the meaning of the result greatly. However, if field C was defined as 10 characters long with six decimal positions, a significant digit to the left of the decimal would be lost. Field C in this case would be 0000.111111; the meaning of the result has greatly changed.

Numeric fields have a maximum length of 15 characters. Alphameric fields can be up to 256 characters long. You can indicate the length of a field that was previously described either in input specifications or in calculation specifications. However, if you do so, you must specify the same field length and number of decimal positions as was previously defined for the field.

If the result field contains the name of a table or array, an entry in these columns is optional. If used, it must agree with the length described by the extension specifications.

COLUMN 52 (DECIMAL POSITIONS)

Entry	Explanation
Blank	Alphameric or numeric field described elsewhere.
0-9	Number of decimal places in a numeric result field.

Use column 52 to indicate the number of positions to the right of the decimal in a numeric result field. If the numeric result field contains no decimal positions, enter a 0 (zero). This column must be left blank if the result field is alphameric. This column can be left blank if the result field is numeric but was described by input or calculations specifications. In this case, field length (columns 49-51) must also be left blank.

The number of decimal positions must never be greater than the length of the field. The number can, however, be larger or smaller than the number of decimal positions that actually result from an operation. If the number of decimal positions specified is greater than the number of decimal places that actually result from an operation, zeros are filled in to the right. If the number specified is smaller than the number that results from the operation, the rightmost digits are dropped.

Figure 76 shows how the contents of a result field after a multiplication operation can change according to the decimal positions (column 52) and field length (columns 49-51) specifications.

Multiplication: 98.76 x 1.234 = 121.86984

Decimal Positions for Result Field (column 52)	Result Field Length (columns 49–51)									
	10	9	8	7	6	5	4	3	2	1
9	1.869840000	.869840000								
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

Not permitted

Permitted but inaccurate

Recommended

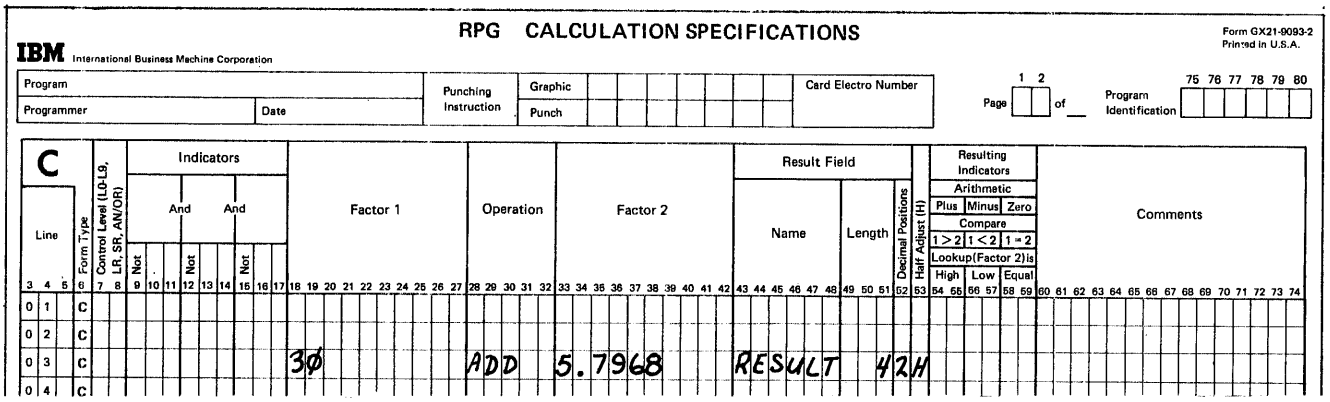
Figure 76. Result Field Contents Based on Various Field Length and Decimal Position Specification

COLUMN 53 (HALF ADJUST)

Entry	Explanation
Blank	Do not half adjust
H	Half adjust

Use column 53 to indicate that the contents of the result field are to be half adjusted (rounded). Half adjusting is done by adding 5 (-5 if the field is negative) to the number at the right of the last decimal position specified for this field. All decimal positions to the right of the position specified for that field are then dropped (Figure 77).

The half adjust entry is allowed only with arithmetic operations. See *Columns 28-32 (Operation)*. This entry cannot be specified for an MVR operation or for a DIV operation followed by an MVR operation.



This calculation line shows a result field being half adjusted to two decimal positions (2 in column 52 and H in column 53).

2nd Position
↓

35.7968	Result of an add operation.
5	Add 5 to the number at the right of the last decimal position specified.
35.80xx	Drop all decimal positions to the right at the position specified.
35.80	Result after half adjusting.

Figure 77. Half Adjust

COLUMNS 54-59 (RESULTING INDICATORS)

Entry	Explanation
01-99	Any 2-digit number
KA-KN, KP, KQ	Any command key indicator
H1-H9	Any halt indicator
L1-L9	Any control level indicator
LR	Last record indicator
OA-OG, OV	Any overflow indicator
U1-U8	Any external indicator

Columns 54-59 have three purposes:

- To test the value of the result field after an arithmetic operation or after a CHAIN, KEY, LOKUP, COMP, READ, TESTB, or TESTZ operation. See *Operation Codes* under Part 2, *RPG II Programmer's Guide* for more information on each specific operation.
- To specify which command keys can be pressed for a SET operation.
- To specify which indicators to turn on or off using the SETON and SETOF operations.

Note: Command key indicators can be used as resulting indicators only if the operation is SET or SETOF.

Test Results

By entering an indicator in columns 54-59, you specify that the result field is to be tested after the operation specified in columns 28-32 is performed. Normally, only indicators 01-99 and H1-H9 are used for testing. The indicator specified is turned on only if the result field satisfies the condition being tested for. If the condition tested for is not met, the indicator is turned off. This indicator can then be used to condition following calculations or output operations (see *Example*). If the same indicator is used to test the result of more than one operation, the operation last performed determines the setting of the indicator.

Notice that three fields (columns 54-55, 56-57, and 58-59) can be used for this purpose. Each field is used to test for different conditions: columns 54-55, plus or high; columns 56-57, minus or low; columns 58-59, zero or equal. You can test for any or all conditions at the same time.

Columns 54-55 (Plus or High): An indicator in these columns when testing the result field to find:

- If the result field in an arithmetic operation is positive.
- If factor 1 is higher than factor 2 in a compare operation.
- If factor 2 is higher than factor 1 in a table array look-up operation.
- If a CHAIN operation is not successful.
- If each bit named in factor 2 is off for a TESTB operation.
- If the character tested in a TESTZ operation is one of the characters &, A-I.
- If the numeric field entered in a KEY operation is positive.

Columns 56-57 (Minus or Low): Place an indicator in these columns when testing the result field to find:

- If the result field in an arithmetic operation is negative.
- If factor 1 is lower than factor 2 in a compare operation.
- If factor 2 is lower than factor 1 in a table or array look-up operation.
- If the bits named in factor 2 are of mixed status (some bits on, some bits off) for a TESTB operation.
- If the characters tested in a TESTZ operation is one of the characters -, J-R.
- If the numeric field entered in a KEY operation is negative.

Columns 58-59 (Zero or Equal): Place an indicator in these columns when testing the result field to find:

- If the result field in an arithmetic operation is zero.
- If factor 1 is equal to factor 2 in a compare operation.
- If factor 2 is equal to factor 1 in a table or array look-up operation.
- If an end-of-file condition is reached for the demand file using the READ operation.
- If each bit named in factor 2 is on for a TESTB operation.
- If the character tested in a TESTZ operation is any character other than &, A-I, or –, J-R.
- If the numeric field entered in a KEY operation is zero or if an alphameric field is blank.

Allowing Command Keys To Be Pressed (SET)

Columns 54-59 can contain command key indicators (KA-KN, KP, KQ) for a SET operation.

When a SET operation occurs, only the command keys in columns 54-59 for that SET operation can be pressed at that time. From one to three command keys can be entered for each SET operation. If one or two command keys are specified, they can appear in any of the three sets of columns. See *Operation Codes, SET* under Part 2, *RPG II Programmer's Guide* for complete information on this operation.

Setting Indicators (SETON, SETOF)

You can turn on or off indicators by using the operation codes SETON or SETOF. See *Operation Codes, Setting Indicators* under Part 2, *RPG II Programmer's Guide* for more information on these operations. Any indicators to be turned on or off by the SETON or SETOF operation codes are specified in any of the three resulting indicators fields (Figure 78). Column headings in columns 54-59 have no meaning for SETON or SETOF operations.

RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page ___ of ___ Program Identification

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus, Minus, Zero); Compare (1>2, 1<2, 1=2); Lookup(Factor 2)is (High, Low, Equal)	Comments
01	C				SETON						0245	
02	C											
03	C											
04	C				SETON						22	
05	C											
06	C											
07	C				SETOF						45	
08	C											
09	C											

Figure 78. Setting Indicators

Example

Figure 79 shows the entry of two indicators that are used to test for the different conditions in a compare operation. These indicators are used to condition the calculations which must be performed for a payroll job. Indicator 10 is turned on if the hours worked (HRSWKD) are greater than 40 and is then used to condition all operations necessary to find overtime pay. Indicator 20 is turned on if HRSWKD is less than 40. It is also used to condition other operations. In line 03 if 20 is not on (the employee worked 40 or more hours), regular pay based on a 40 hour week is calculated. In line 06 if 20 is on (employee worked less than 40 hours), pay based on less than a 40 hour week is calculated.

COLUMNS 60-74 (COMMENTS)

Enter in columns 60-74 any meaningful information you want. The comments you use should help you understand what you are doing on each specification line. Comments are not instructions to the RPG II program. They serve only as a means of documenting your program.

COLUMNS 75-80 (PROGRAM IDENTIFICATION)

See *Common Entries.*

RPG CALCULATION SPECIFICATIONS

IBM International Business Machine Corporation

Form GX21-9093-2 Printed in U.S.A.

Program | Programmer | Date | Punching Instruction | Graphic Punch | Card Electro Number | Page 1 2 of __ | Program Identification 75 76 77 78 79 80

C

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / And: Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus	Minus	Zero; Compare: 1>2	1<2	1=2; Lookup(Factor 2)is: High	Low	Equal	Comments
01	C			HRSWKD	COMP	40										10	20	
02	C		10	HRSWKD	SUB	40	OVERTM	3	0									
03	C		N20	PAYRATE	MULT	40	PAY	6	2	H								
04	C		10	OVERTM	MULT	OVERRATE	OVRPAY	6	2	H								
05	C		10	OVERPAY	ADD	PAY	PAY											
06	C		20	PAYRATE	MULT	HRSWKD	PAY											
07	C																	

Figure 79. Conditioning Operations (Resulting Indicators)

Output specifications describe your output records. These specifications can be divided into two general categories:

- Record description entries (columns 7-31) which describe the output file record.
- Field description entries (columns 23-74) which indicate the position and the format of data on the output record.

Write the specifications on the output specifications sheet (Figure 80). The field description entries start one line lower than record description entries.

COLUMNS 1-2 (PAGE)

See *Common Entries.*

COLUMNS 3-5 (LINE)

See *Common Entries.*

COLUMN 6 (FORM TYPE)

An O must appear in column 6.

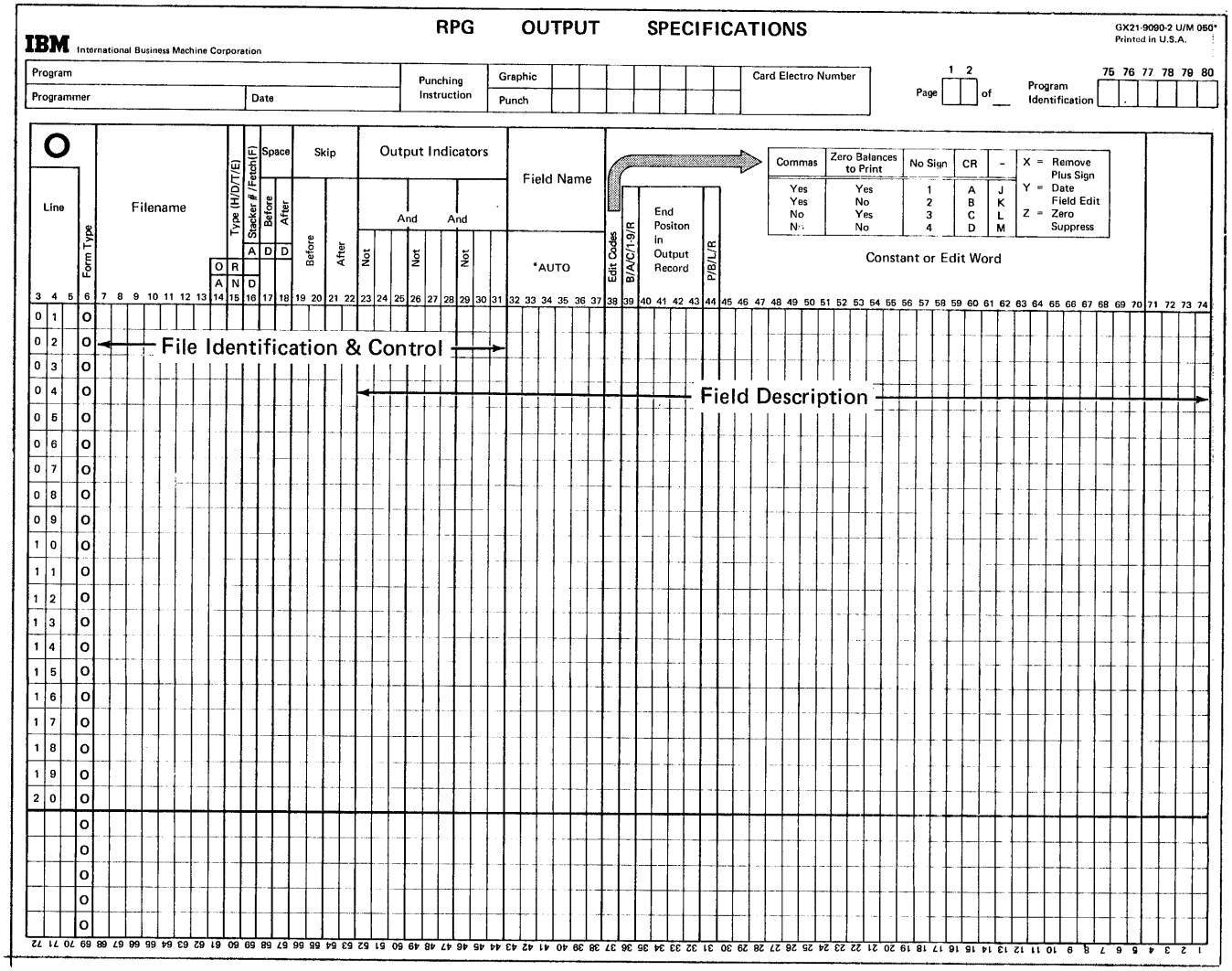


Figure 80. Output Specifications Sheet

COLUMNS 7-14 (FILENAME)

Use columns 7-14 to identify the output file you will be using. The filename must begin in column 7. Use the same filename given by the file description specifications.

When writing the specifications for output records, the filename need be specified only on the first line for the file. On subsequent lines for the same file, the filename need not be specified. However, the filename must be repeated if another output file is specified and you then want to continue with further specifications from the first output file (Figure 80).

COLUMNS 14-16

Use columns 14-16 to specify AND/OR lines for output operations. For further information, see *Columns 23-31 (Output Indicators), AND and OR Lines* under *Output Specifications.*

COLUMN 15 (TYPE)

Entry	Explanation
H	Heading records
D	Detail records
T	Total records
E	Exception records (lines to be written during calculation time)

Use column 15 to indicate the type of record to be written. Column 15 must have an entry for every output record. This record can be printed, written on the disk, or written on the display screen. We suggest you describe output files by entering the records for each file in this order: heading, detail, total, and exception (Figure 81, insert A).

Another method of describing output files is to enter all heading records for all output files, then all detail records for all output files, etc (Figure 81, insert B).

Heading Records (H)

Heading records usually contain unchanging identifying information such as column headings and page and date information.

Detail Records (D)

Detail records are closely connected with input data. Most data in a detail record comes directly from the input record or is the result of calculations performed on data from the input record.

Total Records (T)

Total records usually contain data that is the end result of specific calculations on several detail records. Total output cannot be specified for primary or secondary update files.

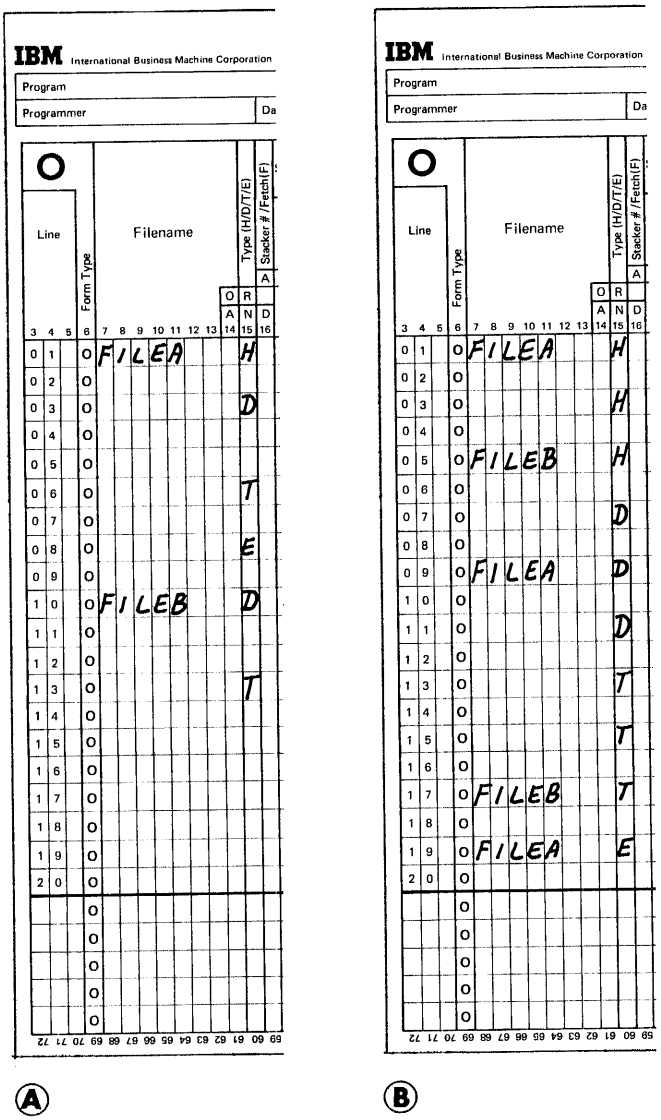


Figure 81. Order of Output Record Types

Exception Records (E)

Exception records are written on disk or printed during calculation time. This can be indicated only when the operation code EXCPT is used. See *Operation Codes* under Part 2, *RPG II Programmer's Guide* for further information on the EXCPT operation code.

COLUMNS 16-18 (ADD A RECORD)

Entry	Explanation
ADD	Add a record

Use columns 16-18 to specify that a record is to be added to an indexed or sequential file defined as an input, output, or update file. In addition to the ADD entry in columns 16-18, you must enter A in column 66 of the file description specifications sheet for the file to which records are being added. The output device for these files must be a disk.

The ADD entry must not be specified in an OR line. An ADD entry in columns 16-18 of the previous line also applies to the record in the OR relationship.

COLUMN 16 (FETCH OVERFLOW)

Entry	Explanation
F	Fetch overflow

Fetch overflow applies to the printer only. When the fetch overflow routine is not used and overflow is specified for the file, the following occurs when the overflow line is sensed:

1. All remaining detail lines in that program cycle are printed.
2. All remaining total lines in that program cycle are printed.
3. All lines conditioned by an overflow indicator are printed.
4. Forms advance to a new page if a skip to a new page is specified.

When the fetch overflow routine is not used and the overflow is not specified for the file, the following occurs when the overflow line is sensed:

1. All remaining detail lines in that program cycle are printed (if a printer operation spaced or skipped to the overflow area).
2. All remaining total lines in that program cycle are printed.
3. A skip to line 06 occurs.

If you do not want all of the remaining detail and total lines printed on the page before overflow lines are printed and forms advance to the new page, you can cause overflow lines to be printed ahead of the usual time. This is known as *fetching the overflow routine* and is indicated by the entry in column 16. Overflow is fetched only if all conditions specified by the indicators in columns 23-31 are met and an overflow has occurred. See *Overflow Indicators* under Part 2, *RPG II Programmer's Guide* for detailed information and examples of a fetched overflow routine. The fetched overflow routine does not automatically cause forms to advance. A skip to a new page (01-06 in columns 21 and 22 of the output specifications sheet) must also be specified on a line conditioned by the overflow indicator.

You must enter an F in column 16 of each OR line if you want to fetch the overflow routine for each record in the OR relationship.

COLUMNS 17-22 (SPACING AND SKIPPING)

Use columns 17-22 to specify spacing and line skipping for printer and display screen files. Figure 82 shows the possible spacing and skipping entries for the various files.

If columns 17-22 are blank, single spacing occurs after each line is printed. Otherwise, the system performs exactly the spacing and skipping you specify. There can be as many as six spaces (three before, three after) between two lines of printing on the printer.

You can save time by specifying that spacing or skipping should be done after printing. This means that the output file does not have to wait for paper movement before it can print.

Files	Space Before Column 17	Space After Column 18	Skip Before[1] Columns 19-20	Skip After[1] Columns 21-22
Printer	0-3	0-3	01-84	01-84

[1]The skip entries you specify in columns 19-22 must not exceed the forms length specified in line counter specifications or must not exceed 66 if no line counter specifications are supplied.

Figure 82. Possible Spacing and Skipping Entries

You can specify different spacing and skipping on OR lines. If there are no spacing or skipping entries in the OR line, spacing, and skipping are done according to the entries in the line preceding the OR line.

Note: If an incorrect entry is made in these columns, the compiler drops the entry and assumes blank. The specification is then processed as discussed in the preceding paragraphs.

You may want to specify a skip to a new page at the beginning of each job to prevent OCL statements from printing on the same page as output data. When specifying spacing and skipping, remember that if the last output line for the job is printed on line 01 of a page and no printer movement is specified, the system begins printing OCL statements for the next job on the next line. If this is not desirable, specify the appropriate printer movement (space 1 after).

Column 17 (Space Before)

Entry	Explanation
0-3	Number of lines to be spaced for display screen or printer files.

Column 18 (Space After)

Entry	Explanation
0-3	Number of lines to be spaced for display screen files or printer files.

Columns 19-20 (Skip Before)

Entry	Explanation
01	Display screen is to be blanked immediately.
01-84	Skip to lines 01-84 for printer files.

Columns 21-22 (Skip After)

Entry	Explanation
01-84	Skip to lines 01-84 for printer files.

Spacing and Skipping for Printer Files

Line spacing and skipping can be specified both before and after printing of a line. If both spacing and skipping are specified on the same line, they are done in this order:

1. Skip before
2. Space before
3. Skip after
4. Space after

Spacing or skipping to the overflow line or past the overflow line causes the overflow indicator to turn on. Skipping past the overflow line to a line on the next page, however, does not cause the overflow indicator to turn on.

If you want to turn on the overflow indicator to condition overflow operations when you skip to a lower line number (higher position) on the next page from a line above the overflow line, you can use a SETON operation. This is necessary because the overflow indicator is not turned on if the skip to a new page occurs on a nonoverflow line.

Skipping before and after refers to jumping from one printing line to another without stopping at lines between. This is usually done when a new page is needed. A skip to a lower line number means advance to a new page. Skipping can also be used, however, when a great deal of space is needed between lines. The entry must be the 2-digit number which indicates the number of the next line to be printed. You can indicate that skipping should be done before (columns 19-20) or after (columns 21-22) a line is printed. If you specify a skip to the same line number the forms are positioned on, no movement of the paper occurs.

Spacing and Skipping for Display Screen Files

The following rules apply to spacing and skipping for display screen files:

- A space before entry (0-3) can be made in column 17.
- A space after entry (0-3) can be made in column 18. If columns 17-22 are blank, the system assumes an entry of 1. If there are no spacing or skipping entries in an OR line, spacing and skipping are done according to the entries in the line preceding the OR line.
- A skip before to line 01 only can be specified in columns 19-20. Enter 01 in these columns whenever you want to clear the display screen.
- A skip after entry (columns 21-22) must not be specified for display screen files.

COLUMNS 23-31 (OUTPUT INDICATORS)

Entry	Explanation
01-99	Any resulting indicator, field indicator, or record identifying indicator previously specified.
KA-KN, KP, KQ	Any command key indicator previously specified.
L0-L9	Any control level indicators previously specified.
H1-H9	Any halt indicators previously specified.
U1-U8	Any external indicator set prior to program execution.
OA-OG, OV	Any overflow indicator previously assigned to this file.
MR	Matching record indicator.
LR	Last record indicator.
1P	First page indicator.

Use output indicators to give the conditions under which output operations are to be done. More specifically, use them to tell:

- When you want to output a line (see *Examples, Example 1*).
- When you want to output a field (see *Examples, Example 2*).

When you use an indicator to condition an entire line of print, place it on the line which specified the type of record (Figure 83, insert A). Place an indicator which conditions when a field is to be printed on the same line as the field name (Figure 83, insert B).

There are three separate output indicator fields (columns 23-25, 26-28, and 29-31). One indicator can be entered in each field. If these indicators are on, the output operation is done. An N in the column (23, 26, or 29) preceding each indicator means that the output operation is done only if the indicator is not on. No output line should be conditioned by all negative indicators (at least one of the indicators used should be positive). If all negative indicators do condition a heading or detail operation, the operation is performed at the beginning of the program cycle when 1P lines are written.

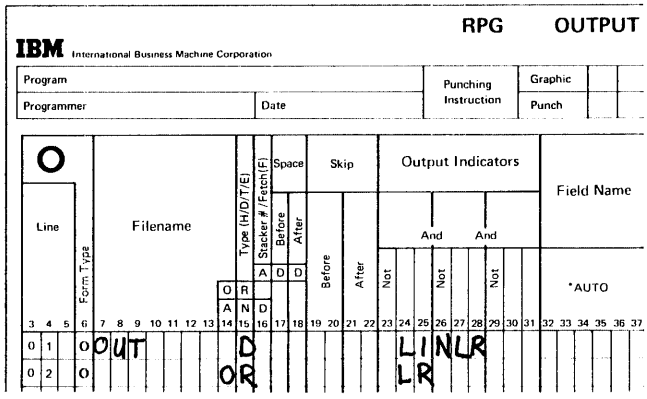
AND and OR Lines

If you need more than three indicators to condition an output operation, use an AND line. Enter the word AND in columns 14-16 and as many indicators as needed. The condition for all indicators in an AND relationship must be satisfied before the output operation is done. A maximum of 20 AND lines can be used for an output operation if no OR lines are used.

Output indicators can also be in an OR relationship. If one or the other condition is met, the output operation is done. A maximum of 20 OR lines can be used for an output operation if no AND lines are used. If AND and OR lines are combined, the total number of AND and OR lines for an output operation cannot exceed 20.

AND and OR lines are used to condition entire output lines; they must not be used to condition fields (see *Examples, Example 3*).

The use of an L0-L9 indicator in an OR relationship with an LR indicator can result in the specified operation being done twice when LR is on. One operation is performed during LR processing and the other at detail or total time. The following example shows how to correctly use the L0-L9 indicators in an OR relationship.



Command Key Indicators (KA-KN, KP, KQ)

Use command key indicators in columns 23-31 to condition output operations; however, any command keys entered in these columns must also be specified in columns 54-59 of the calculation specifications sheet for a SET or SETOF operation. See *Operation Codes* under Part 2, *RPG II Programmer's Guide* for complete information on each specific operation.

Overflow Indicators (OA-OG, OV)

Overflow indicators condition output operations on the printer. The operations conditioned by the overflow indicator are done only after the overflow line (end of page) has been reached.

If you have not assigned an overflow indicator to the printer file in the file description specifications, you cannot use an overflow indicator in the output specifications. In this case, advancing the forms to a new page is handled by the compiler, even though no overflow indicator is assigned.

If an overflow indicator is assigned by file description specifications and a specification line not conditioned by an overflow indicator specifies a skip to a line on a new page, the overflow indicator turns off before forms advance to a new page.

An overflow indicator can appear on either AND or OR lines. However, only one overflow indicator can be associated with one group of output indicators. That overflow indicator must also be the same indicator associated with the file by the file description specifications.

When the overflow indicator is used in an AND relationship with a record identifying indicator, unusual results are often obtained. This is because the record type might not be the one read when overflow has occurred. Thus, the record type indicator is not on and all lines conditioned by both overflow and record type indicators do not print.

An overflow indicator cannot condition an exception line (E in column 15) but can condition fields within the exception record.

First Page Indicator (1P)

The first page (1P) indicator is usually used to allow printing on the first page. It can also be used in connection with the overflow indicator to allow printing on every page (see *Examples, Example 4*). The information printed on the line conditioned by the 1P indicator must be constant information used as headings or fields for reserved words, such as PAGE and UDATE. The constant information is specified on the output specifications sheet.

The 1P indicator is used only with heading or detail output lines. It cannot be used to condition total or exception output lines. The 1P indicator cannot be used in an AND relationship with control level indicators. See *Column 41 (1P Forms Position)* under *Control Specifications* for information on forms alignment of the first page.

Halt Indicators (H1-H9)

On certain error conditions, you may not want output performed. Halt indicators can be used to prevent the data that caused the error from being used (see *Examples, Example 5*).

External Indicators (U1-U8)

A file named in the output specifications can be conditioned by an external indicator in the file description specifications. In this case, every output record for that file must be conditioned by the same external indicator used in the file description specifications.

Examples

Example 1: Figure 83, insert A, shows the use of one indicator to condition an entire line of printing. When 44 is on, the fields named INVOIC, AMOUNT, CUSTR, and SALSMN are all printed.

Example 2: Figure 83, insert B, shows the use of a control level indicator to condition when one field should be printed. When indicator 44 is on, fields INVOIC, AMOUNT, and CUSTR are always printed. However, SALSMN is printed for the first record of a new control group only if 44 and L1 are on.

Example 3: The use of indicators in both AND and OR lines to condition an output line is shown by Figure 84, insert A. The specifications in lines 01-04 say that the detail line is printed if either one of two sets of conditions is met. If 21, 40, 01 and 16 are all on, the line is printed; if 21 and 40 are on and 01 and 16 are off, the line is also printed.

A maximum of three indicators can be used on the output specifications sheet to condition a field since AND and OR lines cannot be used to condition an output field (Figure 84, insert B).

However, you can condition an output field with more than three indicators by using the SETON operation in calculations. For instance, indicators 10, 12, 14, 16, and 18 are used to condition an output field named PAY. In calculation specifications, you can SETON indicator 20 if indicators 10, 12, and 14 are on; then condition the output field PAY on indicators 20, 16, and 18 on the output specifications sheet.

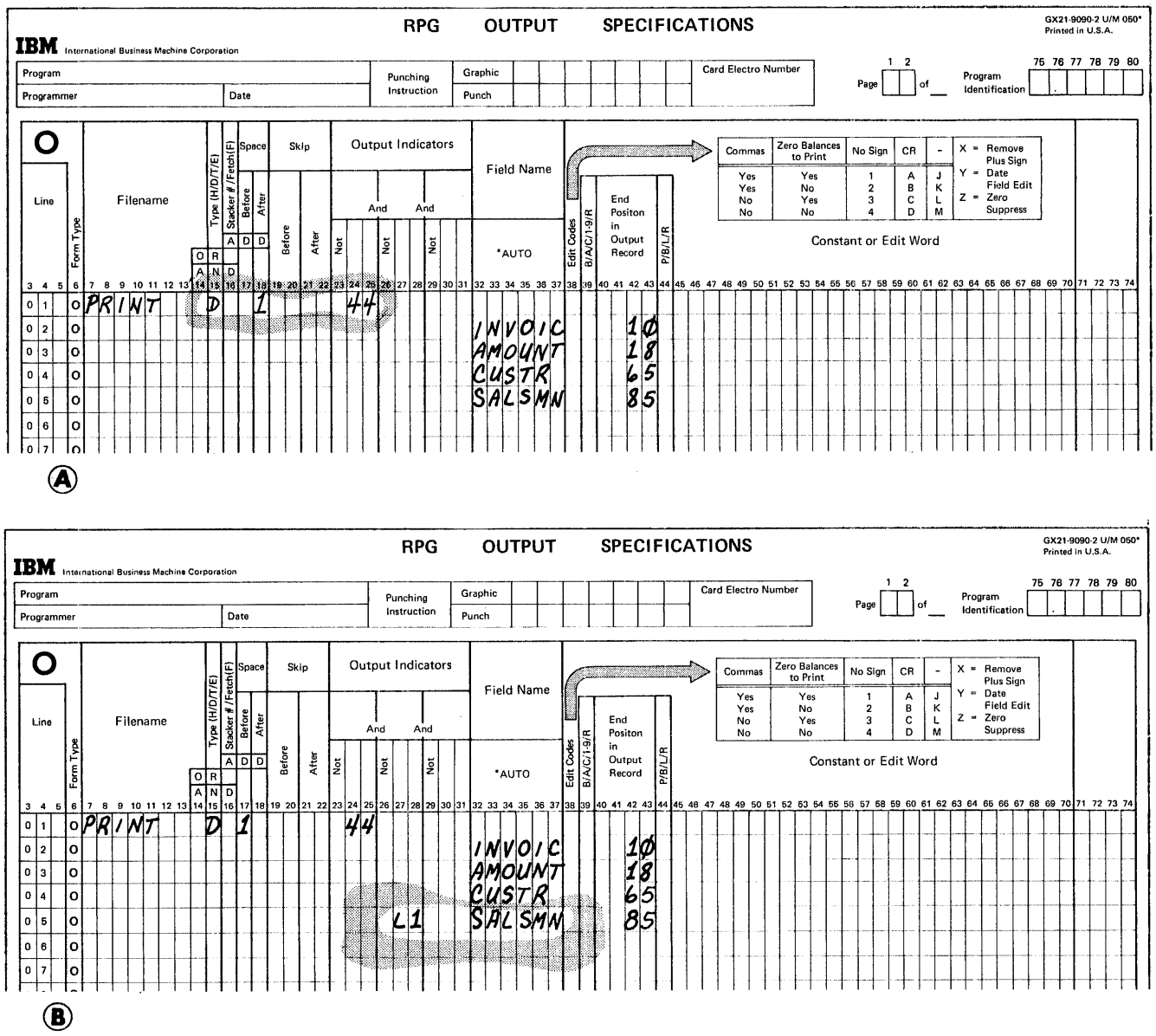


Figure 83. Output Indicators

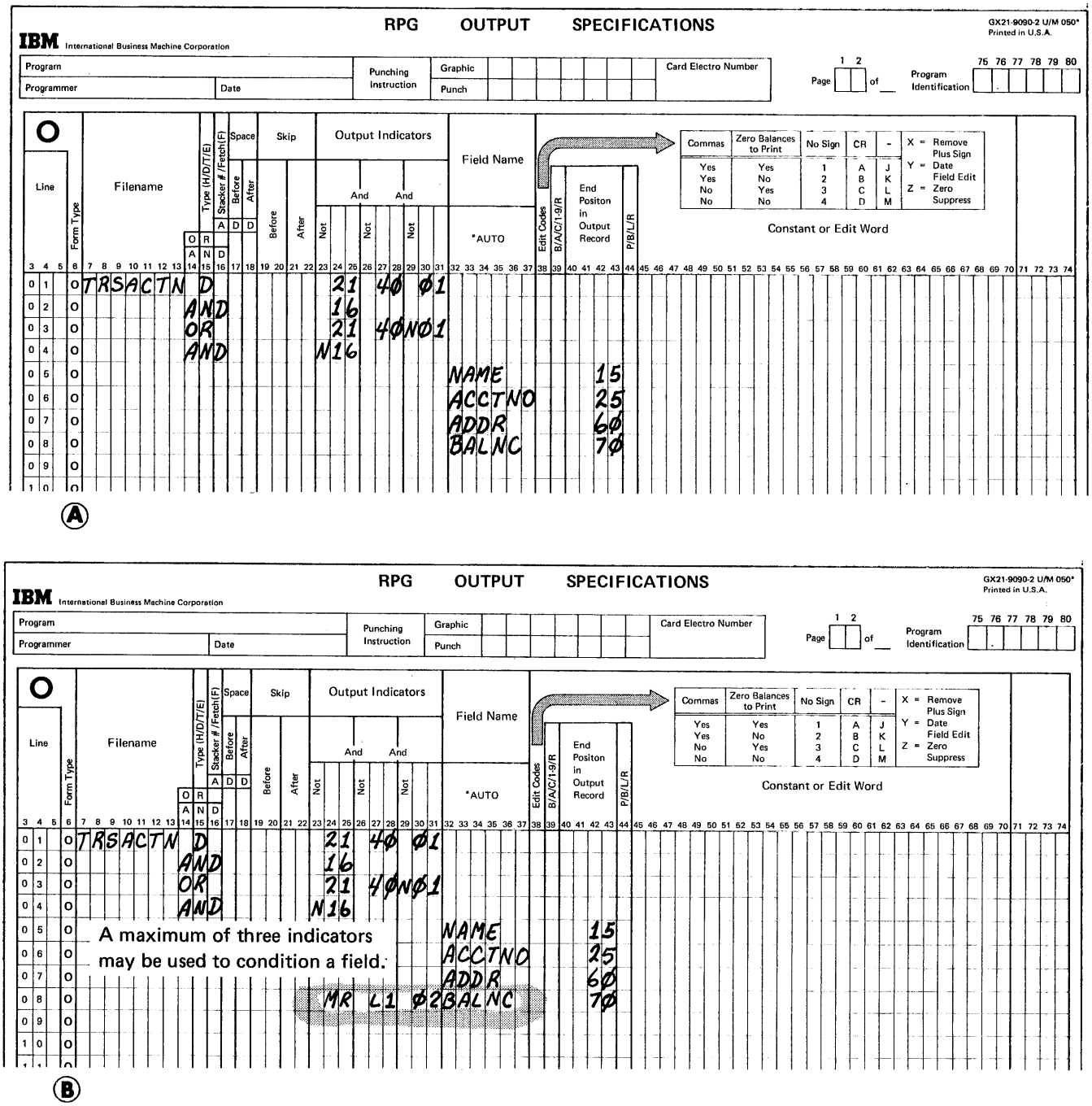


Figure 84. Output Indicators in AND and OR Lines

Example 4: Figure 85, insert A, shows how the 1P indicator is used when headings are to be printed on the first page only. Figure 85, insert B, shows the use of the 1P indicator and overflow indicator to print headings on every page.

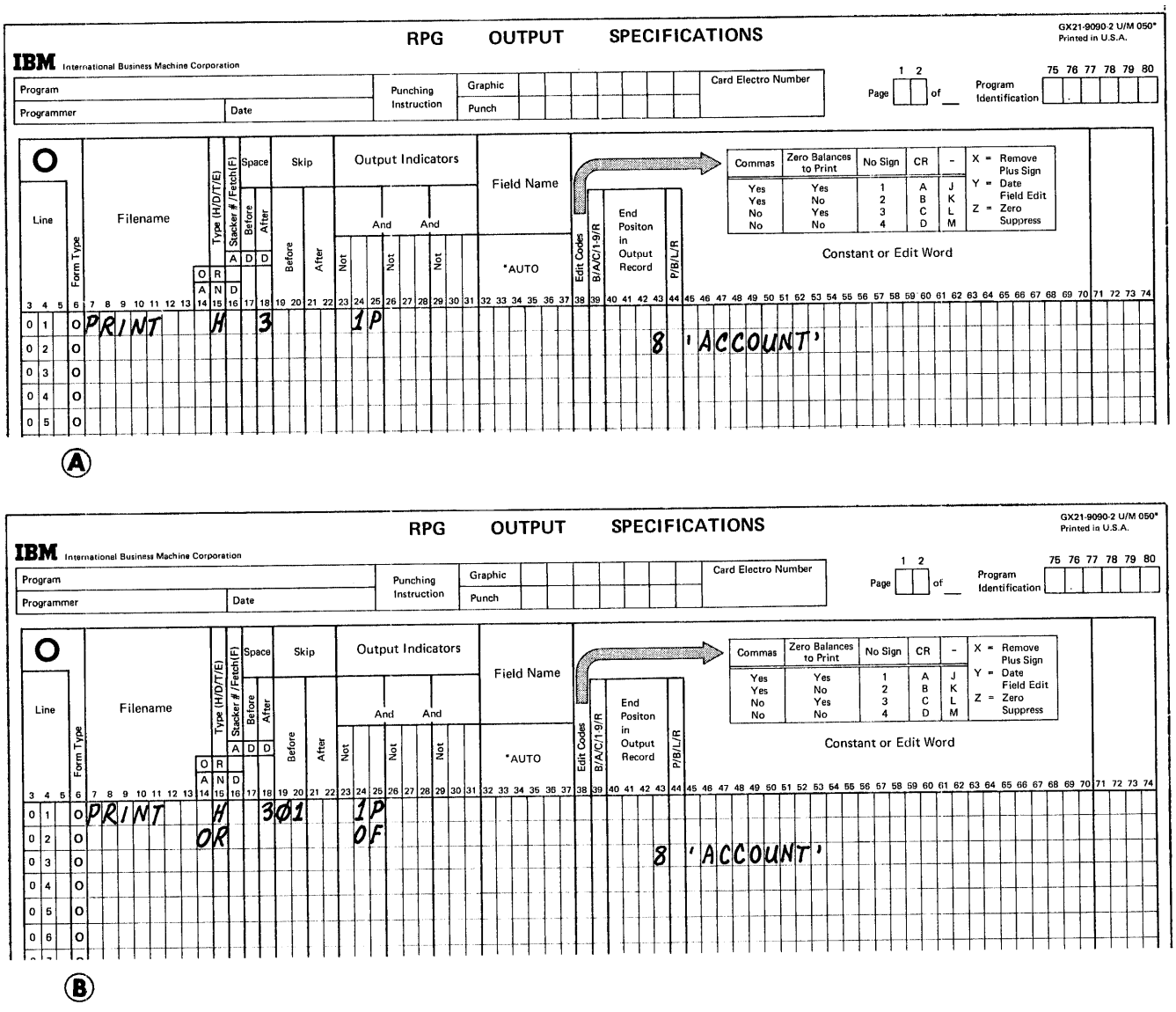


Figure 85. 1P Indicators

Example 5: Figure 86 shows coding necessary to check for an error condition and to stop processing on and printing from the record in error. If FIELDB contains all zeros, halt indicator H1 turns on (line 03 of Figure 86, insert A). In the calculation specifications, if H1 is on, all calculations are bypassed (line 01 of Figure 86, insert B).

In the output specifications, FIELDA and FIELDB are printed only if H1 is not on (lines 04 and 05 of Figure 86, insert C). Therefore, if indicator H1 is on, fields A and B are not printed. Use this general format when you do not want information that is in error to be printed.

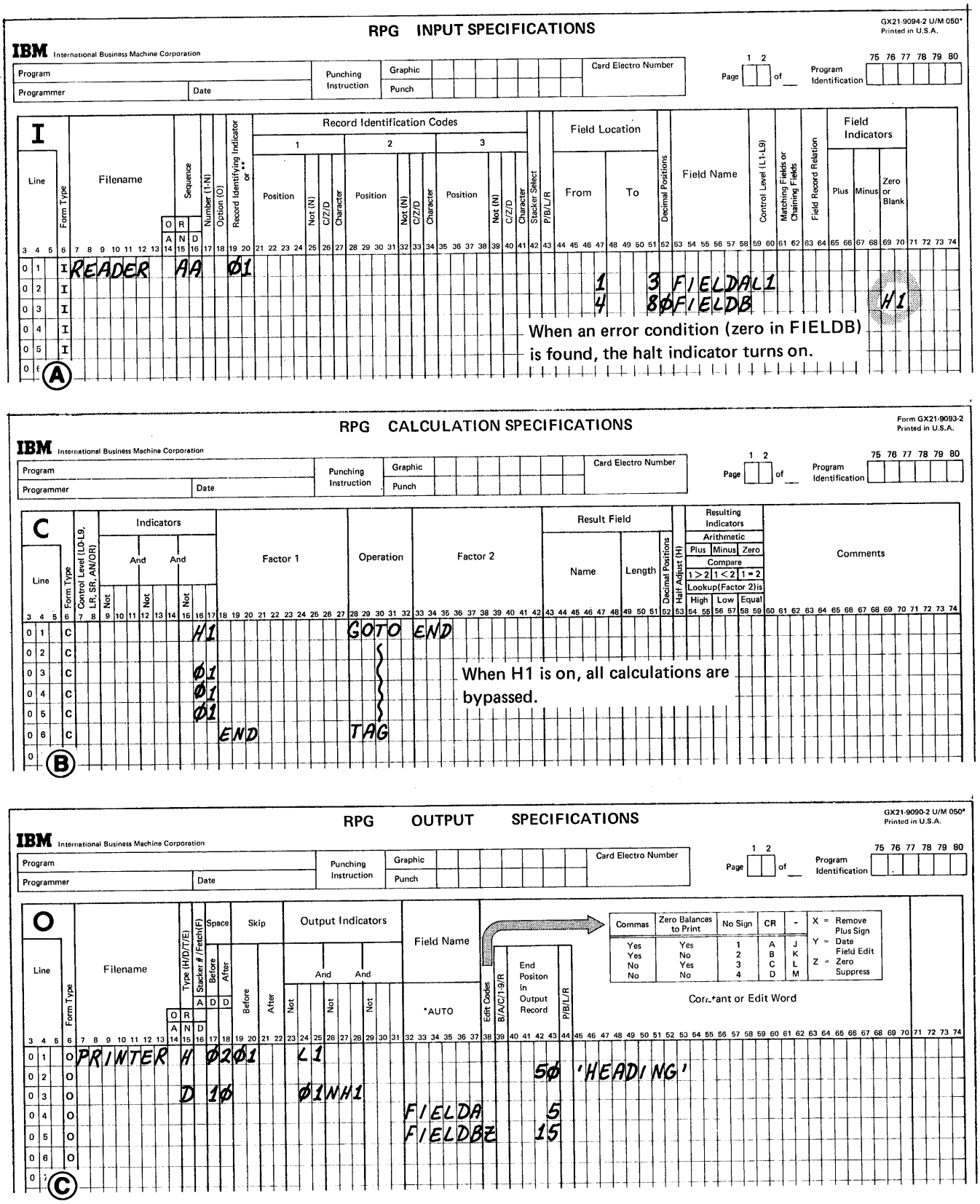


Figure 86. Preventing Fields From Printing

COLUMNS 32-37 (FIELD NAME)

In columns 32-37, use one of the following to name every field that is to be written out:

- Any field name previously used in this program.
- The special words PAGE, PAGE1, PAGE2, *PLACE, UDATE, UDAY, UMONTH, and UYEAR.
- A table name, array name, or array element.

Field Names

The field names used are the same as the field names on the input specifications sheet (columns 53-58) or the calculation specifications sheet (columns 43-48). Do not use these columns if a constant is used. See *Columns 45-70 (Constant or Edit Word)*. If a field name is entered in columns 32-37, columns 7-22 must be blank.

Fields can be listed on the sheet in any order since the sequence in which they appear on the printed form is determined by the entry in columns 40-43. However, they are usually listed sequentially. If fields overlap, the last field specified is the only field completely printed.

The sign (+ or –) of a numeric field is in the units position (rightmost digit). The units position prints as a letter unless the field is edited. See *Column 38 (Edit Codes).*

Special Words

Page Numbering (PAGE, PAGE1, PAGE2)

PAGE is a special word that, when used, causes automatic numbering of your pages. Enter the word PAGE, PAGE1, or PAGE2 in these columns if you want pages to be numbered. When a PAGE field is named in these columns without being defined elsewhere, it is assumed to be a 4-digit, numeric field with no decimal position. Leading zeros are suppressed automatically.

The page number starts with 0001 unless otherwise specified, and 1 is automatically added for each new page. See *Columns 53-58 (Field Name)* under *Input Specifications* for information concerning page numbering starting at a number other than 1.

It is possible at any point in your job to restart the page numbering sequence. To do this, set the PAGE field to zero before it is printed. One method of setting the PAGE field to zero is to use blank after (column 39). Another way is to use an output indicator. If the status of the indicator is as specified, the PAGE field is reset to zero (Figure 88). Remember that 1 is added to the PAGE field before it is printed (see *Examples, Example 1)*

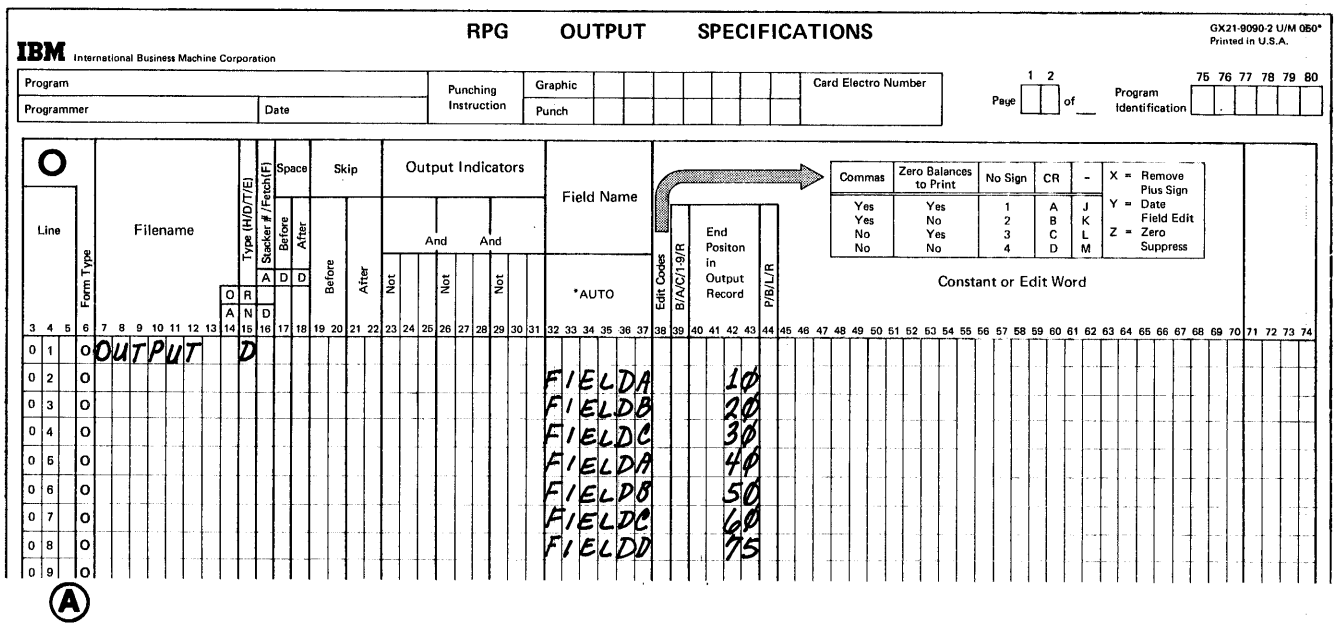
The three possible PAGE entries (PAGE, PAGE1, and PAGE2) may be needed for numbering different types of output pages.

Note: A PAGE field named only in output specifications is four digits long and need not be defined elsewhere. However, a PAGE field can be defined in input or calculation specifications and may be of any length. These PAGE fields are treated exactly as if they were names only in output specifications except for the difference in field length.

*Repeating Output Fields (*PLACE)*

*PLACE is a special RPG II word which makes it possible to write the same field in several locations on one record without having to name the field and give its end position each time the field is to be written. The fields are written in the same relative positions ending in the column specified by *PLACE. For example, if you want FIELDS A, B, and C to appear twice on one record, you can specify this in two ways:

- Define each field and its corresponding end position each time it is to be printed or written on disk (Figure 87, insert A, and Figure 89).
- Use the special word *PLACE (see Figure 87, insert B).



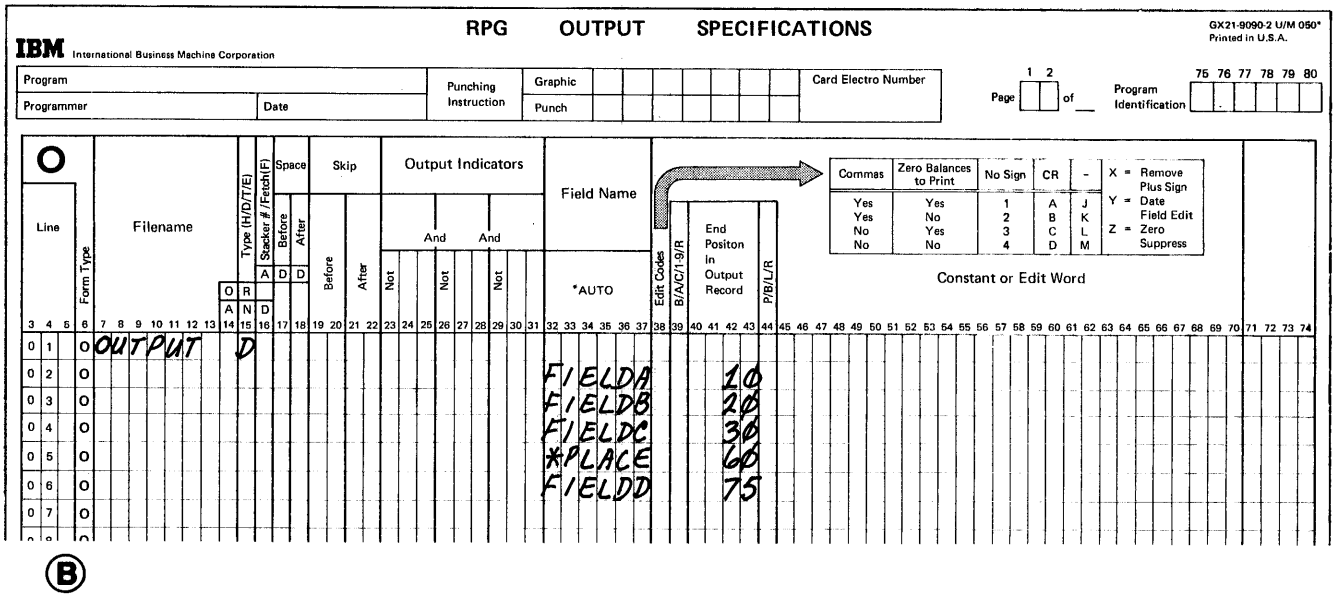


Figure 87. Writing Fields Twice on the Same Record

Both coding methods produce a record which looks like this:

Record Positions	10	20	30	40	50	60	75
Fields	FIELDA	FIELDB	FIELDC	FIELDA	FIELDB	FIELDC	FIELDD

However, it is easy to see that using the special word *PLACE saves extra coding.

When using *PLACE, all fields named for each record type (H/D/T/E) are written as usual in the location specified. The entry *PLACE then causes all of these same fields to be written ending at the position specified in the *PLACE statement. When using *PLACE, remember:

- *PLACE must be specified after the field names which are to be placed in different positions in one record (see *Examples, Example 2*).
- *PLACE causes all fields (in a record type) above the *PLACE entry to be written, not just the field named on the line above *PLACE.
- *PLACE must appear on a separate specification line for every additional time you want the field or group of fields written.
- An end position no greater than 256 must be specified for every *PLACE line. Allow enough space for all fields prior to the *PLACE to be printed again (see *Examples, Example 2*). Otherwise, overlapping occurs.
- Multiple or successive *PLACE entries can be specified for repetition of the fields or positions the first *PLACE repeats (see *Examples, Example 2).*
- The leftmost position of the fields to be moved by the *PLACE specification is always assumed to be position 1.
- The high end position to be used by *PLACE cannot be defined by a whole array. If a whole array does have the highest end position of all fields preceding the *PLACE, a field must be defined which has an end position greater than the end position of the whole array. This field can be a 1-position blank constant.
- Additional fields or constants can be specified after *PLACE. These fields are not affected by any *PLACE above them.

Note: Attempts to use the *PLACE function for other than its defined purpose may produce unpredictable results.

Date Fields (UDATE, UMONTH, UDAY, UYEAR)

Often you want the date to appear on your printed report or program listing. Use special words UDATE, UMONTH, UDAY, and UYEAR to get the date field you desire. The date fields are established at job setup time. The following rules apply to date fields:

- UDATE gives a 6-character numeric date field in one of three formats:

  Month/Day/Year
  Year/Month/Day
  Day/Month/Year

  Use columns 19 and 20 of the control specifications sheet to specify the date format and the editing desired. If columns 19-20 are blank, the date format is based on the contents of column 21.
- Use UDAY for days only, UMONTH for months only, and UYEAR for years only.
- These fields cannot be changed by any operations specified in the program. Thus, these fields are usually used only in compare and test operations.

Examples

Example 1: Figure 88 shows how an output indicator can be used to reset a PAGE field to zero. When indicator 15 is on, the PAGE field is reset to zero and a 1 added before the field is printed. When 15 is off, a 1 is added to the contents of the PAGE field before it is printed.

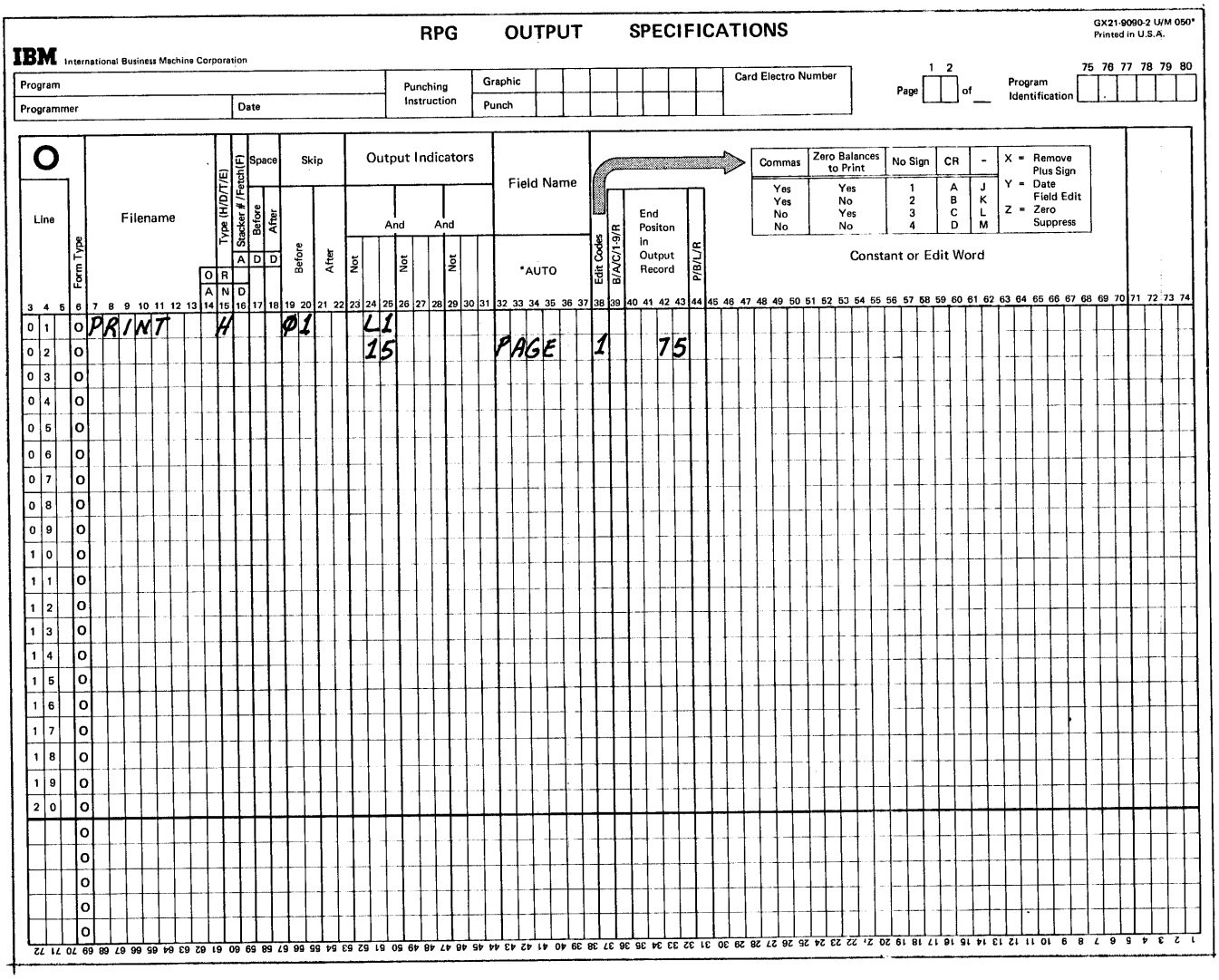
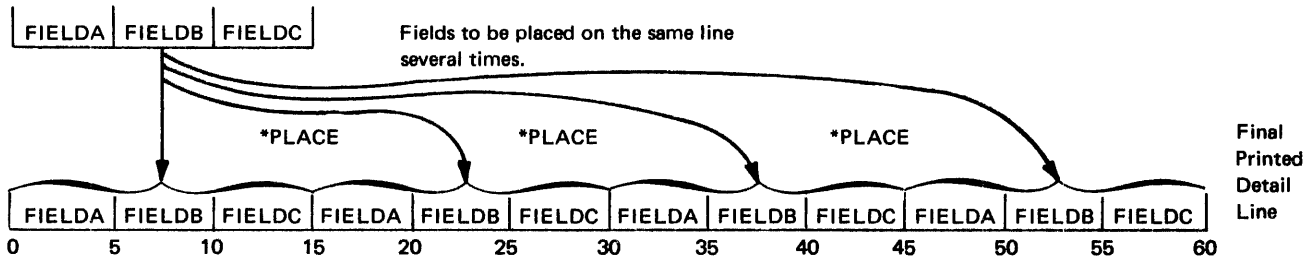


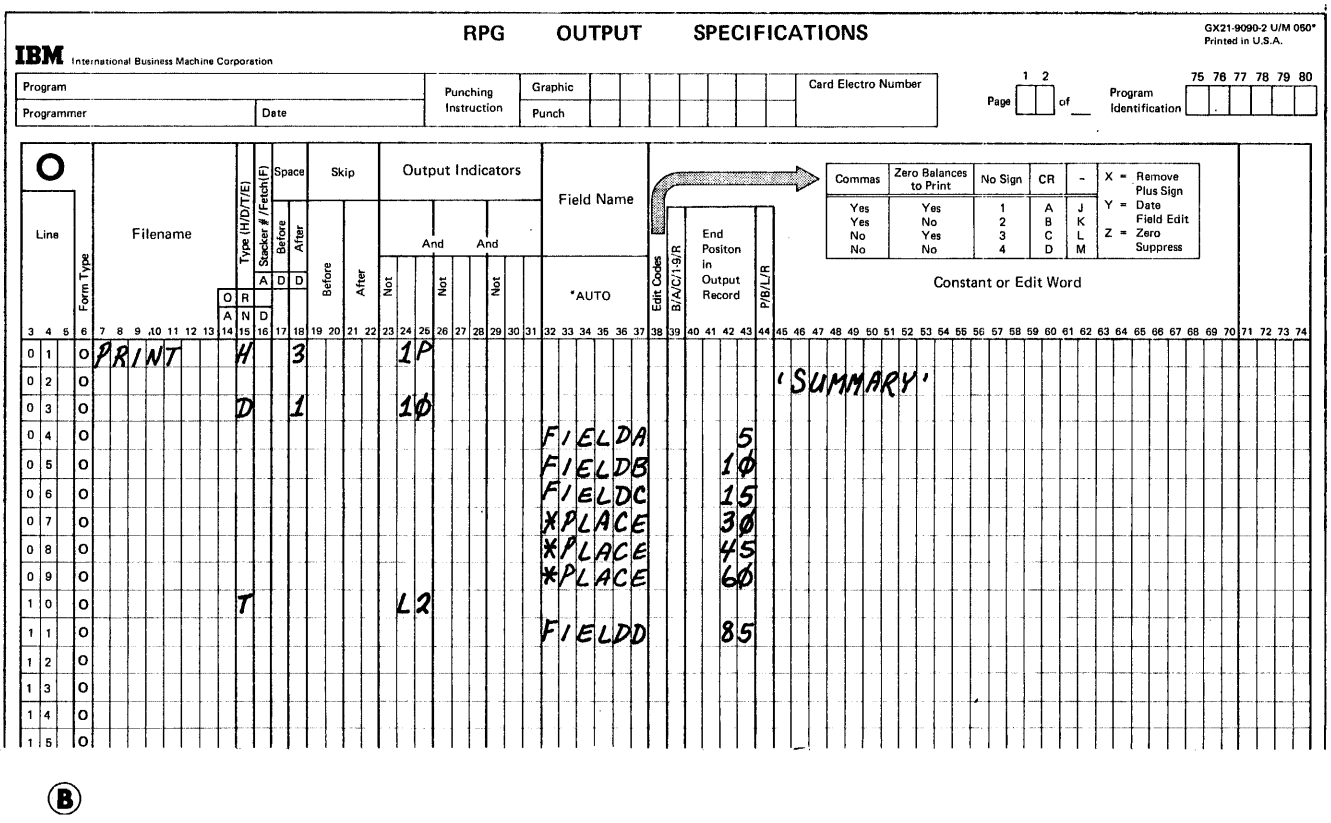
Figure 88. Resetting the PAGE Fields to Zero

Example 2: Figure 89 shows the use of the special word *PLACE to print the same fields several times on the same line. FIELDS A, B, and C are to be printed four times on one line (Figure 89, insert A). They are printed once when they are named and once for every *PLACE entry. In Figure 89, insert B, *PLACE is specified after the fields which are to be printed several times on the same line. All fields to which *PLACE applies appear on the same record. FIELD D, which appears on the total record, is not affected by *PLACE.

Notice also that an end position is given for every *PLACE. FIELDS A, B, and C have a total length of 15 characters. Thus the end positions given for the *PLACE entries allow room for the printing of 15 characters. This eliminates any overlapping.



Ⓐ



Ⓑ

Figure 89. *PLACE

COLUMN 38 (EDIT CODES)

Use column 38 when you want to:

- Suppress leading zeros of a numeric field.
- Omit a sign from the low-order position of a numeric field.
- Punctuate a numeric field without setting up your own edit word.

A table summarizing the edit codes that can be used is printed above columns 45-70 on the output specifications sheet. Each edit code punctuates differently. If you use an edit code in column 38, columns 45-70 must be blank except for the following condition: if asterisk fill or a floating dollar sign is required, enter '*' or '$' in columns 45-47. When an edit code is used to punctuate an array, two spaces are automatically left between fields of the array to the left of each element. Only unpacked numeric fields can be edited.

Note: Editing fields of nonprinter files is not recommended. If you do edit fields of a nonprinter file, you must be aware of the contents of the edited fields and the effects of any operations you want to do on them. For example, if you add an unedited field to an edited field, erroneous results will occur.

Figure 90 shows the edit codes and the options they provide. Figure 91 illustrates how data looks when it is edited by edit codes. Each code punctuates the field a little differently. All codes suppress leading zeros, except when J is entered in column 21 of the control specifications sheet. For this J entry all zero balances and balances with zero values to the left of the decimal comma are always written with one leading zero (such as 0,00 or 0,04). If an edit code is specified by the output specifications and the edit code is to print zero balances, a zero balance field always has a zero to the left of the decimal comma. The edit code cannot suppress this zero.

Figure 92 shows editing for date fields.

Edit Code	Commas	Decimal Point	Sign For Negative Balance			Entry in Column 21 of Control Specifications			Zero Suppress
			No Sign	CR	- (Minus)	D or Blank	I	J	
1	Yes	Yes	No Sign			.00 or 0	,00 or 0	0,00 or 0	Yes
2	Yes	Yes	No Sign			Blanks	Blanks	Blanks	Yes
3		Yes	No Sign			.00 or 0	,00 or 0	0,00 or 0	Yes
4		Yes	No Sign			Blanks	Blanks	Blanks	Yes
A	Yes	Yes		CR		.00 or 0	,00 or 0	0,00 or 0	Yes
B	Yes	Yes		CR		Blanks	Blanks	Blanks	Yes
C		Yes		CR		.00 or 0	,00 or 0	0,00 or 0	Yes
D		Yes		CR		Blanks	Blanks	Blanks	Yes
J	Yes	Yes			-	.00 or 0	,00 or 0	0,00 or 0	Yes
K	Yes	Yes			-	Blanks	Blanks	Blanks	Yes
L		Yes			-	.00 or 0	,00 or 0	0,00 or 0	Yes
M		Yes			-	Blanks	Blanks	Blanks	Yes
X[1]									
Y[2]									Yes
Z									Yes

[1] The X code performs no editing.

[2] The Y code suppresses the leftmost zero only. The Y code edits a three to six digit field according to the following pattern:

nn/n
nn/nn
nn/nn/n
nn/nn/nn

Figure 90. Edit Codes

Edit Codes	Positive Number — 2 Decimal Positions	Positive Number — No Decimal Positions	Negative Number[1] — 3 Decimal Positions	Negative Number[1] — No Decimal Positions	Zero Balance— Two Decimal Positions			Zero Balance — No Decimal Positions
					Entry in Column 21 of Control Specifications			
					D or Blank	I	J	
Unedited	1234567	1234567	00012	00012	000000	000000	000000	000000
1	12,345.67	1,234,567	.120	120	.00	,00	0,00	0
2	12,345.67	1,234,567	.120	120				
3	12345.67	1234567	.120	120	.00	,00	0,00	0
4	12345.67	1234567	.120	120				
A	12,345.67	1,234,567	.120CR	120CR	.00	,00	0,00	0
B	12,345.67	1,234,567	.120CR	120CR				
C	12345.67	1234567	.120CR	120CR	.00	,00	0,00	0
D	12345.67	1234567	.120CR	120CR				
J	12,345.67	1,234,567	.120-	120-	.00	,00	0,00	0
K	12,345.67	1,234,567	.120-	120-				
L	12345.67	1234567	.120-	120-	.00	,00	0,00	0
M	12345.67	1234567	.120-	120-				
X	1234567	1234567	00012	00012	000000	000000	000000	000000
Y								0/00/00
Z	1234567	1234567	120	120				

[1] The character } is a negative zero.

Figure 91. Examples of Edit Code Usage

UDATE	Edit Code	Contents of Column 19	Contents of Column 20	Contents of Column 21			
				Blank	D	I	J
Jan 30, 1974	Y	Blank	Blank	1/30/74	30/01/74	30.01.74	30.01.74
			—	1-30-74	30-01-74	30-01-74	30-01-74
		M	Blank	1/30/74	1/30/74	1.30.74	1.30.74
			—	1-30-74	1-30-74	1-30-74	1-30-74
		D	Blank	30/01/74	30/01/74	30.01.74	30.01.74
			—	30-01-74	30-01-74	30-01-74	30-01-74
		Y	Blank	74/01/30	74/01/30	74.01.30	74.01.30
			—	74-01-30	74-01-30	74-01-30	74-01-30

Figure 92. Date Fields

Normally, when you use an edit code in column 38, you cannot define an edit word in columns 45-70; however, there are two exceptions:

- If you want leading zeros replaced by asterisks, enter '*' in columns 45-47 of the line containing the edit code.
- If you want a dollar sign to appear before the first digit in the field (floating dollar sign), enter '$' in columns 45-47 of the line containing the edit code.

Asterisk fill and the floating dollar sign are not allowed with the X, Y, and Z edit codes.

It is also possible to have a dollar sign appear before the asterisk fill (fixed dollar sign). This is done in the following manner:

1. Place a dollar sign constant one space before the beginning of the edited field.
2. Place '*' in columns 45-47 of the line containing the edit code.

Figure 93 shows the effect the different edit codes have on the same field with a specified end position for output.

Edit Codes	Negative Number – 2 Decimal Positions End Position Specified as 10									
	Output Print Positions									
	3	4	5	6	7	8	9	10	11	
Unedited				0	0	4	1	K[1]		
1					4	.	1	2		
2					4	.	1	2		
3					4	.	1	2		
4					4	.	1	2		
A			4	.	1	2	C	R		
B			4	.	1	2	C	R		
C			4	.	1	2	C	R		
D			4	.	1	2	C	R		
J				4	.	1	2	-		
K				4	.	1	2	-		
L				4	.	1	2	-		
M				4	.	1	2	-		
X				0	0	4	1	K[1]		
Y		.	0	/	4	1	/	2		
Z						4	1	2		

[1] K represents a negative 2.

Figure 93. Effect of Edit Codes on End Position

COLUMN 39 (BLANK AFTER)

Entry	Explanation
Blank	Field is not to be reset.
B	Field specified in columns 32-37 is to be reset after the output operation is complete.

Use column 39 to reset a field to zeros or blanks. Numeric fields are set to zero and alphameric fields are set to blanks. This column must be blank for look ahead and UDATE fields.

Resetting fields to zeros is useful when you are accumulating and printing totals for each control group. After finding the total for one group and printing it, you want to start accumulating totals for the next group. Before you do this, however, you want your total field to start with zeros, not with the total it had for the previous group. A B-entry in column 39 resets the total field to zero after it is printed.

Note: If blank after (column 39) is specified for a field to be printed or written more than once, the B should be entered on the last line specifying output for that field.

COLUMNS 40-43 (END POSITION IN OUTPUT RECORD)

Entry	Explanation
1-4096	End position for disk or SPECIAL.
1-132	End position for 132-position printer.
1-40	End position for display screen.

Columns 40-43 apply to all output devices. Use these columns to define the end position on the output record of a field or constant. All entries in these columns must end in column 43. Enter only the number of the end position of the rightmost character in the field or constant.

Repeating Output Fields (*PLACE)

When *PLACE is specified for the printer, end position indicates the end position of the last field of the group that is to be printed. See *Columns 32-37 (Field Name).* Thus you must be sure you have indicated an end position that allows enough room for all specified fields to be printed. Be sure to allow enough space (as indicated by end position entries) on your output record to hold edited fields.

Note: Overlapped edited fields may result in undesirable output.

COLUMN 44 (PACKED OR BINARY FIELD)

Entry	Explanation
Blank	Field is unpacked numeric data, alphameric data, or to be printed.
P	Field is to be written on disk in the packed decimal format.
B	Field is to be written on disk in the binary format.

Column 44 must have an entry if a numeric field (decimal number) is to be written on disk in packed decimal or binary format. Packed decimal and binary fields cannot be displayed or printed; these fields can only be written on disk. Column 44 must be blank for *PLACE.

After decimal numbers are processed, they can be left in the unpacked format. However, for efficient use of disk space, decimal numbers can be converted into packed decimal or binary format. When binary output is specified, a numeric field one to four digits long (unpacked in storage) is converted into a 2-byte binary field when it is written on disk; a numeric field five to nine digits long is converted into a 4-byte binary field. When packed decimal output is specified, a byte of disk storage (except for the low-order byte) can obtain two decimal numbers. See *Column 43 (Packed or Binary Field)* under *Input Specifications* for a description of how data fields are represented in unpacked decimal, packed decimal, and binary formats.

Note: Although packed and binary fields require less disk storage space, the conversion routines needed to handle such data increase the object program size.

COLUMNS 45-70 (CONSTANT OR EDIT WORD)

Use columns 45-70 to specify a constant or an edit word. If you are using edit codes, you can also use columns 45-47 to specify a floating dollar sign or asterisk fill.

Constants

A constant is any unchanging information that is entered by a specification. Constants are usually words used for report headings or column headings.

The following rules apply to constants (Figure 94 contains examples):

- Field name (columns 32-37) must be blank.
- A constant must be enclosed in apostrophes. Enter the leading apostrophe in column 45.
- An apostrophe in a constant must be represented by two apostrophes. For example, if the word you're appears in a constant it must be coded as 'YOU''RE'.
- Numeric data can be used as a constant.

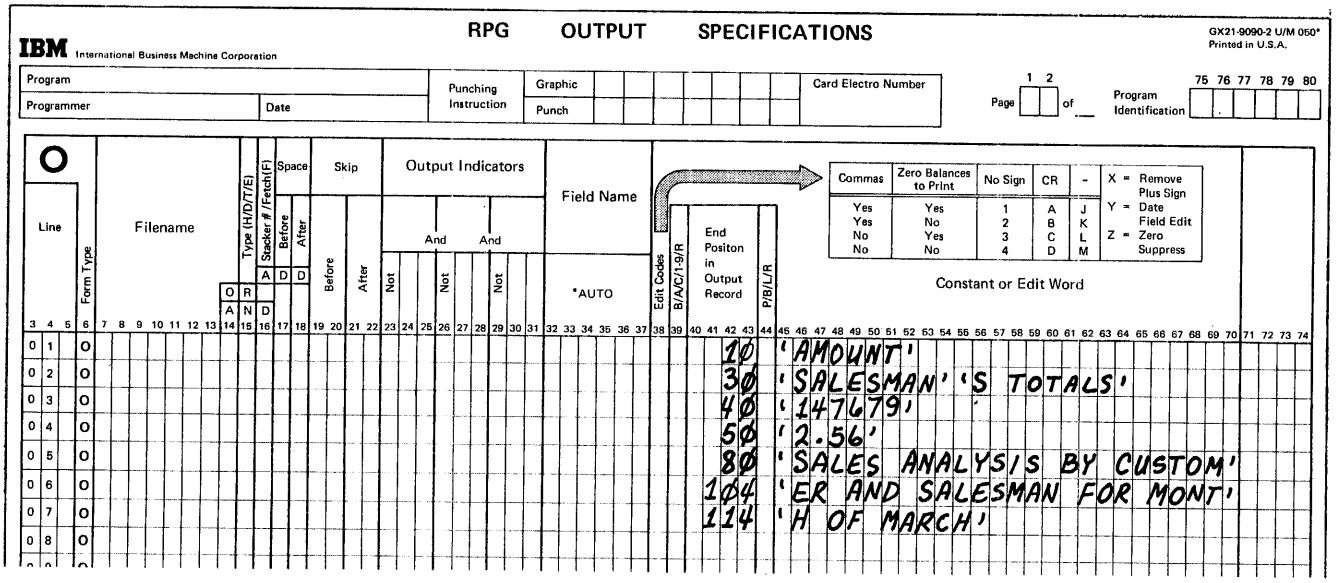


Figure 94. Examples of Output Constants

- Up to 24 characters of constant information can be placed in one line. You can use additional lines, but each line must be treated as a separate line of constants. The end position of each line must appear in columns 40-43.

Edit Codes

If you use an edit code in column 38, columns 45-70 must be blank except for the following condition: if asterisk fill or a floating dollar sign is required, enter '*' or '$' in columns 45-47. When '*' is entered in columns 45-47, asterisks replace all leading zeros (**NN). When '$' is entered in columns 45-47, the dollar sign appears before the first digit in the field ($N.NN). For more information on edit codes, see *Column 38 (Edit Codes)* under *Output Specifications*.

Note: Asterisk fill and the floating dollar sign cannot be used with the X, Y, and Z edit codes.

Edit Words

An edit word gives you more flexibility in punctuating a numeric field than an edit code. You specify directly:

- If commas, decimal points, and zero suppression are needed.
- If the negative sign should print.
- If the output is dollars and cents, whether you want a dollar sign and leading asterisks.

The following rules apply to edit words:

- Column 38 (edit codes) must not be used.
- Columns 32-37 (field name) must contain the name of a numeric field.
- Columns 40-43 (end position in output record) must contain an entry.
- An edit word must be enclosed in apostrophes. Enter leading apostrophe in column 45. The edit word itself must begin in column 46.
- Any printable character is valid, but certain characters in certain positions have special uses. See *Editing Considerations* in the following text.
- An edit word cannot be longer than 24 characters.
- The number of replaceable characters in the edit word must be equal to the length of the field to be edited. See *Editing Considerations* in the following text.
- All leading zeros are suppressed unless a zero or asterisk is specified in the edit word. The zero or asterisk indicates the last leading zero in the field to be replaced by a blank or asterisk.
- Any zeros or asterisks following the leftmost zero or asterisk are treated as constants; they are not replaceable characters.

Editing Considerations

One important thing to keep in mind is that you must leave exactly enough room on the printed form for the edited word. If the field to be edited is seven characters long on the input record, consider whether seven positions allow enough space for it to print on the report. By the time the field is edited, it may contain many more characters than seven.

When computing the length of an edited output field, determine how many of the editing characters are replaceable. The number of replaceable characters in the edit word must be equal to the length of the field to be edited. A replaceable character is a part of the edit word that does not require a position in the output file. The replaceable characters are:

0 (if used for zero suppression)

* (if used for asterisk fill)

b (blank)

$ (if it appears immediately to the left of zero suppress: a floating dollar sign)

A fixed dollar sign, decimal points, floating dollar sign, commas, ampersands (representing blanks), negative signs (- or CR), and constant information all require space in the output field.

Note: There are two exceptions to the rule that the number of replaceable characters in the edit word must be equal to the length of the field to be edited. The exceptions are:

- An extra space must be left in the edit word for the floating dollar sign. This ensures a print position for the dollar sign if the output field is full.

Unedited Field	Edit Word	Edited Field	Unedited Field Length	Replaceable Characters in Edit Word
72432N	'␣␣,␣$Ø.␣␣&-'	$7,243.25␣-	6	7

- An extra space can be left in the edit word if the first character in the edit word is a zero. In this case, the field to be edited is not zero suppressed by the leading zero, but all other specified editing is performed.

Unedited Field	Edit Word	Edited Field	Unedited Field Length	Replaceable Characters in Edit Word
ØØ746J	'Ø␣␣␣,␣␣␣'	ØØ7,461	6	7

Formatting Edit Words

The printer spacing chart is helpful when forming edit words. Figure 95 shows how an output line can be formatted using this chart. Note that Xs and zeros are used to show field positions. A zero indicates where zero suppression stops. An X indicates any number can appear in the position. Use blanks in place of the Xs when writing the edit words. Two additional Xs are provided for percent profit or loss since a negative value must be recognizable.

If it is necessary to show a negative number, a sign must be included in the edit word. You can use either the minus sign (-), or the letters CR. These print only for a negative number; however, the character positions they require must be taken into consideration when entering the end position of the field on the output specifications sheet. Figure 95 shows that for the field PERCPL, CR is to be printed for a negative balance. Assume the field PERCPL contains the negative data 2N (which is -25). The printed output looks like this: 25CR. If PERCPL is positive, CR does not print and the same field appears as: 25.

You can also use a minus sign to indicate a negative balance. If you want to leave a space between the number and the negative sign, place an ampersand (&) in the edit word before the minus sign. PERCPL then prints as: 25␣-.

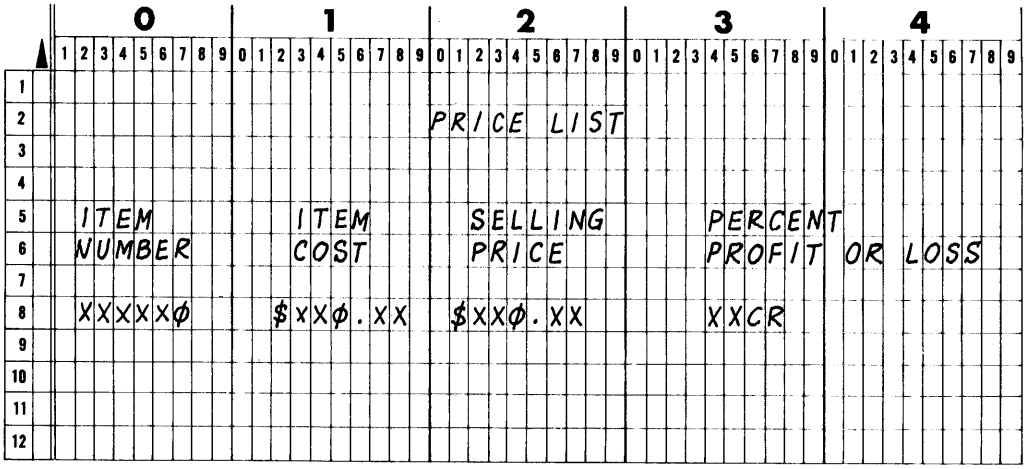
If you want to print a dollar sign, you also indicate this in your edit word. To print a dollar sign at the left of the field called SPRICE, put the dollar sign ($) next to the first quote mark and then put in the necessary blanks and punctuation. A dollar sign in this position is called a fixed dollar sign. The SPRICE field in Figure 96, line A, can look like any of the following (N stands for any number):

```
$NNN.NN
$ NN.NN
$  N.NN
$   .NN
```

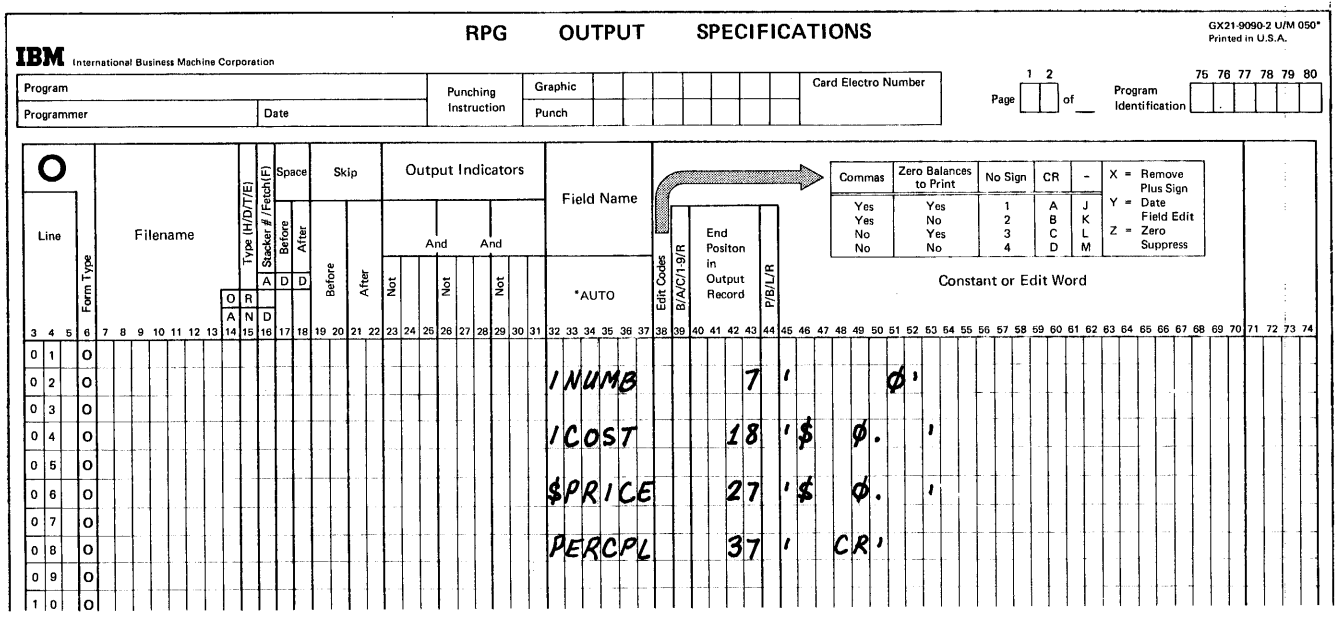
Unedited Data

Item number	000241
Item cost	02000
Selling price	02200
% profit or loss	25

Printer Spacing Chart



Output Sheet



Edited Output

241 $ 20.00 $ 22.00 25

Figure 95. Using the Printer Spacing Chart to Format Data

Suppose, however, you do not want a lot of empty space between the dollar sign and the first digit when zero suppression occurs. (This is commonly the case when writing checks.) You can fill in this empty space with asterisks (*). Instead of using 0 to indicate zero suppression, use the asterisk (*) to indicate that all extra spaces should be filled with asterisks. The SPRICE field in Figure 96, line B, can look like any of the following (N stands for any number):

```
$NNN.NN
$*NN.NN
$** N.NN
$***  .NN
```

You may want the dollar sign to always be next to the leftmost digit instead of filling in the space with asterisks or leaving extra blanks. This is indicated in the edit word by placing the $ next to the zero suppress 0. A dollar sign which changes positions depending upon the number of positions zero suppressed is known as a floating dollar sign. When printed, the SPRICE field in Figure 96, line C, can look like any of the following:

```
$NNN.NN
 $NN.NN
  $N.NN
   $.NN
```

Note that an extra space must be left in the edit word for the floating dollar sign. This ensures a print position for the dollar sign if the output field is full.

Examples of Edit Words

Figure 97, inserts A and B, shows examples of edit words. All examples assume that column 38 is blank. To avoid confusion about the number of blank positions in an edited data field, the symbol ⌀ is used to indicate where blank spaces appear. Zeros are not slashed where no confusion with the letter O is likely to result.

Examples numbered 1-9 are sample edit words for some of the most frequently desired output formats. The examples numbered 10-71 show possible ways of handling many of the editing situations you might have.

The numbers under the heading *Example Number* in Figure 97, inserts A and B, refer to the numbers in the following text:

1. Normal method of editing an amount field. Decimal point appears between dollars and cents; commas offset appears every three positions in the dollar portion of the field. The symbol CR appears in the edited data field when the data is negative; otherwise, it is replaced by blanks. Since zero suppression occurs through the unit-dollars position (zero in the edit word just left of the decimal point), blanks replace leading zeros and constants until a significant digit is encountered or until the specified zero is encountered. Thus, the decimal point and data to its right always appear in the edited data. Notice that, since zero suppression occurs through the position of the zero in the edit word, that zero is replaced

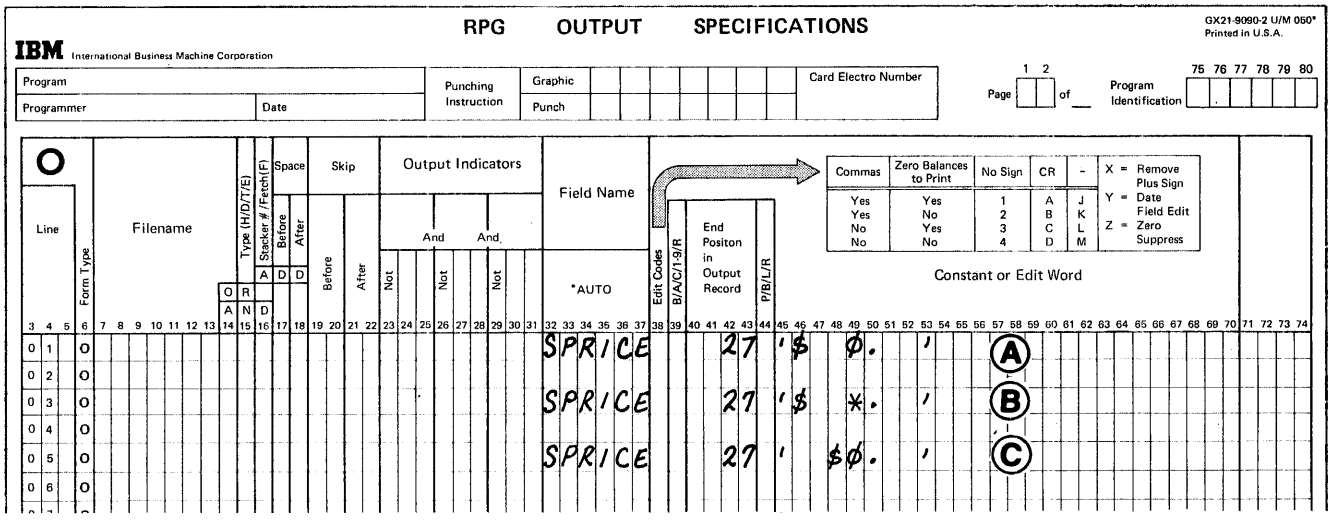


Figure 96. Different Edit Words Used on the Same Field

by a blank when no significant digit appears in the data field.

2. Normal method of punctuating a quantity field. Leading zeros and constants are replaced by blanks through the position of the zero suppression 0 (the next-to-last position in the edit word). Thus, if the entire data field is zero, a zero appears only in the low-order position of the edited data. A minus sign appears in the edited data if the field is negative; if not, the minus sign is replaced by a blank. The constant ON HAND always appears in the edited data as it is specified in the edit word, regardless of whether the minus sign appears as specified or as a blank.

3. Normal editing of an amount field. Because the zero suppression 0 appears in the 10-dollar position of the edit word, leading zeros and constants are retained starting with the unit-dollars position. Because the dollar sign is placed just left of the zero suppression 0, it becomes a floating dollar sign. In an edited data field, the floating dollar sign always appears to the immediate left of the first digit or retained constant. Notice that an extra position is allowed in the high-order portion of the edit word to accommodate the floating dollar sign. The minus sign appears if the field is negative; the asterisk always appears as a constant since a zero is specified to the left of it.

Edit Word (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)	Example Number	Source Data	Appears in Output Record as:
, , 0. &CR	1	0000000005 -	ØØØØØØØØØØ.05ØCR
, , 0 - &ON&HAND	2	00000000	ØØØØØØØØØ0ØØONØHAND
, , $0. - *	3	0000000005 +	ØØØØØØØØØ$0.05Ø*
, , $0. CR**	4	0034567890 -	ØØØ$345,678.90CR**
$, , 0.	5	0000000000	$ØØØØØØØØØØ.00
$& , , 0. &-&GROSS	6	1234567890 -	$Ø12,345,678.90Ø-ØGROSS
, , . -	7	00000000123 -	ØØØØØØØØØ1.23-
, , . CR	8	0000000000	ØØØØØØØØØØØØØØCR
, , * . &-	9	0000135792 -	*****1,357.92ØØ
	10	0000135678	0000135678
	11	0000135678 +	0000135678
	12	0000135678 -	000013567Q
	13	0000000000	ØØØØØØØØØØ
	14	0000135678 +	ØØØØ135678
	15	0000135678 -	ØØØØ135678
0	16	0000135678 -	ØØØØ135678
0	17	0000135678 +	Ø000135678
&CR&NET	18	0000135678 +	ØØØØ135678ØØØØNET
&CR&NET	19	0000135678 -	ØØØØ135678ØCRØNET
&-&&NET	20	0000135678 -	ØØØØ135678Ø-ØØNET
&NET&CR	21	0000135678	ØØØØØ1,356.78ØØØØ-NET
&NET&CR	22	0000135678 -	ØØØØ135678ØNETØCR
&&PROFIT	23	0000135678	ØØØØ135678ØØPROFIT
$ &-&NET	24	0000135678 +	$ØØØØ135678ØØØNET
$ &- &NET	25	0000135678 -	$ØØØØ135678Ø-ØNET
$0 -&NET	26	0000135678	Ø$000135678ØØNET
$0 &CR	27	0000135678 -	ØØØØ$135678ØCR
$0 &CR	28	1234567809 -	$1234567809ØCR
&-&TOTAL	29	0000000000 -	ØØØØØØØØØØØ-ØTOTAL
0&-&TOTAL	30	0000000000 -	ØØØØØØØØØØØ-ØTOTAL
$ 0 &CR&GROSS	31	0000000000 +	$ØØØØØØØØ00ØØØØGROSS
$0 &CR&GROSS	32	0000000000 -	ØØØØØØØØ$00ØCRØGROSS
$0	33	0000000000 -	ØØØØØØØØØØ$

Ⓐ

Figure 97 (Part 1 of 2). Examples of Edit Words

Edit Word (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)	Example Number	Source Data	Appears in Output Record as:
*&CR'	34	0000000000 -	**********ØCR
*&CR'	35	0000000000 -	********00ØCR
* '	36	0000135678 -	*000135678
* '	37	1234567890 +	1234567890
* '	38	0000135678 -	****135678
, , . &CR&&NET'	39	0000135678 -	ØØØØØ1,356.78ØCRØØNET
, , . &CR&-NET'	40	0000135678	ØØØØØ1,356.78ØØØØ-NET
$&Ø , , . &NET'	41	0000135678 +	$ØØ0,001,356.78ØNET
, $Ø . &NET'	42	0000000005	ØØØØØØØØØ$0.05ØNET
, $Ø. '	43	0000000005	ØØØØØØØØØØ$.05
, $Ø. -'	44	1234567890 -	$12,345,678.90-
, $Ø. CR'	45	0001356789 -	ØØØØ$13,567.89CR
, *. *CR**'	46	0000135678 +	*****1,356.78ØØØ**
, Ø. &CR*'	47	00000000 -	ØØØØØØØ.00ØCR*
, , *. -'	48	0000000000 -	**********.00-
, $Ø, . -SALES'	49	0000001234	ØØØØØØ$,012.34ØSALES
$& , , Ø. CR'	50	1234567890 -	$Ø12,345,678.90CR
, , , -OLD&BALMCE'	51	1234567890 -	1,234,567,890-OLDØBALANCE
, , , Ø -OLD&BALNCE'	52	0000000000 +	ØØØØØØØØØØØØØ0ØOLDØBALANCE
, , DOLLARS CENTS'	53	0000135678	ØØØØØ1,356DOLLARS78CENTS
, DOLLARS CENTS'	54	000000 +	ØØØØØØØØØØØØØØØØCENTS
, Ø DOLLARS CENTS&CR'	55	000000	ØØØØØDOLLARS00ØØØØØØØØØ
Ø LBS.& OZ.TARE&-'	56	000002 +	ØØØ0LBS.Ø2ØØØØØØØØØ
ØLBS. OZ.TARE&-'	57	000002 -	ØØØØLBS.02OZ.TAREØ-
Ø - - '	58	095140036	Ø95-14-0036
Ø HRS. MINS.&O'' CLOCK'	59	0042	Ø0HRS.42MINS.ØO'CLOCK
, Ø. '	60	000000	ØØØØØ.00
, .Ø '	61	000000	ØØØØØØØØ0
, $,Ø . '	62	00123456	ØØØ1$,234.56
, , , Ø * '	63	0000000000	ØØØØØØØØØ0*00
Ø, ,Ø '	64	001234	Ø,012,034
, *, * '	65	0000001234	******,012*34
& ,* Ø, '	66	013579	***130,579
- - &LATER'	67	093066	Ø9-30-66ØLATER
& & &LATER'	68	093066	Ø9Ø30Ø66ØLATER
/ / '	69	100166	10/01/66
, , . -'	70	000000015 -	ØØØØØØØØØ15-
, , Ø . -'	71	000000005	ØØØØØØØØ0.05Ø

(B)

Figure 97 (Part 2 of 2). Examples of Edit Words

4. Similar to example 3, except that (1) zero suppression is allowed up to the decimal point, (2) CR is used to indicate a negative value, and (3) two asterisks are printed at the end of the edited data. In the edited data shown, the dollar sign has *floated* to the left to precede the first significant digit. If the unedited data were all zeros, it would appear in the output record as $.00bb**. Note the extra position in the leftmost portion of the edit word to allow for the dollar sign.

5. Similar to example 4, except that (1) no symbol indicates a negative value, and (2) the edit word includes a fixed dollar sign. Because the dollar sign is placed in the extreme left position of the word and is not followed immediately by a zero suppression 0, it is a fixed dollar sign. The fixed dollar sign always appears in the leftmost position to the edited data field.

6. This example shows that a space can be left in the edited data field between a fixed dollar sign and the first digit, even when the entire field contains significant digits. An ampersand (&) in an edit word becomes a blank in the edited field. The minus sign appears in the edited data if the field is negative; the constant GROSS always appears in the edited data.

7. By not specifying a zero or asterisk, zero suppression can occur throughout the field; thus, edited data begins with the first significant digit. If the unedited data field only contained zeros, the entire edit word, except the minus, would be replaced by blanks when the field was edited.

8. Zero suppression can occur throughout the field. The symbol CR appears in the edited data field when the field contains a minus zero.

9. This example shows the use of asterisk fill. Asterisks replace all positions in the edit word to the left of the first significant digit. If the asterisk were in the rightmost position of the edit word, the entire edited field contains asterisks when the data is all zeros.

10, 11, and 12. No edit word. The data in the output record has the same format as the unedited data. Notice that the low-order position of the output field is printed as an alphabetic character (J-R) if the source data field is negative.

13, 14, and 15. A blank edit word. All leading zeros are blanked and any sign in the low-order position of the unedited field is removed when the data is edited. Negative values are not identified.

16. The effect is the same as shown in examples 13, 14, and 15.

17. Although the zero suppression 0 appears in the high-order position of the edit word, suppression of the first leading zero cannot be avoided. See the note under *Editing Considerations* for a discussion of this exception.

18 and 19. An ampersand appears as a blank in the edited data. The symbol CR appears in the edited data if the field is negative; it is replaced by blanks if the field is positive. The constant NET always appears in the edited data field.

20. An ampersand appears as a blank in the edited data. A minus sign, instead of CR, indicates negative values.

21 and 22. NET CR indicates when the edited data field is negative. Therefore, when the edited field is positive, CR appears as blanks.

23. The constant PROFIT appears in the edited data field. Negative values are not identified.

24 and 25. Similar to example 20, except that a fixed dollar sign is shown. An extra position is added to the edit word to allow for the dollar sign.

26. When the dollar sign appears to the immediate left of the zero suppression 0, it becomes a floating dollar sign, even when the dollar sign is entered in the leftmost position of the edit word.

27 and 28. The floating dollar sign is shown for different numbers of leading zeros. Note the extra position in the high-order portion of the edit word to allow for the dollar sign.

29 and 30. Even if there is no zero in the edit word, the minus sign appears in the edited field when the contents of the data field are minus zero. The constant TOTAL always appears in the edited field.

31. This example shows how some zeros can appear in the edited field when the entire field is zero. Zero suppression occurs through the position of the zero in the edit word. This leaves two positions in which zeros can appear in the edited field.

32. Similar to example 31, except that this example uses a floating dollar sign to replace the last suppressed zero.

33. Because the dollar sign is adjacent to the zero in the low-order position, it is a floating dollar sign. The floating dollar sign appears in the low-order position of an all-zero data field. This gives full protection with a floating dollar sign, even when all leading zeros are suppressed.

34. Because the asterisk appears in the low-order position of the edit word, asterisks appear throughout the edited field when the contents of that field are zero. This gives full protection with an asterisk, even when all leading zeros are suppressed.

35. This example shows asterisk protection to a certain position; thereafter, any additional leading zeros appear in the edited field.

36 and 37. This example shows asterisk protection and zero suppression for a single position. Note that the asterisk is replaced by a significant digit in the position. Negative values are not identified.

38. Asterisk protection and zero suppression for an entire field. Asterisks are replaced by significant digits.

39 and 40. A method of editing an amount field. Punctuation and zeros to the left of the first significant digit are blanked. The decimal point is also lost when there are fewer than three significant digits. The CR symbol is printed for an all-zero negative field; the constants NET or -NET always appear in the edited field.

41. The ampersand, which appears in the edited field as a space, makes it possible to keep the dollar sign fixed while limiting zero suppression to the minimum one position. All punctuation is retained regardless of leading zeros because the zero in the edit word is placed to the left of the first comma.

42 through 45. Standard methods for placing the floating dollar sign so that at least the decimal point is retained regardless of the number of leading zeros. The extra position appears in the leftmost position of the edit word to compensate for the floating dollar sign.

46. Asterisk protection and zero suppression to the decimal point. The decimal point is retained regardless of the number of leading zeros. Note that asterisks replace punctuation when leading zeros are suppressed. The second asterisk appears only when the edited data field is negative; the third and fourth asterisks always appear in the edited field.

47. This example shows a standard programming technique for retaining the decimal point while suppressing all leading zeros. The edited data shown is a minus zero value.

48. Asterisk protection and zero suppression to the decimal point. The decimal point is retained regardless of the number of leading zeros. A minus sign appears in the edited data if the field is negative.

49. This example shows that a constant (in this case, a comma) follows the dollar sign in the edited data if the floating dollar sign and the zero suppression 0 immediately precede a constant. This applies if there is a number of leading zeros. In the case of a comma, this has an awkward-looking effect; in the case of a decimal point, it is a normal approach (see *Example 43*).

50. This example shows how to insert a space between a fixed dollar sign and the first data digit when all digits in the field are significant. An ampersand in an edit word appears as a space in the edited data field.

51. Normal punctuation of a quantity field. In this example, all leading zeros, including the units position, are suppressed (compare with example 52).

52. Normal method of showing a single zero in the edited data field when the data field contains only zeros.

53 through 57. Constants in the edit word are handled the same as punctuation marks. That is, only constants to the right of the first significant digit or the zero suppression 0 appear in the edited data. Examples 55-56 show how more edit word constants, other than the CR or minus, can be blanked on a positive field. Examples 55-57 also show the effect that the position of the zero suppression 0 has on constants. In example 56, an ampersand placed after the first constant provides a space following that constant in the edited data.

58. Possible method for editing a social security number field. A hyphen (-) is used within the edit word. In the example shown, the initial zero is suppressed. However, if you want the initial zero to appear in the edited data, you must leave an extra position in the edit word. See the note under *Editing Considerations* for a discussion of this exception.

59. This example shows the use of constants in the edit word. In this example the constant is an apostrophe.

60 and 61. This example shows the effect that the position of the zero suppression 0 has on the decimal point (or any other constants) and following zeros.

62. This example shows that a dollar sign separated from the zero suppression 0, even if only by a comma, is not a floating dollar sign, but a constant.

63 through 66. Any zero or asterisk to the right of the high-order zero or asterisk is a constant, not a zero suppression 0 or asterisk-protection symbol. Examples 65 and 66 also show that asterisk protection replaces not only blanks but also other constants to the left of the first significant digit.

67 through 69. These are three examples of editing a date field. Since month numbers have at most one leading zero, it is not necessary to specify a zero suppression 0. Example 68 shows the use of an ampersand to retain a blank space in the edited data.

70. This example shows what happens to the decimal point when no zero suppression 0 is specified for a field which has fewer than three significant digits. This applies if the field is more than three digits long.

71. This example shows how to retain the decimal point in a data field which has fewer than three significant digits. This applies if the field is more than three digits long.

COLUMNS 71-74

Columns 71-74 are not used. Leave them blank.

COLUMNS 75-80 (PROGRAM IDENTIFICATION)

See *Common Entries.*

Part 2

RPG II Programmer's Guide

Indicators are used to signal when certain conditions occur or do not occur. After you have assigned an indicator (on one of the specification sheets) to signal a certain condition, the indicator assigned should be associated with that one condition throughout the entire program.

Many times you want operations to be performed only when certain conditions occur. Because the indicator associated with the condition tells whether or not the condition has occurred, you can use the indicator to signal whether or not the operation should be done. In this way, indicators condition operations.

The status (on or off) of an indicator assigned on a specification line is determined by the results of processing the instruction on that specification line. If the condition is satisfied, the indicator turns on; if it is not, the indicator turns off.

Usually indicators are set on or off by the condition in the program itself. However, you can also set certain indicators by the SETON, SETOF, and SET operations. At the start of each program all indicators are off except the 1P indicator, L0 indicator, and any external indicators which have been set on. All indicators which you can use are shown in Figure 98. A summary of all indicators is shown in Figure 99.

Note: Only record identifying indicators and control level indicators (L1-L9) are turned off at the beginning of each program cycle. All other indicators are left unchanged.

Indicators	File Description Specifications		Input Specifications			Calculation Specifications			Output Specifications
	Overflow (33-34)	File Conditioning (71-72)	[1]Record Identifying (19-20)	[1]Field Record Relation (63-64)	Field (65-70)	Control Level (7-8)	Conditioning (9-17)	Resulting (54-59)	Conditioning (23-31)
01-99			X	X	X		X	X	X
H1-H9			X	X	X		X	X	X
1P									X[3]
MR				X[2]			X		X
OA-OG, OV	X						X	X	X[4]
L0						X			X
L1-L9			X	X[2]		X	X	X	X
LR			X			X	X	X	X
U1-U8		X[5]		X			X	X	X
KA-KN, KP, KQ							X	X	X

Note: X denotes the indicators that can be used.

1 Not valid on look-ahead fields.
2 When field named is not a match field or a control field.
3 Only for detail or heading lines.
4 Cannot condition an exception line, but can condition fields within the exception record.
5 Not valid for table input files.

Figure 98. Valid Indicators

Location on Specification Sheets	Indicators	Where Located	Where Normally Used as Conditioning Indicators	Normally Turned On		Normally Turned Off	
				By	When	By	When
	Record identifying indicator	Input sheet cols 19-20	Input: field record relation (cols 63-64) Calculation: indicators (cols 9-17) Output: output indicators (cols 23-31)	Record identification	Before total-time calculations	Different record type	Before total-time calculations
	Field indicators: Plus/minus	Input sheet Cols 65-68: Numeric data only	Calculation: indicators (cols 9-17)	Data field with plus or minus balance	Before detail-time calculations	Data field without a plus or minus balance	
	Zero/blank	Cols 69-70	Output: output indicators (cols 23-31)	Data field with zero or blank balance	Initially upon blank-after, and before detail-time calculations when field is zero or blank	Data field without a zero or blank balance	Before detail-time calculations
	Control level (L1-L9)	Input sheet cols 59-60	Calculation: control level (cols 7-8) Calculation: indicators (cols 9-17) Output: output indicators (cols 23-31)	Control break of that or higher level	Before total-time calculations	A control field with the same contents as the control field of previous record	After detail-time output
	Matching records (MR)—based on Matching fields	Input sheet cols 61-62: M1-M9 control MR	Calculation: indicators (cols 9-17) Output: output indicators (cols 23-31)	Matching of primary with any secondary record	Before detail-time calculations	Nonmatch between primary and other records	Before detail-time calculations
	Calculation resulting indicators Arth ops: Plus, Minus, Zero COMP: High, Low, Equal	Calculation sheet cols 54-59	Calculation: indicators (cols 9-17) Output: output indicators (cols 23-31)	Plus result Minus result Field contents Zero* Factor 1 > Factor 2 Factor 1 < Factor 2 Factor 1 = Factor 2	Immediately when the specified condition is met upon execution of the operation	Failure to satisfy the assigned condition when the specifications in the line are executed	Immediately

Figure 99 (Part 1 of 2). A Summary of Indicators

Location on Specification Sheets	Indicators	Where Located	Where Normally Used as Conditioning Indicators	Normally Turned On		Normally Turned Off	
				By	When	By	When
	TESTZ: Plus			Presence of a C zone			
	TESTZ: Minus			Presence of a D zone			
	TESTZ: Blank			Any zone other than a C or D zone			
	LOKUP: High			Factor 1 < Factor 2			
	LOKUP: Low			Factor 1 > Factor 2			
	LOKUP: Equal			Factor 1 = Factor 2			
	KEY: Plus			Plus result	Immediately when the specified condition is met after the field is keyed	Failure to satisfy the assigned condition when the field is keyed	Immediately
	KEY: Minus			Minus result			
	KEY: Zero			Field contents Zero or blank			
	KEY: Blank						

Name/Number	Indicators	Where Normally Specified to be Turned On or Off	Turned On by Program Itself	Turned Off by Program Itself
	L1-L9 (control level)	Control level: cols 59-60—input sheet	Before total time upon control break	After each detail-output time
	L0 (level zero)	Nowhere	Initially and after each detail-output time	Never
	LR (last record total)	Nowhere	Before total time following last data record (after / * ⍉) *Note:* Zero or blank indicators for arithmetic and TESTZ operations are on initially and following Blank-After	After detail-output time (unless LR terminated job)
	1P (first page)	Nowhere	At beginning of program execution	After first detail-time output
	OA-OG, OV (overflow)	Nowhere	When end of page is reached	After next detail-time output unless fetch overflow is specified
	H1-H9 (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
	KA-KN, KP, KQ	Resulting indicators	Never	Never
	U1-U8	External or resulting indicators	During calculation	During calculation

Figure 99 (Part 2 of 2). A Summary of Indicators

01-99 (FIELD INDICATORS, RECORD IDENTIFYING INDICATORS, RESULTING INDICATORS, AND CONDITIONING INDICATORS)

You can assign any of the numbers 01-99 to indicate such things as:

- The type of record read. See *Columns 19-20 (Record Identifying Indicator,**)* under *Input Specifications.*
- The status (plus, minus, zero/blank) of an input field. See *Columns 65-70 (Field Indicators)* under *Input Specifications.*
- The results of a calculation operation. See *Columns 54-59 (Resulting Indicators)* under *Calculation Specifications.* See *Examples, Example 1* and *Example 2.*

Any of these indicators which you have assigned in those columns or used in SETON or SETOF operations can also be used to:

- Establish field record relations. See *Columns 63-64 (Field Record Relationship)* under *Input Specifications.*
- Condition calculation operations. See *Columns 9-17 (Indicators)* under *Calculation Specifications.*
- Condition output operations. See *Columns 23-31 (Output Indicators)* under *Output Specifications.*

Indicators reflect only one condition at a time. When one indicator is used to reflect two or more conditions, it is always set to reflect the condition in the last operation performed (see *Examples, Example 3*).

If any indicator 01-99 is set on or off by the operation codes SETON or SETOF, it remains on or off until an instruction in a specification line containing that same indicator is performed. The indicator is then set to reflect a condition from the operation performed.

Good Programming Practice

When assigning two or more numeric indicators (01-99), use indicators that are close to each other numerically. The system requires less main storage to check the status of indicators 02, 03, and 04 than it does to check 02, 15, and 22.

Examples

Example 1: Figure 100, insert A, shows that resulting indicator 10 is assigned to signal when a minus condition occurs. Indicator 10 turns on if the result is negative after the subtraction operation is performed. It then remains on (or off depending upon the result) until the same operation is performed again. The indicator is always set to reflect the result of the subtraction operation each time it is done.

RPG CALCULATION SPECIFICATIONS

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic	Card Electro Number	Page 1
Programmer	Date		Punch		

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus	Minus	Zero; Compare 1>2	1<2	1=2; Lookup(Factor 2)is High	Low	Equal	
01	C			SALES	SUB	RETRNS	NETSLS	92					10					A
02	C																	
03	C		01	SALES	SUB	RETRNS	NETSLS	92					10					B
04	C																	
05	C			SALES	SUB	RETRNS	NETSLS	92					10					C
06	C			NETSLS	ADD	TOTSLS	TOTSLS	102										
07	C			PROFIT	SUB	RTRPRF	NETPRF	82					10					
08	C																	

Figure 100. Indicators 01-99

Example 2: Figure 100, insert B, shows the same operation as insert A. However, this operation is conditioned by indicator 01. The operation is done only when indicator 01 is on. Resulting indicator 10 is set on only when the result of the operation is negative.

Example 3: Figure 100, insert C, shows the use of the same indicator (10) in two lines. The status of this indicator reflects the result of each operation. For instance, indicator 10 turns on after the operation in line 05 is done if the result of the operation is negative. However, if the result of the operation in line 07 is positive or zero, indicator 10 turns off. It is then reset only when the operation in line 05 is done again.

KA-KN, KP, KQ (COMMAND KEY INDICATORS)

Command key indicators can be assigned to indicate what command keys the operator can press for a SET operation. See *Operation Codes, SET* for further information.

Any of the indicators which are used in a SET or SETOF operation can then also be used to:

- Condition calculation operations in columns 9-17 of the calculation specifications sheet.
- Condition output operations in columns 23-31 of the output specifications sheet.

The command key indicators used as conditioning indicators in calculations (columns 9-17) and in output operations (columns 23-31) are turned on and off in the following manner:

- The indicators are turned on by pressing the appropriate command key or keys for a SET operation.
- The indicators are turned off when the SET operation in which they are specified is performed or when a SETOF operation is performed.

H1-H9 (HALT INDICATORS)

Use any halt indicators to:

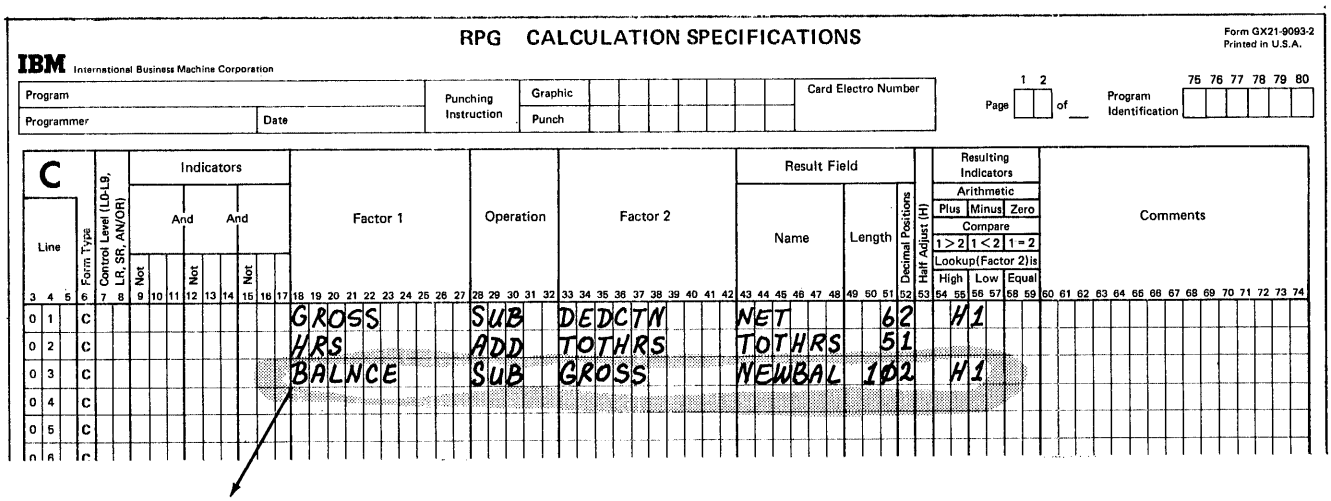
- Cause the program to stop after finding an unacceptable condition.
- Condition calculation or output operations that are not to be performed when such an unacceptable condition occurs. This is necessary because all calculation and detail output operations are still performed for the record that caused the unacceptable condition before processing stops.
- Establish field record relations. See *Columns 63-64 (Field Record Relation)* under *Input Specifications.*

Using the same indicator to test for two or more error conditions is not usually good practice. For example, Figure 101, insert A, shows the use of H1 in two different specification lines. If the result of the calculation operation in line 01 is negative, H1 turns on. This is an error condition. Processing continues, however, until this program cycle is completed. Thus, the operation in line 03 is done. If the result of this subtraction operation is positive, H1 turns off. The program does not stop because H1 is not on, even though an error condition was found in line 01.

The use of two different halt indicators as shown in Figure 101, insert B, does not allow a situation like the one just described to occur.

Any halt indicator assigned to test for zero or blank is off at the beginning of the program.

Note: If a halt indicator stops processing, it is turned off when the system is restarted. If more than one halt indicator turns on during a program cycle, each halt indicator causes a separate program halt.



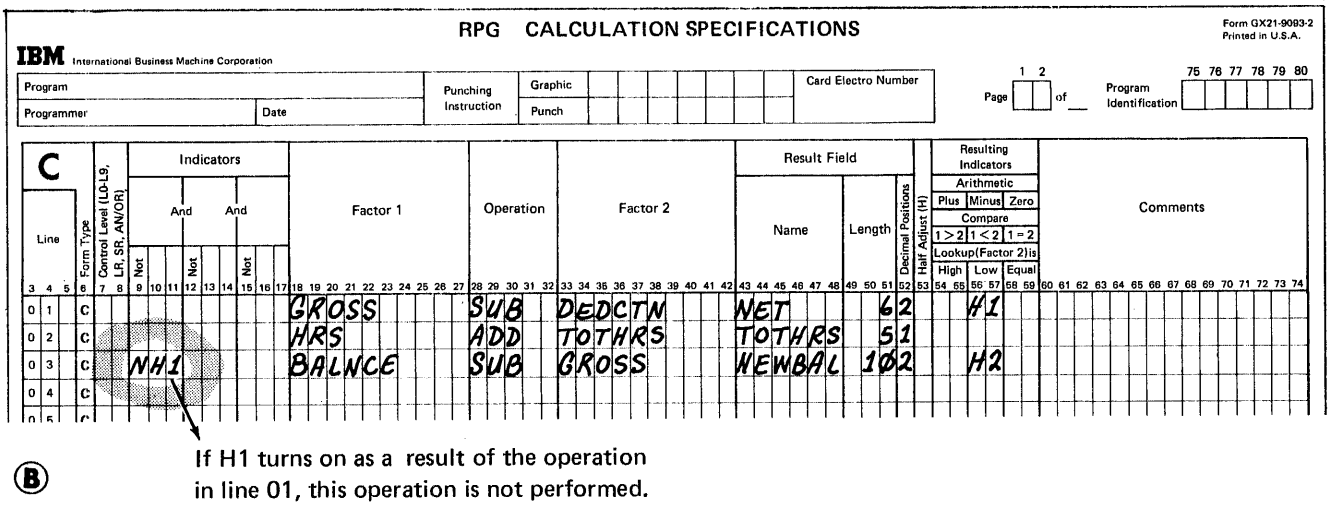


Figure 101. One Halt Indicator Testing for Two Conditions

1P (FIRST PAGE INDICATOR)

Use the first page indicator to condition those lines which are to be printed on only the first page. These lines are usually heading lines. Data is provided for lines conditioned by the 1P indicator by constants entered in columns 45-70 of the output specifications sheet.

All lines conditioned by the 1P indicator are printed even before the first record from input file is processed. Therefore, do not condition output fields which are based upon data from input records by the 1P indicator. You get meaningless output if you do.

Calculation operations cannot be conditioned by the 1P indicator. This indicator is on at the beginning of the program and turns off after the detail output has been performed on the first page output.

MR (MATCHING RECORD INDICATOR)

Use the MR indicator to condition calculation and output operations which are to be done only when records match. The MR indicator turns on when a primary file record matches any secondary file record on the basis of the matching fields indicated by M1-M9. The matching record indicator is always set to reflect the match or nonmatch condition before any detail calculation operations are performed. If all primary file records match all secondary file records, the MR indicator is always on.

When a control level indicator (L1-L9) is specified in columns 7-8 (total time) and MR is specified in columns 9-17, MR indicates the matching condition of the previous record and not the one just read that caused the control break. After all operations conditioned by control level indicators (specified in columns 7-8 of the calculation specifications sheet) are done, MR then indicates the matching condition of the record just read.

If record types are read for which no matching fields are specified, they are processed immediately as if they belong to the same match group as the record previously processed. MR is always off for these types.

OA-OG, OV (OVERFLOW INDICATORS)

Use overflow indicators for printer files. Use them primarily to condition the printing of heading lines. If you use an overflow indicator to condition output lines, you must assign an overflow indicator to the printer file on the file description specifications sheet (columns 33-34). This same indicator must then condition all lines that are to be written only when overflow occurs.

If the destination of a space/skip or print operation falls within the form overflow area, the overflow indicator is turned on and remains on until all overflow lines are printed. However, if a skip is specified that advances the form past the overflow line to the first line or past the first line on a new page, the overflow indicator does not turn on. Certainly you do not want the overflow indicator on to signal a need for a new page when you just skipped to a new page.

If an overflow indicator is used as a conditioning indicator, it indicates that output is to be performed at overflow time. This applies regardless of whether or not the line conditioned by the indicator is in an AND or OR relationship with other indicators.

When an overflow indicator is used, a forms skip specification usually is made on the first line conditioned by an overflow indicator. Otherwise, forms do not advance. Remember, they advance automatically if you do not use overflow indicators.

The overflow indicator can be set by the SETON or SETOF operation code. After all total records are written, however, the indicator is set as it normally is in accord with the overflow line. See *Overflow Indicators* in this section for further information.

L1-L9 (CONTROL LEVEL INDICATORS)

Control level indicators signal when a change in a control field has occurred. Because they turn on when the information in a control field changes, they can condition operations (such as finding totals) that are to be performed only when all records having the same information in the control field have been read. They can also be used to condition total printing (last record of a control group), or to condition detail printing (first record in a control group). Control level indicators always turn on after the first record of a control group is read. Control level indicators can be used in three different types of specifications: input, calculation, and output.

Input Specifications

If a control level indicator is entered in columns 59-60 of this sheet, the field described in columns 53-58 is a control field. This means that the field on each record read is matched against the same field on the previous record. If the information is not the same, the control level indicator turns on. All lower level indicators turn on when a higher level indicator turns on. For example, if L8 turns on, L1-L7 also turn on.

When a control level indicator is used on the input specifications sheet in columns 63-64 (field record relation), the data from the field named in columns 53-58 is accepted and used only when the control level indicator is on.

If record types without a control field are read, they are treated as if they belong to the same control group as the preceding record. No control level indicator is set for them. Control level indicators can also be used to establish field record relations. See *Columns 63-64 (Field Record Relation)* under *Input Specifications*.

Calculation Specifications

When a control level indicator is entered in columns 7-8 of this sheet (total calculations), it conditions the operation so that it is done only when a control field changes. If any control level indicator appears in columns 9-17 (detail calculations), the operation is done only on the first record of a new control group.

A control level indicator can be turned on or off by operation codes SETON and SETOF. However, these operations do not cause all control level indicators lower than the one specified to turn on or off. For example, when L2 is set on, L1 does not automatically turn on.

Output Specifications

Control level indicators entered in columns 23-31 of this sheet specify when output records are written:

- If the control level indicator is entered along with a T in column 15 and no overflow indicator is used, the record is written only after the last record of a control group is processed.
- If the indicator is entered along with an H or D in column 15 and no overflow indicator is used, the record is written only after the first record of the new control group is processed.
- If the control level indicator is entered along with an overflow indicator, the record is written after the overflow line is sensed (provided a control break has also occurred).

L0 INDICATOR

The L0 indicator is never assigned, but it is always automatically on. It is normally used only in columns 7-8 of the calculation specifications sheet to specify that the calculation is to be done at total time.

LR (LAST RECORD INDICATOR)

Use the LR indicator to condition all operations that are to be done only at the end of the job. For disk or console files, the LR indicator is normally turned on when the last record is detected. No record identifying indicators can be on while last record processing is performed for these files. When LR is turned on, all other control level indicators (L1-L9) used also turn on automatically.

For keyboard files, the LR indicator must be turned on at the appropriate time in calculation specifications. Record identifying indicators may be on while last record processing is performed for these files. When LR is turned on in detail calculations, all other control level indicators are automatically turned on at the beginning of the next cycle.

Do not specify an L0-L9 indicator in an OR relationship with an LR indicator. If you do, the specified operations are done twice at LR time.

In System/32 RPG II, all total lines conditioned by LR are performed last. The job ends after all total records are written. The LR indicator cannot be turned off by a SETOF operation.

U1-U8 (EXTERNAL INDICATORS)

Indicators U1-U8 are external indicators. This means they are normally set prior to processing by an operation control language (OCL) statement. Their setting can be changed during processing, allowing the program to alter the status of these indicators.

You can use these indicators as file conditioning indicators. They tell whether or not a certain file is to be used for a job. For example, you may have a job which one time requires the use of two output (or input) files and another time the use of only one.

Instead of writing two different programs (one using one file, the other two), you can condition a file (in columns 71-72 of the file description specifications sheet) by an external indicator. When the indicator is on, the file is used; when it is off, the file is not used.

If a file is conditioned by an external indicator, all output data handled by the file must also be conditioned by the same indicator. Any calculation operations which are not done when the file is not in use should also be conditioned by the same indicator.

In addition to using these indicators as file conditioning indicators, you can use them:

- To condition calculation operations.
- To condition output operations.
- As field record relation indicators (columns 63-64 of the input specifications sheet).

An RPG II program processes one record at a time. Normally only the information from the record being processed is available for use. However, the RPG II look ahead function enables you to use information from records that follow the one being processed. The fields containing the information are called look ahead fields.

LOOK AHEAD FIELDS

Look ahead can be used only with input or update files. Chained files and demand files cannot use look ahead. To use look ahead, you must describe the look ahead fields and reference them as you do normal fields. You can describe any or all of the fields in a record as look ahead fields. The description applies to all records in the file, regardless of their type. All record types should contain the same look ahead fields. (The specifications for describing the fields are given later.)

Look ahead fields always apply to the next record in the file provided the file is not an update file. Thus, if you want to use information both before and after the record is selected for processing, you must describe the field twice, once as a look ahead field and once as a normal field.

For update files the look ahead fields apply to the next record in the file only if the current record selected for processing was read from another file. Therefore, when you are reading from only one file and the file is an update file, look ahead fields always apply to the current record and contain the same information as a normal field.

Figure 102 shows the processing of three records from two input files, one primary and one secondary. The first record from each file is read (Figure 102, insert A). In Figure 102, insert B, record P1 is selected for processing; in Figure 102, insert C, record P2; and in Figure 102, insert D, record S1. The records available for look ahead during the processing of these records are:

Records Being Processed	Records Available For Look Ahead
P1	P2 and S1
P2	P3 and S1
S1	P3 and S2

In general, when the record being processed is from an input file, the next record in the input file is available as are the records which were read, but not selected from the other files.

Figure 103 shows the same files as Figure 102 with one exception: the primary file is an update file. The records available for look ahead during the processing of the three records are:

Records Being Processed	Records Available For Look Ahead
U1	U1 and S1
U2	U2 and S1
S1	U3 and S2

In general, when the record being processed is from an update file, only the records which were read, but not selected from the other files, are available for look ahead. The next record from the update file is not read until after the current record is processed. Therefore, the next record from the update file is not available for look ahead.

After the last record from a file is processed, every look ahead field for the file is automatically filled with 9s. For example, a field three record positions long contains 999. The 9s remain in the fields until the job ends. Note also that blank after (B in column 39 of the output specifications sheet) cannot be used with look ahead fields.

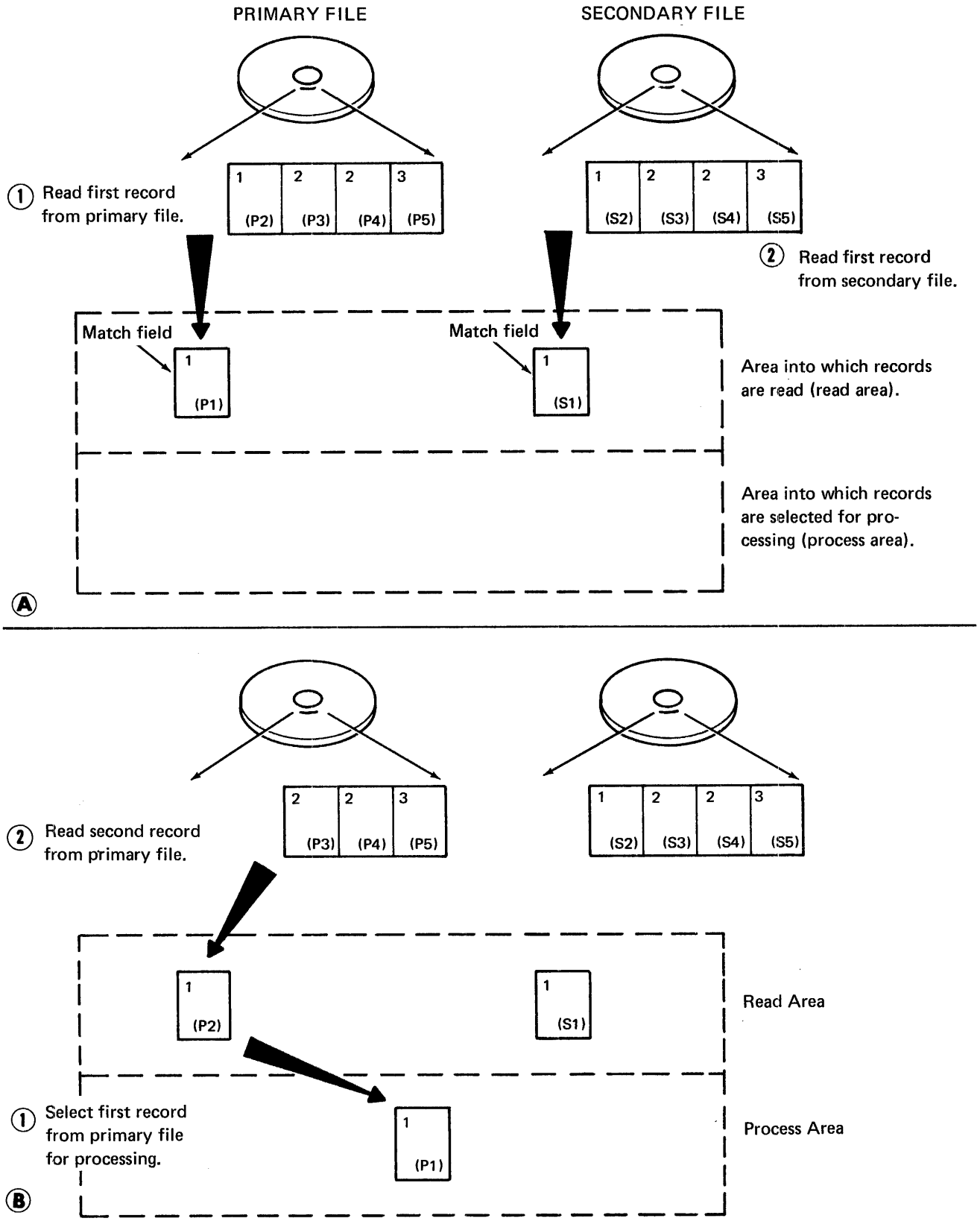


Figure 102 (Part 1 of 2). Available Records: Two Input Files

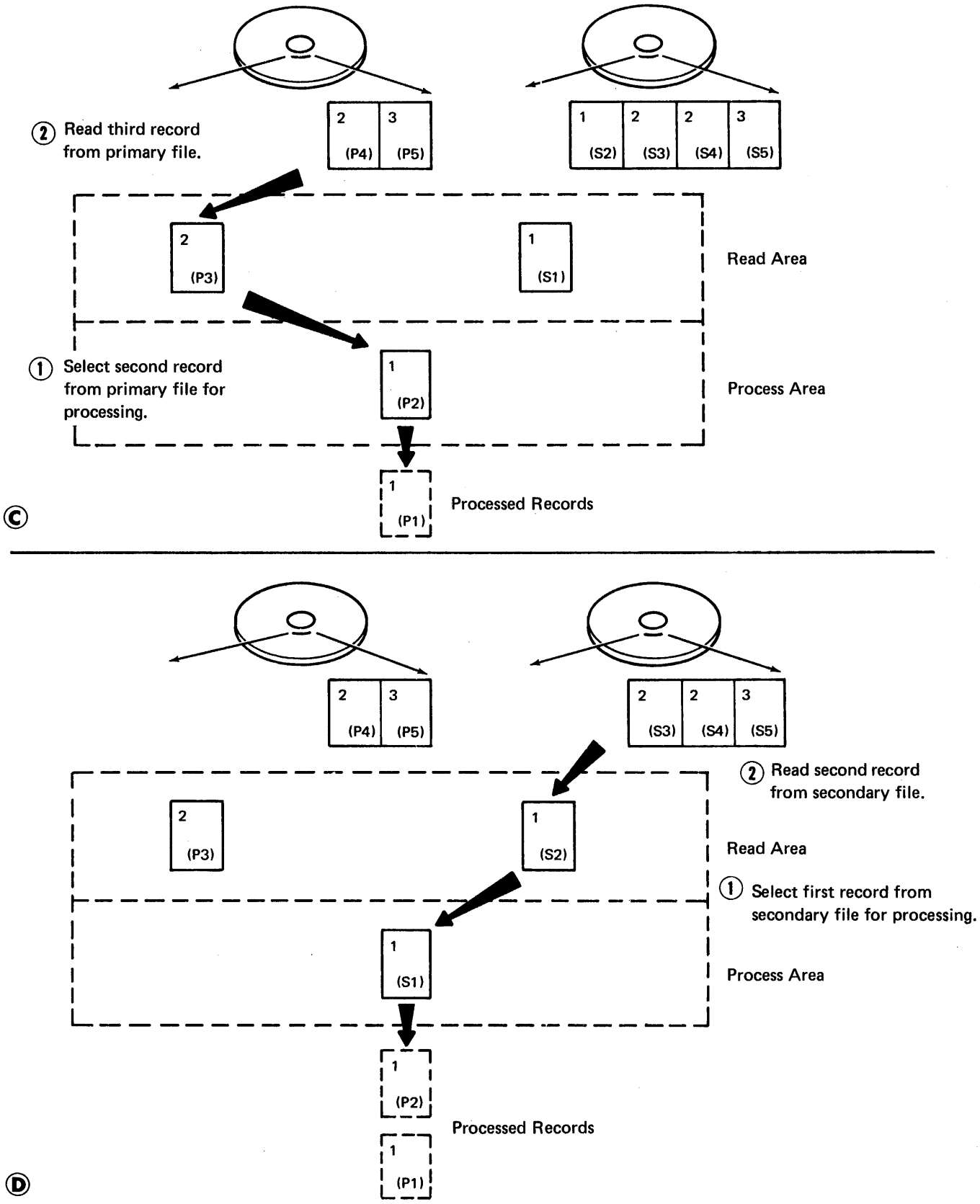


Figure 102 (Part 2 of 2). Available Records: Two Input Files

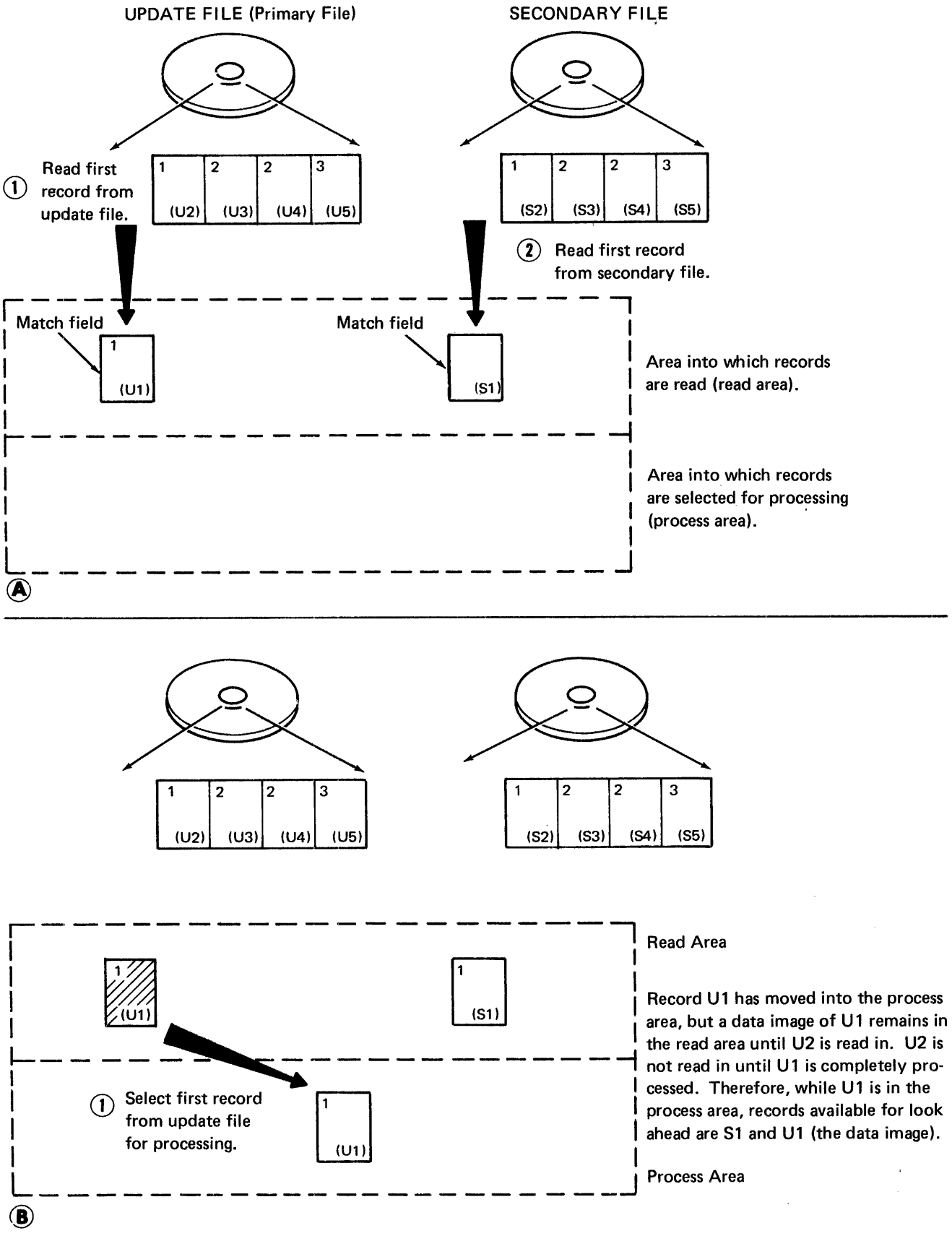


Figure 103 (Part 1 of 3). Available Records: One Input File, One Update File

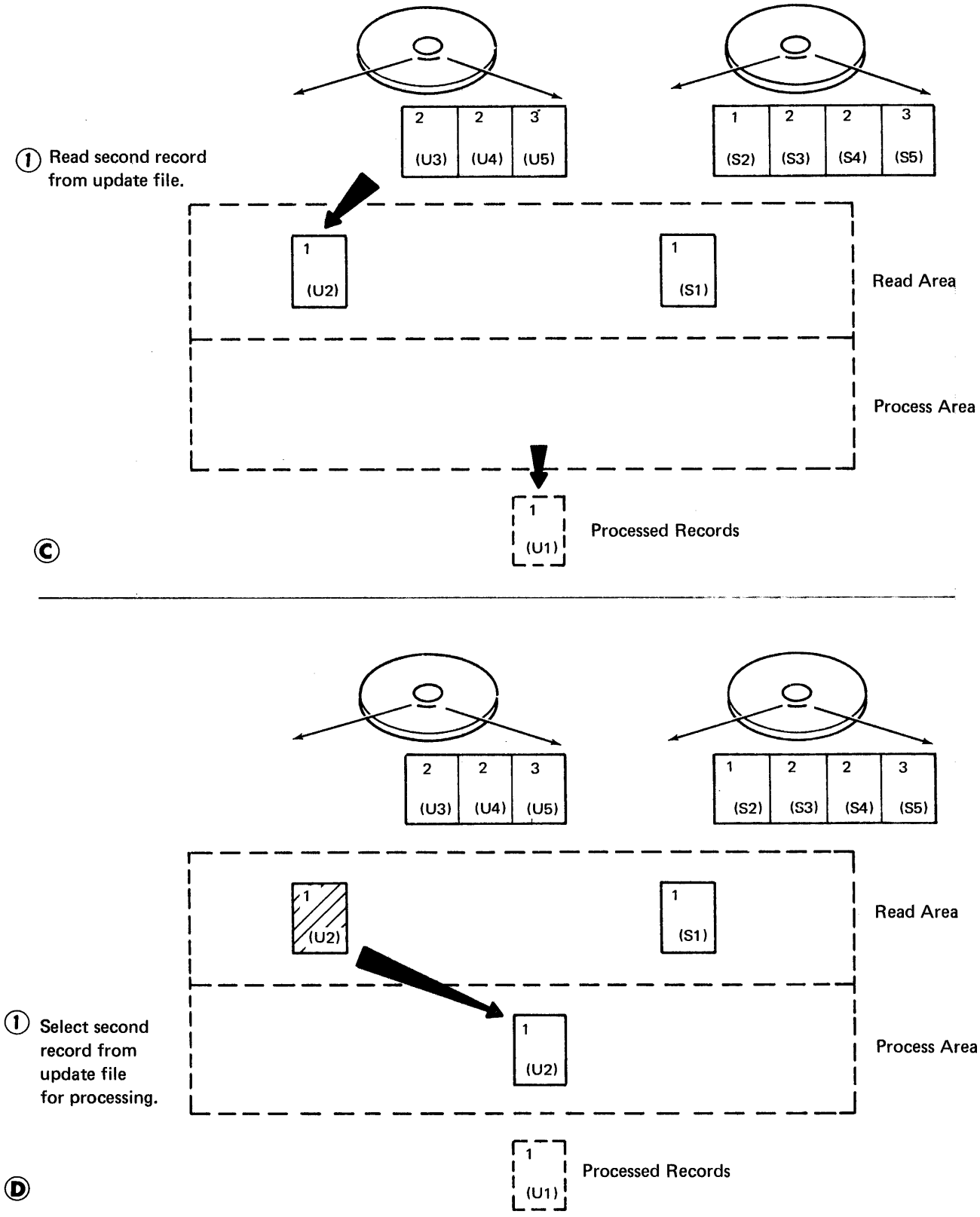


Figure 103 (Part 2 of 3). Available Records: One Input File, One Update File

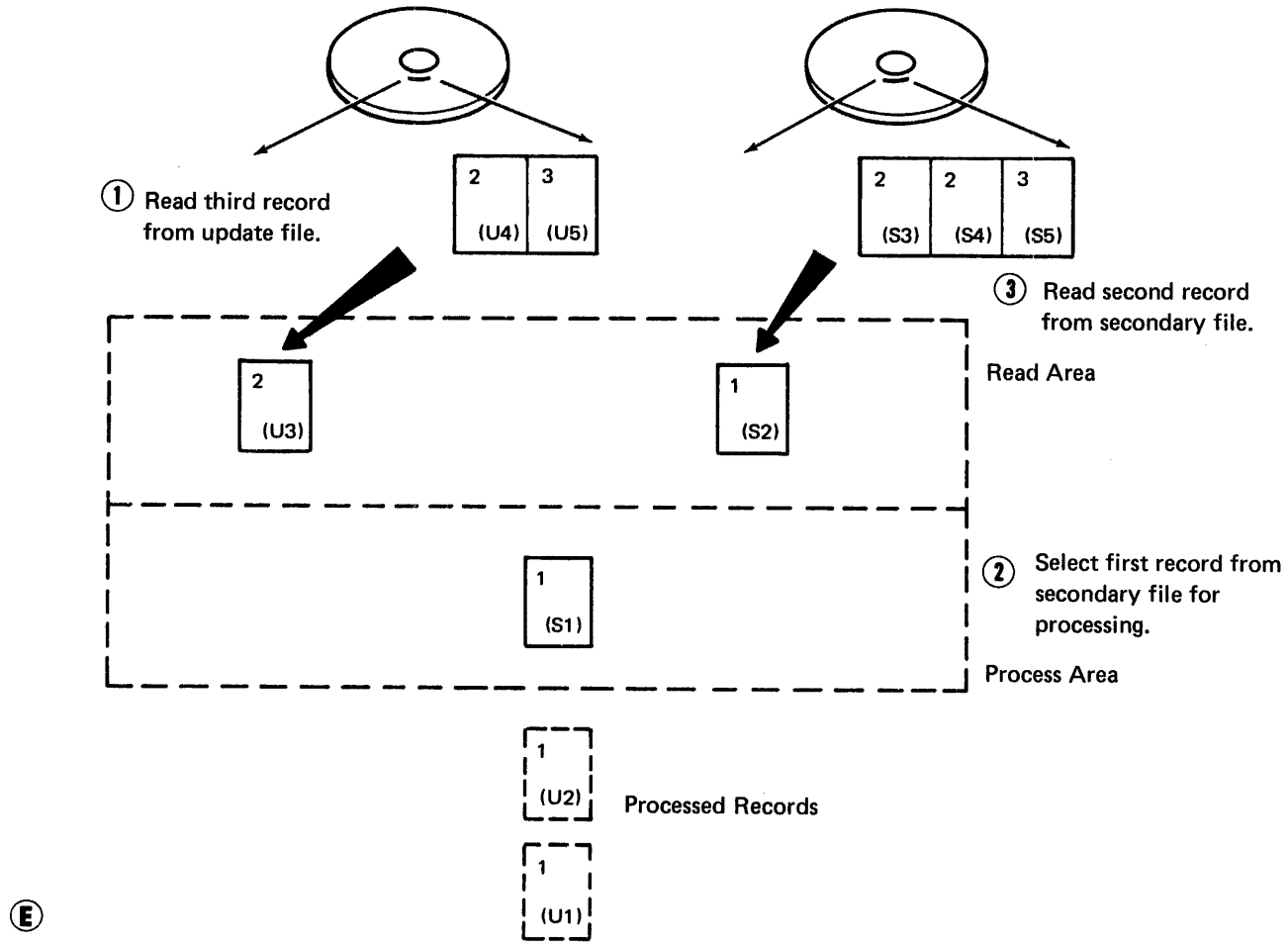


Figure 103 (Part 3 of 3). Available Records: One Input File, One Update File

SPECIFICATIONS

You can describe any or all of the fields in a record as look ahead fields. The description applies to all records in the file, regardless of their type. Look ahead fields must not be described for chained files or demand files and they must not be used as array fields. Describe look ahead fields on the input specifications sheet before or after the field descriptions for any of the records in the file. To indicate that look ahead fields are described in the following specification lines, enter ** in columns 19-20 of a specification line. Enter any alphabetic characters in columns 15-16 of that specification line. Leave columns 7-14, 17-18, and 21-74 blank.

Describe the look ahead fields on separate lines immediately following the ** line (as shown in Figure 104, insert B). Describe each field as follows:

- Leave columns 7-43 blank.
- In columns 44-51, identify the record positions in which the field is located.
- If the field is numeric, enter the number of digits to the right of the decimal point in column 52. If the field is not numeric, leave column 52 blank.
- In columns 53-58, identify the look ahead field by name. If the field is also one of your normal fields, be sure to use a different name here. Use this name to refer to the look ahead field.
- Leave columns 59-74 blank.

EXAMPLE

Figure 104 shows a job which reads records from two disk files. The primary file is named PRIMARY; the secondary file, SECONDRY. If a record from the primary file matches one from the secondary file, the information in positions 1-10 of the secondary file record is placed in positions 31-40 of the primary file record. When there is no match, a 6 is placed in position 1 of the primary file record. The 6 indicates an unmatched record in the primary file.

Because the primary file record is processed first when it matches a secondary file record, the information from the secondary file can only be made available by using a look ahead field.

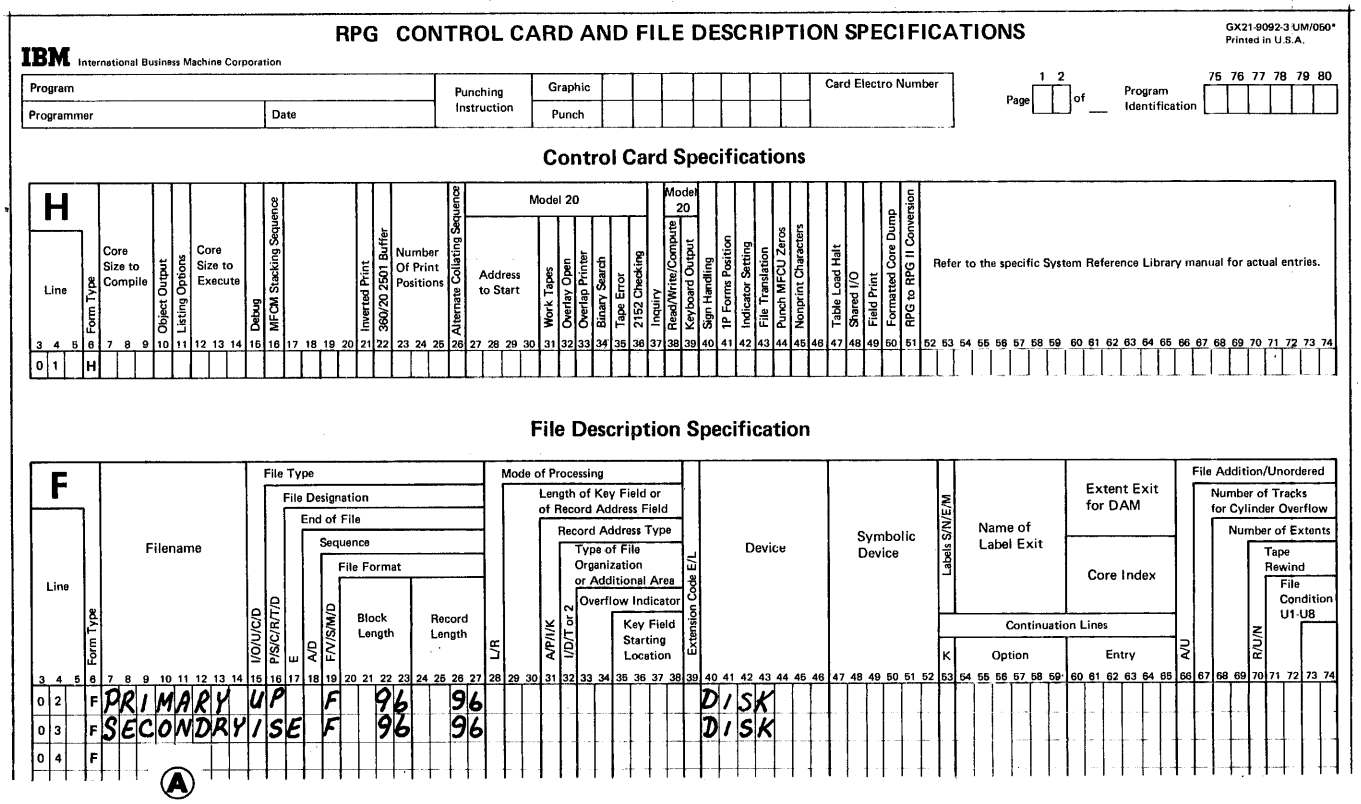
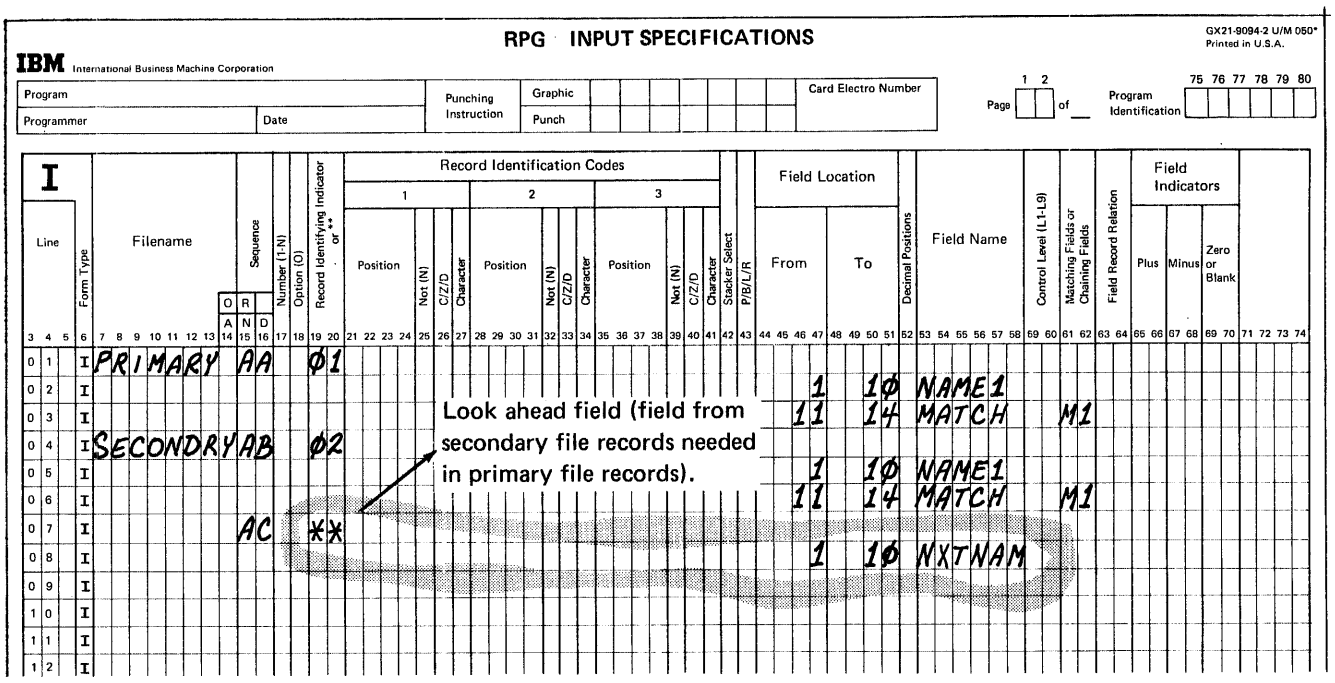
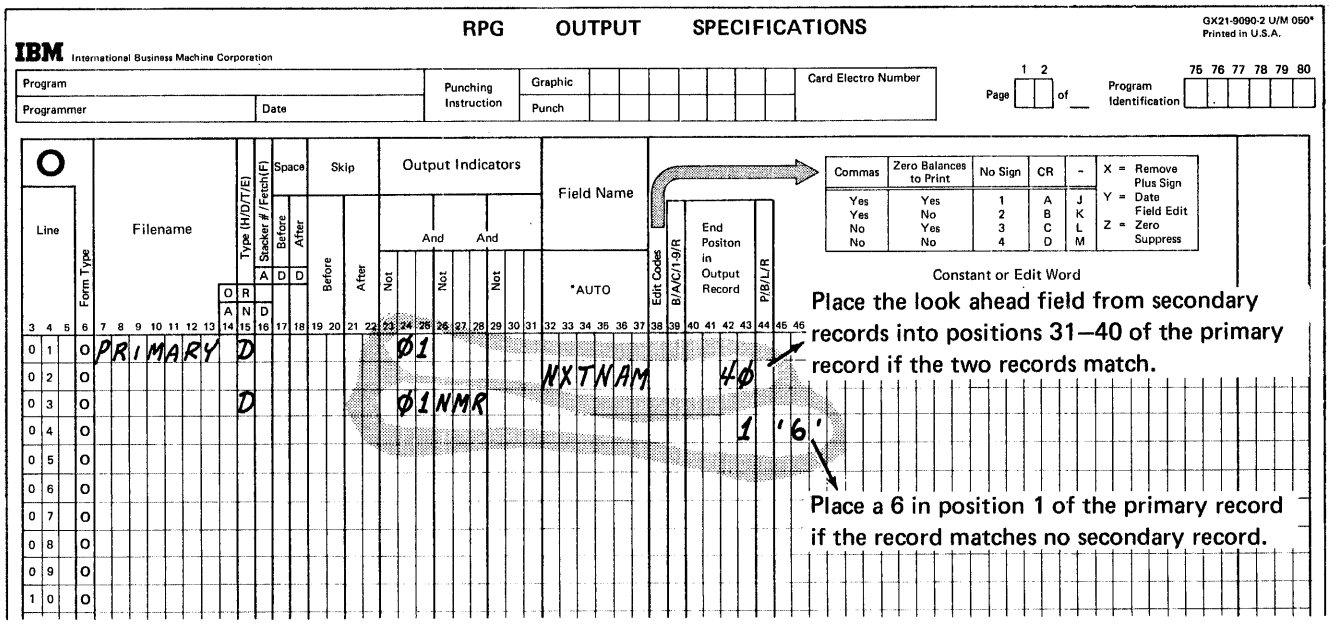


Figure 104 (Part 1 of 2). Look Ahead Fields



Ⓑ



Ⓒ

Figure 104 (Part 2 of 2). Look Ahead Fields

Multifile processing applies to programs that read records from a primary file and one or more secondary files. It is the name given to the methods by which the programs select records for processing. The method used depends upon whether or not match fields are used in the records.

NO MATCH FIELDS

When no match fields are used, records are selected from one file at a time. When the records from one file are all processed, the records from next file are selected. The files are chosen in this order:

1. Primary file.
2. Secondary files in the order in which they are described by the file description specifications.

MATCH FIELDS

When match fields are used, records are selected according to the contents of the match fields. One record is read from every file, and the match fields in the records are compared. If the records are in ascending order, the record with the lowest match field is selected for processing. If the records are in descending order, the record with the highest match field is selected.

When a record is selected from a file, the next record from the file is read. At the beginning of the next program cycle, the new record is compared with the other records in the read area which are waiting for selection, and one is selected.

Records without match fields can be included in the files. Such records are selected before records with match fields. If two or more of the records being compared have no match fields, selection of those records is determined by the priority of the files from which the records came. The priority of files is:

1. Primary file.
2. Secondary files in the order in which they are described by the file description specifications.

When the primary record matches one or more of the secondary records, the MR (matching record) indicator is turned on. The indicator can condition calculations or output for the record that is selected. When one of the matching records must be selected, the selection is determined by the priority of the files from which the records came.

Figure 105 shows when the MR indicator is turned on during the RPG II cycle. For more information on the RPG II cycle, see Part 2, Chapter 6, *RPG II Object Program Logic (Detailed).*

EXAMPLE

Figures 106-108 show the order of record selection from three files using match fields. In Figure 106, the file description specifications sheet shows a primary file and two secondary files.

On the input specifications sheet, two record types are described for each file: one type has a match field assigned match value M1, in the other type the field in the same position is not used as a match field. Figure 107, part 1, shows all the records from the three files. The circled numbers represent the order of selection for the 26 records shown.

Figure 107, part 2, provides a summary of the 26 cycles required to process the records shown in Figure 107, part 1. The summary tells what file is being processed in each cycle, the indicators that are on for the cycle, and a brief explanation of why a particular record was selected.

In Figure 108 a more thorough explanation is given for the selection of the first 10 records from the disk files shown in Figure 107, part 1. The reasons for selection of these 10 records cover most of the situations involved in multifile processing.

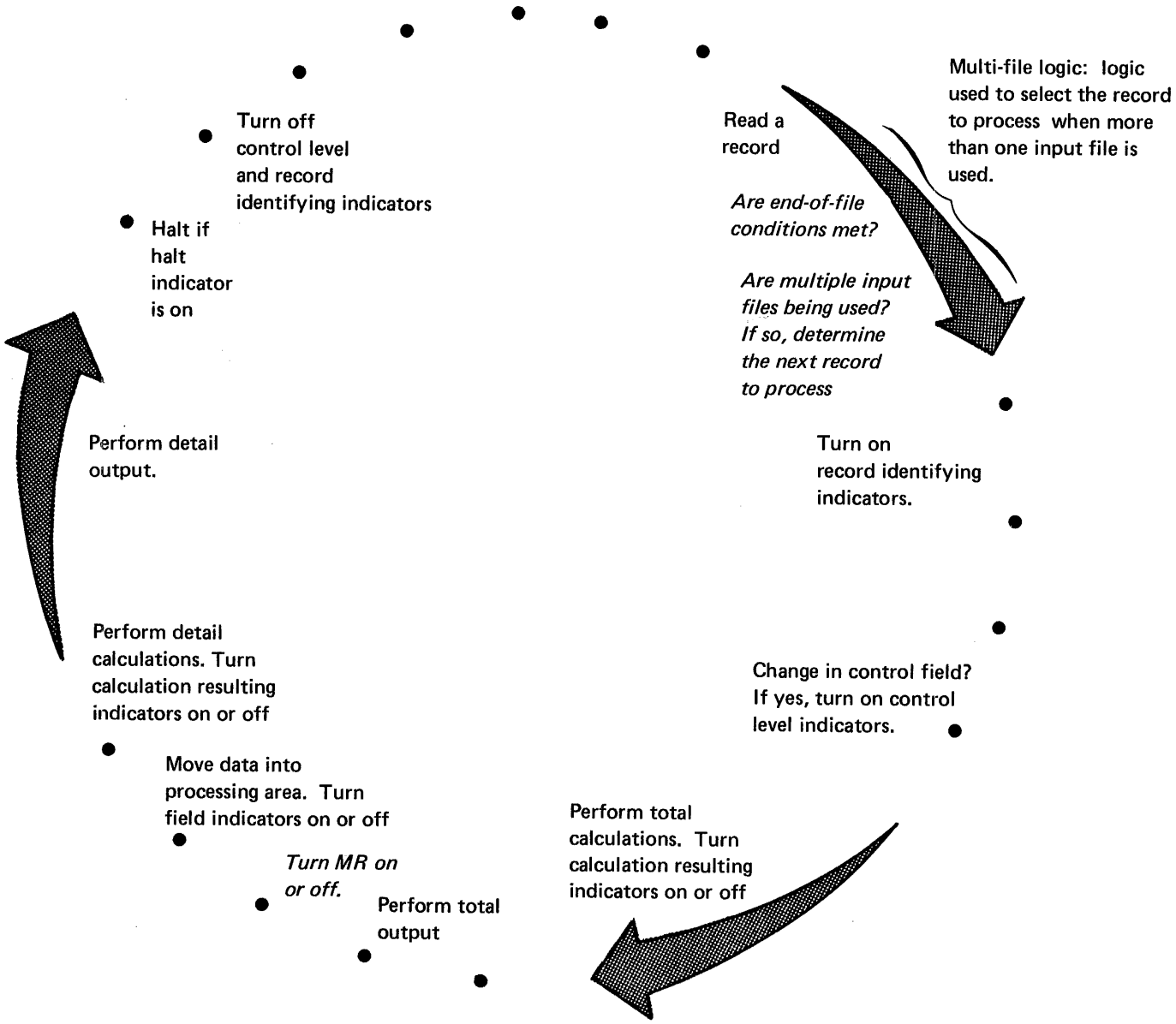


Figure 105. Simplified Matching Record Logic

File Description Specification

F

Line	Form Type	Filename	File Type I/O/U/C/D	P/S/C/R/T/D	File Designation E	End of File A/D	Sequence F/V/S/M/D	File Format Block Length	Record Length	Mode of Processing L/R	Length of Key Field or of Record Address Field AP/II/K	Record Address Type I/D/T or 2	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Core Index	File Addition/Unordered A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind R/U/N	File Condition U1-U8
																				Continuation Lines							
																				K	Option	Entry					
02	F	PRIMARY	I	P	E		A	F	64								DISK										
03	F	FIRSTSEC	I	S			A	F	64								DISK										
04	F	SECSEC	I	S			A	F	64								DISK										
05	F																										
06	F																										
07	F																										
08	F																										

RPG INPUT SPECIFICATIONS

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

IBM International Business Machine Corporation

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

Programmer | Date

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

I

Line	Form Type	Filename	Sequence OR AND	Number (1-N)	Option (O)	Record Identifying Indicator or *	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	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
01	I	PRIMARY	AA			01		1	C	P		2	Z	1																
02	I																				2	3	MATCH		M1				
03	I		BB			02		1	C	P		2	N	Z	1															
04	I																				2	3	NOM						
05	I																													
06	I	FIRSTSEC	AB			03		1	C	S		2		Z	1															
07	I																				2	3	MATCH		M1				
08	I		BC			04		1	C	S		2	N	Z	1															
09	I																				2	3	NOM						
10	I																													
11	I	SECSEC	AC			05		1	C	T		2		Z	1															
12	I																				2	3	MATCH		M1				
13	I		BD			06		1	C	T		2	N	Z	1															
14	I																				2	3	NOM						
15	I																													

Figure 106. Match Field Specifications for Three Disk Files

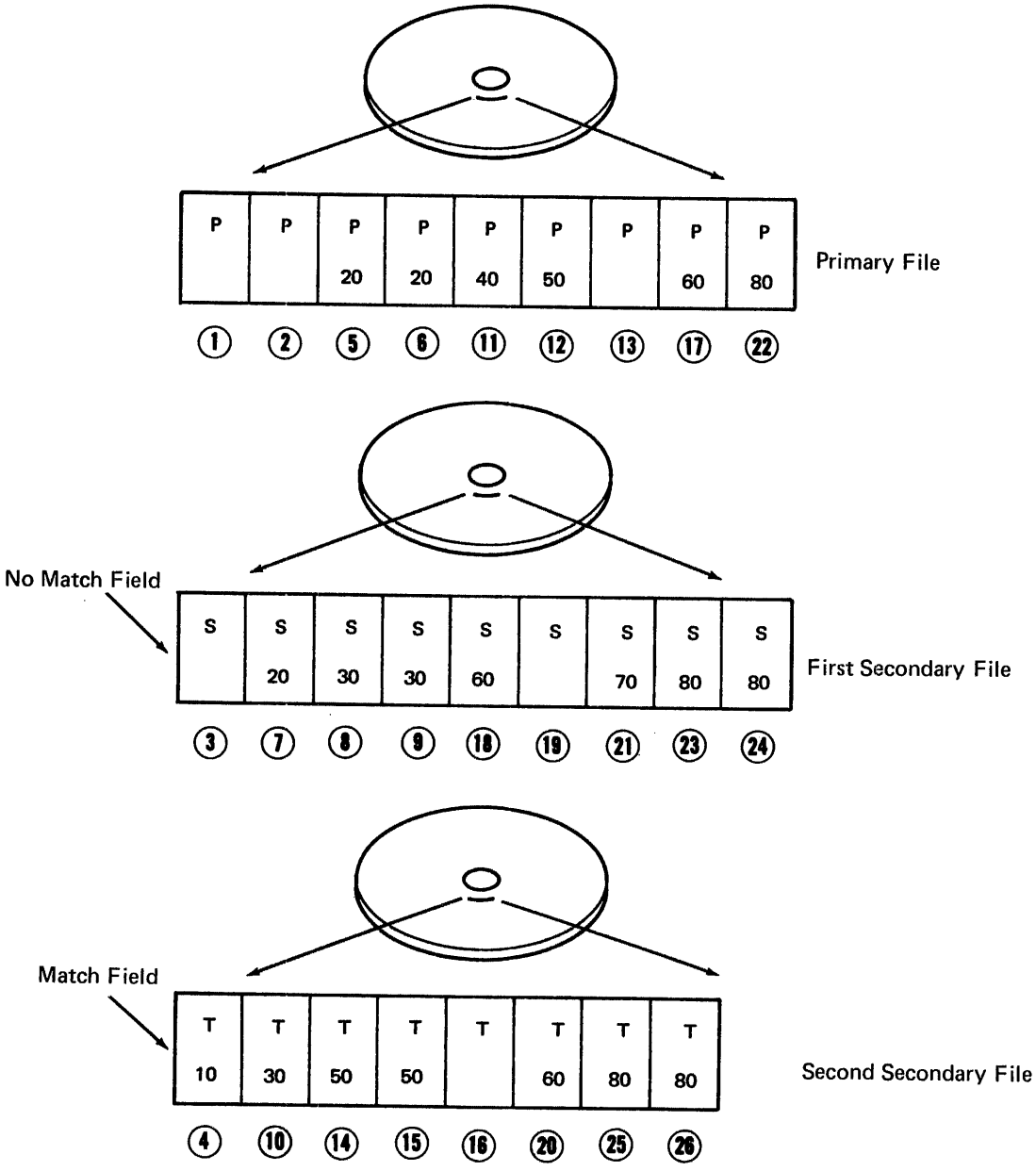


Figure 107 (Part 1 of 2). Normal Record Selection from Three Disk Files

Cycle	File Processed	Indicators On	Reason for Record Selection
1	PRIMARY	02	No match field specified
2	PRIMARY	02	No match field specified
3	FIRST SEC	04	No match field specified
4	SEC SEC	05	Second secondary low No primary match
5	PRIMARY	01, MR	Primary matches first secondary
6	PRIMARY	01, MR	Primary matches first secondary
7	FIRST SEC	03, MR	First secondary matches primary
8	FIRST SEC	03	First secondary low No primary match
9	FIRST SEC	03	First secondary low No primary match
10	SEC SEC	05	Second secondary low No primary match
11	PRIMARY	01	Primary low No secondary match
12	PRIMARY	01, MR	Primary matches second secondary
13	PRIMARY	02	No match field specified
14	SEC SEC	05, MR	Second secondary matches primary
15	SEC SEC	05, MR	Second secondary matches primary
16	SEC SEC	06	No match field specified
17	PRIMARY	01, MR	Primary matches both secondary files
18	FIRST SEC	03, MR	First secondary matches primary
19	FIRST SEC	04	No match field specified
20	SEC SEC	05, MR	Second secondary matches primary
21	FIRST SEC	03	First secondary low No primary match
22	PRIMARY	01, MR	Primary matches both secondary files
23	FIRST SEC	03, MR	First secondary matches primary
24	FIRST SEC	03, MR	First secondary matches primary
25	SEC SEC	05, MR	Second secondary matches primary
26	SEC SEC	05, MR	Second secondary matches primary

Figure 107 (Part 2 of 2). Normal Record Selection from Three Disk Files

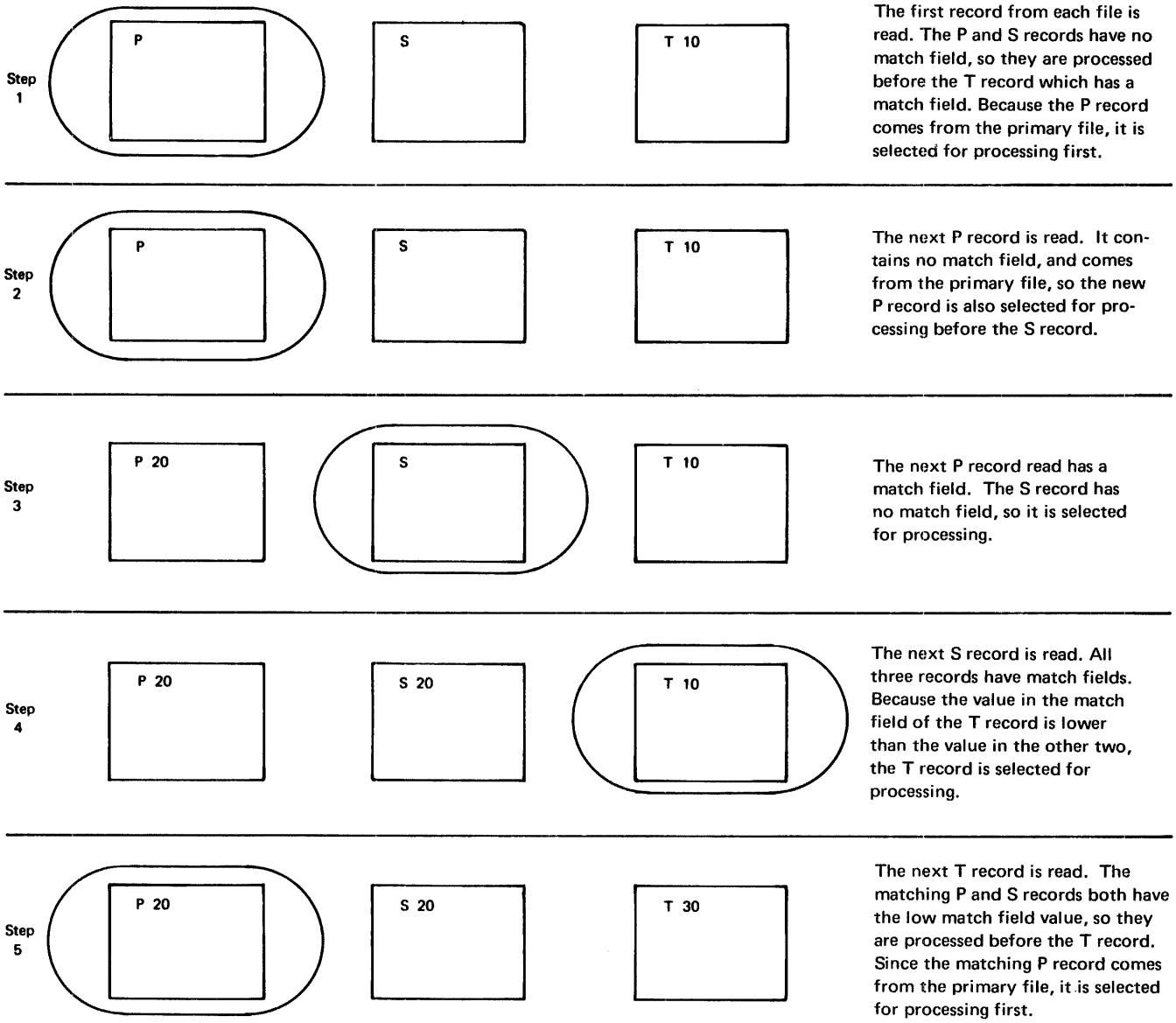


Figure 108 (Part 1 of 2). Normal Record Selection from Three Disk Files

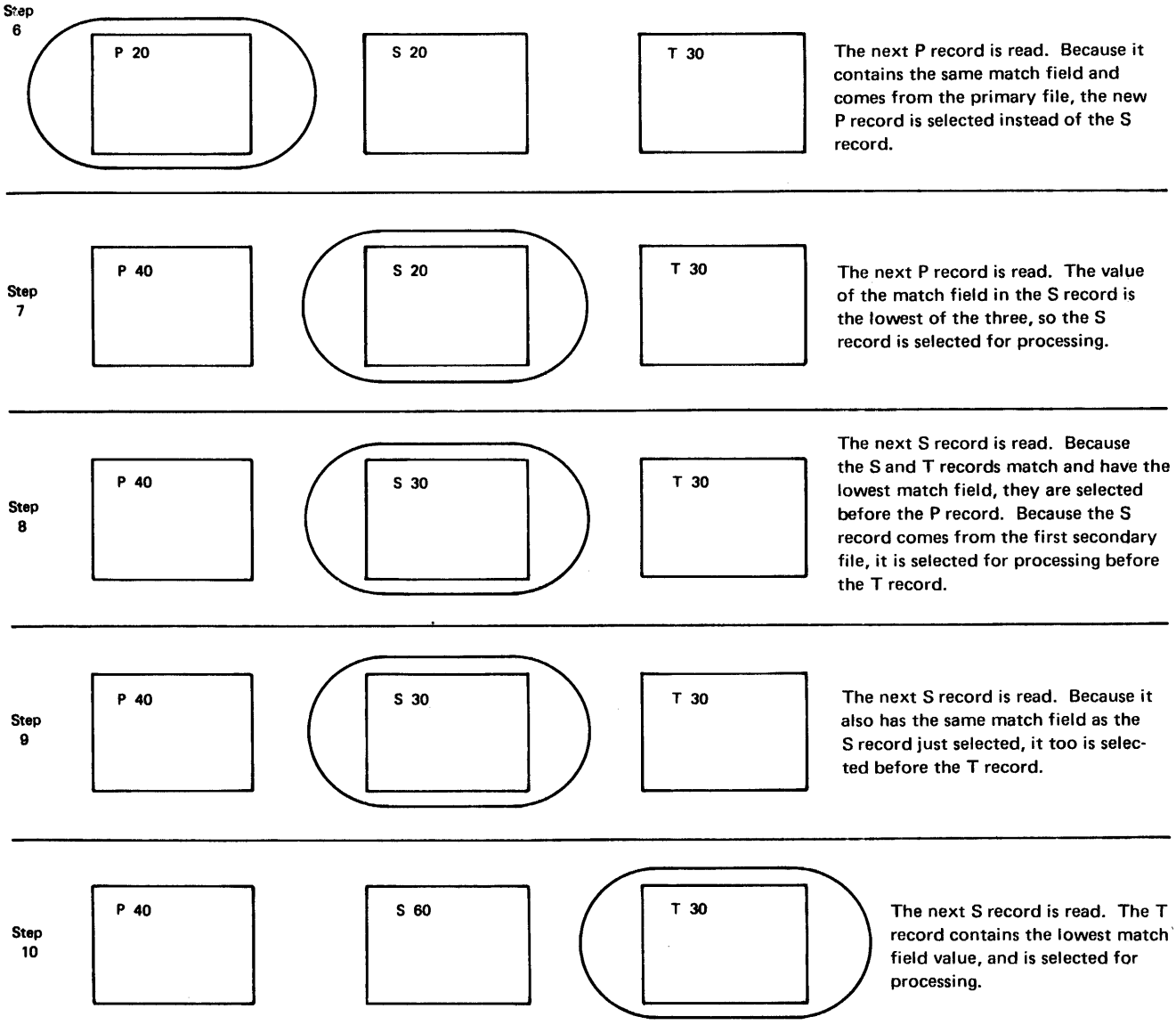


Figure 108 (Part 2 of 2). Normal Record Selection from Three Disk Files

You are able to perform many different types of operations on your data using the RPG II language. Special codes are set up which indicate the operation to be performed. Usually these are just abbreviations of the name of the operation. You must use these codes to specify the operation to be performed.

Operations can be divided into 12 categories; all codes in each category are explained in this section. Examples are also given for many codes. Figure 109 provides a summary of the operation codes. It also shows what other specifications need to be used with each code.

ARITHMETIC OPERATIONS

Arithmetic operations can be performed only on numeric fields or literals. The result field must also be numeric. Decimal alignment is performed for all arithmetic operations. Even though truncation may occur, the position of the decimal point in the result field is not affected. For arithmetic operations in which all three fields are used:

- Factor 1, factor 2, and the result field may all be different fields.
- Factor 1, factor 2, and the result field may all be the same field.
- Factor 1 and factor 2 may be the same field but different from the result field.
- Either factor 1 or factor 2 may be the same as the result field.

The length of any field involved in an arithmetic operation cannot exceed 15 characters. If the result exceeds 15 characters, characters may be dropped from either or both ends depending on the location of the decimal point. The results of all operations are signed (+ or -). Any data placed in the result field replaces the data that was there previously.

Add (ADD)

Factor 2 is added to factor 1. The sum is placed in the result field. Factor 1 and factor 2 are not changed by the operation.

Zero and Add (Z-ADD)

Factor 2 is added to a field of zeros. The sum is placed in the result field. Factor 1 is not used.

Subtract (SUB)

Factor 2 is subtracted from factor 1. The difference is placed in the result field. Factor 1 and factor 2 are not changed by the operation.

Note: Subtracting two fields which are the same is a method of setting the result field to zero.

Zero and Subtract (Z-SUB)

Factor 2 is subtracted from a field of zeros. The difference is placed in the result field. This actually places the negative of factor 2 in the result field. This operation can be used to change the sign of a field. Factor 1 is not used.

Multiply (MULT)

Factor 1 is multiplied by factor 2. The product is then placed in the result field. Factor 1 and factor 2 are not changed. You must be sure the result field is large enough to hold the product. To determine the minimum result field length, use this rule: result field length equals the length of factor 1 plus the length of factor 2 plus one.

Type of Operation	Function of Operation	Operation Code (Columns 28-32)	Control Level	Indicators	Factor 1	Factor 2	Result Field	Field Length	Decimal Position	Half Adjust	Resulting Indicators
Arithmetic Operations	Add factor 2 to factor 1.	ADD	O	O	R	R	R	O	O	O	O
	Clear result field and add factor 2.	Z-ADD	O	O	B	R	R	O	O	O	O
	Subtract factor 2 from factor 1.	SUB	O	O	R	R	R	O	O	O	O
	Clear result field and subtract factor 2.	Z-SUB	O	O	B	R	R	O	O	O	O
	Multiply factor 1 by factor 2.	MULT	O	O	R	R	R	O	O	O	O
	Divide factor 1 by factor 2	DIV	O	O	R	R	R	O	O	O	O
	Move remainder of preceding division to a result field.	MVR	O	O	B	B	R	O	O	O	O
	Sum elements of an array and put sum in result field.	XFOOT	O	O	B	R	R	O	O	O	O
	Derive the square root of factor 2.	SQRT	O	O	B	R	R	O	O	O	B
Move Operation	Move factor 2 into result field, right-justified.	MOVE	O	O	B	R	R	O	O	B	B
	Move factor 2 into result field, left-justified.	MOVEL	O	O	B	R	R	O	O	B	B
	Move factor 2 into result field, left-justified.	MOVEA	O	O	B	R	R	O	B	B	B
Move Zone Operations	Move zone from low-order position of factor 2 to low-order position of result field.	MLLZO	O	O	B	R	R	O	O	B	B
	Move zone from high-order position of alphameric factor 2 to high-order of alphameric result field.	MHHZO	O	O	B	R	R	O	B	B	B
	Move zone from low-order position of factor 2 to high-order position of alphameric result field.	MLHZO	O	O	B	R	R	O	B	B	B
	Move zone from high-order position of alphameric factor 2 to low-order position of result field.	MHLZO	O	O	B	R	R	O	O	B	B
Compare and Zone Testing Operations	Compare factor 1 to factor 2.	COMP	O	O	R	R	B	B	B	B	R
	Identify the zone in the leftmost position of an alphameric result field.	TESTZ	O	O	B	B	R	O	B	B	R
Bit Operations	Set on specified bits.	BITON	O	O	B	R	R	O	B	B	B
	Set off specified bits.	BITOF	O	O	B	R	R	O	B	B	B
	Test specified bits.	TESTB	O	O	B	R	R	O	B	B	R
Setting Indicators	Set one, two, or three specific indicators on.	SETON	O	O	B	B	B	B	B	B	R
	Set one, two, or three specific indicators off.	SETOF	O	O	B	B	B	B	B	B	R
Branching Within RPG II	Branch to another RPG II calculation specification line.	GOTO	O	O	B	R	B	B	B	B	B
	Identify the name in factor 1 as a destination label to which GOTO may branch.	TAG	O	B	R	B	B	B	B	B	B
Branching to External Subroutines	Branch to user-written assembler subroutine.	EXIT	O	O	B	R	B	B	B	B	B
	Transfer data to user-written assembler subroutine.	RLABL	B	B	B	B	R	O	O	B	B
Look up Operations	Table look up.	LOKUP	O	O	R	R	O	O	O	B	R
	Array look up.	LOKUP	O	O	R	R	B	B	B	B	R

Figure 109 (Part 1 of 2). Operation Codes

Type of Operation	Function of Operation	Operation Code (Columns 28-32)	Control Level	Indicators	Factor 1	Factor 2	Result Field	Field Length	Decimal Position	Half Adjust	Resulting Indicators
Subroutine	Beginning of the subroutine.	BEGSR	[1]	B	R	B	B	B	B	B	B
	End of the subroutine.	ENDSR	[1]	B	O	B	B	B	B	B	B
	Call to execute the subroutine.	EXSR	O	O	B	R	B	B	B	B	B
Program Control of Input and Output	Pause for input data from keyboard.	KEYnn[2]	O	O	O	B	R	O	O	B	O
	Control of console buffer clear and command key specification.	SETnn[2]	O	O	O	O	B	B	B	B	O
	Call for immediate input.	READ	O	O	B	R	B	B	B	B	[3]
	Forcing record to be read on next cycle.	FORCE	B	O	B	R	B	B	B	B	B
	Forcing output printing.	EXCPT	O	O	B	B	B	B	B	B	B
	Sets lower limits for indexed sequential files being processed within limits.	SETLL	O	O	R	R	B	B	B	B	B
	A record is read from a disk file.	CHAIN	O	O	R	R	B	B	B	B	[4]
Debug Function	Aid in finding programming errors.	DEBUG	O	O	O	R	O	B	B	B	B

O — Optional
R — Required
B — Blank

[1] Columns 7-8 must have an SR entry for all subroutine lines.

[2] The nn entries in columns 31-32 are for message indicator (MIC) numbers.

[3] Columns 58-59 may contain an indicator for this operation; columns 54-57 must be blank.

[4] A resulting indicator should be entered in columns 54-55; columns 56-59 must be blank.

Figure 109 (Part 2 of 2). Operation Codes

Divide (DIV)

Factor 1 (dividend) is divided by factor 2 (divisor). The quotient (result) is placed in the result field. Factor 1 and factor 2 are not changed.

If factor 1 is 0, the result of the divide operation is 0. Factor 2 cannot be 0. If it is, the job stops immediately. You can continue processing, however, by responding to the halt. When processing is continued, the result and remainder are set to zero.

Any remainder resulting from the divide operation is lost unless the move remainder operation (MVR) is specified as the next operation. If move remainder is the next operation, the result of the divide operation cannot be half-adjusted (rounded).

Move Remainder (MVR)

This operation moves the remainder from the previous divide operation to a separate field named under result field. Factor 1 and factor 2 must not be used. This operation must immediately follow the divide operation.

The maximum length of the remainder (including decimal positions) is 15. The number of significant decimal positions is the greater of:

- The number of decimal positions in factor 1 of the previous divide operation.
- The sum of the decimal positions in factor 2 and the result field of the previous divide operation.

The maximum whole number positions in the remainder is equal to the whole number positions in factor 2 of the previous divide operation. Figure 110 shows the use of the move remainder operation.

RPG CALCULATION SPECIFICATIONS

IBM International Business Machine Corporation

Form GX21-9093-2 Printed in U.S.A.

Program		Punching Instruction	Graphic Punch	Card Electro Number
Programmer	Date			

Page ___ of ___ Program Identification

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And	Indicators: And	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Result Field: Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus	Minus	Zero	Resulting Indicators: Compare: 1>2	1<2	1=2	Resulting Indicators: Lookup(Factor 2)Is: High	Low	Equal	Comments
01	C		14	N02	FIELDA	DIV	FIELDB	SAVE		60											
02	C		14	N02		MVR		STORE		60											
03	C																				
04	C																				
05	C																				

Figure 110. Move Remainder Operation

Square Root (SQRT)

This operation derives the square root of the field named in factor 2. The square root of factor 2 is placed in the result field. Do not use factor 1.

A whole array can be used in a SQRT operation if factor 2 and result field contain array names.

The number of decimal places in the result field can be either less than or greater than the number of decimal places in factor 2. However, the result field should not have less than half the number of decimal places in factor 2. The result of a SQRT operation is always half-adjusted.

If the value of the factor 2 field is negative, the job halts. You can continue processing by responding to the halt. When processing is continued, the result field is set to zero.

Summing The Elements Of An Array (XFOOT)

This operation is used only on numeric arrays. It adds all the elements of the array together and puts the sum into a separate field specified as the result field. Factor 1 is not used. Factor 2 contains the name of the array.

MOVE OPERATIONS

Move operations move part or all of factor 2 to the result field. Factor 2 remains unchanged, but the result field is changed.

Factor 1 is not used in any move operations. It must always be blank. No resulting indicators can be used. Numeric fields can be changed to alphameric fields and alphameric fields can be changed to numeric fields by the move operations. To change a numeric field to an alphameric field, place the name of the numeric field in the factor 2 columns and use an alphameric result field. To change an alphameric field to a numeric field, place the name of the alphameric field in the factor 2 columns and use a numeric result field.

When move operations are specified to move data into numeric fields, decimal positions are ignored. For example, if the data 1.00 is moved into a numeric field with one decimal position, the result is 10.0.

Move (MOVE)

This operation causes characters from factor 2 to be moved to the rightmost positions in the result field. Moving starts with the rightmost characters.

If factor 2 is longer than the result field, the excess leftmost characters of factor 2 are not moved. If the result field is longer than factor 2, the excess leftmost characters in the result field are unchanged.

An alphameric field or constant can be changed into a numeric field by moving it into a numeric field. When this is specified, the digit portion of each character is converted to its corresponding numeric character and then moved to the result field. Blanks are transferred as zeros. However, the zone portion of the rightmost alphameric character is converted to a corresponding sign and is moved to the rightmost position of the numeric field where it becomes the sign of the field. A numeric field can also be changed into an alphameric field by moving it into an alphameric field. The MOVE operation is summarized in Figure 111.

Factor 2 Shorter Than Result Field

	Factor 2		Result Field	
a. Alphameric	P H 4 S N	Before MOVE Operation	1 2 3 4 5 6 7 8 +4	Alphameric
	P H 4 S N	After MOVE Operation	1 2 3 4 P H 4 S N	
b. Alphameric	P H 4 S N	Before MOVE Operation	1 2 3 4 5 6 7 8 +4	Numeric
	P H 4 S N	After MOVE Operation	1 2 3 4 7 8 4 2 −5	
c. Numeric	1 2 7 8 4 2 5	Before MOVE Operation	1 2 3 4 5 6 7 8 9	Numeric
	1 2 7 8 4 2 5	After MOVE Operation	1 2 1 2 7 8 4 2 5	
d. Numeric	1 2 7 8 −4 2 5	Before MOVE Operation	A C F G P H 4 S N	Alphameric
	1 2 7 8 4 2 5	After MOVE Operation	A C 1 2 7 8 4 2 5	

Factor 2 Longer Than Result Field

	Factor 2		Result Field	
a. Alphameric	A C E G P H 4 S N	Before MOVE Operation	5 6 7 8 4	Alphameric
	A C E G P H 4 S N	After MOVE Operation	P H 4 S N	
b. Alphameric	A C E G P H 4 S N	Before MOVE Operation	5 6 7 8 +4	Numeric
	A C E G P H 4 S N	After MOVE Operation	7 8 4 2 −5	
c. Numeric	1 2 7 8 4 2 5	Before MOVE Operation	5 6 7 4 8	Numeric
	1 2 7 8 4 2 5	After MOVE Operation	7 8 4 2 5	
d. Numeric	1 2 7 8 4 2 5	Before MOVE Operation	P H 4 S N	Alphameric
	1 2 7 8 4 2 5	After MOVE Operation	7 8 4 2 5	

Factor 2 and Result Field Same Length

	Factor 2		Result Field	
a. Alphameric	P H 4 S N	Before MOVE Operation	5 6 7 8 4	Alphameric
	P H 4 S N	After MOVE Operation	P H 4 S N	
b. Alphameric	P H 4 S N	Before MOVE Operation	5 6 7 8 4	Numeric
	P H 4 S N	After MOVE Operation	7 8 4 2 −5	
c. Numeric	7 8 4 2 5	Before MOVE Operation	A L T 5 F	Numeric
	7 8 4 2 5	After MOVE Operation	7 8 4 2 −5	
d. Numeric	7 8 4 2 −5	Before MOVE Operation	A L T 5 F	Alphameric
	7 8 4 2 −5	After MOVE Operation	7 8 4 2 N	

$\overset{+}{4}$ = letter D

$\overset{-}{5}$ = letter N

Figure 111. MOVE Operations

Move Left (MOVEL)

This operation causes characters from factor 2 to be moved to the leftmost positions in the result field. Moving begins with the leftmost characters.

If factor 2 is longer than the result field, the excess rightmost characters of factor 2 are not moved. If the result field is longer than factor 2, the excess rightmost characters in the result field are unchanged. In this case the sign of a numeric field is not changed either.

An alphameric field or constant can be changed into a numeric field by moving it into a numeric result field. When this is specified, the digit portion of each character is converted to its corresponding numeric character and then moved into the result field.

Blanks are transferred as zeros. If the rightmost character is moved, the zone is also converted and used as the sign of the field. When the rightmost character is not transferred, the zone is, nevertheless, still transferred and used as the sign of the result field.

A numeric field can also be changed into an alphameric field by moving it into an alphameric field. Both digit and zone portions of the rightmost character are transferred if that character is to be moved. A summary of rules for MOVEL transfers are as follows (Figure 112).

Factor 2 is the same length as the result field.

- Factor 2 and result field numeric: the sign is moved with the rightmost position.
- Factor 2 numeric, result field alphameric: the sign is moved with the rightmost position.
- Factor 2 alphameric, result field numeric: if the zone from the rightmost position of factor 2 is a D (minus zone), a minus zone is moved into the rightmost position of the result field. If the zone from the rightmost position of factor 2 is not a D, a positive zone is moved into the rightmost position of the result field. Digit portions are converted to their corresponding numeric characters.
- Factor 2 and result field alphameric: all characters are moved.

Factor 2 is longer than the result field.

- Factor 2 and result field numeric: the sign from the rightmost position of factor 2 is moved into the rightmost position of the result field.
- Factor 2 numeric, result field alphameric: the result field contains only numeric characters.
- Factor 2 alphameric, result field numeric: if the zone from the rightmost position of factor 2 is a D (minus zone), a minus zone is moved into the rightmost position of the result field. If the zone from the rightmost position of factor 2 is not a D, a positive zone is moved into the rightmost position of the result field. Other result field positions contain only numeric characters.
- Factor 2 and result field alphameric: only the number of characters needed to fill the result field are moved.

Factor 2 is shorter than the result field.

- Factor 2 either numeric or alphameric, result field numeric: digit portion of factor 2 replaces the contents of the leftmost positions of the result field. The sign in the rightmost position of the result field is not changed.
- Factor 2 either numeric or alphameric, result field alphameric: characters in factor 2 replace the equivalent number of leftmost positions in the result field. No change is made in the zone of the rightmost position of the result field.

Factor 2 and Result Field Same Length

	Factor 2		Result Field	
a. Numeric	7 8 4 2 5	Before MOVEL Operation	5 6 7 8 4	Numeric
	7 8 4 2 5	After MOVEL Operation	7 8 4 2 5	
b. Numeric	7 8 4 2 5	Before MOVEL	A K T 4 D	Alphameric
	7 8 4 2 5 (5=letter N)	After MOVEL	7 8 4 2 N	
c. Alphameric	P H 4 S N	Before MOVEL	5 6 7 8 4	Numeric
	P H 4 S N	After MOVEL	7 8 4 2 5	
d. Alphameric	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	

Factor 2 Longer Then Result Field

	Factor 2		Result Field	
a. Numeric	0 0 0 0 0 8 4 2 5	Before MOVEL Operation	5 6 7 8 4	Numeric
	0 0 0 0 0 8 4 2 5	After MOVEL Operation	0 0 0 0 0	
b. Numeric	9 0 3 1 7 8 4 2 5	Before MOVEL	A K T 4 D	Alphameric
	9 0 3 1 7 8 4 2 5	After MOVEL	9 0 3 1 7	
c. Alphameric	B R W C X H 4 S N	Before MOVEL	5 6 7 8 4	Numeric
	B R W C X H 4 S N	After MOVEL	2 9 6 3 7	
d. Alphameric	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	

Factor 2 Shorter Than Result Field

		Factor 2		Result Field	
a.	Numeric	7 8 4 2 5	Before MOVEL Operation	1 3 0 9 4 3 2 1 0	Numeric
		7 8 4 2 5	After MOVEL Operation	7 8 4 2 5 3 2 1 0	
	Alphameric	C P T 5 N	Before MOVEL	1 3 0 9 4 3 2 1 0	Numeric
		C P T 5 N	After MOVEL	3 7 3 5 5 3 2 1 0	
b.	Numeric	7 8 4 2 5	Before MOVEL	B R W C X H 4 S A	Alphameric
		7 8 4 2 5	After MOVEL	7 8 4 2 N H 4 S A	
	Alphameric	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	

The arrow ↓ between numbers indicates a decimal point.

Figure 112. MOVEL Operations

MOVE ZONE OPERATIONS

These operations are used only to move the zone portion of a character. There are four varieties of the move zone operation (Figure 113).

Using a minus (-) sign in a move zone operation does not yield a negative character in the result field, since minus is represented by a X'60' internally, and a D zone is required for a negative character. Characters J-R have D zone representations, and can be used to obtain a negative value (J=X'D1', . . . ,R=X'D9').

Note: Whenever the word *high* is used, the field involved must be alphameric; whenever *low* is used, the field involved can be either alphameric or numeric.

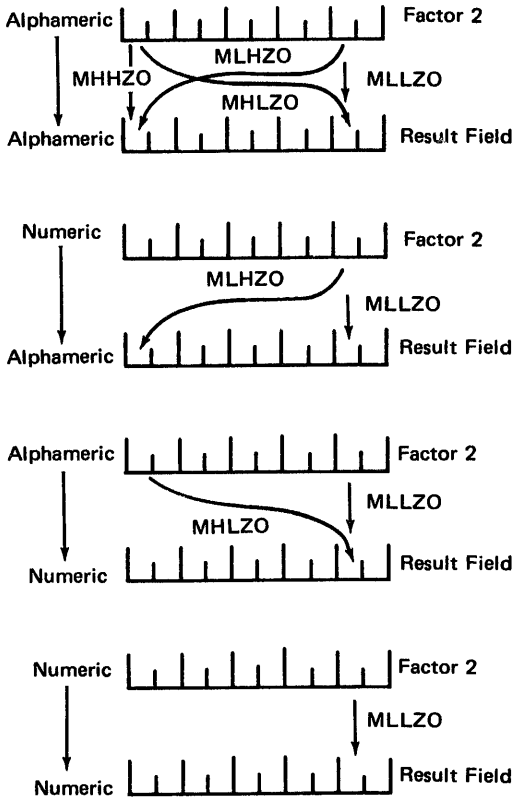


Figure 113. Function of Move Zone Operations

Move High to High Zone (MHHZO)

This operation moves the zone from the leftmost position of factor 2 to the leftmost position of the result field. Factor 2 and the result field must be alphameric.

Move High to Low Zone (MHLZO)

This operation moves the zone from the leftmost position of factor 2 to the rightmost position of the result field. Factor 2 can be only alphameric. The result field can be either alphameric or numeric.

Move Low to Low Zone (MLLZO)

This operation moves the zone from the rightmost position of factor 2 to the rightmost position of the result field. Factor 2 and the result field can be either alphameric or numeric.

Move Low to High Zone (MLHZO)

This operation moves the zone from the rightmost position of factor 2 to the leftmost position of the result field. Factor 2 can be numeric or alphameric, but the result field can only be alphameric.

MOVE ARRAY OPERATION (MOVEA)

The MOVEA operation causes characters from the leftmost positions of factor 2 to be moved to the leftmost positions of the result field. The length of the move is determined by the shorter of the lengths of factor 2 and the result field. If factor 2 is longer than the result field, the excess rightmost characters of factor 2 are not moved; if the result field is longer than factor 2, the rightmost characters in the result field are unchanged.

Factor 2 and/or the result field must reference an alphameric array; however, both cannot reference the same array (regardless of whether the array is indexed). If a field is used with the MOVEA operation, that field must be defined as alphameric.

The MOVEA operation makes it possible to:

- Move several contiguous array elements to a single field.
- Move a single field to several contiguous array elements.
- Move contiguous elements of one array to contiguous elements of another array.

Movement of data starts with the first element of an array (if the array is not indexed) or with the element referenced (if the array is indexed). The movement of data is terminated when the last array element is moved or filled, or when the number of characters moved equals the length of the shorter field specified by factor 2 and the result field. Note that this could cause the move to terminate in the middle of an array element. Figure 114 illustrates the use of the MOVEA operation.

Example: Array to array move. No indexing; different length arrays, same element length.

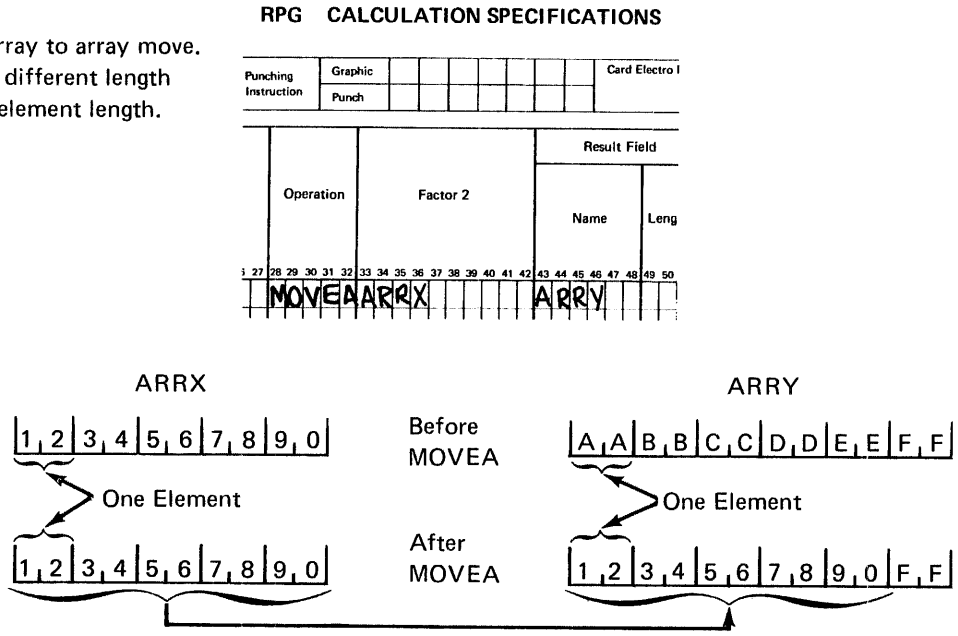
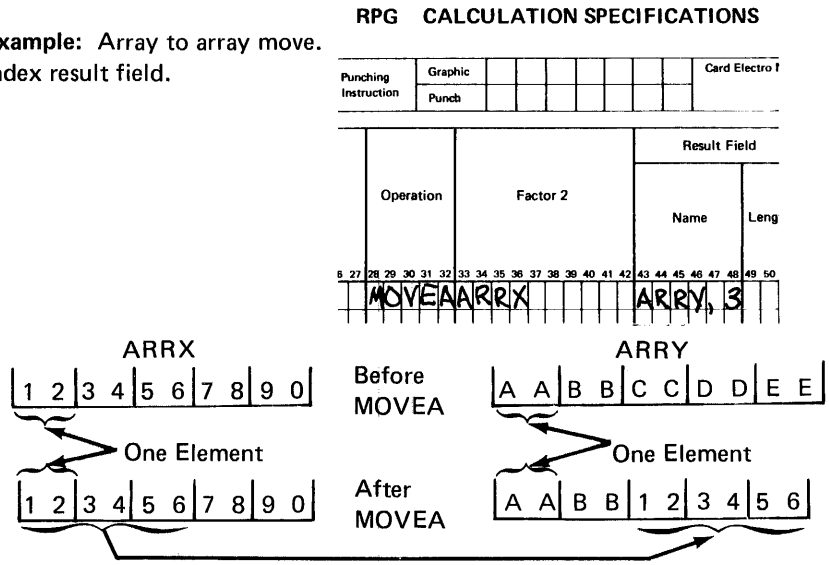
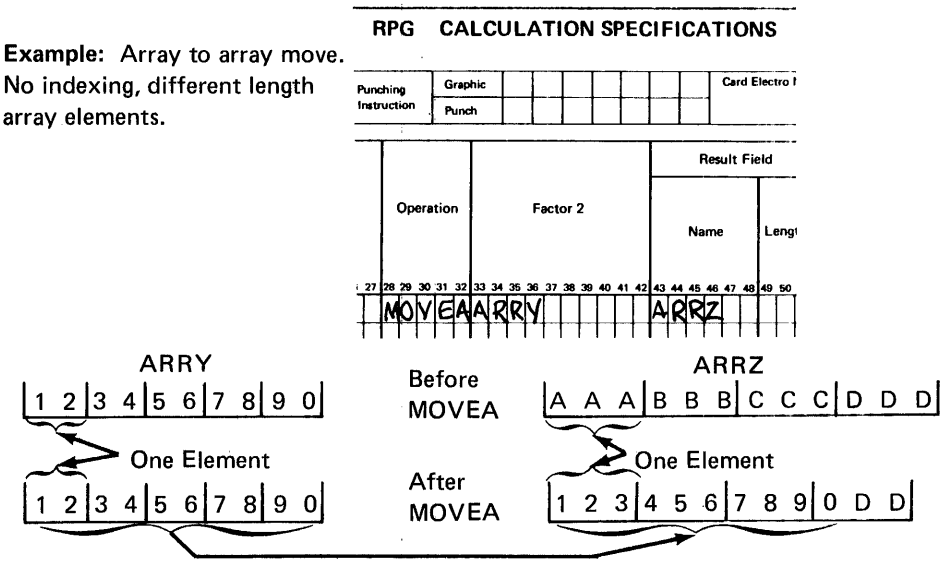


Figure 114 (Part 1 of 3). MOVEA Operation

Example: Array to array move. Index result field.



Example: Array to array move. No indexing, different length array elements.



Example: Array to array move. Index factor 2, different length array elements.

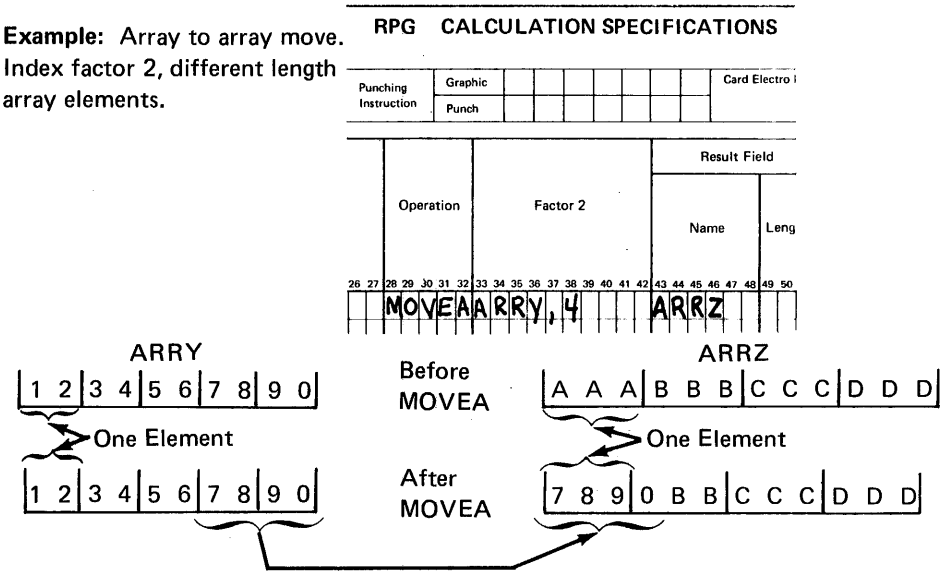
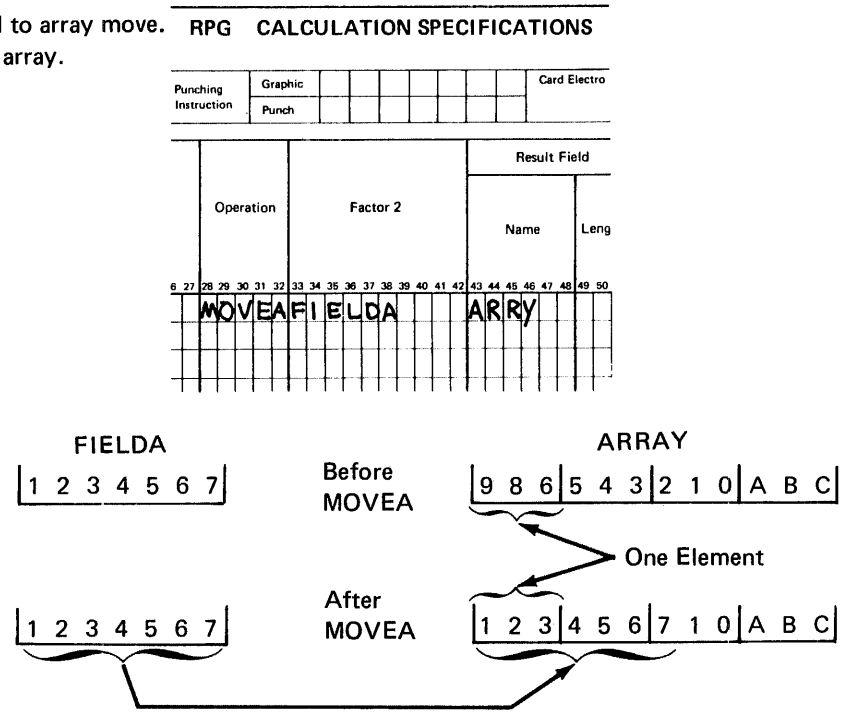


Figure 114 (Part 2 of 3). MOVEA Operation

Example: Field to array move. No indexing on array.



Example: Array to field move. Variable indexing.

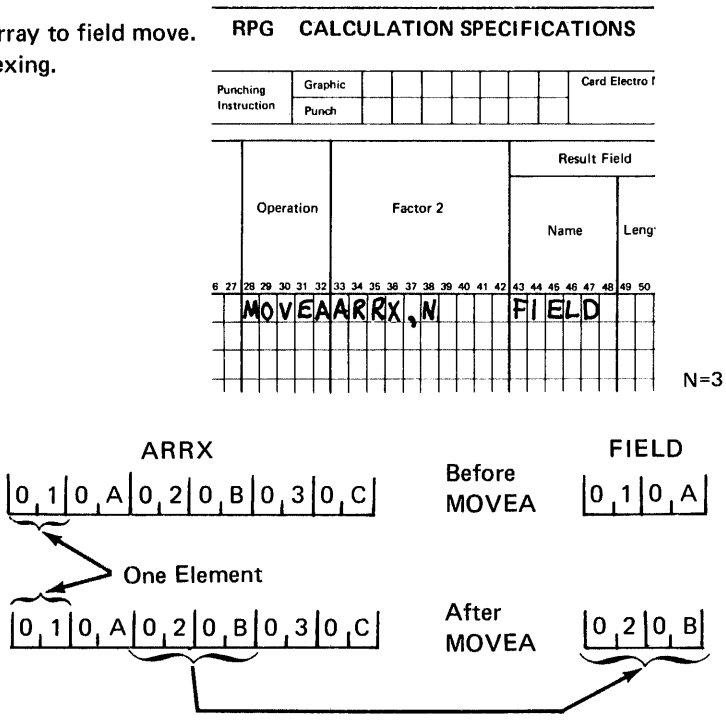


Figure 114 (Part 3 of 3). MOVEA Operation

COMPARE AND TESTING OPERATIONS

These operations test fields for certain conditions. The result of the test is shown by the resulting indicators assigned in columns 54-59. No fields are changed by these operations.

Compare (COMP)

This operation causes factor 1 to be compared with factor 2. As a result of the compare, indicators are turned on as follows:

High	Factor 1 is greater than factor 2.
Low	Factor 1 is less than factor 2.
Equal	Factor 1 equals factor 2.

Factor 1 and factor 2 must either be both alphameric or both numeric.

The fields are automatically aligned before they are compared. If the fields are alphameric, they are aligned to their leftmost character. If one is shorter, the unused positions are filled with blanks (Figure 115). The maximum field length for alphameric fields which are to be compared is 256 characters.

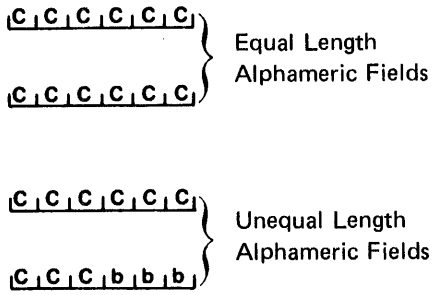


Figure 115. Comparision of Alphameric Fields

If the fields which are to be compared are numeric, they are aligned according to the decimal point. Any missing digits are filled with zeros (Figure 116). The maximum field length for numeric fields which are to be compared is 15 digits.

If an alternate collating sequence is defined, alphameric fields are compared according to the alternate sequence.

Figure 117 shows some specifications for compare operations.

Line 01: The contents of the field SLS69 (1969 sales) are compared with the contents of SLS70. If 1969 sales exceed 1970 sales, resulting indicator 21 turns on; if they are less, indicator 26 turns on; if the two years had equal sales, indicator 30 turns on.

Line 03: The alphameric constant OCTOBER is compared against the contents of the field named MONTH (which must also be defined as alphameric). If the MONTH field does not contain the word OCTOBER, indicator 13 turns on; if it does, indicator 15 turns on after the compare operation.

Line 05: The contents of the field named GRSPAY (which must be defined as numeric) is decimal-aligned with numeric constant 1250.00. If the value in field GRSPAY is greater than or equal to 1250.00, indicator 04 turns on; if its value is less than 1250.00, indicator 05 turns on.

Line 07: The contents of the field NETPAY (which must be defined as numeric) is decimal-aligned with numeric constant 0 and then compared to it. If NETPAY is greater than zero, indicator H1 remains off after the compare operation. If NETPAY is zero or negative, indicator H1 turns on.

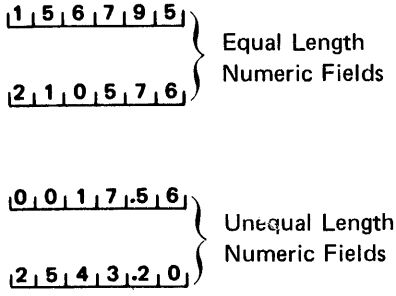
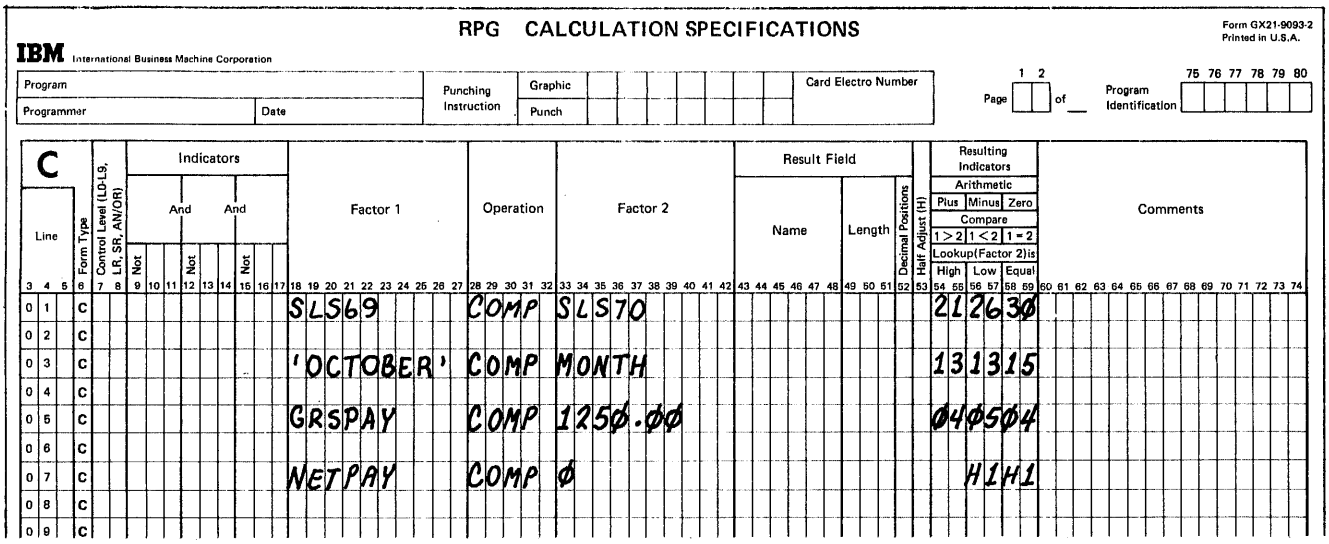


Figure 116. Comparison of Numeric Fields



RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic	Card Electro Number
Programmer	Date		Punch	

Page ___ of ___ Program Identification

C

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / And	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus / Minus / Zero; Compare: 1>2 / 1<2 / 1=2; Lookup(Factor 2)is: High / Low / Equal	Comments
01	C			SLS69	COMP	SLS70					212630	
02	C			'OCTOBER'	COMP	MONTH					131315	
03	C			GRSPAY	COMP	1250.00					040504	
04	C			NETPAY	COMP	0					H1H1	
05	C											
06	C											
07	C											
08	C											
09	C											

Figure 117. Compare Operations

Test Zone (TESTZ)

This operation tests the zone of the leftmost character in the result field. The result field must be alphameric since this operation can be done only on alphameric characters. Resulting indicators are used to determine the results of the test. The zone portion of characters &, A-I causes the plus indicator to turn on. The zone portion of the characters - (minus), J-R causes the minus indicator to turn on. All other characters, when tested, cause the zero indicator (columns 58-59) to turn on. Factor 1 and factor 2 are not used in this operation.

BIT OPERATIONS

Three operation codes BITON, BITOF, and TESTB set and test individual bits. You can use the individual bits as switches in a program, thus allowing a saving of storage for binary type switches.

Note: When using the BITON, BITOF, and TESTB operations, any field named in factor 2 or result field must be a 1-position alphameric field. A field is considered alphameric if there are no entries in the decimal positions columns on input or calculation specifications sheet.

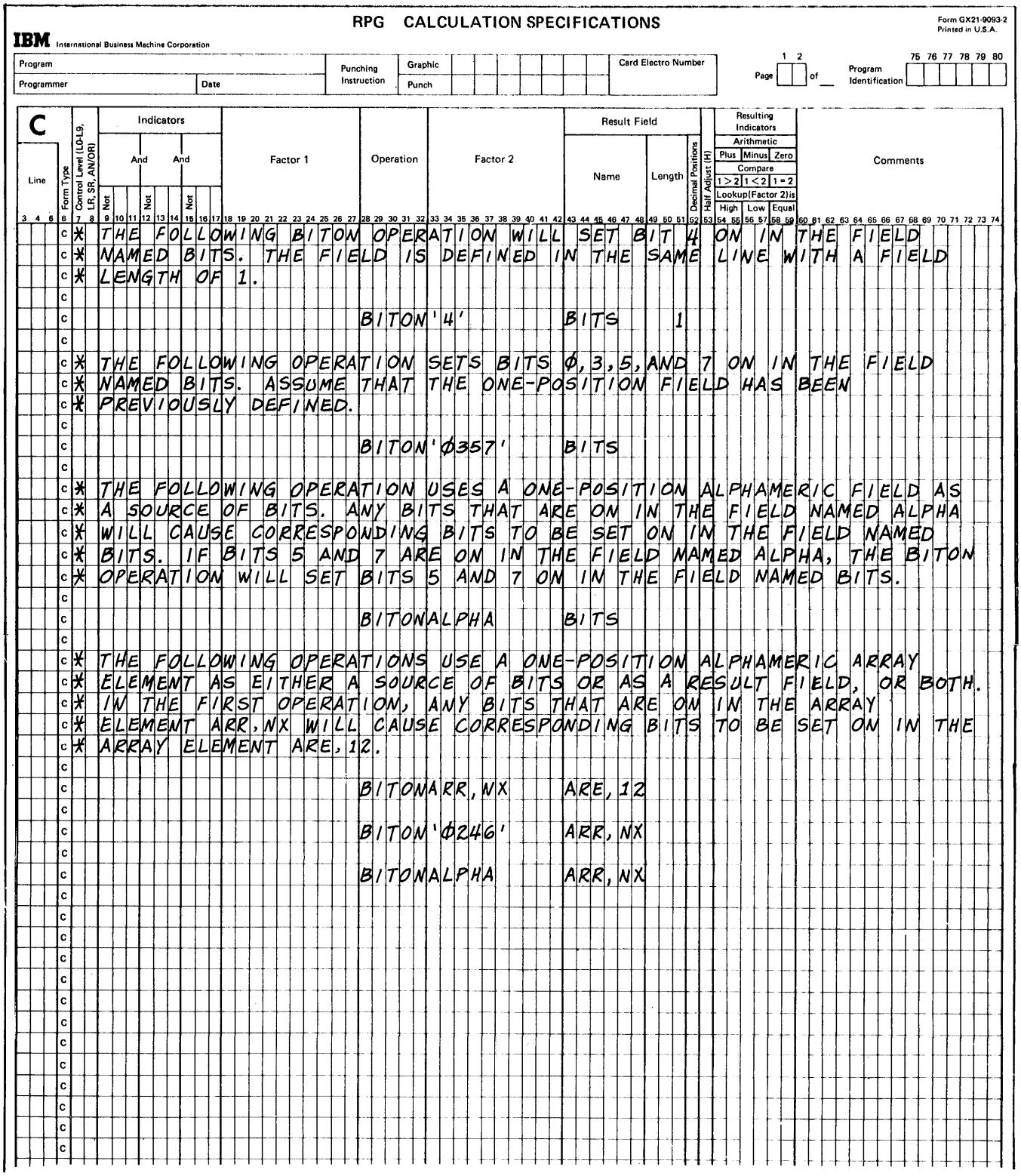
Set Bit On (BITON)

This operation code causes bits identified in factor 2, to turn on (set to 1) in a field named as the result field. Factor 2 is always a source of bits for the result field. The bits to be turned on can be entered as literals in factor 2, or can be contained in a field named in factor 2. If literal values are entered in factor 2, the bits to be turned on are identified by the numbers 0-7 (0 is the first bit number). The bit numbers must be enclosed by apostrophes, and the entry must begin in column 33. From one to eight bits can be identified in factor 2 for a BITON operation. If a field name is entered in factor 2, the field must be a 1-position alphameric field. The bits that are on in the factor 2 field are turned on in the result field. The field in factor 2 can be an array element if each element in the array is a 1-position alphameric element. When factor 2 is a field name or array element, apostrophes are not required, and the entry begins in column 33.

The result field is the field in which the bits are turned on. The field named here must be a 1-position alphameric field. The result field can be an array element, if each element in the array is a 1-position alphameric element. See Figure 118 for a summary of the BITON operation.

The operation code BITON must appear in columns 28-32. Conditioning indicators can be used in columns 7-17, and field length must be 1.

Factor 1, decimal positions, half adjust, and the resulting indicator columns must be blank.



RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

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

C

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus, Minus, Zero); Compare (1>2, 1<2, 1=2); Lookup(Factor 2)is (High, Low, Equal)	Comments
	C	*		THE FOLLOWING BITON OPERATION WILL SET BIT 4 ON IN THE FIELD								
	C	*		NAMED BITS. THE FIELD IS DEFINED IN THE SAME LINE WITH A FIELD								
	C	*		LENGTH OF 1.								
	C											
	C				BITON	'4'	BITS	1				
	C											
	C	*		THE FOLLOWING OPERATION SETS BITS Ø,3,5,AND 7 ON IN THE FIELD								
	C	*		NAMED BITS. ASSUME THAT THE ONE-POSITION FIELD HAS BEEN								
	C	*		PREVIOUSLY DEFINED.								
	C											
	C				BITON	'Ø357'	BITS					
	C											
	C	*		THE FOLLOWING OPERATION USES A ONE-POSITION ALPHAMERIC FIELD AS								
	C	*		A SOURCE OF BITS. ANY BITS THAT ARE ON IN THE FIELD NAMED ALPHA								
	C	*		WILL CAUSE CORRESPONDING BITS TO BE SET ON IN THE FIELD NAMED								
	C	*		BITS. IF BITS 5 AND 7 ARE ON IN THE FIELD NAMED ALPHA, THE BITON								
	C	*		OPERATION WILL SET BITS 5 AND 7 ON IN THE FIELD NAMED BITS.								
	C											
	C				BITONALPHA		BITS					
	C											
	C	*		THE FOLLOWING OPERATIONS USE A ONE-POSITION ALPHAMERIC ARRAY								
	C	*		ELEMENT AS EITHER A SOURCE OF BITS OR AS A RESULT FIELD, OR BOTH.								
	C	*		IN THE FIRST OPERATION, ANY BITS THAT ARE ON IN THE ARRAY								
	C	*		ELEMENT ARR,NX WILL CAUSE CORRESPONDING BITS TO BE SET ON IN THE								
	C	*		ARRAY ELEMENT ARE,12.								
	C											
	C				BITONARR,NX		ARE,12					
	C											
	C				BITON	'Ø246'	ARR,NX					
	C											
	C				BITONALPHA		ARR,NX					

Figure 118. Summary of BITON Operations

Set Bit Off (BITOF)

This operation code causes bits identified in factor 2, to turn off (set to 0) in a field named under result field.

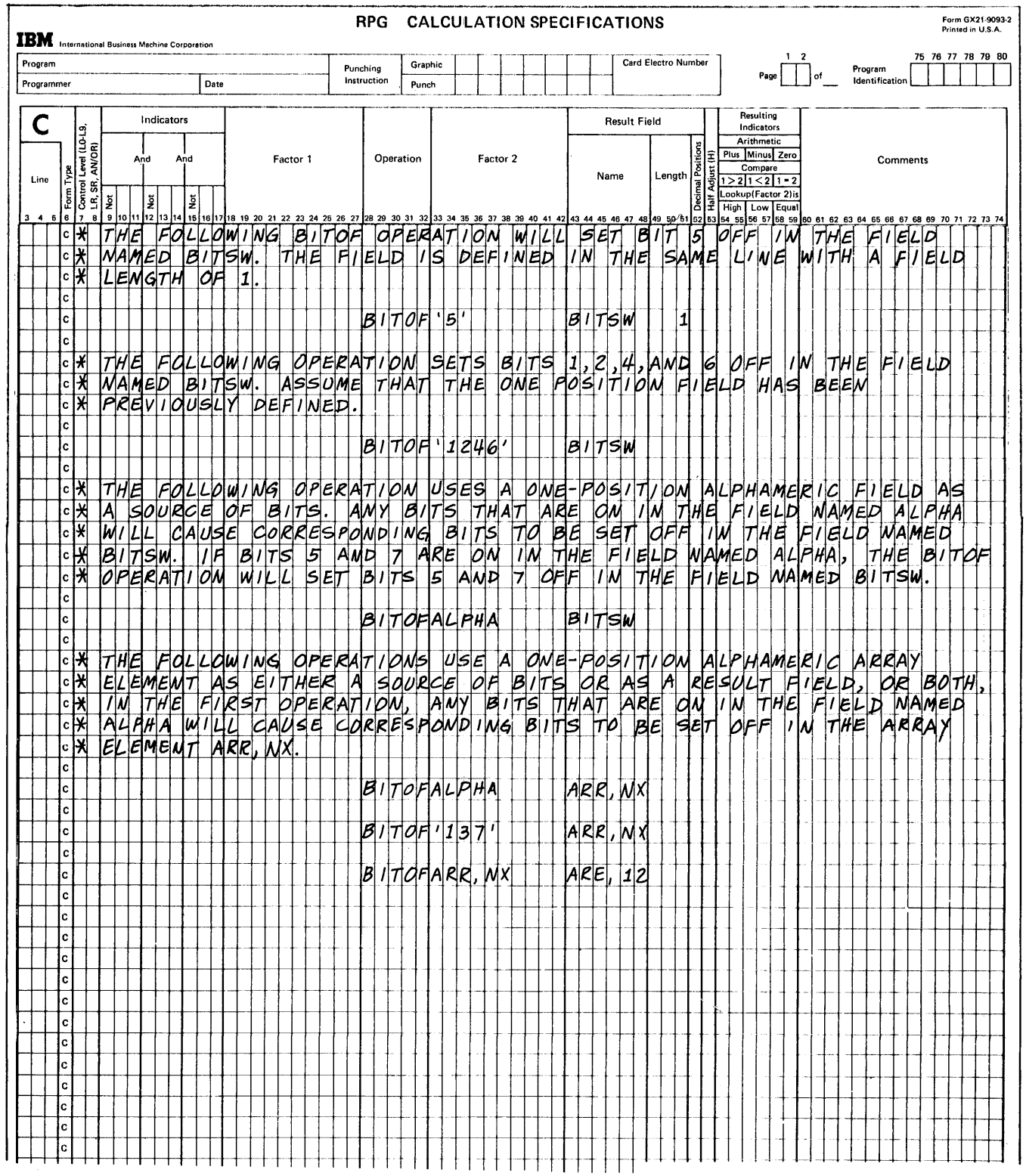
Factor 2 is always used as a source of bits for the result field. The bits to be turned off can be entered as literals in factor 2, or can be contained in a field named in factor 2. If literal values are entered in factor 2, the bits to be turned off are identified by the numbers 0-7 (0 is the first bit number). The bit numbers must be enclosed by apostrophes, and the entry must begin in column 33. From one to eight bits can be specified in factor 2 for a BITON operation. If a field name is entered in factor 2, the field must be a 1-position alphameric field. The bits that are on in the factor 2 field are turned off in the result field. The field in factor 2 can be an array element if each element in the array is a 1-position alphameric element.

When factor 2 is a field name or array element, apostrophes are not required, and the entry begins in column 33.

The result field is the field in which the bits are turned off. The field named here must be a 1-position alphameric field. The result field can be an array element if each element in the array is a 1-position alphameric element. See Figure 119 for a summary of the BITOF operation.

The operation code BITOF must appear in columns 28-32. Conditioning indicators can be used in columns 7-17, and field length must be 1.

Factor 1, decimal positions, half adjust, and the resulting indicator columns must be blank.



RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program | Programmer | Date | Punching Instruction | Graphic | Punch | Card Electro Number | Page 1 2 of __ | Program Identification 75 76 77 78 79 80

C

Line 3 4 5	Form Type 6	Control Level (L0-L9, LR, SR, AN/OR) 7 8	Indicators: And / And — Not 9 10 11 12 13 14 15 16 17	Factor 1 18-27	Operation 28-32	Factor 2 33-42	Result Field: Name 43-48	Length 49-50	Decimal Positions 51	Half Adjust (H) 52	Resulting Indicators: Arithmetic: Plus 53 Minus 54 Zero 55; Compare: 1>2 56 1<2 57 1=2 58; Lookup(Factor 2)is: High 59 Low 60 Equal 61	Comments 62-74
	C	*		THE FOLLOWING BITOF OPERATION WILL SET BIT 5 OFF IN THE FIELD								
	C	*		NAMED BITSW. THE FIELD IS DEFINED IN THE SAME LINE WITH A FIELD								
	C	*		LENGTH OF 1.								
	C											
	C				BITOF	'5'	BITSW	1				
	C											
	C	*		THE FOLLOWING OPERATION SETS BITS 1,2,4,AND 6 OFF IN THE FIELD								
	C	*		NAMED BITSW. ASSUME THAT THE ONE POSITION FIELD HAS BEEN								
	C	*		PREVIOUSLY DEFINED.								
	C											
	C				BITOF	'1246'	BITSW					
	C											
	C	*		THE FOLLOWING OPERATION USES A ONE-POSITION ALPHAMERIC FIELD AS								
	C	*		A SOURCE OF BITS. ANY BITS THAT ARE ON IN THE FIELD NAMED ALPHA								
	C	*		WILL CAUSE CORRESPONDING BITS TO BE SET OFF IN THE FIELD NAMED								
	C	*		BITSW. IF BITS 5 AND 7 ARE ON IN THE FIELD NAMED ALPHA, THE BITOF								
	C	*		OPERATION WILL SET BITS 5 AND 7 OFF IN THE FIELD NAMED BITSW.								
	C											
	C				BITOF	ALPHA	BITSW					
	C											
	C	*		THE FOLLOWING OPERATIONS USE A ONE-POSITION ALPHAMERIC ARRAY								
	C	*		ELEMENT AS EITHER A SOURCE OF BITS OR AS A RESULT FIELD, OR BOTH,								
	C	*		IN THE FIRST OPERATION, ANY BITS THAT ARE ON IN THE FIELD NAMED								
	C	*		ALPHA WILL CAUSE CORRESPONDING BITS TO BE SET OFF IN THE ARRAY								
	C	*		ELEMENT ARR,NX.								
	C											
	C				BITOF	ALPHA	ARR,NX					
	C											
	C				BITOF	'137'	ARR,NX					
	C											
	C				BITOF	ARR,NX	ARE,12					

Figure 119. Summary of BITOF Operations

Test Bit (TESTB)

This operation code causes bits identified in factor 2 to be compared with corresponding bits in a field named as the result field. The condition of the result field bits is known by the use of resulting indicators in columns 54-59.

Factor 2 is always a source of bits for the comparison with the result field. The bits to be tested can be entered as literals in factor 2, or can be contained in a field named in factor 2. If literal values are entered, the bits to be compared are identified by the numbers 0-7 (0 is the first bit number). The bit numbers must be enclosed by apostrophes, and the entry must begin in column 33. From one to eight bits can be identified in factor 2 for a TESTB operation. If a field name is entered in factor 2, the field must be a 1-position alphameric field. The bits that are on in the factor 2 field can be compared with corresponding bits in the result field. The condition of the result field bits is known by the use of resulting indicators in columns 54-59. The field in factor 2 can be an array element if each element in the array is a 1-position alphameric element. When factor 2 is a field name or array element, apostrophes are not required, and the entry begins in column 33.

The result field is the field which corresponding bits are compared from the bits specified in factor 2. The result field must be a 1-position alphameric field. This field can be an array element if each element in the array is a 1-position alphameric field.

See Figure 120 for a summary of the TESTB operation. Indicators can be assigned in columns 54-59 to reflect the status of the result field bits. At least one indicator should be assigned, and as many as three can be assigned for one operation. Two indicators can be the same for a TESTB operation, but not three. A resulting indicator has the following meanings for these columns:

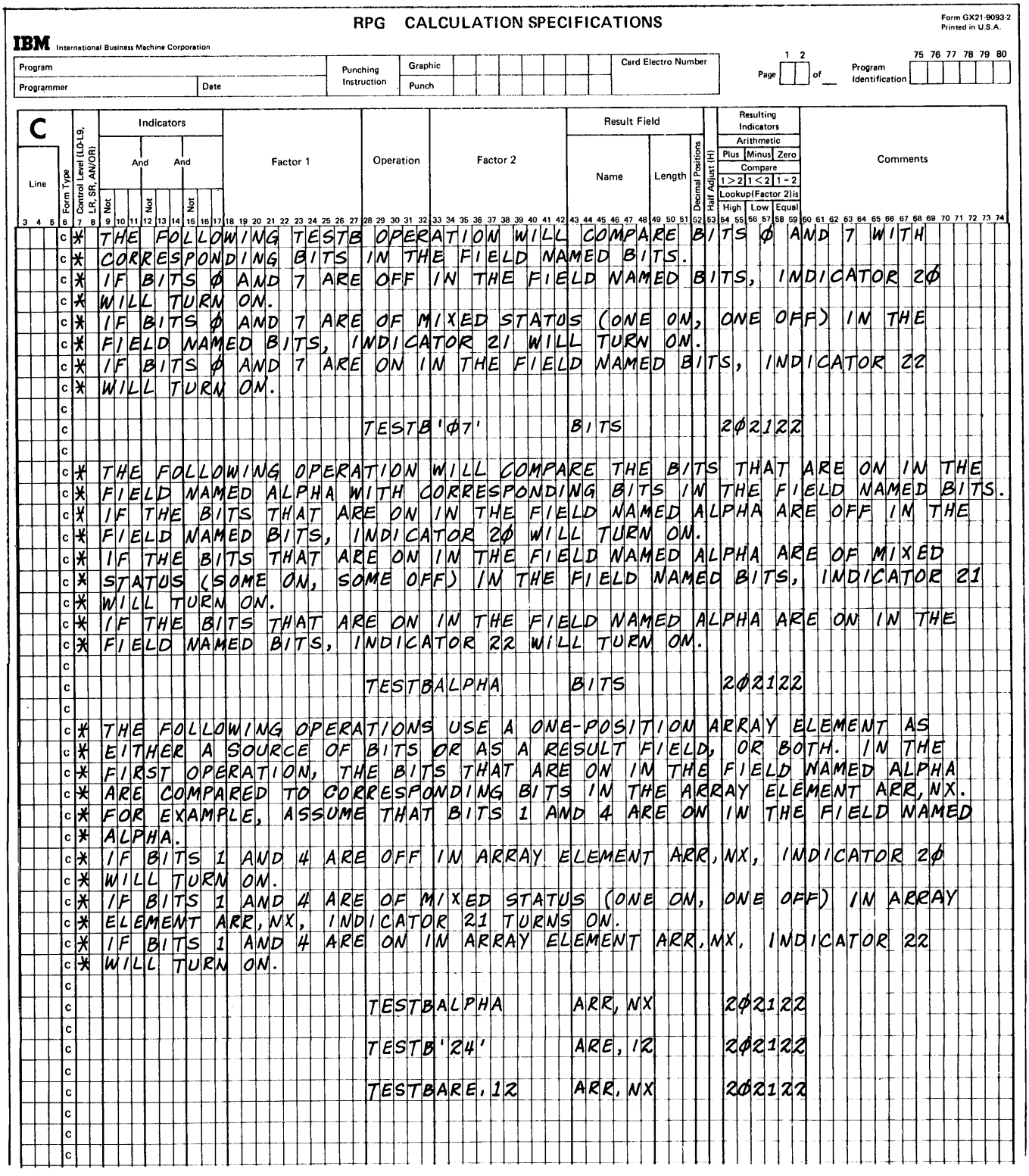
Columns 54-55: An indicator in these columns turns on if each literal bit specified in factor 2, or each bit that is on in the factor 2 field is off in the result field.

Columns 56-57: An indicator in these columns turns on if the literal bits specified in factor 2, or the bits that are on in the factor 2 field are of mixed status (some on, some off) in the result field.

Columns 58-59: An indicator in these columns turns on if each literal bit specified in factor 2, or each bit that is on in the factor 2 field is on in the result field.

The operation code TESTB must appear in columns 28-32. Conditioning indicators can be used in columns 7-17, and field length must be one. At least one resulting indicator should be assigned in columns 54-59. As many as three resulting indicators can be assigned, but not more than two can be the same.

Factor 1, decimal positions, and the half-adjust columns must be blank.



RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 1 2 of __ Program Identification 75 76 77 78 79 80

C

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus / Minus / Zero; Compare 1>2 / 1<2 / 1=2; Lookup(Factor 2)is High / Low / Equal	Comments
3 4 5	6	7 8	9 10 11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27	28 29 30 31 32	33 34 35 36 37 38 39 40 41 42	43 44 45 46 47 48	49 50 51	52	53	54 55 56 57 58 59	60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
	C	*		THE FOLLOWING TESTB OPERATION WILL COMPARE BITS Ø AND 7 WITH								
	C	*		CORRESPONDING BITS IN THE FIELD NAMED BITS.								
	C	*		IF BITS Ø AND 7 ARE OFF IN THE FIELD NAMED BITS, INDICATOR 2Ø								
	C	*		WILL TURN ON.								
	C	*		IF BITS Ø AND 7 ARE OF MIXED STATUS (ONE ON, ONE OFF) IN THE								
	C	*		FIELD NAMED BITS, INDICATOR 21 WILL TURN ON.								
	C	*		IF BITS Ø AND 7 ARE ON IN THE FIELD NAMED BITS, INDICATOR 22								
	C	*		WILL TURN ON.								
	C											
	C				TESTB	'Ø7'	BITS				2Ø2122	
	C											
	C	*		THE FOLLOWING OPERATION WILL COMPARE THE BITS THAT ARE ON IN THE								
	C	*		FIELD NAMED ALPHA WITH CORRESPONDING BITS IN THE FIELD NAMED BITS.								
	C	*		IF THE BITS THAT ARE ON IN THE FIELD NAMED ALPHA ARE OFF IN THE								
	C	*		FIELD NAMED BITS, INDICATOR 2Ø WILL TURN ON.								
	C	*		IF THE BITS THAT ARE ON IN THE FIELD NAMED ALPHA ARE OF MIXED								
	C	*		STATUS (SOME ON, SOME OFF) IN THE FIELD NAMED BITS, INDICATOR 21								
	C	*		WILL TURN ON.								
	C	*		IF THE BITS THAT ARE ON IN THE FIELD NAMED ALPHA ARE ON IN THE								
	C	*		FIELD NAMED BITS, INDICATOR 22 WILL TURN ON.								
	C											
	C				TESTB	ALPHA	BITS				2Ø2122	
	C											
	C	*		THE FOLLOWING OPERATIONS USE A ONE-POSITION ARRAY ELEMENT AS								
	C	*		EITHER A SOURCE OF BITS OR AS A RESULT FIELD, OR BOTH. IN THE								
	C	*		FIRST OPERATION, THE BITS THAT ARE ON IN THE FIELD NAMED ALPHA								
	C	*		ARE COMPARED TO CORRESPONDING BITS IN THE ARRAY ELEMENT ARR,NX.								
	C	*		FOR EXAMPLE, ASSUME THAT BITS 1 AND 4 ARE ON IN THE FIELD NAMED								
	C	*		ALPHA.								
	C	*		IF BITS 1 AND 4 ARE OFF IN ARRAY ELEMENT ARR,NX, INDICATOR 2Ø								
	C	*		WILL TURN ON.								
	C	*		IF BITS 1 AND 4 ARE OF MIXED STATUS (ONE ON, ONE OFF) IN ARRAY								
	C	*		ELEMENT ARR,NX, INDICATOR 21 TURNS ON.								
	C	*		IF BITS 1 AND 4 ARE ON IN ARRAY ELEMENT ARR,NX, INDICATOR 22								
	C	*		WILL TURN ON.								
	C											
	C				TESTB	ALPHA	ARR,NX				2Ø2122	
	C											
	C				TESTB	'24'	ARE,12				2Ø2122	
	C											
	C				TESTB	ARE,12	ARR,NX				2Ø2122	
	C											
	C											
	C											

Figure 120. Summary of TESTB Operations

Setting Indicators

These operation codes turn indicators off or on. Any indicator to be turned on or off is specified in columns 54-59. The headings in the resulting indicators field (plus or high, minus or low, zero or equal) have no meaning in these operations. When setting indicators, remember:

- The following indicators cannot be turned on by the SETON operation: 1P, MR, L0, KA-KN, KP, KQ.
- The following indicators cannot be turned off by the SETOF operation: 1P, MR, L0, and LR.
- If the LR indicator is turned on by a SETON operation which is conditioned with a control level indicator (columns 7-8 of the calculation specifications sheet), processing stops after all total output operations are finished. If it is turned on by a SETON operation at detail time (not conditioned by a control level indicator in columns 7-8), processing stops after the next total output operation is completed.
- If the halt indicators (H1-H9) are set on and not turned off before the detail output operations are complete, the system stops. Processing can be continued by responding to the halt for every halt indicator that is on.
- Setting L1-L9 on or off does not automatically set any lower control level indicators.
- Indicators L1-L9 and the record identifying indicators are always turned off after the next detail output operations are completed regardless of the previous set on or set off operation.
- Whenever a new record is read, record identifying indicators (01-99) and field indicators are set to reflect conditions on the new record. The setting from any previous SETON or SETOF operation does not apply then.
- If a numeric indicator (01-99) is SETON and not changed in other calculations, it remains on until it is set off by another calculation step.

Set On (SETON)

This operation causes any indicators in columns 54-59 to be turned on.

Set Off (SETOF)

This operation causes any indicators in columns 54-59 to be turned off.

BRANCHING WITHIN RPG II

Operations are normally performed in the order that they appear on the calculation specifications sheet. There may be times, however, when you do not want the operations performed in the order they are specified. For example, you may wish to:

- Skip several operations when certain conditions occur.
- Perform certain operations for several, but not all, record types.
- Perform several operations over and over again.

Go To (GOTO)

This operation allows you to skip instructions by specifying some other instruction to go to. You can branch to an earlier line or to a later specification line. You can branch from a detail calculation line to another detail calculation line, and you can branch from a total calculation line to another total calculation line. You cannot branch from a detail calculation line to a total calculation line or vice versa. Neither can you branch from calculations conditioned by L0-L9 to calculations conditioned by LR or vice versa. (A total calculation line is defined as one which is conditioned by a control level indicator in columns 7-8 of calculation specifications sheet.)

Factor 2 must contain the name of the point to which you want to go (identified by the label on a TAG statement). The name in factor 2 is called a label. The label can be from 1-6 characters long, and must begin in column 33 with an alphabetic character. The remaining characters can be any combination of alphabetic or numeric characters.

Blanks must not appear between characters in the label. Factor 1 and the result field are not used in this operation. The GOTO operation can be conditioned by any indicators. If it is not conditioned, the operation is always done. See *Examples* under *Branching Operations* for use of the GOTO operations.

Tag (TAG)

The operation code names the point to which you are branching in the GOTO operation. Factor 1 contains this label. The name must begin in column 18. The same label cannot be used for more than one TAG instruction.

Factor 2 and the result field are not used. No indicators are to be entered in columns 9-17 for a TAG instruction. Control level indicators must be used, however, if branching is to occur only when the information in a control field has changed. (See *Examples* under *Branching Operations* for use of the TAG operation.)

Examples

Example 1: Figure 121 shows how TAG and GOTO can skip operations on certain conditions.

1. If the result of the subtraction in line 01 is minus (indicator 10 is on), a branch is taken to RTN1 (routine 1) named by the TAG operation code in line 09. Notice that both the GOTO (line 02) and TAG (line 09) are not conditioned by control level indicators.

2. If the branch is not taken in line 02, the multiplication in line 03 is performed. Then the branch to RTN1 (line 09) must be taken because this branch is not conditioned by indicators.

3. Operations in lines 10-12 are then done. If the operation in line 12 does not turn indicator 15 on, a branch is taken backwards to RTN2 (line 05).

4. Operations then go in the order specified again from lines 06-12. Nothing is done in line 09 since TAG only gives a name. These same operations are performed again and again until 15 does turn on.

5. When 15 is on, the branch to RTN2 is not taken. The TESTZ operation is then performed. If this operation causes 20 to turn on, a branch is taken to line 17 (GOTO END). If 20 is not on, the operation in line 16 is done.

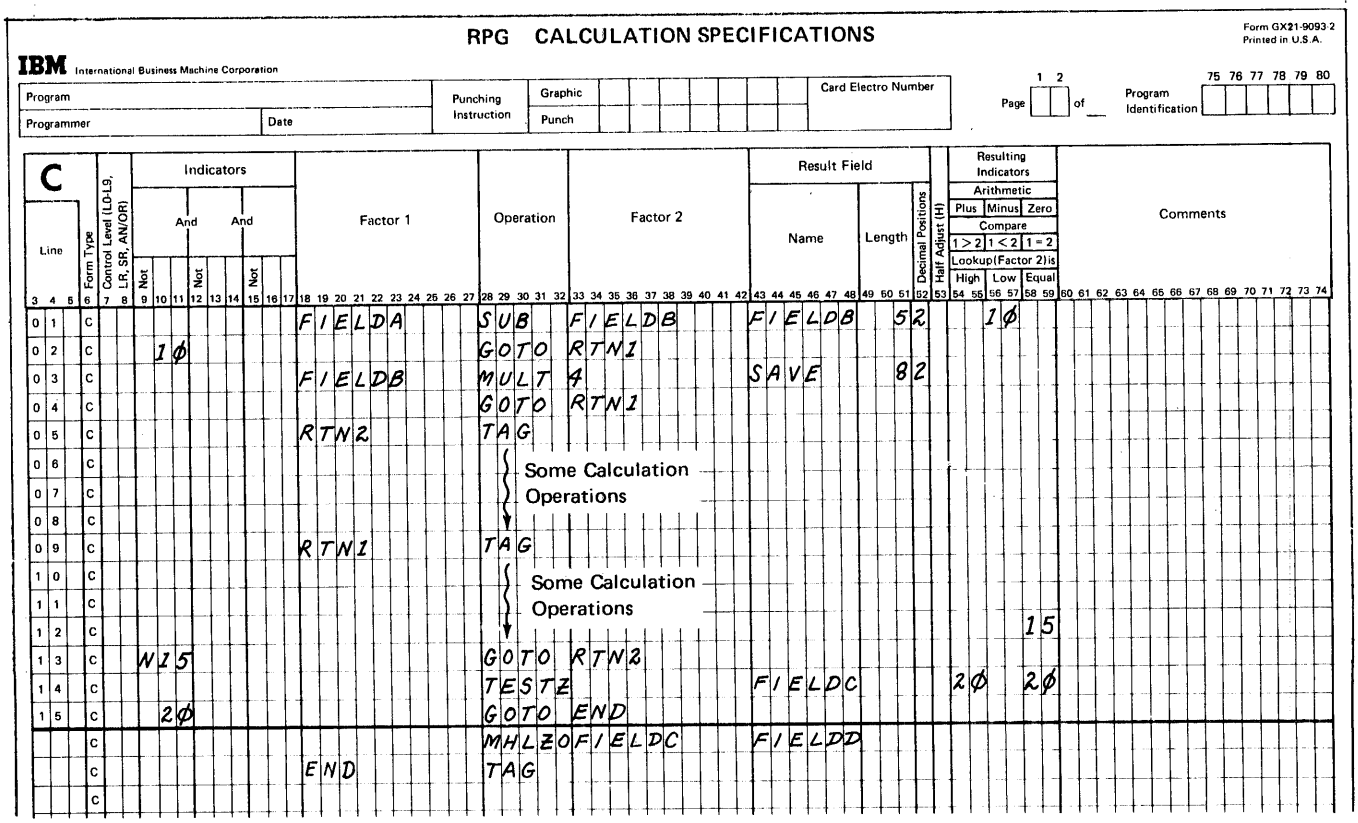
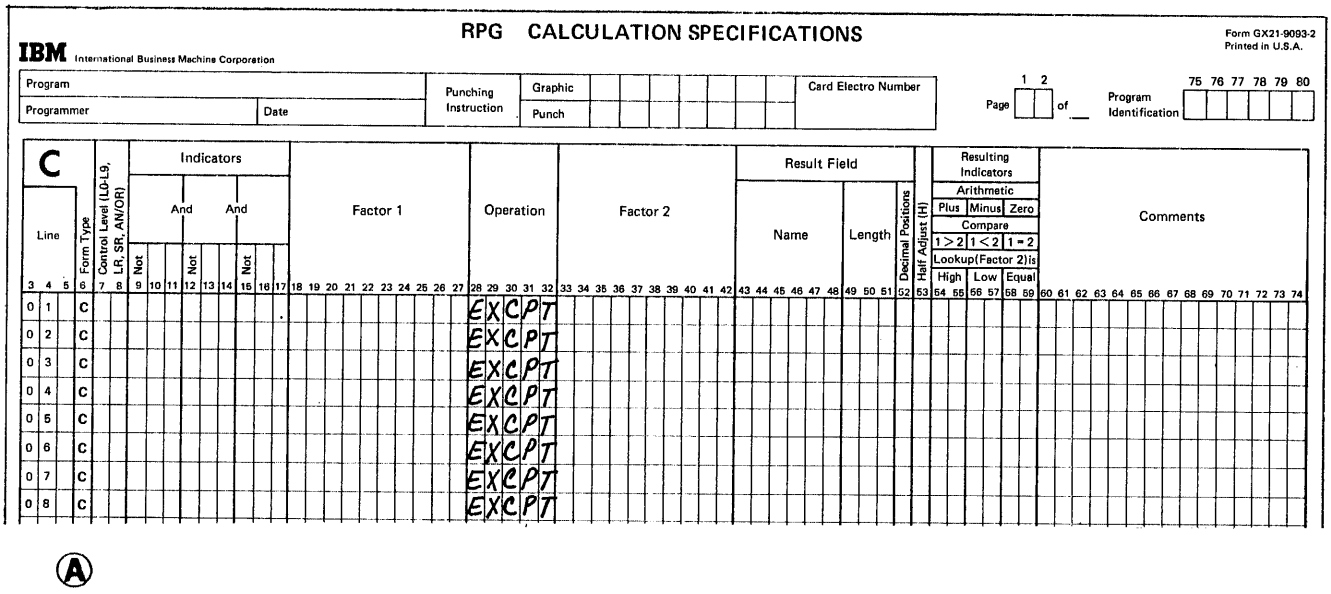


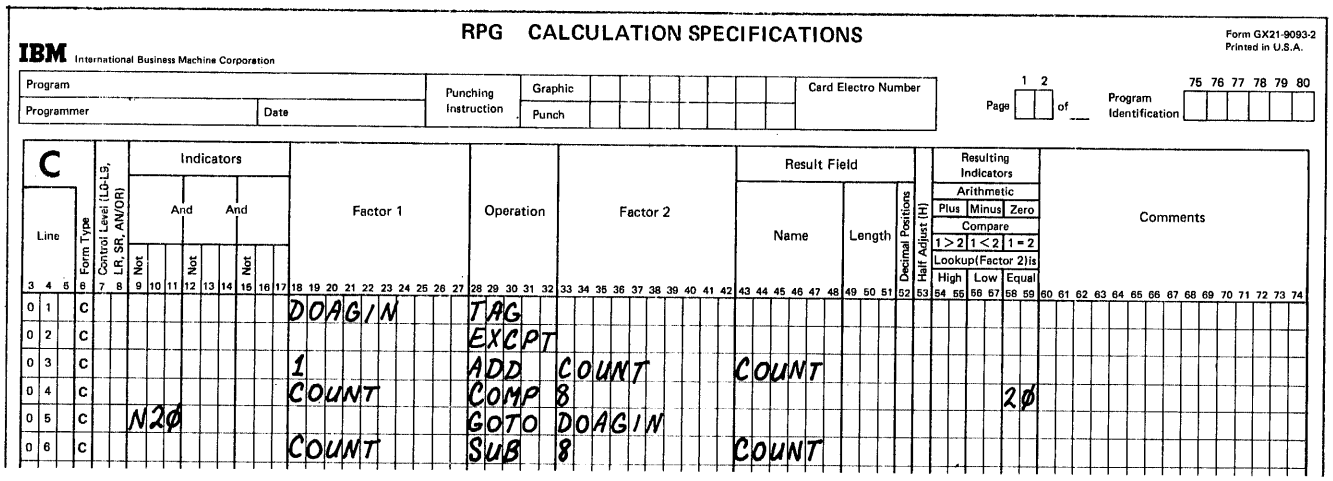
Figure 121. Using GOTO and TAG (Skipping Operations)

Example 2: Figure 122 shows how TAG and GOTO can eliminate coding when several operations have to be performed again and again. Assume you want to make eight mailing labels for every customer you have. The customer's name and address are found on an input record. Since you want to write eight labels for each record, you have to use exception lines and the operation EXCPT. (See *Exception (EXCPT)* under *Programmed Control of Input and Output* for further information.)

The EXCPT operation can be coded as shown in Figure 122, insert A. You have to write the EXCPT operation code for every mailing label. However, by using branching, you can code it all in five lines (Figure 122, insert B). An EXCPT line is printed out. One is added to COUNT to keep track of how many times the line is printed. Then COUNT is compared to 8. If COUNT does not equal 8, a branch is taken back to the beginning (GOTO DOAGIN). If COUNT equals 8, the branch is not taken. Instead 8 is subtracted from the COUNT field so that it is zero for the next cycle.



Ⓐ



Ⓑ

Figure 122. Using GOTO and TAG (Eliminate Duplicate Coding)

BRANCHING TO EXTERNAL SUBROUTINES

Linkage from RPG II to an assembler language subroutine can be done using the EXIT and RLABL operations. Control cannot be transferred from one user assembler subroutine to another user assembler subroutine.

Exit to an External Subroutine (EXIT)

The EXIT operation code designates a point in the calculation specifications where control is to be transferred from RPG II to an assembler language subroutine.

The rules for use of the EXIT operation RPG II calculation specifications are as follows:

Columns	Entry
Operation (28-32)	EXIT
Factor 1 (18-27)	Blank
Factor 2 (33-42)	The name of the subroutine to which control is to be passed. The name must consist of five or six characters, the first four of which are SUBR. The remaining characters must be alphabetic for user-written subroutines. (Numeric characters are reserved for IBM-supplied subroutines.) The module name and entry point name must be the same
Result field (43-48)	Blank
Resulting indicators (54-59)	Blank

The EXIT operation can be conditioned by control level entries (columns 7-8) and indicator entries (columns 9-17). If not conditioned by control level entries, the EXIT operation occurs at detail calculation time.

The position of the EXIT operation in the calculation specifications of the RPG program determines when the actual subroutine execution occurs (Figure 123).

A subroutine for a SPECIAL file can be used on an EXIT calculation entry. It is the responsibility of the user to keep track of the EXIT that he took. On an EXIT, index register 2 does not point to the DTF.

Position	Execution of Subroutine
First detail line in calculation specifications	Immediately following data routine file, that is, after data is extracted from input record.
Last detail line in calculation specifications	Immediately before heading records output time.
First total line in calculation specifications	Immediately following input routine (after determination of record type and testing for control level break).
Last total line in calculation specifications	Immediately before total records output time.
Any other detail/total line in calculation specifications	Immediately following the previous calculation operation.

Figure 123. Relationship Between Position of EXIT Operation and Execution of Subroutine

RPG II Label (RLABL)

The RLABL operation allows the subroutine specified in an EXIT operation to reference a field, table, array, or indicator defined in the RPG II program. RLABL operations must be specified immediately after the EXIT operation which refers to the subroutine using the field, table, array, or indicator in the RLABL statement. All external subroutines become part of the root segment and are not to be put into overlays.

The rules for use of RLABL in RPG II calculation specifications are as follows:

Columns	Entry
Operation (28-32)	RLABL
Result field (43-48)	Field, table or array name, or indicator
Field length (49-51)	Length of field (optional)
Decimal positions (52)	Decimal indication (optional)

A name defined by a TAG, BEGSR, or ENDSR specification cannot be used in an RLABL specification.

An assembler subroutine can reference indicators in the RPG II program to which it is linked. When an indicator is specified in an RLABL operation, you must use the form 1Nxx, where xx is the indicator to be transferred to the subroutine. For example, if the MR indicator is to be transferred to a subroutine, you must specify INMR as the result field for the RLABL operation.

Only RLABL operations specifying a field or a table or array name can have entries for field length (columns 49-51) and decimal positions (column 52).

The following columns must be left blank in an RLABL operation: columns 7-8 (control level), columns 9-17 (indicators), columns 18-27 (factor 1), columns 33-42 (factor 2), column 53 (half adjust), and columns 54-59 (resulting indicators).

Using RLABL Fields in the EXIT Routine

When linkage is effected from RPG II to an assembler subroutine, there are three possible entries in the result field of the RLABL specification: field, table or array, and indicator. Figure 124 shows the RPG II coding for the linkages. (See *RPG Linkage Sample Programs* for further examples.)

The subroutine can refer to a table or array defined in the RPG II program by utilizing the control field created for that table or array. This control field, one of which is created for each table or array built by the RPG II program, is in the following format:

Bytes	Meaning
1-2	Rightmost address of the first entry
3-4	Rightmost address of the last entry
5-6	Initialized to rightmost address of first entry; used at object time for rightmost address of the last looked-up entry
7-8	Length of an entry

The subroutine can obtain the data retrieved from the preceding LOKUP by using the address in bytes 5-6. To access the table or array itself, the address in bytes 1-2 must be used. Data the subroutine uses is left unpacked.

Table

RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program | Punching Instruction | Graphic / Punch | Card Electro Number | Page 1 2 of __ | Program Identification 75 76 77 78 79 80

Programmer | Date

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / And — Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus, Minus, Zero); Compare (1>2, 1<2, 1=2); Lookup(Factor 2)is (High, Low, Equal)	Comments
01	C				EXIT SUBRXX							
02	C				RLABL		TABLE	2				
03	C											

(Table name or array name)

Field

RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program | Punching Instruction | Graphic / Punch | Card Electro Number | Page 1 2 of __ | Program Identification 75 76 77 78 79 80

Programmer | Date

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / And — Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus, Minus, Zero); Compare (1>2, 1<2, 1=2); Lookup(Factor 2)is (High, Low, Equal)	Comments
01	C				EXIT SUBRXX							
02	C				RLABL		FIELD	1				
03	C											

(Field name)

Indicator

RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program | Punching Instruction | Graphic / Punch | Card Electro Number | Page 1 2 of __ | Program Identification 75 76 77 78 79 80

Programmer | Date

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / And — Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus, Minus, Zero); Compare (1>2, 1<2, 1=2); Lookup(Factor 2)is (High, Low, Equal)	Comments
01	C				EXIT SUBRXX							
02	C				RLABL		INYY					
03	C											

(Indicator)

Figure 124. RPG II Coding for RLABL Field Entries

RPG Linkage Sample Programs

Sample Program 1

In this sample program, the RPG II coding uses the EXIT operation code to effect linkage to the assembler language subroutine SUBRA (Figure 125). The RLABL specification names a field called HERE, into which SUBRA moves a character A. When control is returned to the RPG II program, a compare operation is performed to determine which character was placed in the field HERE.

Sample Program 2

In this sample program, the RPG II coding uses the EXIT operation code to effect linkage to the assembler subroutine, SUBRB (Figure 126). The first RLABL specification names a table, TABB, and the second names an indicator, IN44. The subroutine refers to both RLABL entries. It first tests the indicator. If the indicator is off, control is returned to the RPG II program. If the indicator is on, a character C is moved into the last looked-up entry in the table, TABB. When control is returned to the RPG II program, a compare operation is performed to see whether or not the subroutine placed a C in TABB.

LOOK UP OPERATIONS (LOKUP)

Use look up operations when searching through a table or an array to find a specific element. The LOKUP operation code causes a search to be made for a particular item in a table or array. The table or array is named in factor 2. Factor 1 is the search word (data for which you want to find a match in the table or array named). Factor 1, the search word, can be:

- An alphameric or numeric constant
- A field name
- An array element
- A table name

Remember that when a table is named in factor 1, it refers to the element of the table last selected in a LOKUP operation, not to the whole table.

Resulting indicators are always used in connection with LOKUP. They first indicate the type of search desired and then reflect the result of the search. A resulting indicator assigned to equal (columns 58-59) instructs the program to search for an entry in the table or array equal to the search word. The indicator turns on only if such an entry is found. If there are several entries identical to the search word, the first one that is encountered is selected.

An indicator assigned to low (columns 56-57) instructs the program to locate an entry in the table that is nearest to, yet lower in sequence than, the search word. The first such entry found causes the indicator assigned to low to turn on.

The indicator assigned to high (columns 54-55) instructs the program to find the entry that is nearest to, yet higher in sequence then, the search word. The first higher entry found causes the indicator assigned to high to turn on. In all cases the resulting indicator turns on only if the search is successful.

At least one resulting indicator must be assigned, but no more than two can be used. Resulting indicators can be assigned to equal and high or to equal and low. The program searches for an entry that satisfied either condition with equal given precedence; that is, if no equal entry can be found, the nearest lower or nearest higher entry is selected. If resulting indicators are assigned both to high and low, the indicator assigned to low is ignored. When using the LOKUP operation, remember:

- The search word and each table or array item must have the same length and the same format (alphameric or numeric).
- You can search on high, low, high and equal, or low and equal only if your table or array is in sequence.
- No resulting indicator turns on if the entry searched for is not found.

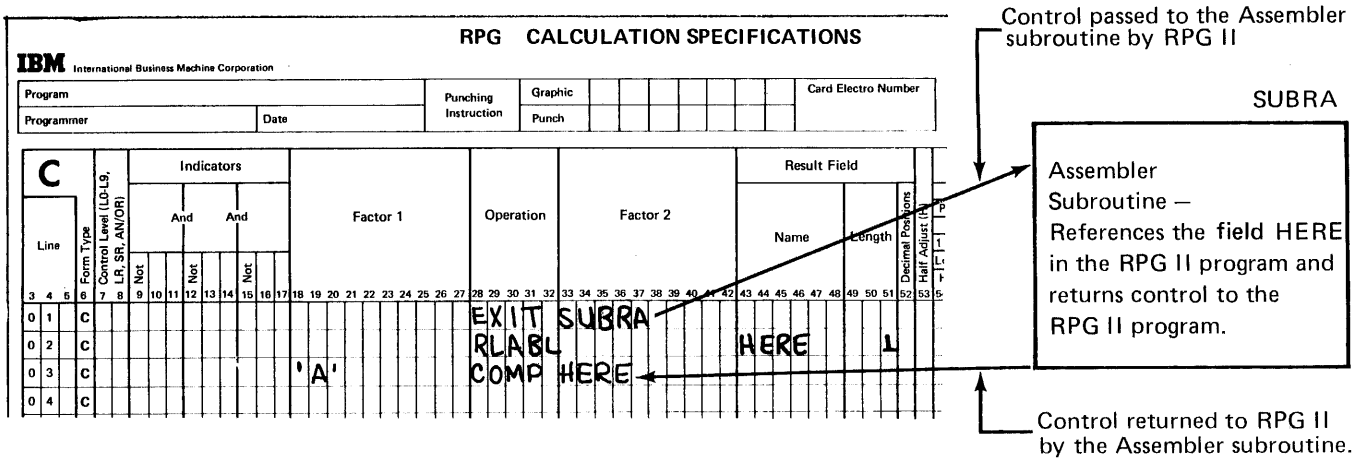


Figure 125. RPG II Coding for Sample Program 1

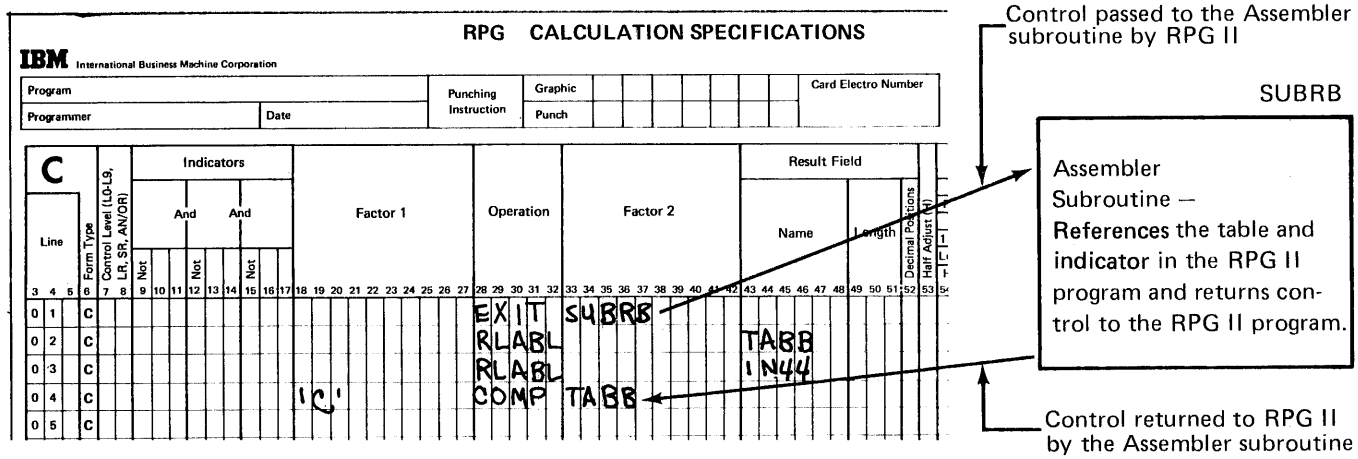


Figure 126. RPG II Coding for Sample Program 2

LOKUP With One Table

When searching a single table, factor 1, factor 2, and at least one resulting indicator must be specified. Conditioning indicators (specified in columns 7-17) can also be used.

Whenever a table item is found that satisfies the type of search being made (equal, high, low), a copy of that table item is placed in a special storage area. Every time a search is successful, the newly found table item is placed in this area, destroying what was there before. If the search is not successful, no table item is placed in the storage area. Instead, the area keeps the contents it had before the unsuccessful search.

Resulting indicators reflect the result of the search. If the indicator is on, showing a successful search, you know that a copy of the item searched for is in the special storage area.

LOKUP With Two Tables

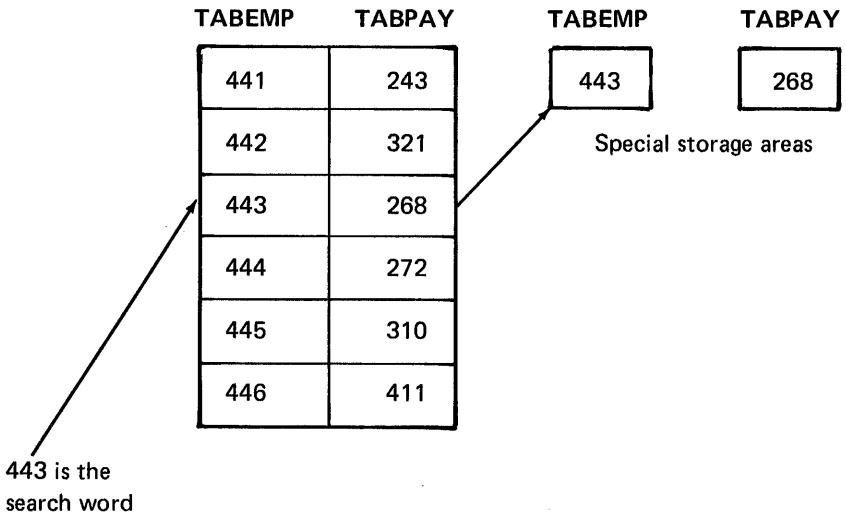
When two related tables are used in a search, only one is actually searched. When the search condition (high, low, equal) is satisfied, the corresponding data items from both tables are placed in their respective special storage areas and are made available for use.

Factor 1 must be the search word and factor 2 must name the table to be searched. The result field must name the related table from which data is made available for use. Resulting indicators must also be used. You can specify conditioning indicators in columns 7-17 if needed.

The two tables involved should be the same length. If the table that is searched is longer than its related table, it is possible to satisfy the search condition. However, there will not be a data table item in the second table which corresponds to the item found in the search table. Unpredictable results may occur.

Note: Using a table name in an operation (other than LOKUP) before a successful LOKUP occurs may lead to unpredictable results, since you do not know what is in the special storage area referenced by the table name.

In Figure 127, insert A, the related tables TABEMP and TABPAY are read into storage. Assume that an input record is read with 443 in the EMPNUM field. With 443 as the search word, the table TABEMP can be searched for an equal entry. When the correct entry is found, the table item 443 is moved into the special storage area for TABEMP. At the same time, the corresponding item 268 is moved into the special storage area for TABPAY. The contents of the storage areas can now be referenced in subsequent calculation operations by the appropriate table name. Insert B of Figure 127 shows the coding needed to perform the LOKUP operation described in insert A. It also shows how to reference the contents of the special storage area after a successful LOKUP operation.



Ⓐ

RPG CALCULATION SPECIFICATIONS

IBM International Business Machine Corporation

Form GX21-9093-2
Printed in U.S.A.

Program | Punching Instruction: Graphic / Punch | Card Electro Number | Page __ of __ | Program Identification

Programmer | Date

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus / Minus / Zero); Compare (1>2 / 1<2 / 1=2); Lookup(Factor 2)is (High / Low / Equal)	Comments
01	C			EMPNUM	LOKUP	TABEMP	TABPAY				09	
02	C											
03	C	X		THE ABOVE OPERATION WILL SEARCH TABEMP FOR AN ENTRY THAT IS EQUAL								
04	C	X		TO THE CONTENTS OF THE FIELD NAMED EMPNUM. IF THE CORRECT ENTRY								
05	C	X		IS FOUND IN TABEMP, 09 TURNS ON, AND THE TABEMP ENTRY AND ITS								
06	C	X		RELATED ENTRY IN TABPAY ARE MOVED INTO THEIR SEPARATE STORAGE								
07	C	X		AREAS.								
08	C											
09	C			HRSWKD	MULT	TABPAY	AMT		62	H		
10	C											
11	C	X		THE ABOVE OPERATION WILL MULTIPLY THE CONTENTS OF THE FIELD NAMED								
12	C	X		HRSWKD BY THE CONTENTS OF THE SPECIAL STORAGE AREA FOR TABPAY. THE								
13	C	X		SPECIAL STORAGE AREA FOR TABPAY CONTAINS THE RESULTS OF THE LAST								
14	C	X		SUCCESSFUL LOKUP OPERATION INVOLVING TABPAY.								
15	C											

Ⓑ

Figure 127. LOKUP with Related Tables

Referencing the Table Item Found in a LOKUP Operation

Whenever a table name is used in an operation other than LOKUP, the table name really refers to the data placed in the special table storage area by the last successful search. Thus, by specifying the table name in this fashion, you can use data items from a table in calculation operations.

If the table is used as factor 1 in a LOKUP operation, the contents of the special table storage area is used as the search word. In this way a data item from a table can itself become a search word.

The table can also be used as the result field in operations other than the LOKUP operation. In this case the contents of the special table storage area is changed by the calculation operation. The corresponding table item in the table in main storage is also changed. This is a way in which you can modify the contents of the table by calculation operations (Figure 128).

Example

Figures 129 through 132 show the use of the LOKUP operation. Figure 129, insert A, shows the content of four tables: table A, table B, table C, and table D (loaded at compile time). Each table has five entries.

Figure 129, insert B, shows the extension specifications for these tables. Tables A and B are described separately and are, therefore, entered separately. Tables C and D are related tables and are entered in alternating format on the table input records. Figure 130 shows the order in which the table input records are loaded into the system at compile time.

Figure 131 shows 15 different LOKUP operations using these four tables. The results of these operations are shown in Figure 132. Figure 132 tells if the entry searched for was found, and if so, what indicator is on to indicate the result of the search and what item was taken from a related table (when one is used).

RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 1 2 of __ Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And	Indicators: And	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Result Field: Decimal Positions	Result Field: Half Adjust (H)	Resulting Indicators: Arithmetic: Plus	Resulting Indicators: Arithmetic: Minus	Resulting Indicators: Arithmetic: Zero	Resulting Indicators: Compare: 1>2	Resulting Indicators: Compare: 1<2	Resulting Indicators: Compare: 1=2	Resulting Indicators: Lookup(Factor 2)is: High	Resulting Indicators: Lookup(Factor 2)is: Low	Resulting Indicators: Lookup(Factor 2)is: Equal	Comments
01	C				ARGMNT	LOKUP	TABLEA														20SEARCH FOR =
02	C		20		TABLEA	MULT	1.5	TABLEA													IF ELEMENT IS
03	C	*																			FOUND, MULTIPLY
04	C	*																			BY 1.5
05	C																				

Figure 128. Referencing the Table Item Found in LOKUP Operation

	First Entry	Second Entry	Third Entry	Fourth Entry	Fifth Entry
Table A	01	05	08	32	96
Table B	06.13	02.12	47.16	28.70	15.16
Table C	WWW	NNN	LLL	GGG	AAA
Table D	7	8	3	2	5

Ⓐ

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

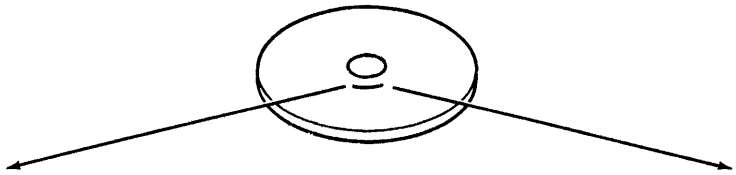
	Punching Instruction	Graphic Punch							Card Electro Number	Page 1 2

Extension Specifications

To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)
	TABLEA	2	5	2			A					
	TABLEB	5	5	4	2							
	TABLEC	1	5	3		D		TABLED	1		Ø	

Ⓑ

Figure 129. Table Look-Up (Tables Used)



These tables are in the library.

Source Program	**	0105	0832	96	**	05130 21247 15287 01516	**	WWW7	NNN8	LLL3	GGG2	AAA5	/*
		TABLEA				TABLEB		TABLEC and TABLED					

Figure 130. Order in Which Tables are Located

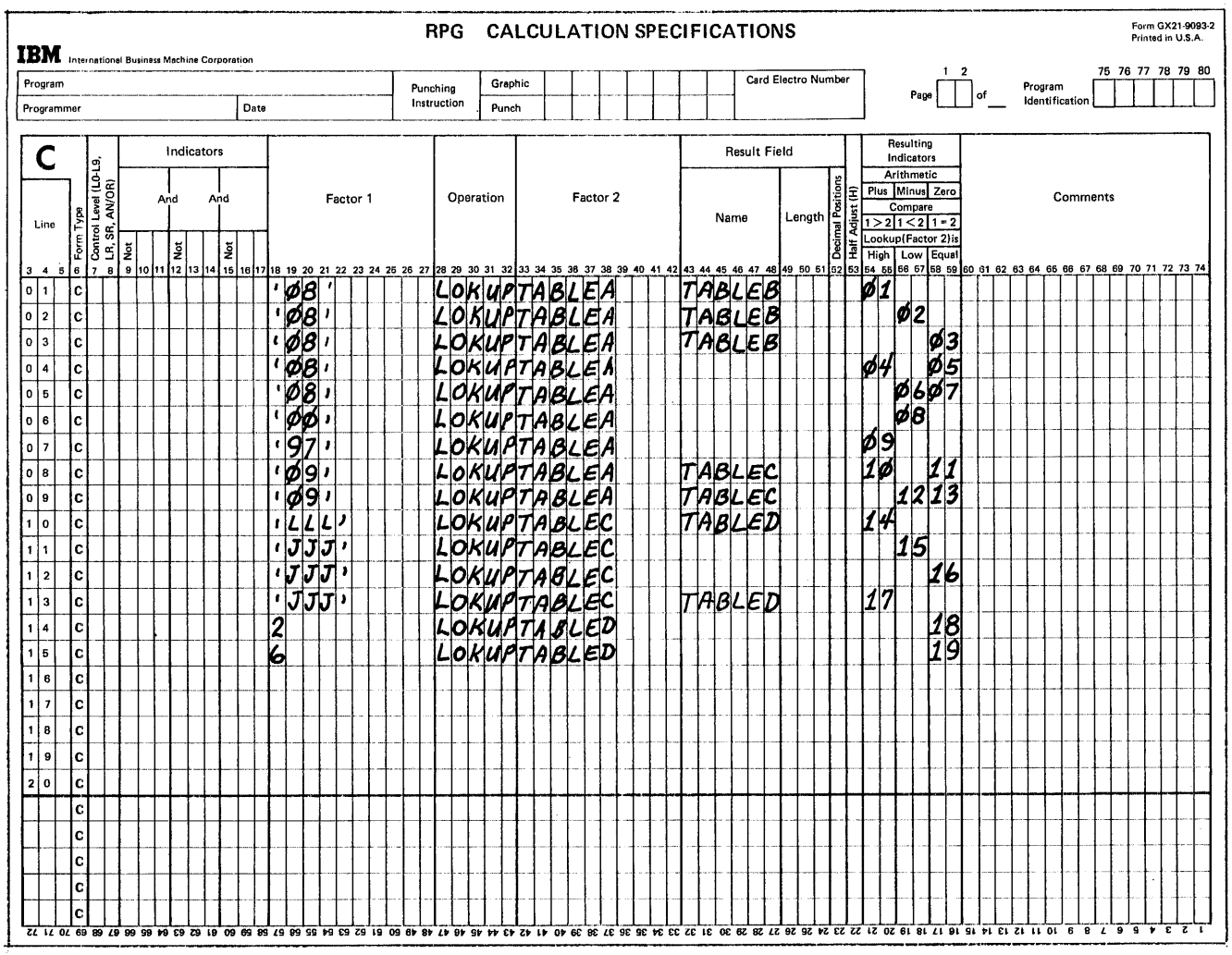


Figure 131. Table LOKUP Operation

Specification Line Number	Entry Found	Indicator On	Table Item Satisfying Search Condition	Table Item Used From Related Table
01	Yes	01	32	28.70
02	Yes	02	05	02.12
03	Yes	03	08	47.15
04	Yes	05	08	
05	Yes	07	08	
06	No			
07	No			
08	Yes	10	32	GGG
09	Yes	12	08	LLL
10	Yes	14	NNN	8
11	Yes	15	GGG	
12	No			
13	Yes	17	LLL	3
14	Yes	18	2	
15	No			

Figure 132. Results of LOKUP Operations

Using the LOKUP Operation with Arrays

The LOKUP specifications for arrays are the same as for tables except that if factor 2 is an array, the result field cannot be used. In addition, if the desired item is found, it is not moved to a special holding area since these holding areas are only associated with tables. Instead, the indicators reflect only that the desired item is in the array; the programmer does not have ready access to this item.

Example: Figure 133 shows two LOKUP operations performed with an array. MANNOS, a 2100-element array of employee numbers, is read in at execution-time from file ARRFILE with six 10-position elements per record; the array elements are in ascending order. Line 01 of the calculation specifications sheet shows a LOKUP of array MANNOS to find the element nearest to but higher in sequence than the search word 100336. If this desired element is found in the array, indicator 20 turns on and the GOTO in line 02 is performed. Notice that the result of this LOKUP indicates only whether or not the desired element exists in the array.

Line 05 of the calculation specifications sheet shows essentially the same LOKUP operation. Indicator 20 turns on when the first element higher in sequence than 100336 is found. Note, however, that in this LOKUP operation, the array MANNOS is indexed by the field INX. This index field was set to 1 in line 04 so the LOKUP begins at the first element of MANNOS. If the desired element is found, the number of this element (not its contents) is placed in the field INX. In this way, the actual element which satisfied the LOKUP can be used in subsequent calculation operations, as in line 07. If no element was found to satisfy the LOKUP, the field INX is reset to 1. Refer to *Starting the Search at a Particular Array Item* under *Look Up Operations (LOKUP)* for more information on indexing an array in a LOKUP operation.

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

Form X21-9091-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 1 2 of __ Program Identification 75 76 77 78 79 80

Extension Specifications

Line	Form Type	Record Sequence of the Chaining File / Number of the Chaining Field / From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
3 4 5	6	7 8 9 10 11 12 13 14 15 16 17 18	19 20 21 22 23 24 25 26	27 28 29 30 31 32	33 34 35	36 37 38 39	40 41 42	43	44	45	46 47 48 49 50 51	52 53 54	55	56	57	58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
01	E	ARRFILE		MANNOS	10	210	6			ØA						
02	E															
03	E															

RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 1 2 of __ Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus / Minus / Zero); Compare (1>2 / 1<2 / 1=2); Lookup(Factor 2)is (High / Low / Equal)	Comments
3 4 5	6	7 8	9 10 11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27	28 29 30 31 32	33 34 35 36 37 38 39 40 41 42	43 44 45 46 47 48	49 50 51	52	53	54 55 56 57 58 59	60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
01	C			100336	LOKUPMANNOS						20	
02	C		20		GOTO NEXT							
03	C											
04	C				Z-ADD1	INX		40				
05	C			100336	LOKUPMANNOS,INX						20	
06	C		N20		GOTO END							
07	C				MOVE MANNOS,INXSAVE			60				
08	C			NEXT	TAG							
09	C											
10	C											
11	C											
12	C											
13	C											
14	C											

Calculation Operations

Figure 133. LOKUP with an Array

Starting the Search at a Particular Array Item

It is possible, to save processing time, to start the LOKUP search at a particular item in the array. This type of search is indicated by additional entries in columns 33-42. Enter the name of the array to be searched in these columns followed by a comma and a numeric literal or the name of a numeric field (with no decimal positions). The numeric literal or numeric field tells the number of the item at which you want to start the search (Figure 134). This numeric literal or field is known as the index because it points to a certain item in the array. All other columns are used as previously described for the normal look up operation.

The search starts at the specified item and continues until the desired item is found or until the end of the array is reached. When an index field is used, an unsuccessful search causes the index field to contain the value of one.

If, however, an item is found which satisfies the conditions of the LOKUP operation, the number of that array item (counting from the first item) is placed in the index field. A numeric literal used as an index is not changed to reflect the result of the search.

Note: If a literal or field index for an array is zero or greater than the number of elements in the array, the following results:

- For a literal index, a severe error occurs, and compilation ceases.
- For a field index the job halts, allowing the operator to cancel or restart the program. If the program is restarted, the field index is given a value of one.

RPG CALCULATION SPECIFICATIONS

IBM International Business Machine Corporation

Form GX21-9093-2 Printed in U.S.A.

Program		Punching Instruction	Graphic	Card Electro Number
Programmer	Date		Punch	

Page 1 2 of __ Program Identification 75 76 77 78 79 80

Line 3 4 5	Form Type 6	Control Level (L0-L9, LR, SR, AN/OR) 7 8	Indicators: And / And — Not 9 10 11 12 13 14 15 16 17	Factor 1 18-27	Operation 28-32	Factor 2 33-42	Result Field: Name 43-48	Length 49-50	Decimal Positions 51	Half Adjust (H) 52	Resulting Indicators: Arithmetic: Plus / Minus / Zero; Compare: 1>2 / 1<2 / 1=2; Lookup(Factor 2)is: High / Low / Equal 53-59	Comments 60-74
0 1	C			ACCTNO	LOKUPCUST	,INDEX					Ø6	
0 2	C											
0 3	C											

Figure 134. Array Look Up: Starting at a Particular Array Item

SUBROUTINE OPERATIONS

These operation codes are only used for subroutines. See *Subroutines* under *Branching to External Subroutines* for information on external subroutines. All subroutine lines must be written on specification lines following all operations conditioned by control level indicators specified in columns 7-8. Subroutine lines are always identified by an SR in columns 7-8.

Begin Subroutine (BEGSR)

This operation code serves as the beginning point of the subroutine. Factor 1 must contain the name of the subroutine.

End Subroutine (ENDSR)

This operation code must be the last statement of the subroutine. It serves to define the end of the subroutine. Factor 1 can contain a name. This name then serves as a point to which you can branch by a GOTO statement within the subroutine. The ENDSR operation ends the subroutine and automatically causes a branch back to the next statement after the EXSR operation.

Note: Neither BEGSR operations nor ENDSR operations can have conditioning indicators (columns 9-17).

Execute Subroutine (EXSR)

This operation causes operations in the subroutine to be performed. EXSR can appear anywhere in the program. Whenever it appears, the subroutine is executed. After operations in the subroutine are done, the operation in the line following the EXSR operation is performed.

This operation can be conditioned by any indicators, meaning the subroutine is executed only when all conditions are satisfied. Factor 2 must contain the name of the subroutine that is to be executed. This same name must appear on a BEGSR instruction. A subroutine must not call itself.

PROGRAMMED CONTROL OF INPUT AND OUTPUT

The normal program cycle can be altered to allow input and output operations during calculations. (See *General RPG II Program Logic* under *Introduction* for a brief description of the program cycle.) The following operations provide this capability:

- Exception (EXCPT)
- Read (READ)
- Force (FORCE)
- Chain (CHAIN)
- Key (KEY)
- Set (SET)
- Set lower limits (SETLL)

Exception (EXCPT)

This operation allows records to be written during calculations. Use this primarily when you want to have a variable number of similar or identical records (either detail or total) written in one program cycle. (Remember that normally only the exact number of records specified by the output specifications are written on a file in one program cycle.) For example, you might use EXCPT to produce a variable number of identical mailing labels, to write out contents of a table, or to produce a number of records having the same information.

When the EXCPT operation is used, EXCPT is entered in columns 28-32. Columns 7-17 can also have entries. All other columns must be blank. The line or lines which are to be written during calculation time are indicated by an E in column 15 of the output specifications sheet. Figure 135 shows the use of the EXCPT operation to produce a variable number of records on a printer file having the same information.

Records in the input file have two fields, NAME and COUNT. The NAME field is to be entered into a certain number of records. That number is indicated in the COUNT field.

The first calculation line compares COUNT to zero. If COUNT is equal to zero, indicator 02 turns on and the GOTO operation avoids further calculations. If the COUNT field is one or more, the EXCPT operation is performed, and the exception record indicated by the E in column 15 of the output specifications sheet is printed. The field CONSEC is used to keep track of the number of records printed. Each time an exception record is printed, one is added to CONSEC. CONSEC is then compared with COUNT, the field that tells how many records should be printed. If they are not equal (indicator 20 is not on), a branch is taken back to DOAGIN. Another record is printed. One is added to CONSEC and CONSEC is compared to COUNT. If these fields are equal, another input record is read. If not, the same operations are done again. Whenever CONSEC equals COUNT, enough records are printed. CONSEC is then subtracted from itself, making it zero. This last operation is necessary so that an accurate count can be kept for the next record.

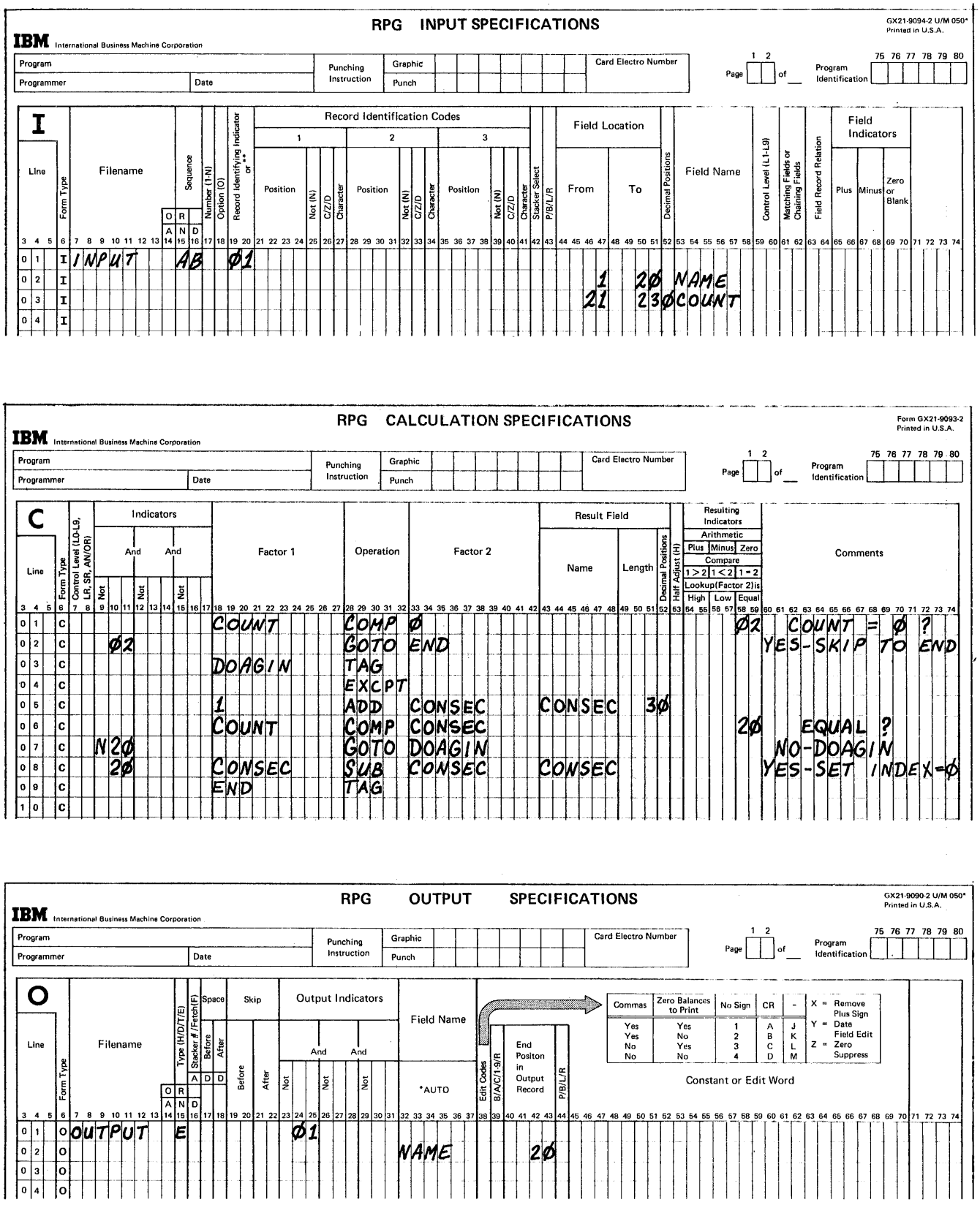


Figure 135. EXCPT Operation (Producing a Variable Number of Identical Records)

Read (READ)

The read operation calls for immediate input from a demand file during the calculation phase of the program cycle. This operation differs from the force operation because force calls for certain input on the next program cycle, not the present one.

The operation code READ must appear in columns 28-32. Factor 2 contains the name of the file from which a record should be read immediately. An indicator can be used in columns 58-59. An indicator specified in these columns turns on when an end-of-file condition is reached for the demand file, or on each READ operation after an end-of-file condition is reached. If columns 58-59 are left blank, a halt occurs on an end-of-file condition and on subsequent read operations after the end-of-file condition is reached. Indicators can be specified in columns 7-17.

The following columns must remain blank for a READ operation: columns 18-27 (factor 1), columns 43-48 (result field), columns 49-51 (field length), column 52 (decimal positions), column 53 (half adjust), and columns 54-55 and 56-57 (resulting indicators).

The following files can appear as factor 2 in a READ operation:

- Sequential disk files processed consecutively, and specified as input or update files.
- Indexed disk files processed sequentially by key, and specified as input or update files.
- Indexed disk files processed sequentially by limits, and specified as input or update files.
- Direct files processed consecutively as input or update files.

Remember these points when using the READ operation:

- All demand files, except those assigned to the KEYBORD, must be processed by the READ operation.
- Control levels, matching fields, and look ahead fields are not allowed with demand files.
- Numeric sequence testing on input specifications sheet is not allowed for demand files.
- Columns 63-64 (field record relation) on input specifications sheet cannot use the MR indicator for demand files.

Note: When a program is doing multiple read operations from one or more demand files during the same RPG II cycle, the record identifying indicators assigned to the file or files remain on throughout the cycle if the previous READ operations were executed successfully.

Force (FORCE)

FORCE statements enable you to select the file from which the next record is to be taken for processing. They apply to primary or secondary input and update files. The FORCE operation cannot be used to read from files assigned to the KEYBORD.

Factor 2 in a FORCE statement identifies the file from which the next record is to be selected. If the statement is executed, the record is read at the start of the next program cycle. If more than one FORCE statement is executed during the same program cycle, all but the last is ignored. FORCE should not be specified at total time.

FORCE statements override the multifile processing method by which the program normally selects records. However, the first record to be processed is always selected by the normal method. The remaining records can be selected by FORCE statements.

Example: Figure 136 shows how the FORCE operation can be used to control input from primary and secondary files.

Three disk files are defined: a primary file, a secondary file, and an output file. Each primary record has a numeric field named NBR. This field contains the number of secondary records to be written after each primary record is read. If the field is zero, no primary or secondary records are written. If the field contains a 2, for example, then the primary record is to be written followed by two secondary records. When the correct amount of secondary records are written, the primary file is read again.

When a primary record is read, indicator 01 turns on. In calculations, NBR is compared to 0. If NBR is lower than or equal to 0, a halt occurs. No primary or secondary records are written. Processing begins with the next primary record, according to normal record selection.

Assume that the NBR field in the next primary record is 2. This indicates that two secondary records are written after the primary record is written. The SUB operation in line 02 acts as a counter. The NBR field is reduced by one and tested. In this case, it is not negative. This indicates that more than one secondary record is to be written and the program moves to line 03. This FORCE operation calls for input on the next program cycle from the secondary file. Since indicator 03 is not on, line 04 is skipped. In the output specifications the fields are written for the primary record because indicator 01 is on.

On the next cycle, the secondary file is selected as a result of the FORCE operation. Indicator 02 turns on. In calculations, the SUB operation in line 02 is performed, reducing the NBR field to zero. It is not negative so indicator 03 is not on and the program moves to line 03. Again, the FORCE operation calls for input from the secondary file on the next program cycle. In output specifications, the fields from the secondary file are written because indicator 02 is on.

On the next cycle, the secondary file is again selected as a result of the FORCE operation. Indicator 02 turns on. In calculations, the NBR field is reduced by one, and is now negative (-1). Indicator 03 turns on as a result of the negative condition, and the program moves to line 04. In this line the FORCE operation calls for primary input on the next cycle, since the requirements of two secondary records written for the last primary record are met after the present cycle. In output specifications, the fields from the secondary record are written because indicator 02 is on. This is the second and last secondary record written after the primary record was selected.

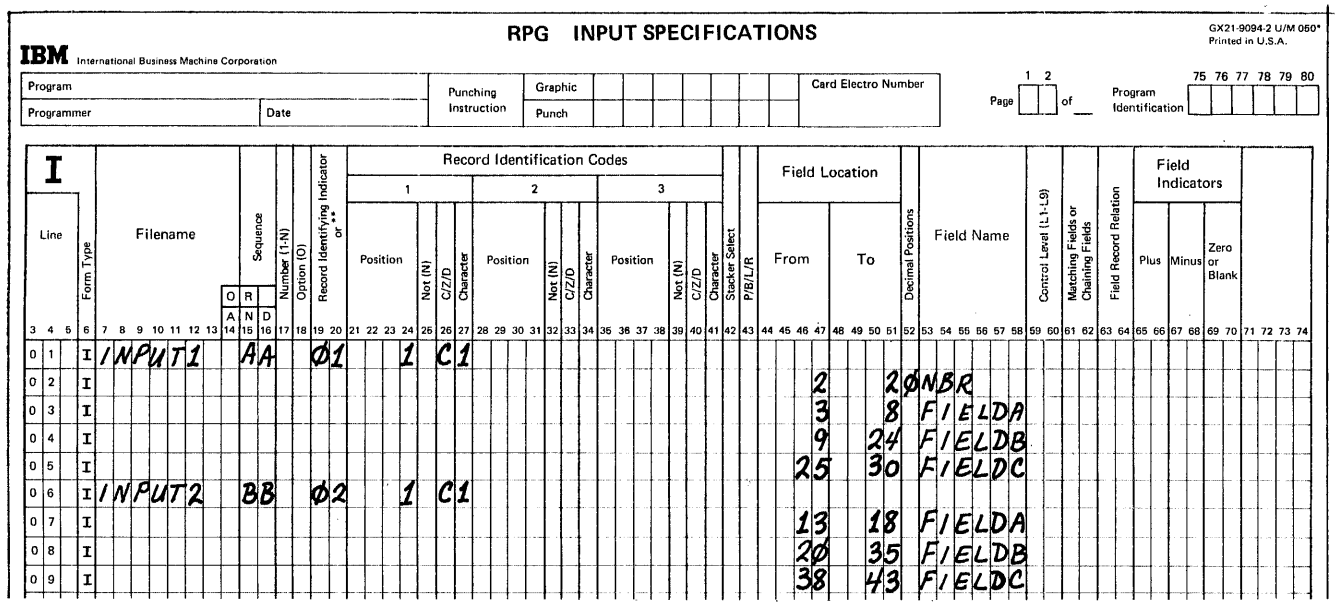
On the next cycle, the primary file is selected as a result of the last FORCE operation. The appropriate number of secondary records are written for that primary record using FORCE to control the input from the secondary file.

File Description Specification

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Core Index	File Addition/Unordered	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	File Condition U1-U8	A/U	R/U/N
	F																												
02	F	INPUT1	I	PE			F	64	64								DISK												
03	F	INPUT2	I	S			F	64	64								DISK												
04	F	OUTPUT	O				F	64	64								DISK												
05	F																												
06	F																												
07	F																												
08	F																												
09	F																												
10	F																												
	F																												
	F																												

Continuation Lines: K, Option, Entry

Figure 136 (Part 1 of 2). FORCE Operation



RPG INPUT SPECIFICATIONS

IBM International Business Machine Corporation

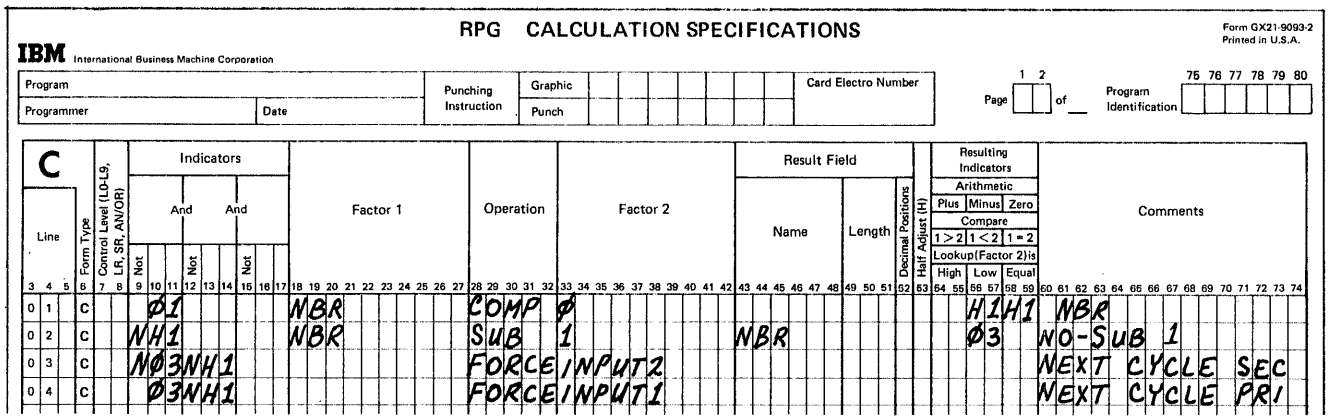
GX21-9094-2 U/M 050* Printed in U.S.A.

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 1 2 of __ Program Identification 75 76 77 78 79 80

I

Line	Form Type	Filename	Sequence (O A / R N / D)	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	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
01	I	INPUT1	AA			01		1	C	1																				
02	I																				2	20		NBR						
03	I																				3	8		FIELDA						
04	I																				9	24		FIELDB						
05	I																				25	30		FIELDC						
06	I	INPUT2	BB			02		1	C	1																				
07	I																				13	18		FIELDA						
08	I																				20	35		FIELDB						
09	I																				38	43		FIELDC						



RPG CALCULATION SPECIFICATIONS

IBM International Business Machine Corporation

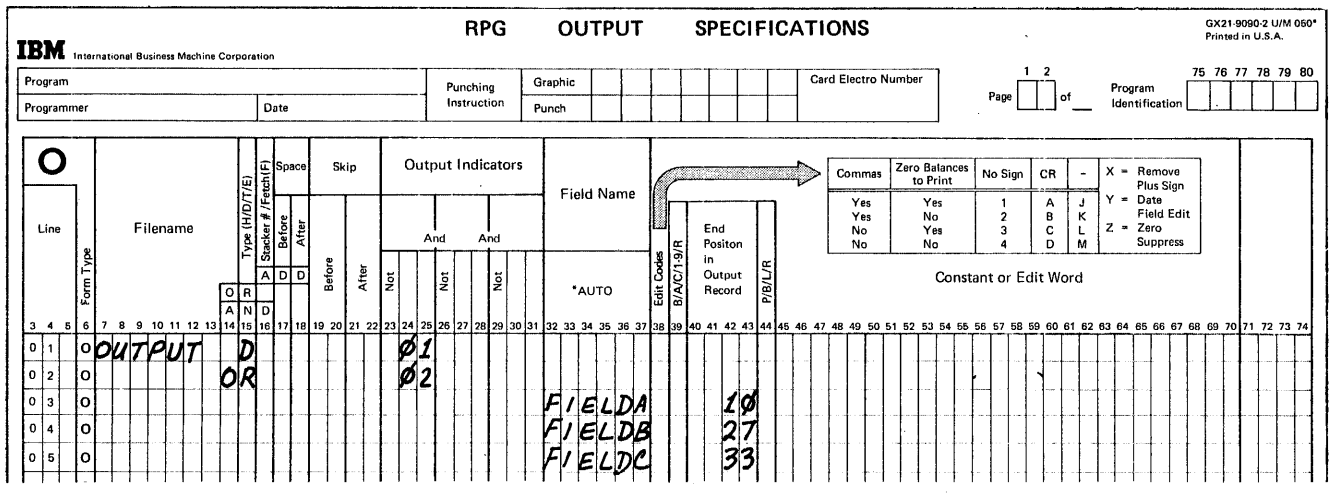
Form GX21-9093-2 Printed in U.S.A.

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 1 2 of __ Program Identification 75 76 77 78 79 80

C

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators And Not	And Not	And Not	Factor 1	Operation	Factor 2	Result Field Name	Result Field Length	Result Field Decimal Positions	Result Field Half Adjust (H)	Resulting Indicators Arithmetic Plus	Arithmetic Minus	Arithmetic Zero	Compare 1>2	Compare 1<2	Compare 1=2	Lookup(Factor 2)is High	Lookup(Factor 2)is Low	Lookup(Factor 2)is Equal	Comments
01	C		01			NBR	COMP 0													H1	H1	NBR
02	C		NH1			NBR	SUB 1		NBR											03		NO-SUB 1
03	C		N03NH1				FORCE	INPUT2														NEXT CYCLE SEC
04	C		03NH1				FORCE	INPUT1														NEXT CYCLE PRI



RPG OUTPUT SPECIFICATIONS

IBM International Business Machine Corporation

GX21-9090-2 U/M 060* Printed in U.S.A.

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 1 2 of __ Program Identification 75 76 77 78 79 80

O

Line	Form Type	Filename	Type (H/D/T/E) (O A / R N / D)	Stacker #/Fetch(F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And Not	And Not	And Not	Field Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R
01	O	OUTPUT	D						01							
02	O		OR						02							
03	O											FIELDA			10	
04	O											FIELDB			27	
05	O											FIELDC			33	

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

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

Constant or Edit Word

Figure 136 (Part 2 of 2). FORCE Operation

Chain (CHAIN)

The chain operation causes a record to be read from a disk file during calculations. This operation allows one record to be read in when the operation code CHAIN appears in columns 28-32 of the calculation specifications sheet. Factor 1 (columns 18-27) defines the record to be selected for processing. Factor 2 (columns 33-42) specifies the name of the chained file.

You can use indicators in columns 7-17, but result field, field length, decimal position, and half adjust (columns 43-53) must be blank. If the chained file is conditioned in the file description specifications by an external indicator, the CHAIN statement should be conditioned by that same external indicator.

Columns 54-55 should contain an entry. If the record is not found, the indicator specified in these columns turns on. No update is permitted to a chained update file when the specified record is not found; however, addition to a file is allowed. Columns 56-59 must always be blank for chain operations.

If an indicator is not specified in columns 54-55 and the record is not found, the program halts. Processing can be continued by pressing ENTER. If LR processing is already initiated, the bypass and begin new cycle option is not allowed. If the controlled cancel option is taken, files are closed but the rest of LR processing does not occur.

When chaining to a file with packed record keys, factor 1 of the CHAIN operation must have a packed length which is the same as the length of the key field in the chained file. Packed key fields can be up to eight bytes long. The following chart shows the packed equivalents for unpacked fields up to 15 bytes long:

Unpacked Length In Bytes	Packed Length In Bytes
15	8
14	8
13	7
12	7
11	6
10	6
9	5
8	5
7	4
6	4
5	3
4	3
3	2
2	2
1	1

The chain operation has two purposes:

- Random processing of an indexed, sequential, or direct file.
- Loading a direct file.

Note: When chaining to one or more files during the same RPG II cycle, record identifying indicators assigned to the chained file or files remain on throughout the cycle if the previous chain operations were executed successfully. When chaining to the same file more than once during an RPG II cycle, only the last record processed is updated during output time unless an exception output is associated with each chain operation.

Random Processing

To read a record from a sequential or direct file, the record must be identified by relative record number. To read a record from an indexed file, a record key is used for identification. The relative record number or key can be contained in a field specified for that purpose.

The chain operation requires the operation code CHAIN in columns 28-32 of the calculation specifications sheet. Factor 1 entries must be a relative record number or key, or the name of a numeric field that contains a relative record number or key. Factor 2 must contain the name of the file from which the record is read. This file is referred to as the file that is chained to or the chained file. It is this file that must be defined with a C entry in column 16 of the file description specifications sheet (see *Example*).

Direct File Load

A direct file load is defined by specifying the disk file to be loaded as a chained output file on the file description specifications sheet. In the calculation specifications, factor 1 must be a relative record number, columns 28-32 must contain the operation code CHAIN, and factor 2 must be the name of the direct disk file to be loaded. The relative record number of the input record defines the record position for each record in the direct disk file. The relative number can be all or part of a field in the input records. Such fields are used for record identification of the input record, as well as for the disk records after the disk file is loaded.

When a direct file is loaded, the disk space required for the file is filled with blanks. When a record is read in, the relative record number is used to chain to the corresponding relative record position in the disk file. The blanks at that record position are read in, and the information contained in the input record is then written on disk, replacing the blanks with data. If a record is missing from the input file when a direct file is loaded, the space reserved for that record in the disk file remains blank (until the proper record is read in later).

Once the direct file is loaded, records are inserted or changed in the file by defining the direct file as an update file processed consecutively or by the CHAIN operation. (Remember that any file defined as a chained output file is cleared entirely to blanks before any records are processed.)

You may have to allow for *synonyms* when you load a direct file. Synonyms are two or more records with the same relative record number. If you have synonyms, you can load the file in one of two ways, using multiple passes:

- Define the disk file as a direct file and clear it to blanks in your first job (by defining it as a chained output file). Once the file is cleared, one or more subsequent jobs can be run using the update function to read record locations and check for synonyms while loading the file.

- Load the direct file with records without synonyms, then run another job using the update function to identify synonyms and load them into the file.

Note: The insertion of records in direct disk files is very different from record addition to sequential or indexed files. For sequential disk files, the new record is added in at the first available position at the end of the file. The same process occurs for an indexed file, except that the record key and disk address are added to the file index. Any new records inserted in a direct disk file already have a space reserved for them. Hence, the record is inserted in its proper place, not merely added to the physical end of the file.

Example

Figure 137 shows the coding necessary to chain to and update an indexed file, MASTINV. The RECIN file consists of records sorted by item number, each record representing some quantity ordered. Item number is used as a control field. When all the quantities for one item number are added, a control break occurs. At this point in calculations, the master record for that item number must be found and updated. ITEMNO is a field containing the item number of the records presently being worked on. The chain operation uses ITEMNO to find the master record for that item number. If it is not found, a SET operation displays on the display screen the item number of the records. Note that indicator (20) turns on when the records are not found.

If the master record is found (20 not on) the total quantity for the item number is subtracted from the quantity on hand. After the total calculations, the QOH field in the master record is updated.

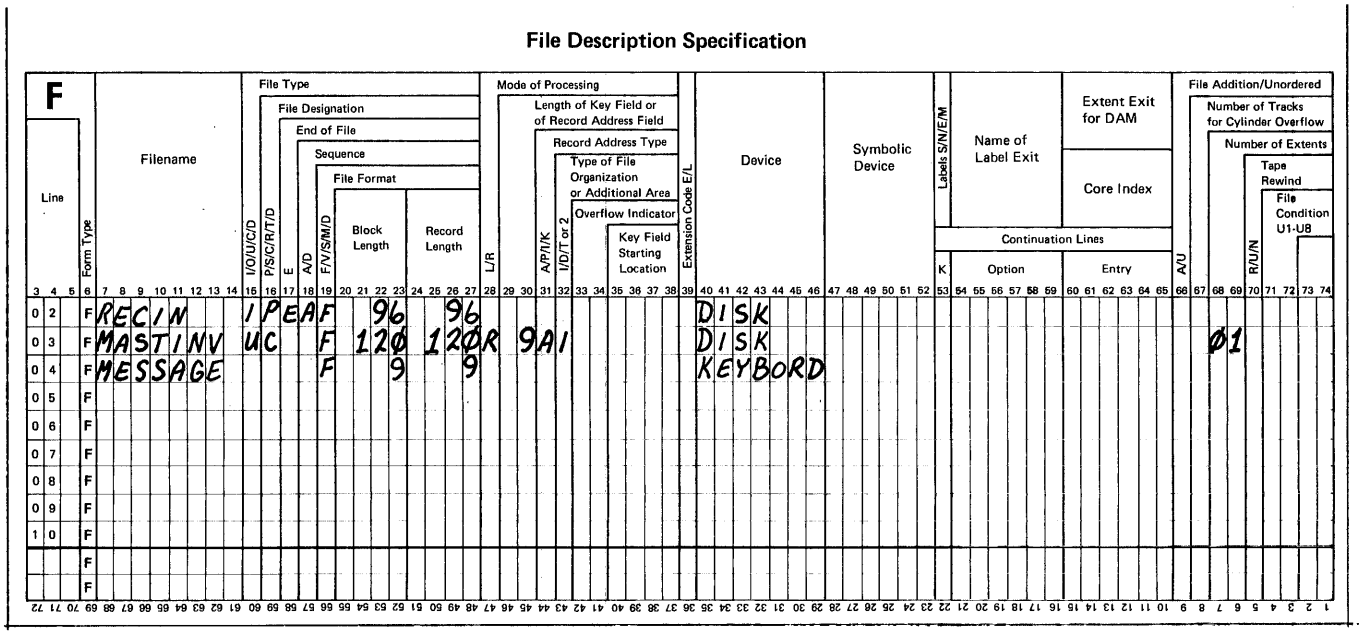
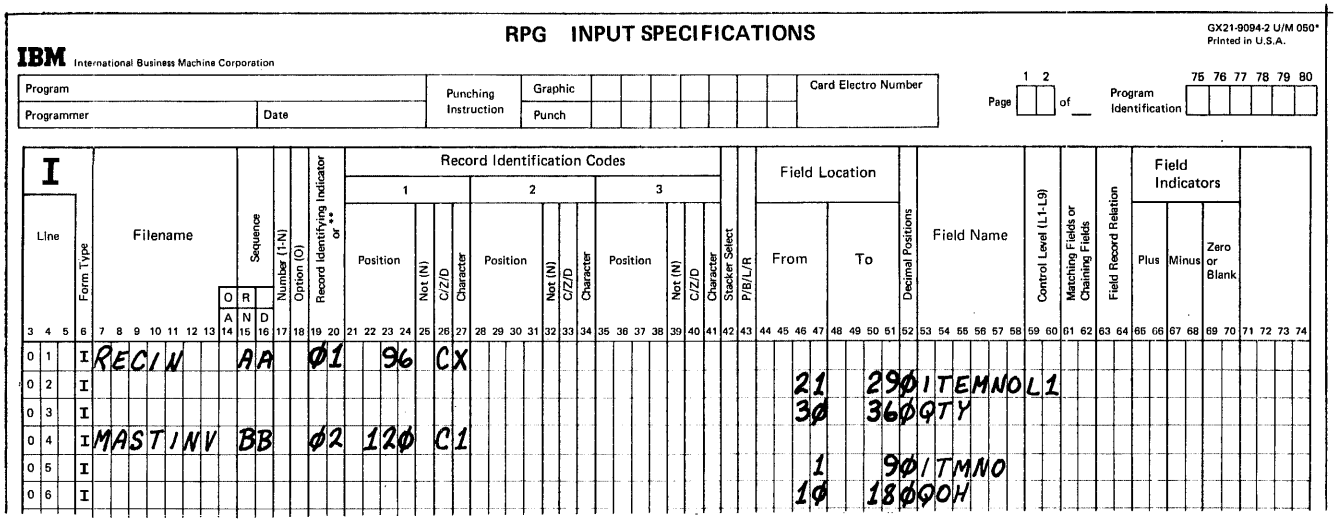
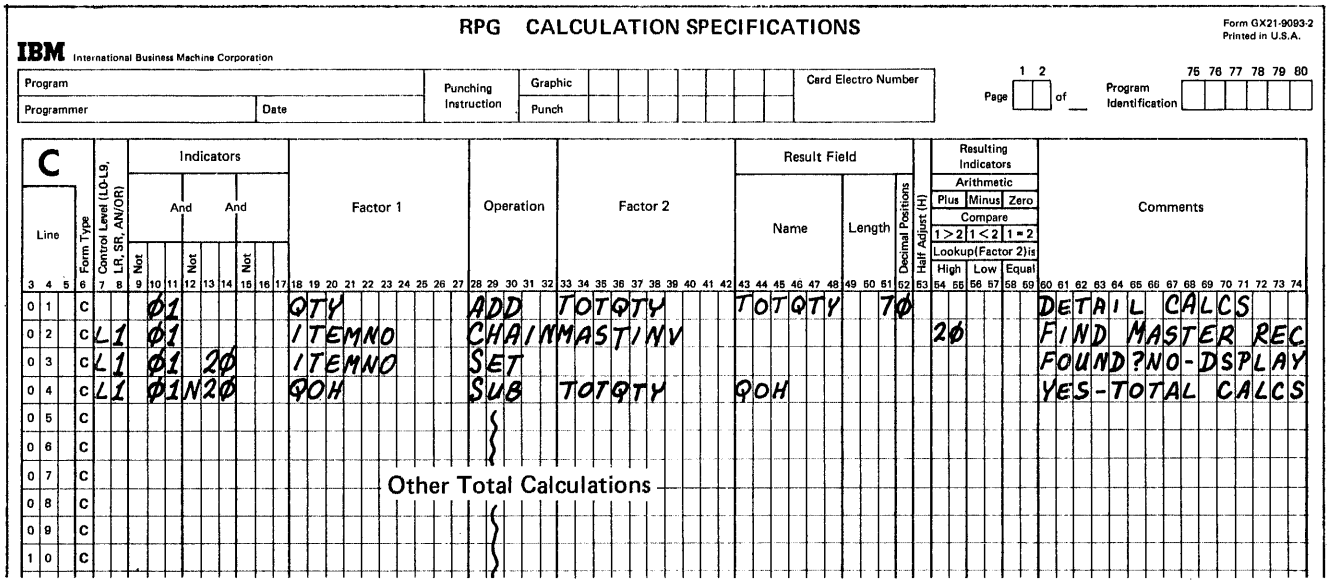


Figure 137 (Part 1 of 2). Chain Operation





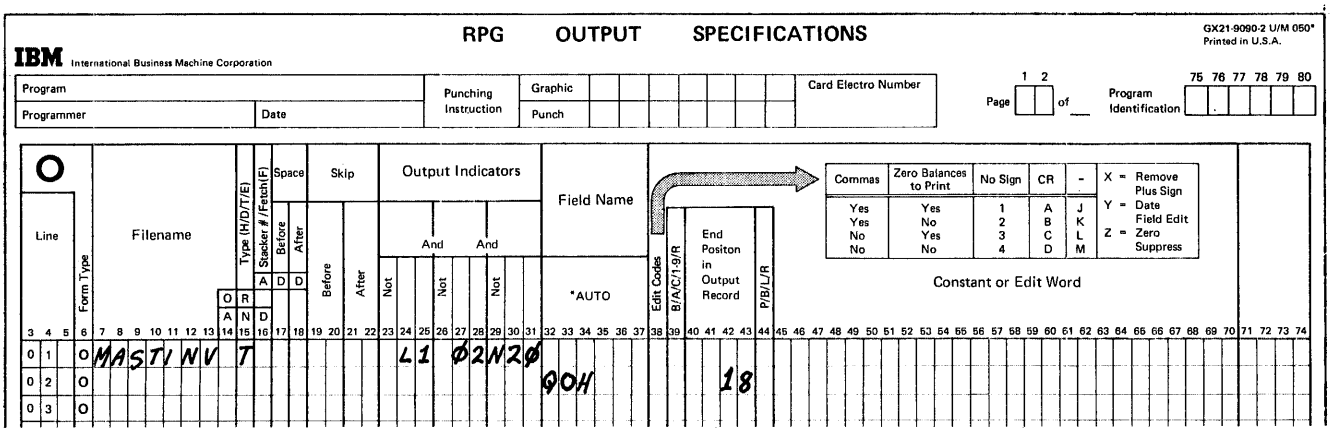


Figure 137 (Part 2 of 2). Chain Operation

Key (KEY)

The KEY operation causes a pause in calculations during which the operator can enter data from the keyboard (KEYBORD). As the data is keyed by the operator, it is displayed on the display screen. When fields are entered from the keyboard using the KEY operation and not all positions of a field are keyed:

- Numeric fields are right-justified and padded to the left with zeros.
- Alphameric fields are left-justified and padded to the right with blanks.

If no data is entered, a numeric field contains zeros and an alphameric field contains blanks.

Bypassing a Key Operation

When the KEY operation causes a pause, the operator has the option of going to the next calculation operation without keying any data. If this is necessary, the operator simply presses the ENTER key when the program is at the KEY operation. This causes any data in the KEY operation result field to be changed to zero or blank. After each KEY operation (regardless of whether data is entered or not), the operator must press ENTER before the next calculation operation can be done. See *Operation Codes, Set (SET)* for the special situation which allows the SET and KEY operations to be combined with only one depression of the ENTER key.

Note: The keying operations described in this section apply to entering data from the device KEYBORD, not to entering data from the device CONSOLE. To perform the operations described here, you must first specify a keyboard file on the file description specifications sheet.

Calculation Specifications Used for KEY Operations

Columns 7-8: Enter any valid conditioning indicator. Leave these columns blank if the KEY operation is not a part of a subroutine, or if it is to be performed only at detail time. You can also use the AN/OR relationship.

Columns 9-17: Enter any valid conditioning indicators. Leave these columns blank if the KEY operation is to be performed on every program cycle. Command key indicators can be used here if they have been specified in a SET or SETOF operation.

Columns 18-27: Enter the constant, literal, field name, table, or array element to be displayed on the display screen.

Columns 28-30: Enter the operation code KEY.

Columns 31-32: Enter the message identification code (MIC) corresponding to the message in the user message member file you want displayed on the display screen to prompt the operator to perform a KEY operation. Valid entries are 01-99.

An entry is required when columns 18-27 are blank or when a special combination of SET and KEY operations are desired (see *Special Combinations of the SET and KEY Operations*). If no user message member is specified, the system prompt 'MESSAGE INDICATOR 00nn' is displayed, where nn is the contents of columns 31-32. If columns 18-27 already contain an entry by which the keying operation is being prompted, the contents of columns 31-32 are neglected.

Columns 33-42: Leave these columns blank.

Columns 43-48: Enter the name of the field to be keyed. A field name can be from one to six characters long and must begin in column 43. The first character must be alphabetic; the remaining characters can be any combination of alphabetic and numeric characters (special characters are not allowed). Blanks must not appear between characters in the field name.

Columns 49-51: Enter the length of the keyed field if that field is not defined elsewhere. The maximum length for a numeric field is 15; the maximum length for an alphameric field is 40.

Column 52: Leave this column blank for alphameric fields. For numeric fields, enter the number of decimal positions (0-9) in the keyed field if that field is not defined elsewhere.

Column 53: Leave this column blank.

Columns 54-59: Use these columns to test the condition of numeric fields. You can test for three conditions: plus (columns 54-55), minus (columns 56-57), zero or blank (columns 58-59).

Columns 54-55 (Plus): Any valid indicator entered here is turned on if the numeric field named in columns 43-48 is greater than zero.

Columns 56-57 (Minus): Any valid indicator entered here is turned on if the numeric field named in columns 43-48 is less than zero.

Columns 58-59 (Zero or Blank): Any valid indicator entered here is turned on if a numeric field named in columns 43-48 is all zeros or if an alphameric field is all blanks.

Columns 60-74: Enter any meaningful comments you want. Figure 138 shows the different keying operations that can be performed.

The KEY operation is normally used with the SET operation. See *Operation Codes, Set (SET)* for further information on this topic.

RPG CALCULATION SPECIFICATIONS

IBM International Business Machine Corporation

Form GX21-9093-2 Printed in U.S.A.

Program | Punching Instruction | Graphic / Punch | Card Electro Number | Page __ of __ | Program Identification

Programmer | Date

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus / Minus / Zero); Compare (1>2 / 1<2 / 1=2); Lookup (Factor 2) is (High / Low / Equal)	Comments
01	C	*		KEYING OPERATIONS WITH USER MESSAGE MEMBER PROMPTS								*****
02	C											
03	C	*		THE FOLLOWING OPERATIONS WILL ALLOW THE OPERATOR TO KEY A NUMERIC								
04	C	*		FIELD (FIELDA) AND AN ALPHABETIC FIELD (FIELDB). THESE FIELDS HAVE								
05	C	*		NOT BEEN DEFINED PREVIOUSLY. THE OPERATIONS WILL BE PROMPTED BY								
06	C	*		MESSAGES ØØØ1 AND ØØØ2 FROM THE USER MESSAGE MEMBER, RESPECTIVELY-								
07	C											
08	C				KEYØ1		FIELDA	5Ø				
09	C				KEYØ2		FIELDB	12				
10	C											
11	C	*		THE FOLLOWING OPERATION WILL ALLOW THE OPERATOR TO KEY A NUMERIC								
12	C	*		FIELD WHICH HAS BEEN DEFINED PREVIOUSLY. THIS FIELD WILL BE								
13	C	*		TESTED FOR A PLUS, MINUS, OR ZERO CONDITION. THE OPERATION WILL BE								
14	C	*		PROMPTED BY USER MESSAGE ØØ3Ø.								
15	C											
16	C				KEY3Ø		AMOUNT		Ø1Ø2Ø3			
17	C											
18	C	*****		DISPLAY KEYING OPERATIONS WITH FACTOR 1 PROMPTS								*****
19	C											
20	C	*		THE FOLLOWING OPERATIONS CAUSE THE PREVIOUSLY DEFINED FIELD ,								
	C	*		(FIELDC) IN FACTOR 1 TO BE DISPLAYED ON THE DISPLAY SCREEN AND								
	C	*		THEN ALLOWS THE OPERATOR TO KEY A NUMERIC FIELD (FIELDA). THE								
	C											
	C											
	C											

Figure 138 (Part 1 of 2). Possible Keying Operations

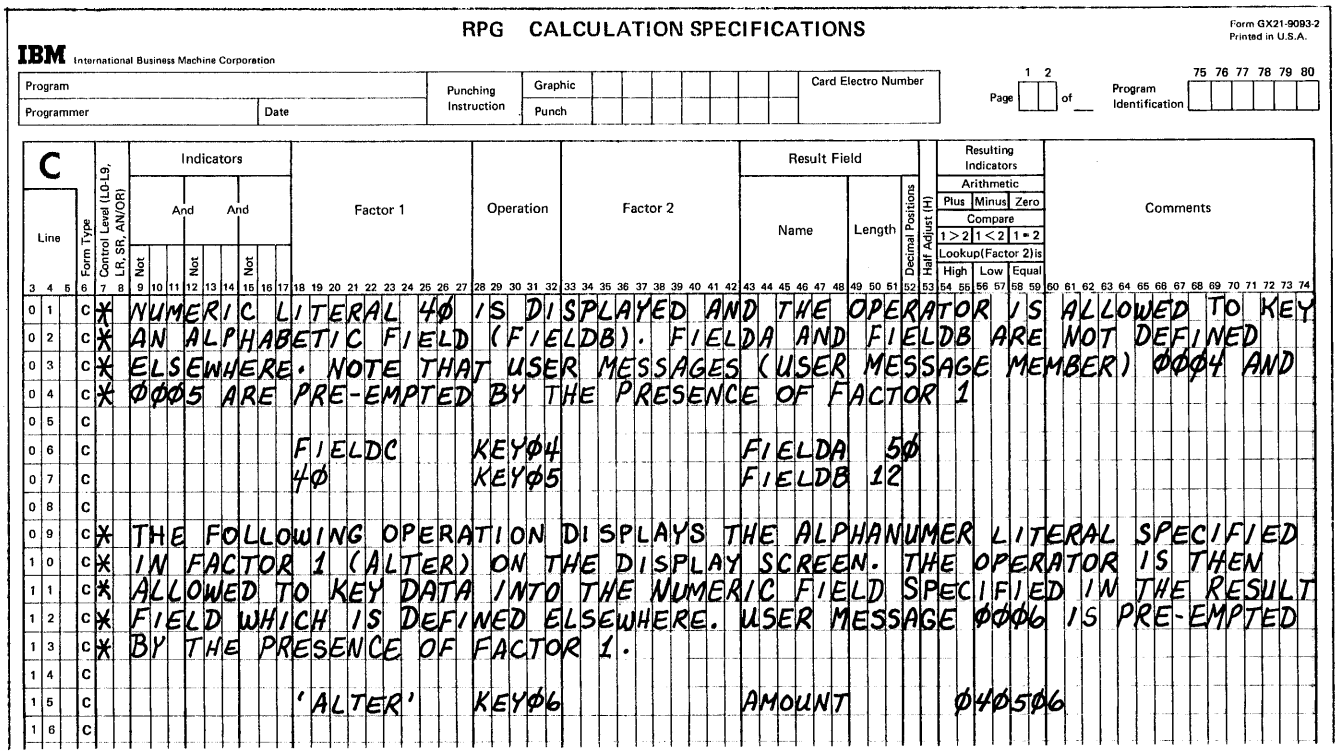


Figure 138 (Part 2 of 2). Possible Keying Operations

Using KEY and SET Operations in Subroutines

Sometimes it is necessary to write a program performing the same KEY operations at several different points in the program. Instead of writing these KEY operations and related SET operations every time they are needed, it is easier and less time consuming if they are written just once in a subroutine. Then, you can refer to the subroutine each time it is needed (see *Subroutines* for information on specifying and using subroutines).

User Message Member

The System/32 system control program (SCP) lets you create your own message members. These message members are called user message members and can contain prompts or informational messages to be displayed during your RPG II program.

You can display messages contained in user message members by using the H1-H9 halt indicators or the message indicator option of the SET and KEY operation codes. Messages displayed using the halt indicators or the SET and KEY operation codes must be formatted so MIC numbers 0001-0109 are assigned to the specific function as follows:

MIC	Function
0001-0099	Message to be displayed as specified by the nn portion of the SET and KEY operation codes (SETnn, KEYnn, where nn = 01-99).
0100	Message to be displayed at the end of a RPG II cycle when the system is finished processing outstanding halt indicators.
0101-0109	Message to be displayed at the end of a RPG II cycle in which the system has encountered H1-H9 halt indicators (0101-0109 correspond to H1-H9 respectively).

When using the SETnn and KEYnn operation codes, if a user message member is not specified prior to execution (with // MEMBER USER1 - library name statement), the system prompt 'MESSAGE INDICATOR 00nn' is displayed where nn is as defined in columns 31-32 of the SET or KEY operation and 00nn is the MIC (message indicator) to be displayed.

The first level message corresponding to the H0-H9 halt indicators may be described in more detail by using an associated second level message member. The text of a second level message may be up to 200 bytes long.

Note: No second level messages are displayed for programs using BSCA.

For information on creating user message members, see *IBM System/32 System Control Programming Reference Manual,* GC21-7593.

Set (SET)

The SET operation allows any or a combination of the following functions:

- Allows command keys to be pressed.
- To cause the field, literal, table or array element specified in factor 1 to be displayed on the display screen.
- Allows user messages (from USER1 message member) 0001-0099 to be displayed by specifying numbers 01-99 respectively in the nn portion of the SETnn and KEYnn operation codes.
- To blank an interactive data entry buffer by using ERASE in the result field.

Allowing Command Keys to be Pressed

This function of the SET operation allows you to specify command keys that the operator is allowed to press at this point in the program. When the operator presses a command key, the corresponding command key indicator is turned on. These command key indicators can be used to condition subsequent calculation or output operations. Command key indicators remain on until they are used again in a SET operation or until they are turned off using the SETOF operation code. When the program is at a particular specification line, you can give the operator the option of pressing one to three command keys. For each command key to be pressed, the operator first presses the CMD key and then presses the digit key corresponding to the command key indicator (KA-KN, or KQ). After all command responses have been entered, the operator presses the ENTER key.

If you do not want to press any command keys, the operator responds to the SET operation by pressing only the ENTER key. This causes the indicators to be turned off and is called a *null response*. Using this null response in your programs is not recommended because of the possibility of an accidental null response. For example, if the operator neglects to press the CMD key before pressing the appropriate digit key, a null response occurs.

Calculation Specifications Used for SET Operations

The following specifications describe all SET operations. The calculation specifications required for a SET operation vary depending upon which function, or combination of functions, you want to perform (see Figure 139 for a summary of these specifications).

Columns 7-8: Enter any valid conditioning indicator. Leave these columns blank if the SET operation is not a part of a subroutine, or if it is to be performed only at detail time.

Columns 9-17: Enter any valid conditioning indicators for any SET operation. Leave these columns blank if the SET operation is to be performed on every program cycle.

Columns 18-27: Enter the constant, literal, field name, table or array element to be displayed on the display screen.

Columns 28-30: Enter the operation code SET.

Columns 31-32: These columns may contain a message indicator for prompting to a SET operation. An entry is required when command keys are specified in columns 54-59 and columns 18-27 are blank. Valid entries are 01-99. An entry indicates the message identification code (MIC) in the user message member that prompts the SET operation on the display screen. If no user message member is specified, the system prompt 'MESSAGE INDICATOR 00nn' is displayed, where nn is the contents of columns 31-32.

Columns 33-42: Leave these columns blank when the only function performed by the SET operation is allowing command keys to be pressed or displaying factor 1 or a MIC 01-99 on the display screen. When the erase function specified in the result field is to be performed on a console file, enter the console filename.

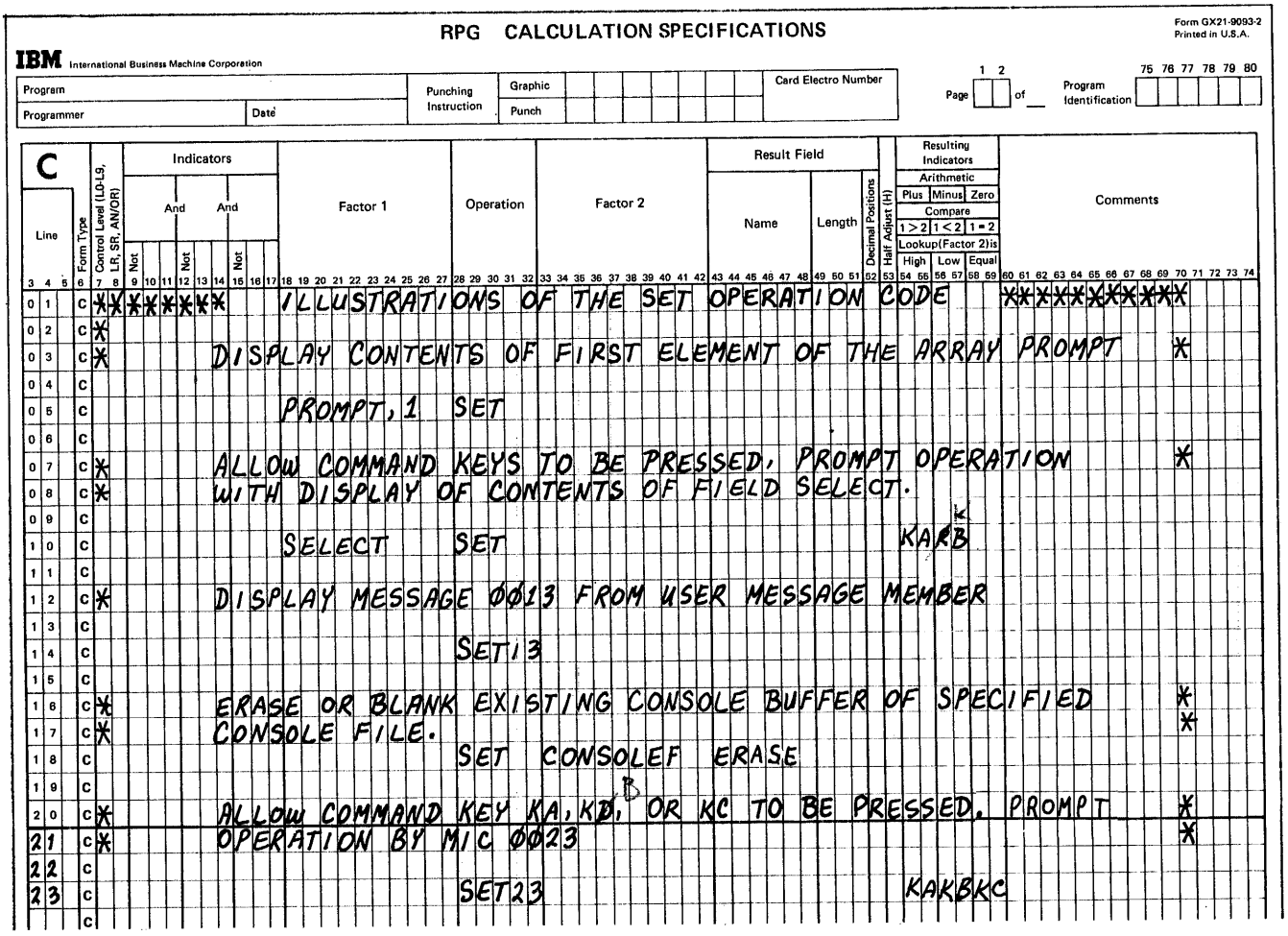


Figure 139. Summary of Calculation Specifications for SET Operations

Columns 43-48: Leave these columns blank when the only function performed by the SET operation is allowing command keys to be pressed or displaying factor 1 or MICs 01-99 on the display screen. Enter erase to clear input from the console file.

Columns 49-53: Leave these columns blank.

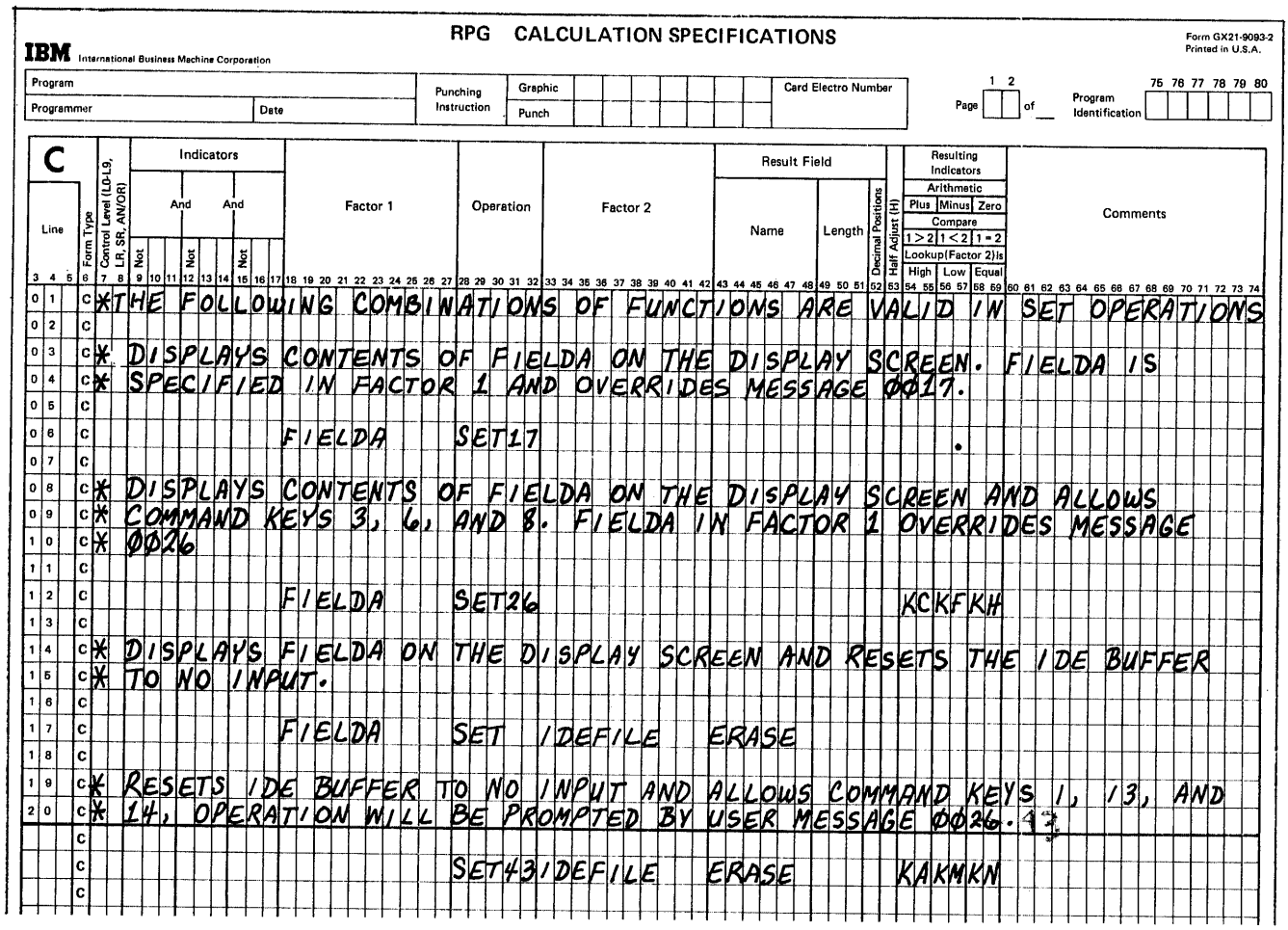
Columns 54-59: Enter the command keys (KA-KN, KP, KQ) that the operator is allowed to press when the program has paused at this specification line. You can specify one to three command keys. If only one or two command keys are specified, they can be entered in any of the three sets of columns. When the operator presses a command key entered in these columns, that command key indicator remains on until it is used again in a SET operation or until it is turned off using the SETOF operation code. A halt occurs if the operator presses a command key other than those specified in columns 54-59 of a SET operation.

Either factor 1 or message indicators in columns 31-32 must be specified on a SET operation when command key indicators are entered in columns 54-59. If both factor 1 and message indicators are present, the message indicators are ignored.

Several lines of prompt can be displayed on the display screen before the system halts for input by stacking set operations with a factor 1 (or MIC) and no command key entries. The system does not halt until a keyboard function is encountered.

Columns 60-74: Enter any meaningful comments.

Examples: Figure 139 shows the coding needed to perform the SET operation. Figure 140 shows the possible combinations of these functions.



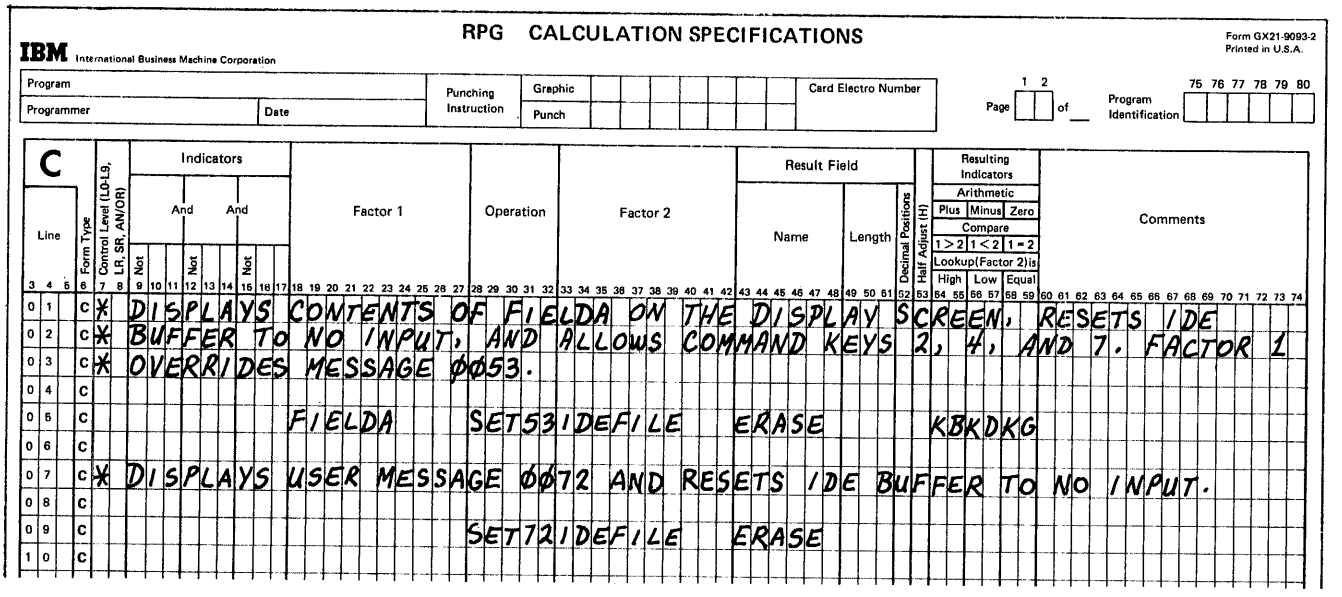


Figure 140. Possible Combinations of Functions Performed by SET Operation

Special Combinations of the SET and KEY Operations

Normally, the operator must press the ENTER key after each KEY operation or after command keys specified in a SET operation are pressed. However, it is possible to combine these operations so that the operator can press command keys (specified in columns 54-59 of a SET operation) and key a field (specified in a KEY operation) with only one depression of the ENTER key. This is only possible if:

- The SET operation immediately precedes the KEY operation.
- The SET and KEY operations are conditioned by the same indicators (columns 7-17). Indicators for both operations must be specified in the same order.
- The SET and KEY operations contain the same message indicators (Figure 141).

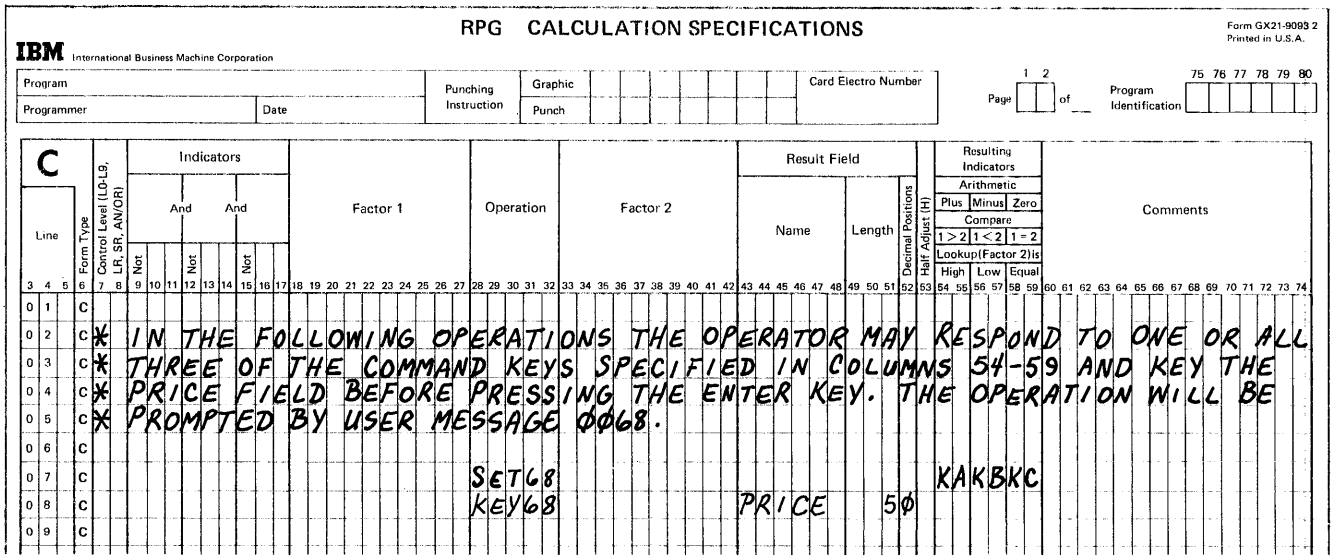


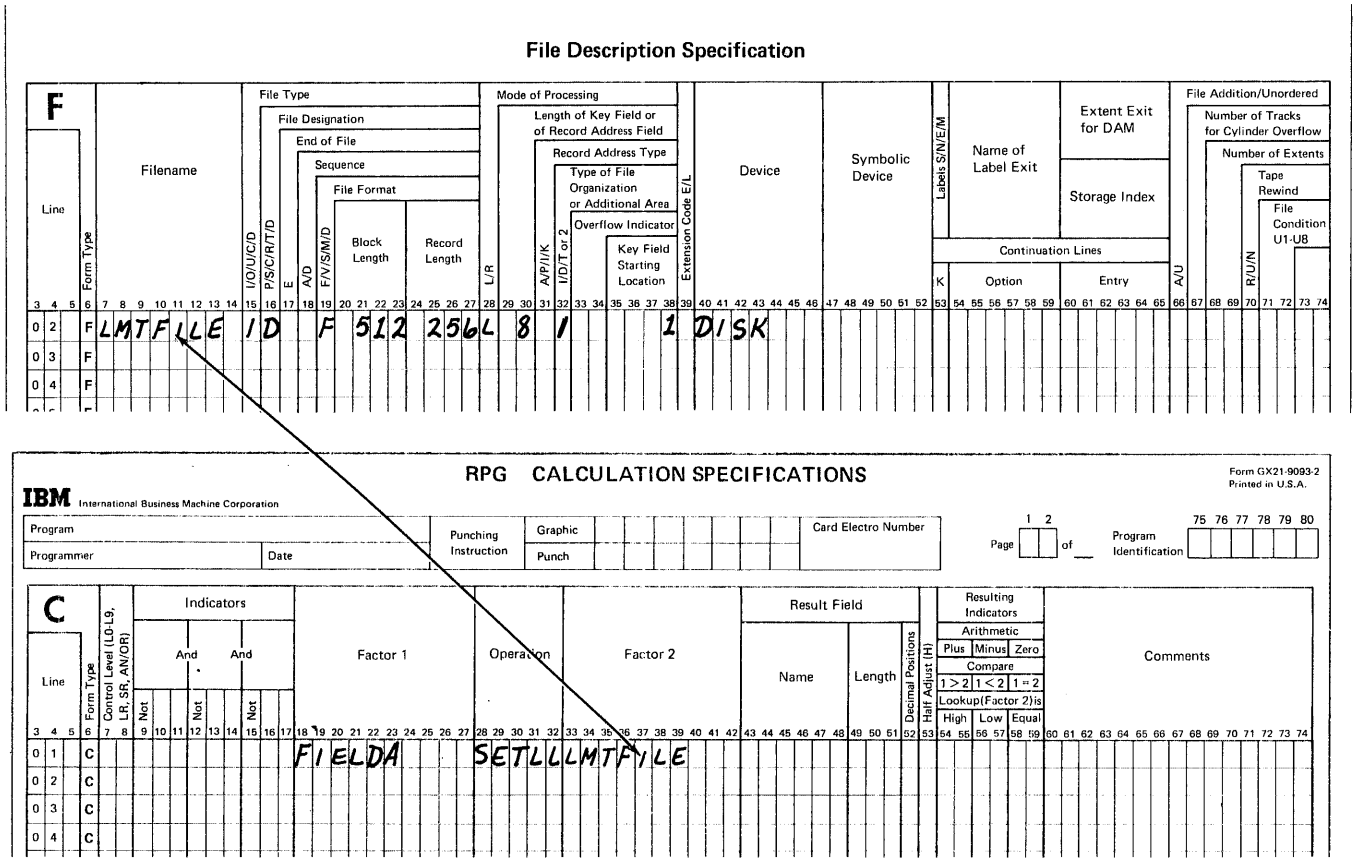
Figure 141. Special Combination of SET and KEY Operations

Set Lower Limits Operation (SETLL)

The SETLL operation allows the lower limits for an indexed sequential demand file being processed within limits to be set during calculations.

Factor 1 must contain a field name or literal representing the value of the lower limit being set. The length of the field or literal must be equal to the length of the key specified on the filename in factor 2.

Factor 2 must contain the name of the file for which the lower limit is to be set. In the event a read is performed to the file prior to a SETLL instruction being issued, the record with the lowest key in the file is fetched. Refer to Figure 142 for an example of SETLL coding.



Note: FIELDA would be defined as an 8-position alphameric field.

Figure 142. SETLL Operation Coding

DEBUG OPERATION

The debug operation is an RPG II function that helps you find errors in a program which is not working properly. This code causes either one or two records to be printed. They contain information which is helpful for finding programming errors.

Debug (DEBUG)

The DEBUG operation code can be placed at any point or at several points in the calculation operations. Whenever it is encountered, either one or two records are printed depending upon the specifications entered. One record contains a list of all indicators which are on at the time the DEBUG code was encountered. The other shows the contents of any one field.

Note: The decimal point is not printed.

Specifications

Factor 1 is optional; it can contain a literal or the name of a field of one to eight characters to help identify the particular DEBUG operation. The name entered here is printed on record 1. Factor 2 must contain the name of the output file on which the records are written. The same output filename must appear in factor 2 for all DEBUG statements in a program. The result field can be a field or array whose contents you want to appear on record 2. Any valid indicator can be used in columns 7-17. Columns 49-59 must be blank.

The operation code produces results only if the proper entry (1 in column 15) was made in the control specifications. If the control specifications entry was not made, the operation code DEBUG and the indicators conditioning it are treated as a comment. See *Column 15 (DEBUG)* under *Control Specifications* for more information.

Records Printed for Debug

Record 1 is required. It is printed in the following format:

Print Positions	Information
1-8	DEBUG-
9-16	Literal or contents of field entered in factor 1 (optional)
17	Blank
18-32	The words INDICATORS ON-
33-any position (depending on length of field)	The names of all indicators which are on, each separated by a blank; the word NONE, if no indicators are on. More than one record may be needed.

Record 2 is optional and is printed only when there is a result field. The record is printed in the following format:

Print Positions	Information
1-14	The words FIELD VALUE-
15-any position (depending on length of field)	The contents of the result field (up to 256 characters). If the result field is an array, more than one record may be needed to contain the array.

Overflow occurs when a record is printed on the overflow line or when a space or skip instruction causes forms movement past the overflow line. When overflow occurs, the overflow indicator you specified on the file description specifications sheet is turned on. This indicator remains on for the rest of the program cycle; it is turned off after all lines conditioned by an overflow indicator have been printed.

According to RPG II logic, there are three times in the program cycle when an overflow indicator can be turned on: (1) at total time, (2) at detail time, and (3) at calculation time if exception output is used. There is only one time in the program cycle, however, when the RPG II program checks to see if the overflow indicator is on. This is right after all total records are printed.

If overflow occurs when a total record is printed, the appropriate overflow indicator turns on, and the remainder of the total lines are printed. A test then determines whether or not the overflow indicator is on. If it is, all total, detail, and heading lines conditioned by the overflow indicator are printed.

If overflow occurs when a detail record is printed, the appropriate overflow indicator turns on, and the remainder of the detail lines are printed. The next record is read. If this record causes a control break, total calculations and total output operations are performed, followed by overflow output.

When the printer has reached the end of a printed page, the RPG II language allows you to do one of four things:

- Advance to the top of the next page and continue printing.
- Ignore the fact that the end of the page is reached and continue printing.
- Print special lines at the bottom of the page and at the top of the new page.
- Alter the basic RPG II overflow logic by fetching the overflow routine at detail time, total time, or during calculations using exception output.

AUTOMATIC PAGE FORMATTING

In the first case, RPG II automatically advances to the next page and continues printing once overflow has occurred. Line counter specifications can be entered, but they are optional. Do not specify an overflow indicator on either the file description specifications sheet or the output specifications sheet.

When RPG II handles overflow automatically, printing is set to begin on line 06 once the operator has positioned the first page. This means that detail lines begin on line 06 for all pages except the first. When RPG II handles overflow, printing ends on line 60 or on the defined overflow line.

While it is convenient and easy to allow RPG II to handle overflow automatically, it allows you little control over how the report looks.

CONTINUOUS LISTINGS

In the second case mentioned, you want to print some reports in the form of a continuous listing. In such a case, you must make an entry which causes the automatic handling of overflow and advancing of forms to be discontinued. To do this, make the following entry on the file description specifications sheet. Assign an overflow indicator in columns 33-34 of the specification line used to describe the printer file. This entry causes overflow to be ignored and a continuous listing to result. No entries are required on either the line counter specifications sheet or the output specifications sheet.

USING OVERFLOW INDICATORS TO CONTROL PAGE FORMATTING

In the third case, RPG II allows you to print special lines at the bottom of the page and at the top of a new page once overflow has occurred. To do this, code the following specifications:

File Description Specifications

Assign an overflow indicator in columns 33-34 of the specification line used to describe the printer file (Figure 143).

Line Counter Specifications (Optional)

- Enter the appropriate filename in columns 7-14.
- Specify the number of lines per page in columns 15-17.
- Enter FL in columns 18-19.
- Specify the print line to be used as the overflow line in columns 20-22.
- Enter OL in columns 23-24 (Figure 144).

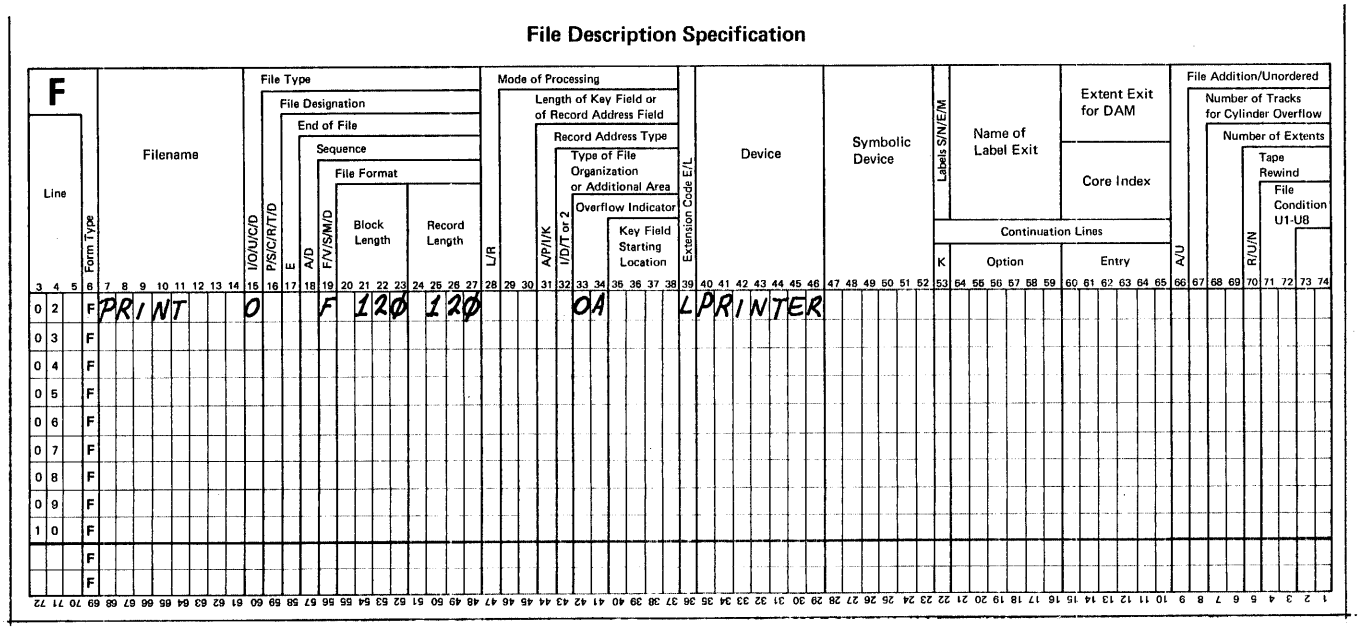


Figure 143. Assigning an Overflow Indicator to a Printer File

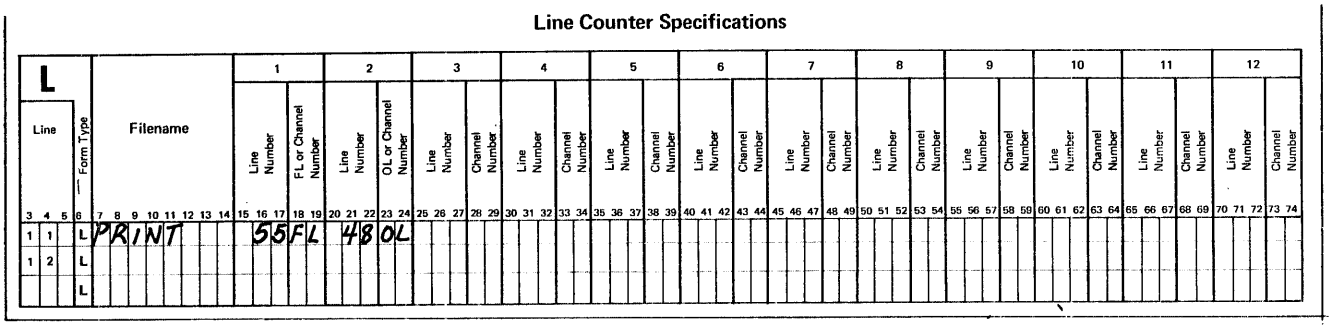


Figure 144. Page Formatting (Line Counter Specifications)

Output Specifications

- Use the overflow indicator assigned by the file description specifications sheet to condition those lines you want printed on each page. Usually these lines are total lines which must be printed at the bottom of each page or heading lines which must be printed at the top of each new page (Figure 145).

- The normal sequence is:

  1. Print page totals
  2. Skip to new page
  3. Print heading lines

If you have specified an overflow indicator and are using it to condition output lines, the following steps occur when the overflow line (end of a page) is sensed:

1. Detail lines are printed (if that part of the program cycle is not already completed).
2. Total lines are printed.
3. Total overflow lines are printed if conditioned by the overflow indicator.
4. Forms advance to the next page if indicated by the skip specifications on a heading line or total line.
5. Heading and detail lines are printed if conditioned by overflow indicators.

USING OVERFLOW INDICATORS AND CONTROL LEVEL INDICATORS

Often you want each page to contain information from only one control group. (Information from one group may require several printed pages, however.) You might also want each page to have headings identifying the type of information on the page. For these cases use both type control level indicators and the overflow indicators. Together they condition when both headings and group information are to be printed.

A new page should advance either when the overflow line is reached (the overflow indicator you assigned is on) or when there is a change in a control field (control level indicator is on). You must specify that each indicator causes a new page to be advanced by specifying a skip to the appropriate printing line on a page. If the control level has changed and the overflow condition has occurred at the same time, it is possible to duplicate an output line (one called for by the overflow indicator, the other by the control level indicator). A blank page can appear in your report as a result.

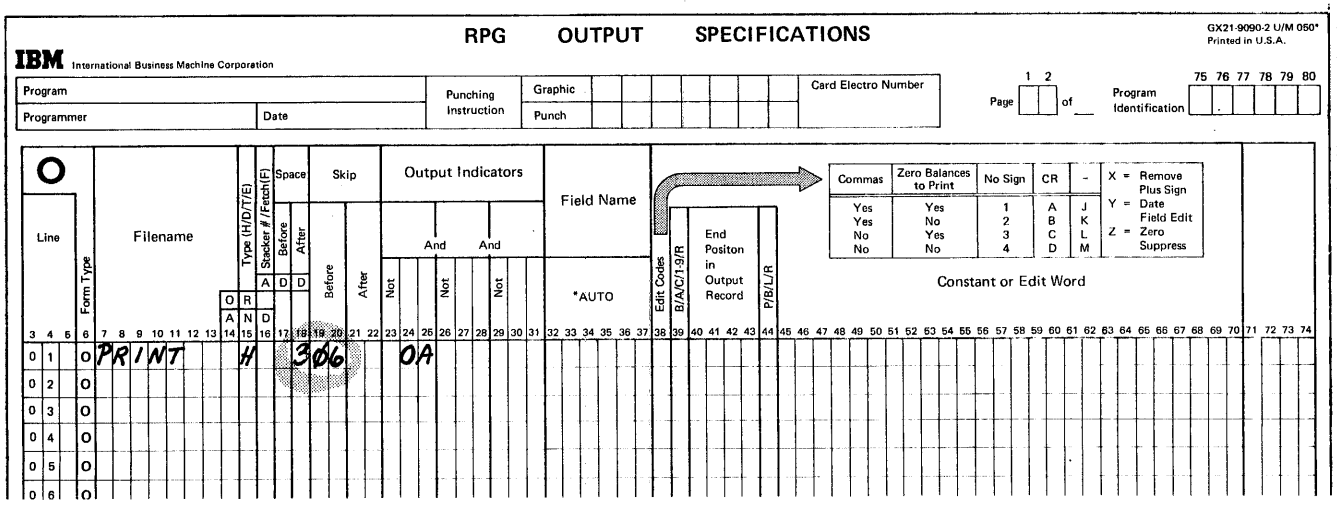


Figure 145. Page Formatting (Output Specifications)

Figure 146 shows the coding necessary for printing headings on every page: first page, every overflow page, and each new page to be started because of a change in control fields (L2 is on). Line 01 allows the headings to be printed at the top of a new page (skip to 01) only when an overflow occurs (OA is on and L2 is not on).

Line 02 allows printing of headings on the new page only at the beginning of a new control group (L2 is on). This way, duplicate headings caused by both L2 and OA being on at the same time do not occur. Line 02 allows headings to be printed on the first page after the first record is read. This is true because the first record always causes a control break (L2 turns on), if control fields are specified on the record.

Figure 147 shows the necessary coding for the printing of certain fields on every page; a skip to 06 is done either on an overflow condition or on a change in control level (L2). The NL2 indicator in line 01 prevents the line from printing and skipping twice in the same cycle.

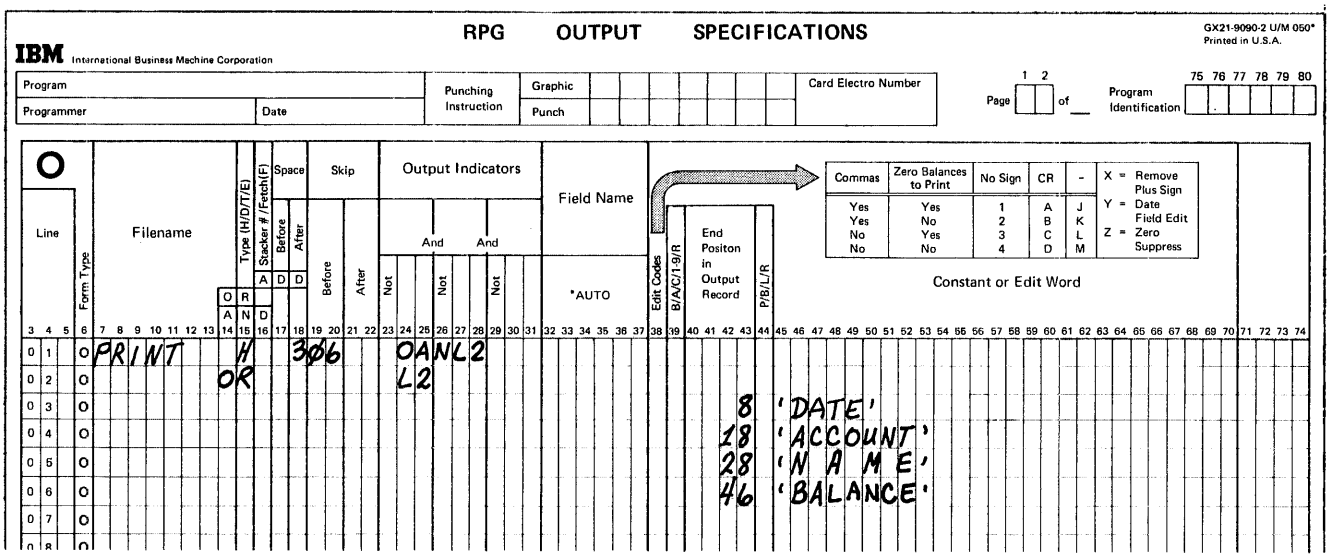


Figure 146. Printing Headings on Every Page

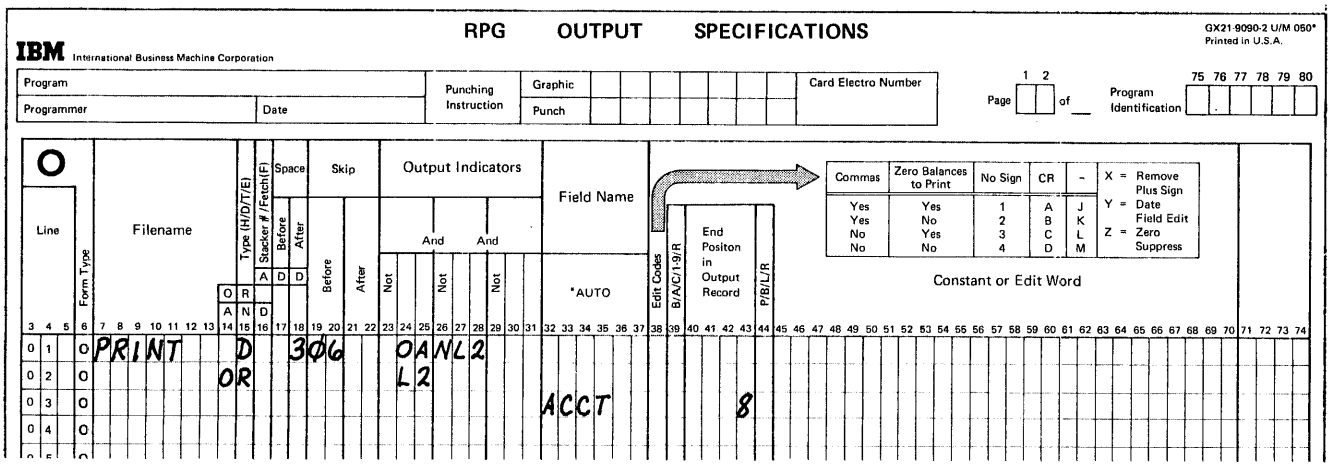


Figure 147. Printing Fields on Every Page

FETCHING THE OVERFLOW ROUTINE

When the overflow line is reached during detail time, the remaining detail lines, total lines, and overflow total lines (total lines conditioned by the overflow indicator) are printed on the page even after overflow has occurred. Therefore, you must leave enough space between the overflow line and the actual end of page for all these lines to print.

However, you can run into problems when you do this. If a different number of detail or total lines can be printed depending upon when overflow occurs, you may find that you have allowed too little or too much space below the overflow line. Assume that your job involves printing two detail records or four total records in a program cycle. Sometimes, both detail and total records may be printed in the same program cycle. In this case, the overflow indicator can be turned on: (1) when any one of the detail records is printed or (2) when any one of the total records is printed.

If overflow occurs when the first detail record is printed, the second detail line and all total lines (provided a control break has occurred) will print before forms advance. Printing is done on the perforation and run over onto the next page unless you specified the overflow line high enough on the page.

On the other hand, suppose overflow was caused by the third total record instead of the first detail record. Only one more total line prints before forms advanced. In this instance, you may find that you have allowed so much room between the overflow line and the end of the page that only half a page is actually used.

To prevent printing over the perforation and at the same time use as much of each page as possible, you can fetch the overflow routine. Fetch overflow specifications allow you to alter the basic RPG II overflow logic. You can cause forms to advance when total or detail records are printed instead of waiting for the usual time. To fetch the overflow routine, enter an F in column 16 of the output specifications sheet for any detail or total record.

The fetch overflow specification causes the computer to check if the overflow indicator is on before it prints total or detail records. The test is made each time an F is entered in column 16 of the output specifications sheet. If the overflow indicator is on when the test is made, all operations conditioned by the overflow indicator are performed immediately.

When the overflow routine is fetched (F in column 16 of the output specifications sheet), the following operations are done:

1. All total lines conditioned by the overflow indicator are printed.

2. Forms advance to a new page when a skip to 06 is specified in a line conditioned on an overflow indicator.

3. Heading lines and detail lines conditioned by the overflow indicator are printed.

4. The line that fetched overflow is printed.

5. Any detail and total lines left to be printed for that program cycle are printed.

You should fetch the overflow routine (F in column 16) only when: (1) printing a particular line causes overflow and (2) if it did, there would not be enough space left on the page to print the remaining detail and total output lines and heading lines conditioned by the overflow indicator.

Note: Fetch overflow cannot be specified when an overflow indicator is specified in columns 23-31 on the same specification line. If this condition does occur, fetch overflow is not performed.

To determine when to fetch the overflow routine (F in column 16 of the output specifications sheet), you must study all possible overflow situations. By counting lines and spaces, you can calculate what happens if overflow occurs on each detail and total line.

Figure 148 shows the use of the fetch specification. Total lines 03 and 09 can fetch the overflow routine. They do this, however, only if the overflow is sensed prior to the printing of one of these lines. When it is time for the operation in line 03 to be performed, a test is made to see if the internal overflow indicator is on. If it is, the overflow routine is fetched causing these operations to be performed:

1. All total lines conditioned by the overflow indicator are printed (if a control break has occurred).

2. Forms are advanced provided a skip to 06 has been specified in a line conditioned by the overflow indicator.

3. Heading lines conditioned by the overflow indicator are printed (line 01).

4. An internal overflow indicator is turned off so that overflow processing will not be repeated until another page overflows.

   Note: The external overflow indicator is not turned off at this time.

5. The record specified on line 03 is printed.

IBM International Business Machines Corporation — RPG OUTPUT

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # /Fetch (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators
01	O	PRINTER	H			3	ϕ5		OA
02	O								
03	O		T	F		1			L1
04	O								
05	O		T			1			L1
06	O								
07	O		T			1			L1
08	O								
09	O		T	F		1			L1
10	O								
11	O		T			1			L1
12	O								
13	O		T			1			L1
14	O								
15	O		T			1			L1
16	O								

Figure 148. Uses of Fetch

Another test is made to see if the internal overflow indicator is on because of the fetch specification (F in column 16) in line 09. If line 03 causes the form to advance, the internal overflow indicator is not on at this time. The total record in line 09 prints normally.

However, if the records in line 03, 05, or 07 were printed on the overflow line, the internal overflow indicator would be on. The specification in line 09 then fetches the overflow routine.

OVERFLOW PRINTING WITH EXCPT OPERATION CODE

Overflow indicators cannot condition an exception line, but can condition fields within an exception line. The use of the EXCPT operation code with exception lines (E in column 15 of the output specifications sheet) causes only exception lines to be printed during calculation time. If the overflow line is sensed when an exception line is printed, the overflow indicator turns on as usual, but overflow processing does not occur until another exception line conditioned to print (and with fetch overflow specified) is encountered.

The actual overflow output lines (totals and/or headings) must be coded as H, D, or T types. Using fetch overflow causes the H, D, or T overflow output lines to be printed if the overflow indicator is on. The overflow output lines are printed prior to the printing of the line on which fetch overflow is specified. The user can also force overflow by issuing a SETON of the appropriate overflow indicator prior to the EXCPT operation code, provided fetch overflow is specified.

ASSIGNING OVERFLOW INDICATORS

When using the overflow indicator to condition overflow printing, consider the following:

- Overflow indicators can be turned on and off by the operation codes SETON and SETOF.
- Spacing past the overflow line causes the overflow indicator to turn on.
- Skipping past the overflow line to any line on the same page turns the overflow indicator on.
- Skipping past the overflow line to any line on the new page does not turn the overflow indicator on.
- A skip to a new page specified on a line not conditioned by an overflow indicator causes the overflow indicator to turn off.

Figure 149 shows the setting of overflow indicators during the normal overflow routine and during a fetched overflow routine for both normal output and exception output. The lefthand portion of the graph shows when the indicators are on or off in relation to the general program cycle. For example, during normal output, if a detail line is printed on the line number specified as the overflow line, the overflow indicator turns on. It remains on until the end of the next program cycle. The solid black lines indicate that the indicator is on. The dashes show a connection between the end of one cycle and the start of the next.

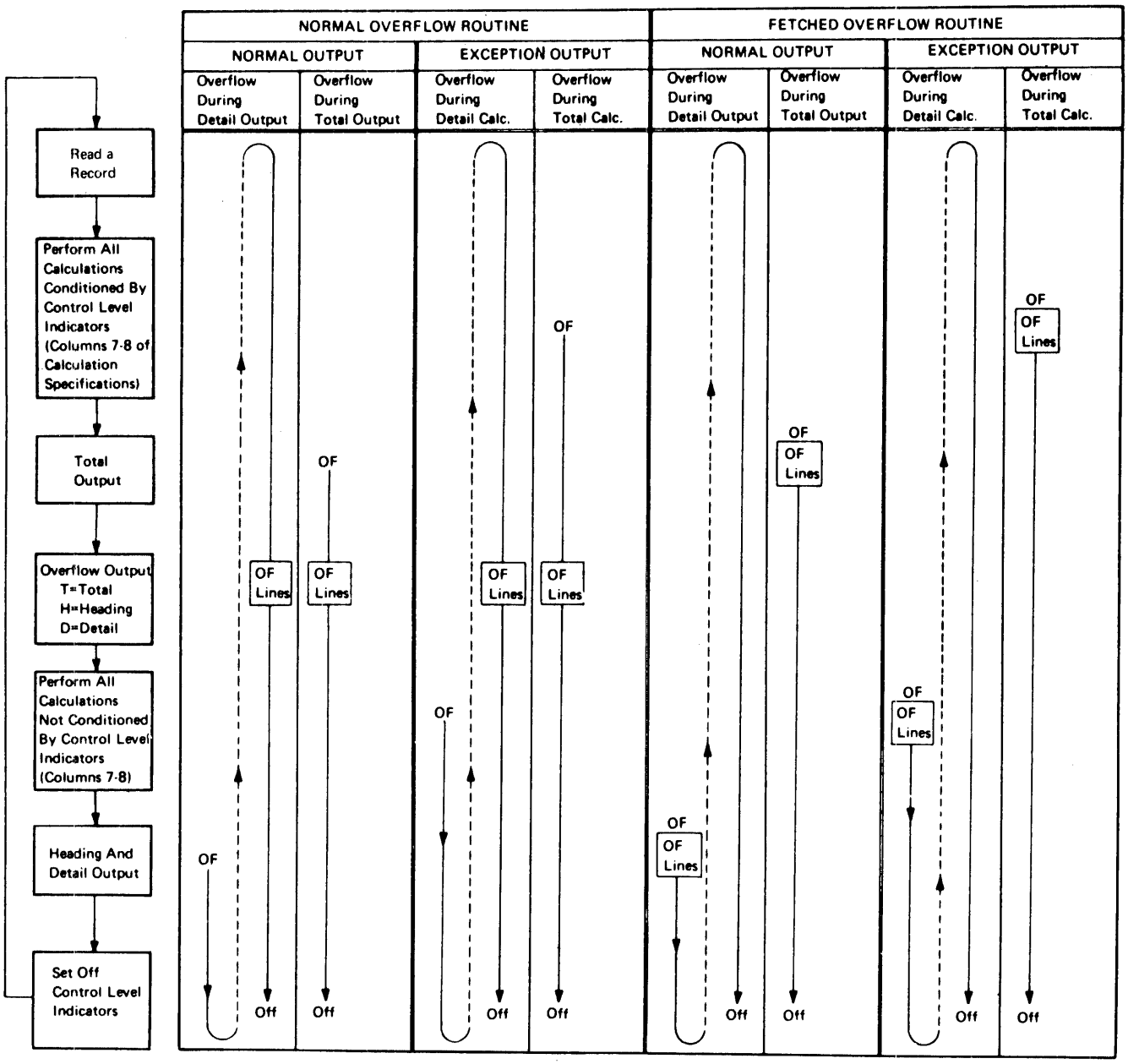


Figure 149. Overflow Printing: Setting of the Overflow Indicator

For each record that is processed, the RPG II object program goes through the same general cycle of operations. Within each program cycle, calculation and output operations can be performed at two different times: total time and detail time. First, total calculation and total output operations (those conditioned by control level indicators) are performed. Second, all detail calculation and detail output operations are performed. (Detail calculation and output operations are those not conditioned by control level indicators in columns 7-8 of the calculation specifications sheet or a T in column 15 of the output specifications sheet.) Total calculation and total output operations are performed on data accumulated for a control group. Detail calculation and detail output operations are performed for individual records as they are read, provided conditioning indicators are satisfied. See *General RPG II Program Logic* under *Introduction* for a general description of the logic flow.

The specific steps taken in one program cycle are shown in Figure 150. The item numbers in the following description refer to the numbers in the figure. A program cycle begins with step 1 and continues through step 26.

1. All data files to be used by the RPG II object program are *opened*; that is, they are prepared to be processed by the object program. Preexecution-time tables and arrays are loaded before the first program cycle.

2. The object program performs all output conditioned by the 1P indicator. This output is performed only once per job and does not fall within the program cycle (steps 3 through 26).

3. The object program performs all specified heading and detail output operations whose conditions are satisfied. This includes specifications that are conditioned by the overflow indicator if the overflow routine has been fetched.

4. The object program tests to determine if the overflow line was encountered during detail calculations in the previous cycle or when heading and detail records were written in the current cycle. If so, the overflow indicator turns on. Otherwise, the indicator turns off, unless the overflow routine was fetched in step 3.

5. The object program tests the halt indicators. If the halt indicators are off, the program branches to step 6.
   a. The execution of the program is stopped once for each halt indicator that is on. The operator selects one of three options: continue, controlled cancel, or immediate cancel.
   b. If the operator desires to continue the job, the program returns to step 5 to test for other halt indicators. If the operator selects one of the cancel options, a branch is taken to step 35.

6. All record identifying indicators and indicators 1P, L1-L9, and H1-H9 are turned off.

7. The program tests to see if the LR indicator is on. If it is, the program branches to step 27.

8. The program tests to see if the keyboard is the primary file. If it is, the program branches to step 18.

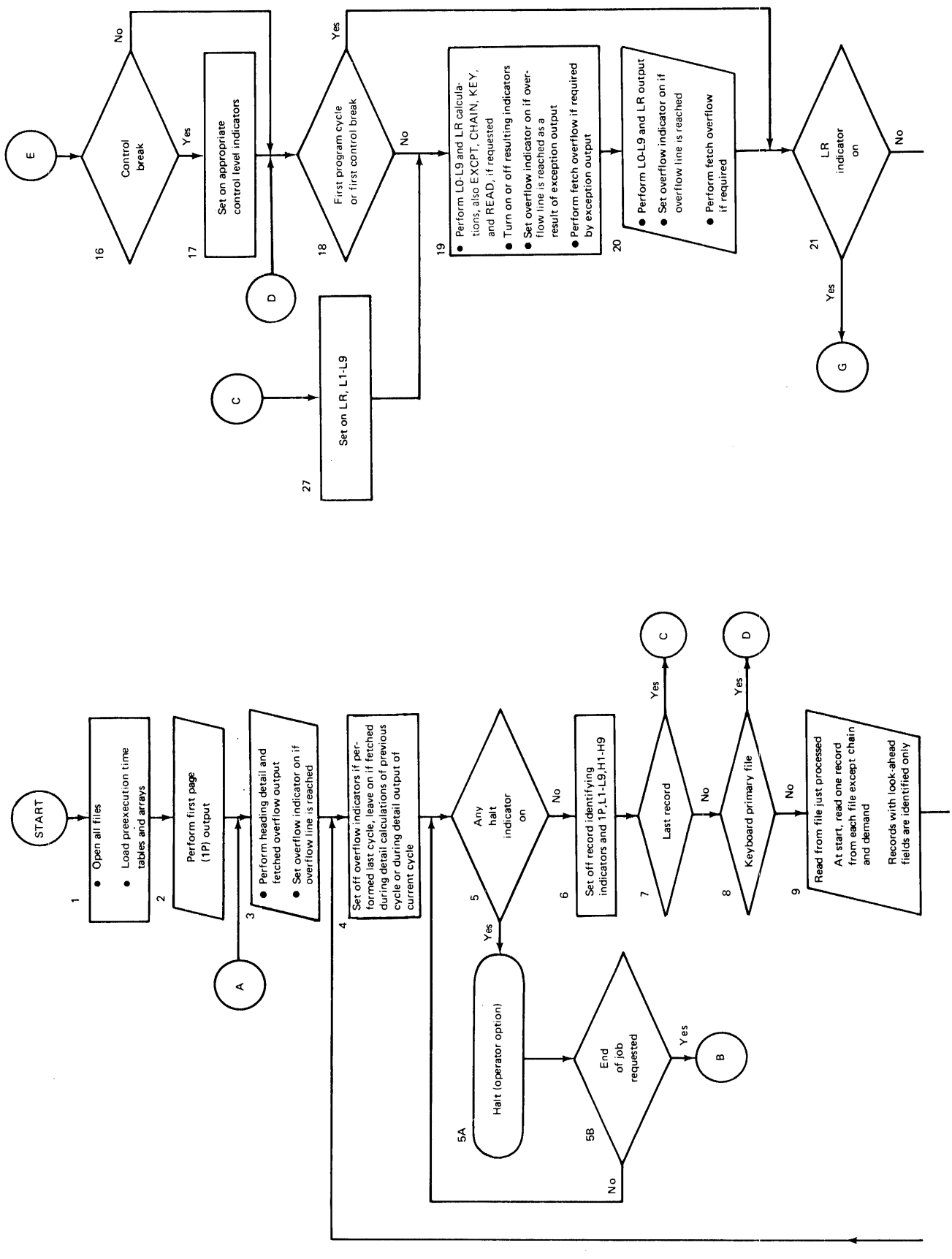
9. The program reads (and translates, if necessary), the next input record. At the beginning of processing, one record from each input file (except forced files and demand files) is read. If the file has look ahead fields, it is read only on the first cycle. After that, records with look ahead fields are identified only.

10. The program tests to determine if the record is an end-of-file record. If an end-of-file condition has occurred, the program branches to step 12.

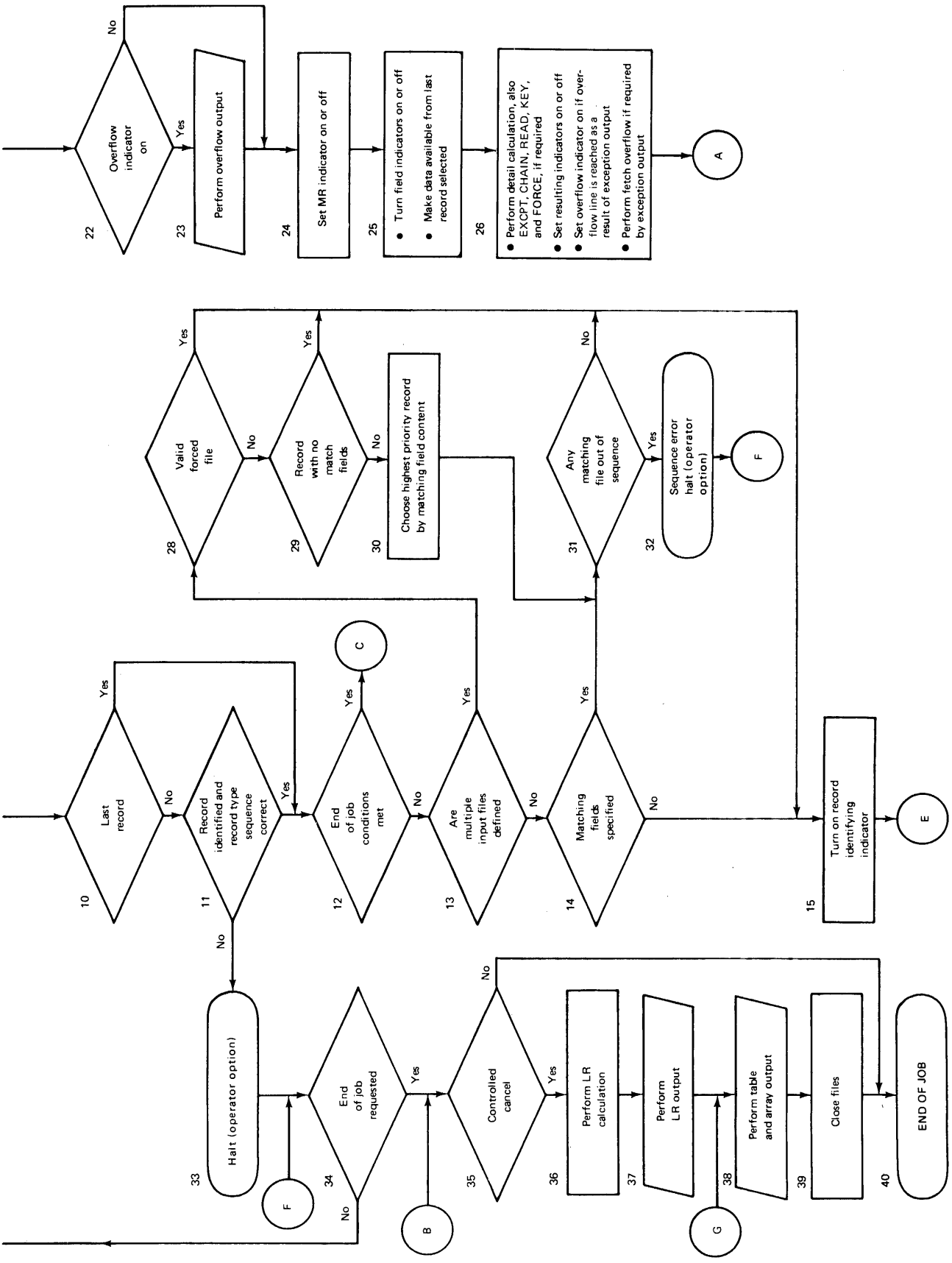
11. If end of file has not occurred, a test determines if the input records are in the sequence specified for them on the input specifications sheet. If the sequence is incorrect, the program branches to step 33. The program also branches to step 33 if nonsequential input records are specified and the record cannot be identified.

12. If end-of-job conditions have been met, a branch is taken to step 27. All files for which an E is specified in column 17 of the file description specifications sheet must be at end of file.

13. When multiple input files are used, it is necessary to select the next record to process. A branch to step 28 is made.



● **Figure 150 (Part 1 of 2). Detailed RPG II Object Program Cycle**



● Figure 150 (Part 2 of 2). Detailed RPG II Object Program Cycle

14. If there is only one input file, no record selection is needed. A test determines if sequence checking is requested. If so, a branch is taken to step 31.

15. The record identifying indicator specified for the current record type turns on. Data from the current record type is not available for processing until step 25.

16. If the record contains control fields, the object program tests to determine if a control break has occurred (the contents of the control field are not equal to the contents of a previously stored field). If a control break has not occurred or control fields are not specified, the program branches to step 18.

17. If a control break has occurred, the control level indicator reflecting the condition is turned on. All lower level indicators are also turned on.

18. If this is the first program cycle or the first control break, the program bypasses all total calculation and output operations, and branches to step 21.

19. All calculations conditioned by control level indicators (columns 7-8 of calculation specifications) are performed and resulting indicators are turned on or off as specified. If the LR indicator is on, calculations conditioned by LR are done after other total calculations. File translation, if specified, is done for exception output, chain, and read operations. Fetch overflow is performed if it is required by exception output. If the overflow line has been reached because of the exception output, the overflow indicator is turned on.

20. All total output that is not conditioned by an overflow indicator is performed. The program tests to determine if an overflow condition has occurred. If an overflow condition has occurred at any time during this cycle, the overflow indicator turns on. If the LR indicator is on, output conditioned by LR is done after other total output. File translation, if specified, is done for total output. Fetch overflow is performed if required.

21. The program tests to determine if the last record indicator (LR) is on. If the indicator is on, the program branches to step 38.

22. The program tests to determine if any overflow indicators are on. If no overflow indicators are on, the program branches to step 24.

23. All output operations conditioned by a positive (no N preceding the indicator) overflow indicator are performed. File translation, if specified, is done for overflow output.

24. The MR indicator turns on if this is a multifile job and the record to be processed is a matching record. Otherwise, the MR indicator turns off.

25. Field indicators are turned on or off as specified. Data from the last record read and from specified look ahead fields is made available for processing.

26. Any calculations not conditioned by control level indicators (columns 7-8 of the calculation specifications) are performed, and resulting indicators are turned on or off as specified. File translation, if specified, is done for exception output, chain, and read operations. Fetch overflow is performed if it is required by exception output. If the overflow line is reached because of the exception output, the overflow indicator is turned on. Processing continues with step 3.

27. The last record indicator (LR) and all control level indicators (L1-L9) are turned on and processing continues with step 19.

28. If a file was forced, the next record in that file is selected for processing and a branch is taken to step 15.

29. If a record with no matching fields is found in a normal input file which is not at end of file, it is selected.

30. When matching fields are specified, the normal file with the highest priority matching record filed is selected. If two or more files have equal and highest priority matching record fields, the highest priority file is selected. (The primary file has the highest file priority, the first specified secondary file is next, and so forth.)

31. The match field value is compared to the match field value of the last record. If it is in sequence, the record is accepted and processing continues with step 15.

32. The execution of the program is stopped because a file with matching fields is out of sequence. The operator's option, indicated in step 34, is to bypass (read the next record from the same file) or cancel the job.

33. The execution of the program is stopped because of a record type sequence error or an unidentified record.

34. Step 34 tests the operator's decision either to bypass the record which causes the error condition (branch to step 4) or to cancel the job.

35. If the operator elects to terminate the job by means of a controlled cancel, steps 36-40 are performed. If the operator selects an immediate cancel, the job is terminated.

36 and 37. All operations conditioned by the LR indicator are done.

38. The program writes any tables or arrays for which a to filename entry is specified on the extension specifications sheet. Output tables or arrays are translated, if necessary.

39. All files used by the program are closed (final termination functions are done).

40. End of job occurs.

A subroutine is a routine that is part of another main routine. A routine is something done over and over again. A program can be called a routine because the instructions in a program are done again and again (the program cycle). A subroutine is a group of instructions in that main routine (program) which can be done several times in one program cycle.

Sometimes it is necessary to write a program which at several points does the same operations. Instead of having to write these instructions every time they are needed, it is easier and less time consuming if they can be written just once and then referred to each time they are needed. You can do this by writing a subroutine which then consists of all those operations you have to do at several points in your program.

CODING SUBROUTINES

Subroutines are coded and used on the calculation specifications sheet. They are entered after all other calculation operations. Every subroutine must have a name, but no two subroutines used in the same program can have the same name. A subroutine name cannot be used as the label of a TAG or ENDSR operation.

Enter the name of the subroutine in factor 1. On the same line, enter the operation code BEGSR (line 10 of Figure 151). The subroutine name can be one to six characters long and must begin in column 18 with an alphabetic character. The remaining characters can be any combination of alphabetic or numeric characters (no special characters). Blanks cannot appear between characters in the name.

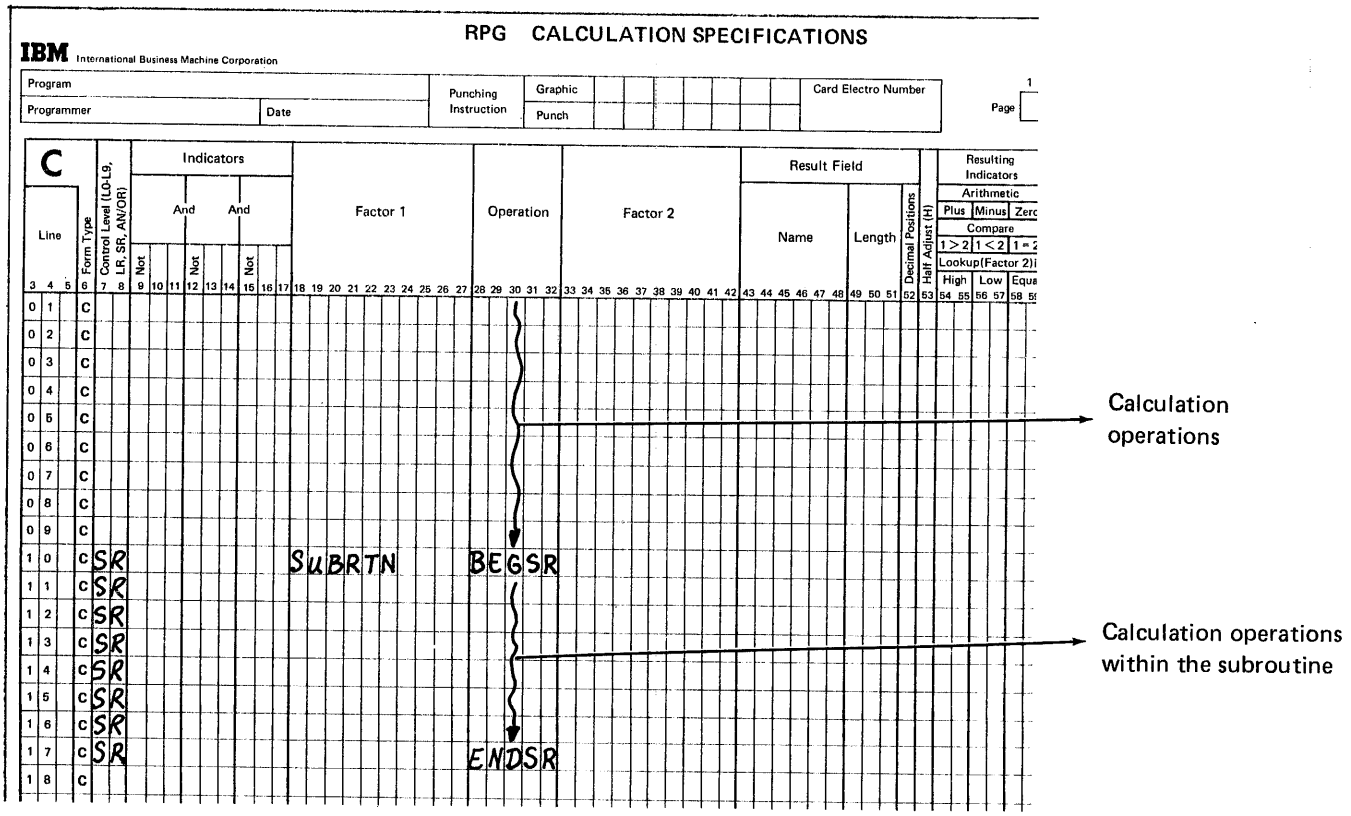


Figure 151. Subroutine Lines (SR)

Each specification line within the subroutine except AN/OR lines must have SR in columns 7-8 to identify it as a subroutine line (Figure 151). The last statement of the subroutine is indicated by the operation code ENDSR (line 17 of Figure 151). Factor 1 of the ENDSR statement can contain a name. This name indicates the point to which a GOTO within the subroutine can branch (Figure 152).

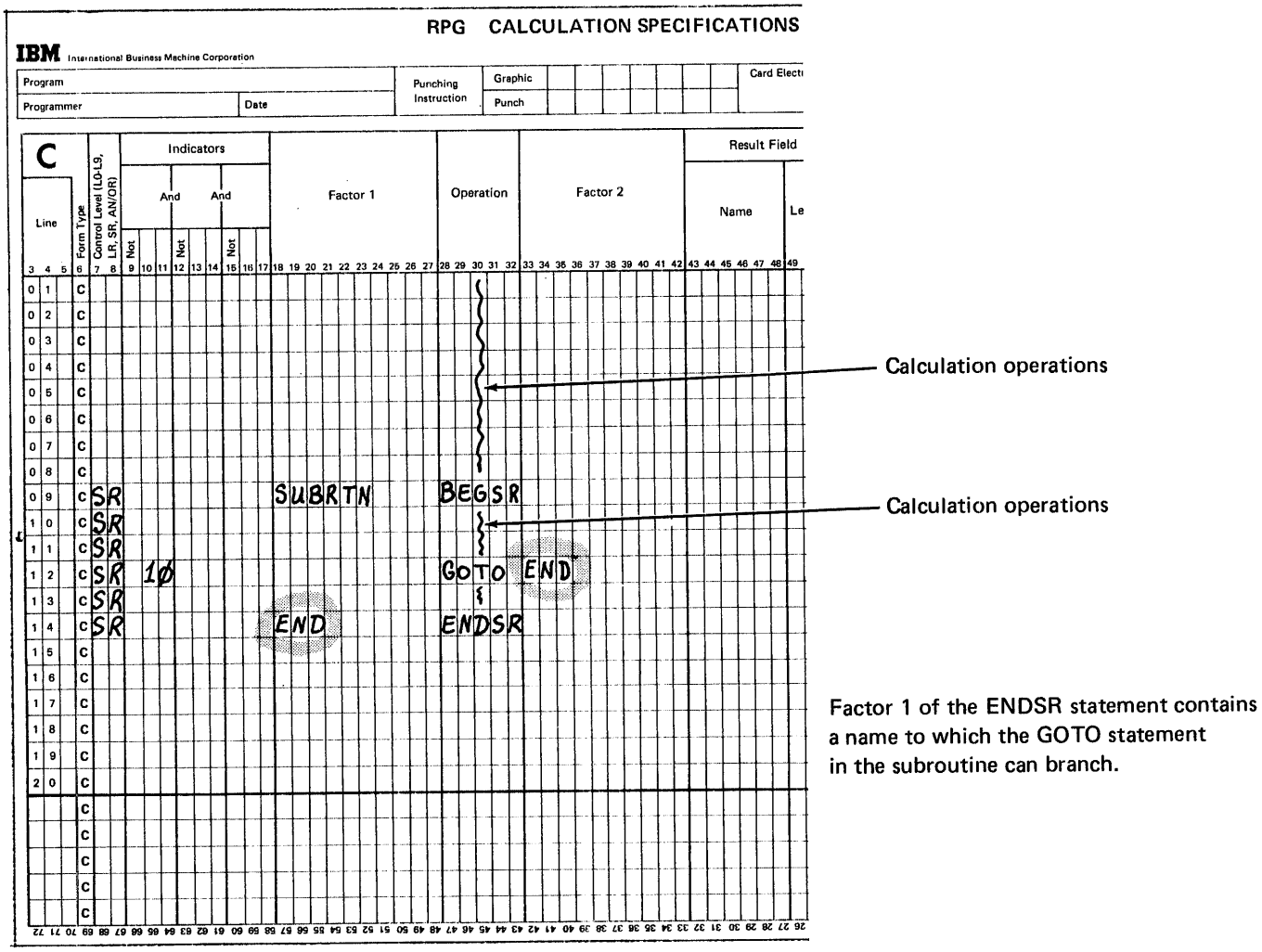


Figure 152. Subroutines (ENDSR)

The subroutine, even though specified last on the calculation specifications sheet, can be performed at any point in the calculation operations. Whenever the subroutine is to be used, enter the operation code EXSR (execute subroutine). The name of the subroutine to be used must also be entered in factor 2 (lines 04 and 08 of Figure 153). Using the EXSR operation is known as calling a subroutine.

The operation code EXSR causes the operations in the subroutine named in factor 2 to be performed. After all calculation operations in the subroutine are done, the next operation after the EXSR is performed. For example, when the EXSR operation (line 04 of Figure 153) is encountered, all subroutine operations (lines 11-15) are done. Then the operation in line 05 is performed. Use indicators with EXSR code to condition when the subroutine should be executed. Any valid indicator can be used in columns 7-17. If no indicators are used, the subroutine is always executed.

All possible RPG II operations can be performed within a subroutine. Operations within the subroutine can be conditioned by any valid indicator in columns 9-17 (Figure 153). Since SR must appear in columns 7-8, control level indicators cannot be used in these columns. This means that individual operations within the subroutine cannot be conditioned by a control level indicator used in columns 7-8. However, entire subroutines can be conditioned by control level indicators. This can be done by using the control level indicator with the EXSR operation (line 08 of Figure 153).

Fields used in the subroutine can be defined either inside or outside the subroutine. In either case, they can be used by both the main routine and the subroutine.

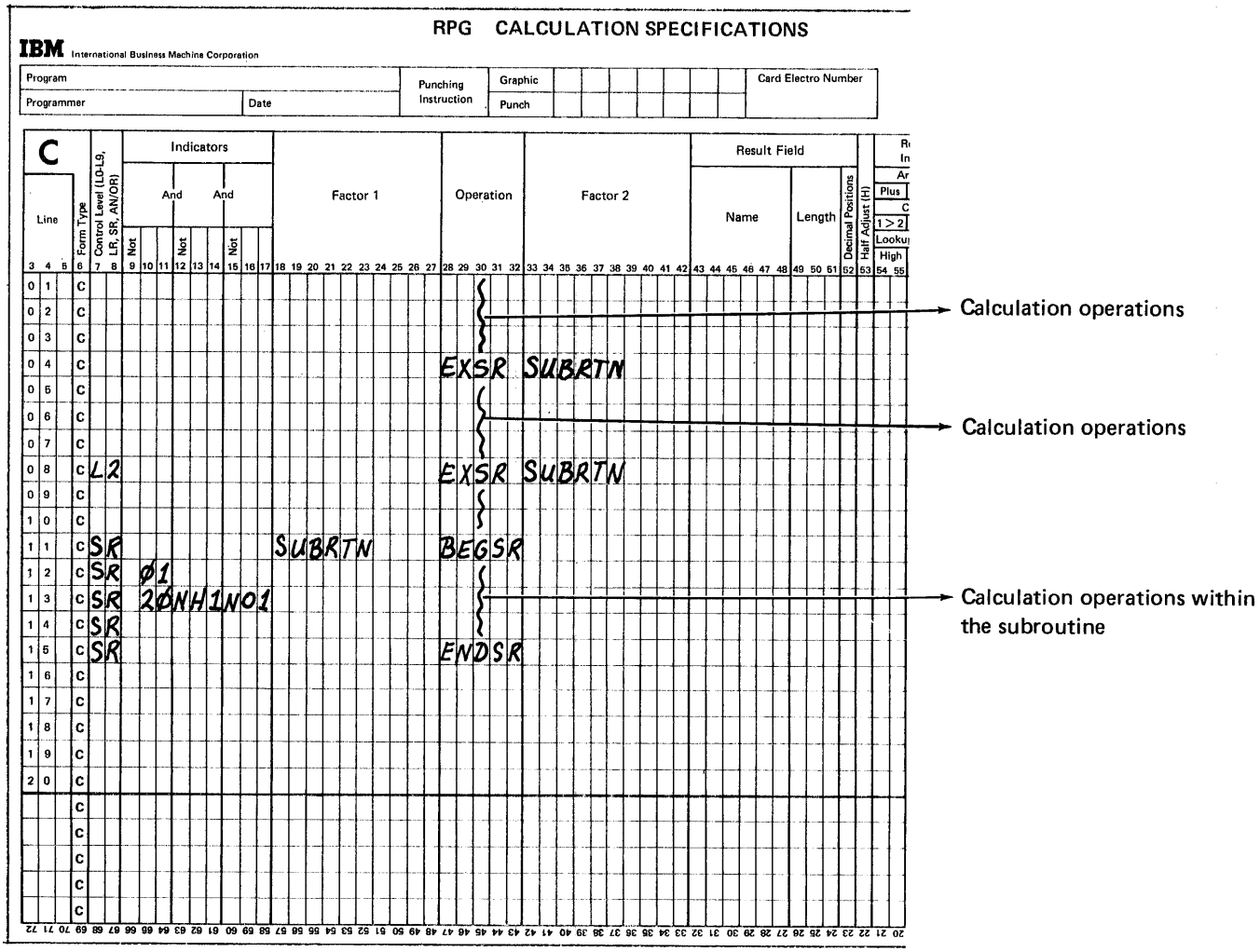


Figure 153. Subroutines (EXSR)

You can use as many subroutines in your main program as you want. However, you cannot write a subroutine within a subroutine. This means that within one subroutine you cannot have the BEGSR and ENDSR operation codes. One subroutine can call another subroutine; however, a subroutine must not call itself. In other words, within a subroutine you can have an EXSR operation (Figure 154).

Subroutines need not be defined in the order in which they are used. However, you must make certain that each one has a different name and a BEGSR and ENDSR operation code.

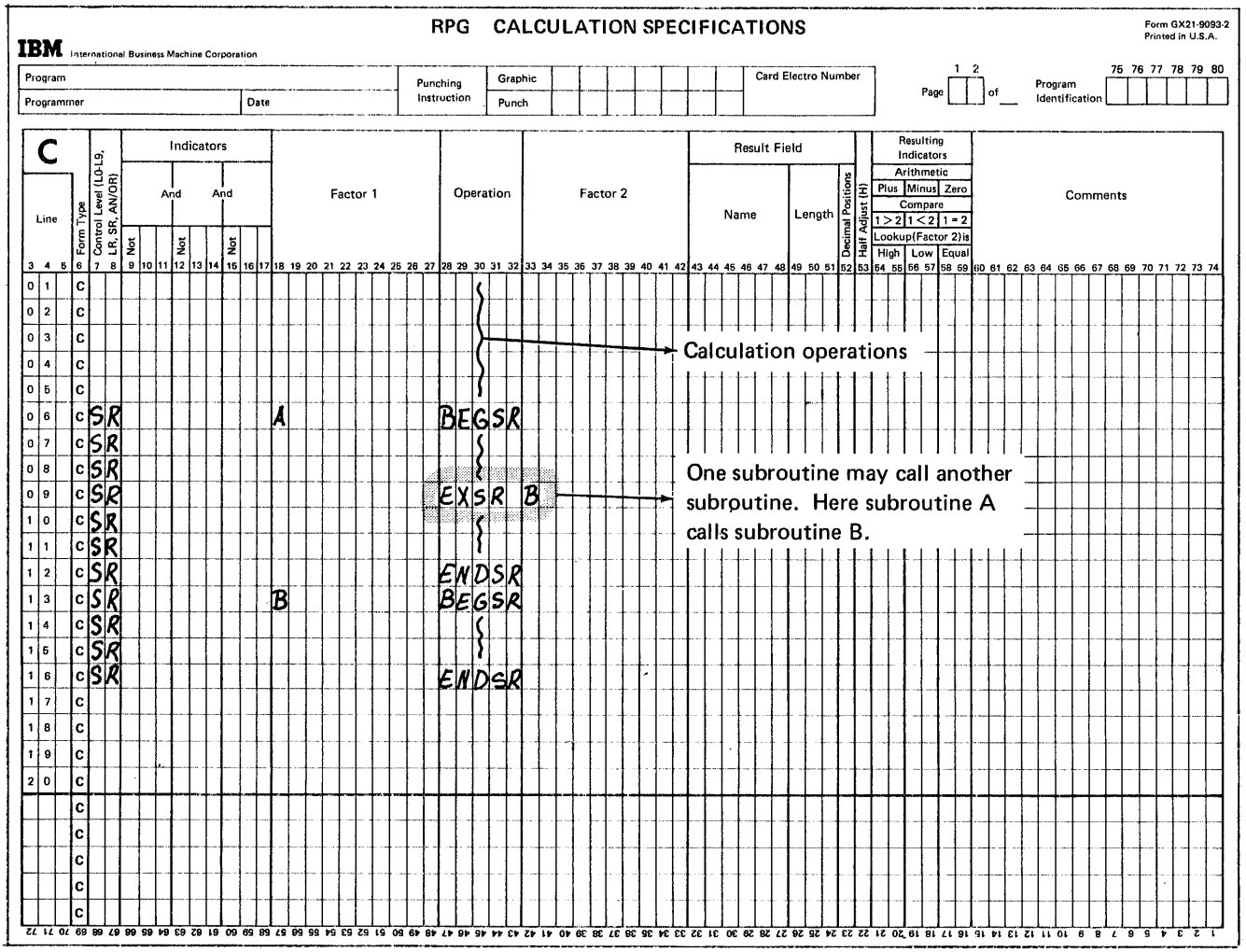


Figure 154. Subroutines: Calling Another Subroutine

When you use a GOTO statement in a subroutine, you can only branch to another statement in that same subroutine. Branching (GOTO) to a statement in another subroutine or outside of a subroutine causes an error condition. You cannot use a GOTO from outside the subroutine to a statement within the subroutine either. Figure 155 shows the correct use of GOTO and TAG within a subroutine.

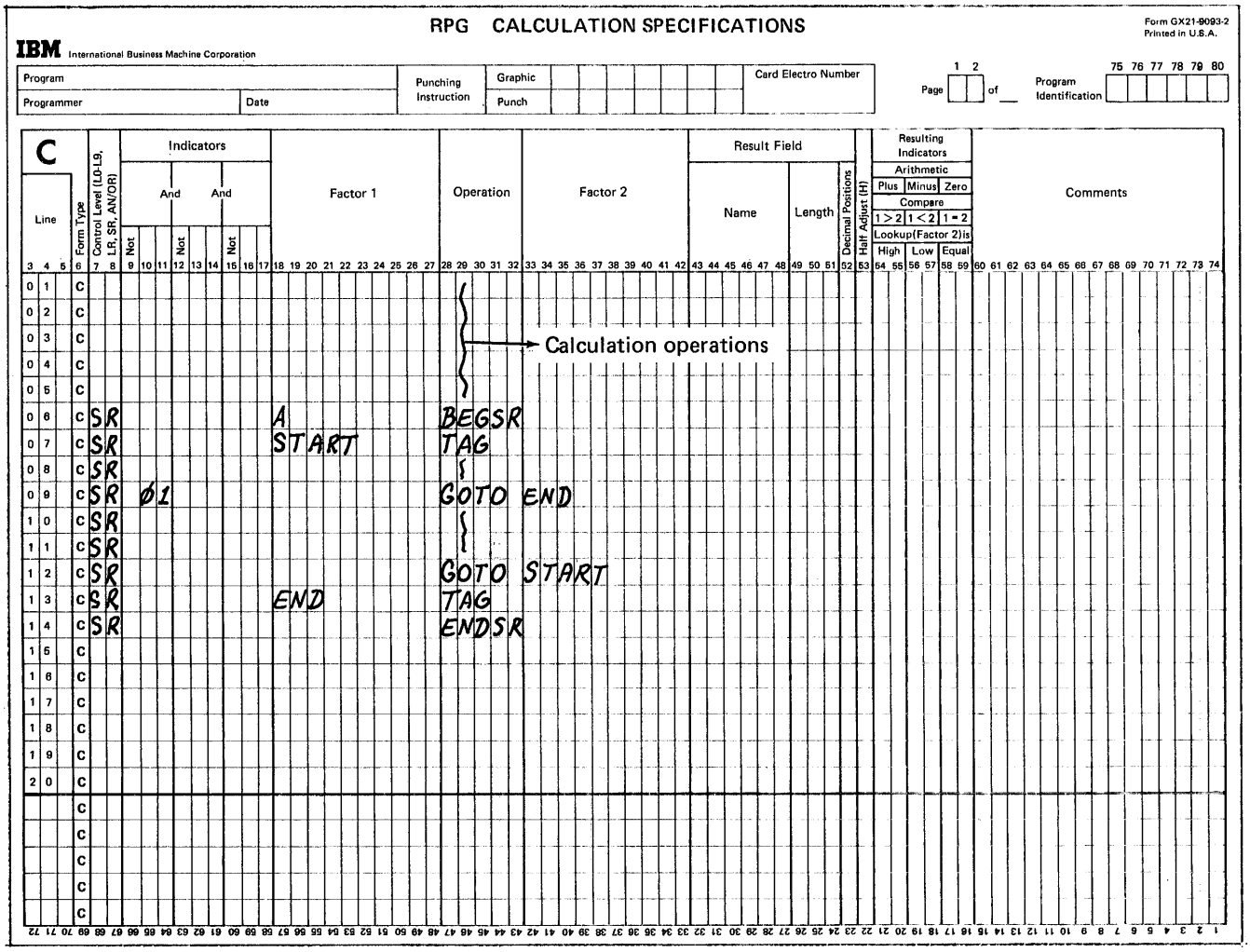


Figure 155. GOTO and TAG Within a Subroutine

Chapter 8. Tables and Arrays

Tables and arrays are systematic arrangements of data items having like characteristics; that is, the same field length, data type (alphameric or numeric), and number of decimal positions. Both tables and arrays are described on the extension specifications sheet. Important differences exist, however, in defining and processing tables and arrays.

Tables are used during the execution of a program much like a shipping clerk uses a rate table for obtaining freight rates. The clerk scans the table for the desired city, then selects the corresponding rate. Tables are referenced by searching the table, one item at a time, for a specific item of data with a unique identifier.

Arrays can also be searched for a uniquely identified data item. Unlike tables, however, array items can also be referenced by their relative position to other items. This is done by indexing to a specific item in the array. Also, an entire array can be processed sequentially by using the array name only once in certain calculation operations.

Several terms are used to describe tables and arrays:

- *Compile-time tables and arrays* are compiled with the source program and become a permanent part of the object program. A compile-time table or array can be permanently changed only by recompiling the source program with the revised table or array.

- *Preexecution-time tables and arrays* are loaded with the object program before actual execution of the RPG II program begins; that is, before any input files are read, calculations are performed, or output functions are performed.

- *Execution-time arrays* are loaded or created by input or calculation specifications. They are loaded after actual execution of your RPG II program has begun (read in as input data or created during calculations in your program). An execution-time array is also described by extension specifications. Tables cannot be specified for execution time.

- *Related tables and arrays* are tables and arrays that are used together. The items in each table or array are called corresponding items; each item in the second table or array gives additional information about its corresponding item in the first. You can specify related tables or related arrays, but you cannot enter specifications to relate tables with arrays or vice versa.

  In Figure 156, table A (TABA) and table B (TABB) are related. An item in table A gives a part number, the corresponding item in table B gives the part cost. Although all items within one table or array must have the same characteristics, corresponding items of related tables or arrays may have different characteristics. Related tables and arrays do not have to have the same number of entries.

- *Short tables and arrays* are those in which not all of the entries contain data. The unused parts of numeric tables and arrays are filled with zeros; the unused parts of alphameric tables and arrays are filled with blanks. You usually create short tables or arrays when you have only a few table or array items available when building the table, but know that more items will soon be included. Short tables and arrays must have at least one entry.

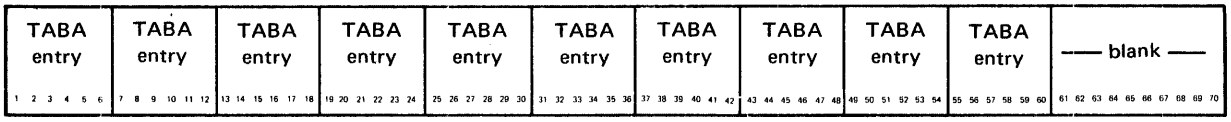
- *Full tables and arrays* are those in which all possible entries contain data.

- *Entry* is one element in a single table or array or corresponding items in related tables or arrays.

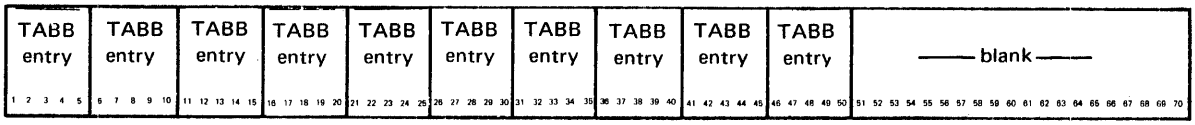
TABA (Part Number)
345126
38A473
39K143
40B125
41C023
42D893
43K832
44H111
45P673
46C732

TABB (Unit Cost)
373
498
1297
93
3998
87
349
679
898
47587

Ⓐ Related Tables

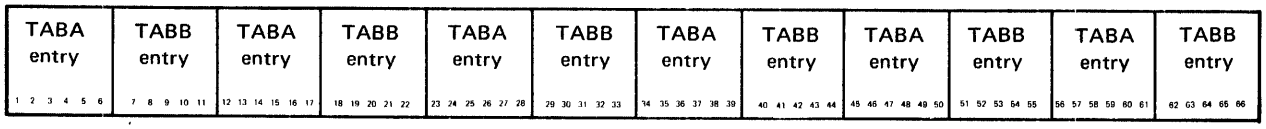


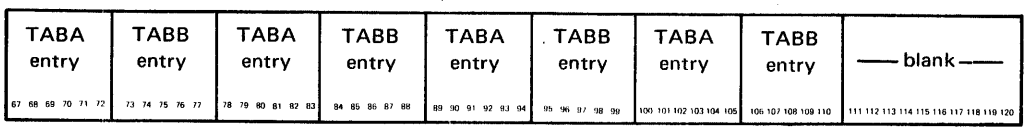
This record contains TABA entries in positions 1-60.



This record contains TABB entries in positions 1-50.

Ⓑ The two tables are described as separate table files.





This record contains TABA and TABB entries in alternating format in positions 1-110.

Ⓒ The two tables are described in alternating format.

Figure 156. Related Tables

RULES FOR CREATING TABLE OR ARRAY INPUT RECORDS

Table and array data must be recorded according to certain rules:

- The first table or array entry for each input record must begin in position 1.
- An entire record need not be filled with entries. If it is not, blanks or comments can be included after the entries (Figure 157).
- Each input record except the last must have the same number of entries. You may want to place just one entry on each record or as many entries as the record can hold.
- Each entry must be entirely on one input record. An entry cannot be split between two records; thus, the length of a single entry is limited to the maximum record length for the input device. If related tables or arrays are used, corresponding items must be on the same input record; together they cannot exceed the maximum record length for the device.
- Related tables or arrays can be described separately or in alternating format. Alternating format means that together, the corresponding items are considered one table or array entry.
- The number of table and/or array names used in a program must be no more than 60.

DEFINING TABLES AND ARRAYS

All tables and arrays are described on the extension specifications sheet. One line describes each set of table or array input records.

If only one table or array is described, columns 11-45 are used. If alternating tables or arrays are described on one set of input records, the second table or array is described in columns 46-57 of the same line as the first table or array. If preexecution-time tables and arrays are being described, entries in columns 11-18 and 27-45 are required. Columns 19-26 are used if the table or array is to be written at the end of the job. Columns 11-18 are not used for compile-time tables and arrays or execution-time arrays.

Tables and arrays can be specified in any sequence. Compile-time and preexecution-time tables and arrays can be mixed. Remember, the sequence in which tables and arrays are specified on the extension specifications sheet determines the order in which they must be loaded at the start of the job.

```
48K16343J64044H12648A47349K34650B1 2551C04352D89373K33274H121
1 2 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

85P67396C79598F37199K704          TABLE OF PART NUMBERS
61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
```

The table entries in this disk record end at position 84. The remaining record positions may remain blank or may be used for comments as in this example.

Figure 157. Table Input Record with Comments

Figure 158 shows the extension specifications required for each type of array.

- Line 1 specifies a compile-time array, ARRAYC. This array has a total of eight elements (three elements per record). Each element has a length of 12 positions, including four decimal places.
- Line 2 specifies a preexecution-time array, ARRAYE, to be read from file DISKIN. ARRAYE has 250 alphameric elements (12 elements per record); each element is five positions long and is equal to or higher than the previous element in collating sequence.
- Line 3 specifies an execution-time array, ARRAYI, to be read from input records. ARRAYI has 10 numeric elements each 10 positions long.

Any of these specifications can include entries in columns 19-26 (to define a filename of a file to which the array is output at end of job) and in columns 46-57 (to define an alternating array).

LOADING TABLES AND ARRAYS

Tables and arrays can be loaded at compilation time or preexecution time. Arrays can also be loaded at execution time.

Compilation-Time Tables and Arrays

Tables and arrays loaded at compilation time are compiled along with the RPG II source program and they become a part of that program. Rules for loading tables and arrays at compile time are as follows:

- Compile-time tables must be placed in the library following the RPG II source program. (Refer to *IBM System/32 System Control Programming Reference Manual*, GC21-7593 for information on system control programs.)
- A record with **Ø in positions 1-3 must be placed before each table entered.
- Tables are loaded in the same order as they are described on the extension specifications sheet.
- A compile-time array must have entries in columns 33-35 of the extension specifications sheet and must not have entries in columns 11-18 of the extension specifications sheet.
- Tables and arrays must not be in packed or binary format. Figure 159 shows the arrangement for the RPG II source program when compile-time tables are loaded.

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

IBM International Business Machine Corporation

Form X21-9091-2 Printed in U.S.A.

Program | Punching Instruction: Graphic / Punch | Card Electro Number | Page __ of __ | Program Identification

Programmer | Date

Extension Specifications

E

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
01	E					ARRAYC	3	8	12		4							
02	E			DISKIN		ARRAYE	12	250	5			A						
03	E					ARRAYI		10	10		Ø							
04	E																	
05	E																	
06	E																	
07	E																	
08	E																	
	E																	
	E																	

Figure 158. Extension Specifications for the Three Types of Arrays

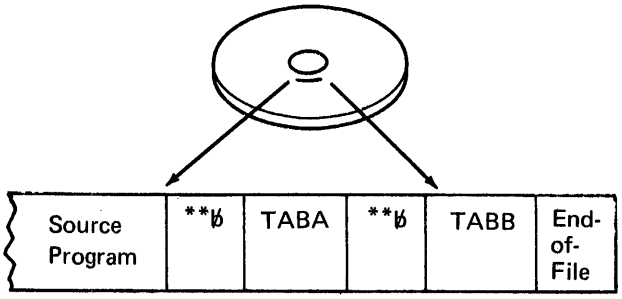


Figure 159. Arrangements of the RPG II Source Program for Compile-Time Tables

Preexecution-Time Tables and Arrays

Preexecution-time tables and arrays are not part of your source program. They are used by the object program like any other data file.

Preexecution-time tables or arrays are loaded from the disk. The table or array file must have been created earlier. OCL statements are used just prior to program execution to identify the table or array file on the disk. If two or more tables or arrays are to be loaded, they must be loaded from different disk files.

Execution-Time Arrays

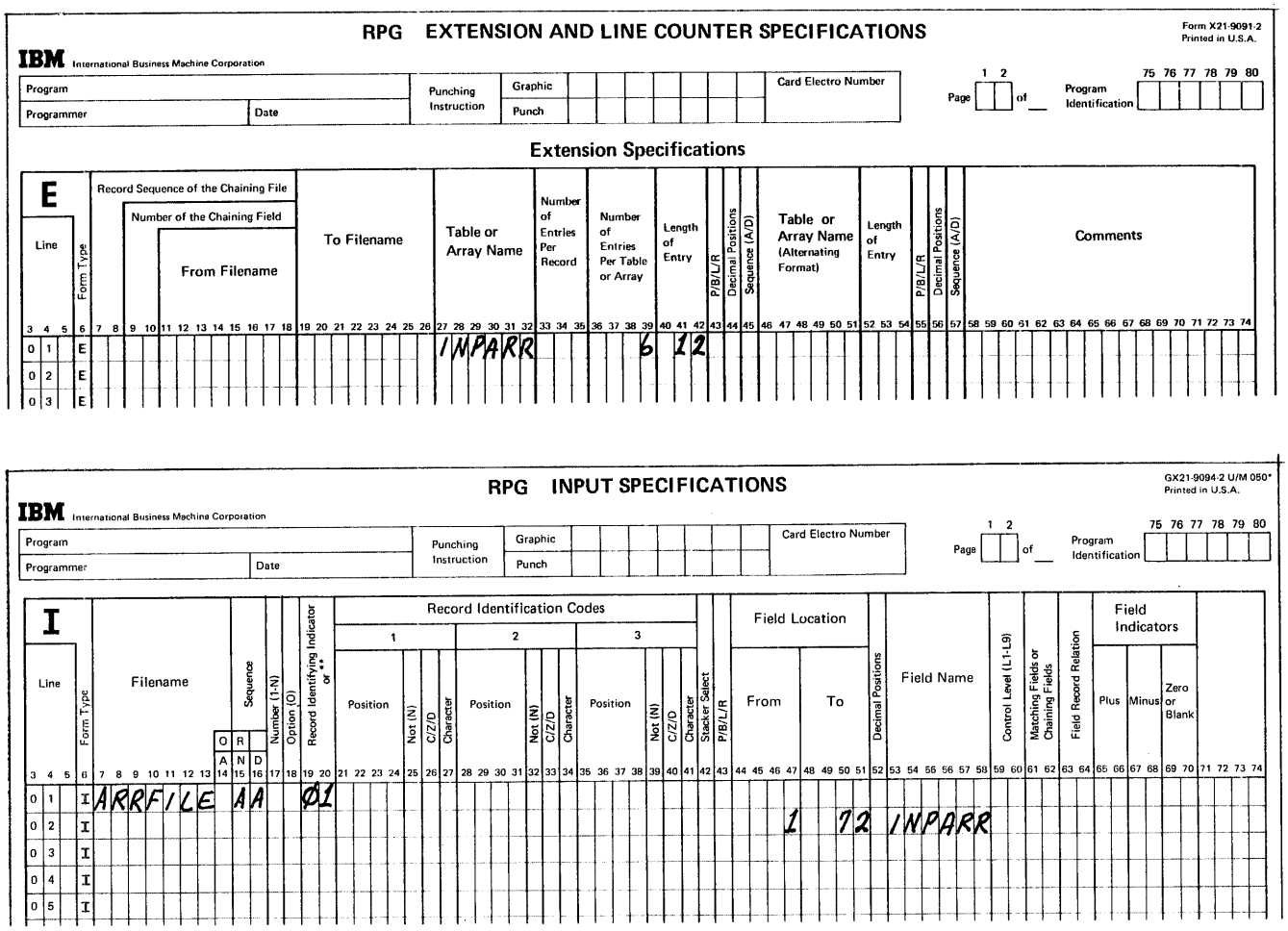
If you are loading an array from information in input records, you must describe that information in your input specifications. The specifications made depend on whether the array information is contained in one or more than one record. Any type of array (compile-time, preexecution-time, execution-time) can be referenced in the input specifications. Execution-time arrays are not sequence checked. However, the array sequence (A or D in column 45) must be specified if high or low LOKUP is used.

Array Information In One Record

If all of the array information is in one record, it can occupy consecutive positions in the record or it can be scattered throughout the record.

If the array elements are consecutive on the input record, they can be loaded with a single input specification. Figure 160 shows an array, INPARR, of six elements (12 positions each) being loaded from a single record from the file ARRFILE.

If the array elements are scattered throughout the record, they can be defined and loaded one at a time, one to a specification line.



RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

Form X21-9091-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program | Programmer | Date | Punching Instruction | Graphic Punch | Card Electro Number | Page 1 2 of __ | Program Identification 75 76 77 78 79 80

Extension Specifications

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
01	E					INPARR			6		12							
02	E																	
03	E																	

RPG INPUT SPECIFICATIONS

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

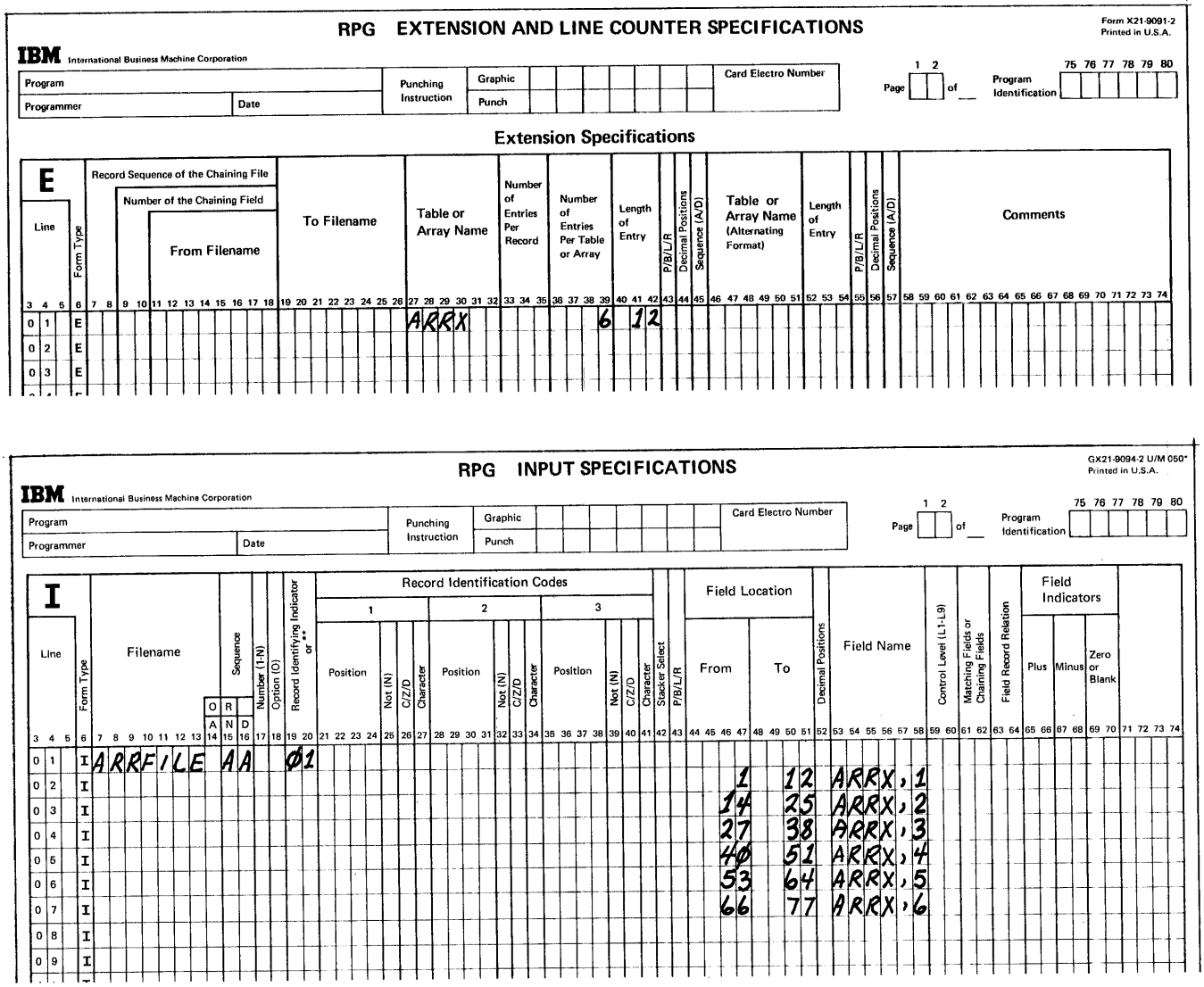
IBM International Business Machine Corporation

Program | Programmer | Date | Punching Instruction | Graphic Punch | Card Electro Number | Page 1 2 of __ | Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence (O R / A N D)	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	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
01	I	ARRFILE	AA			01															1	72		INPARR						
02	I																													
03	I																													
04	I																													
05	I																													

Figure 160. Defining an Execution-Time Array with Consecutive Elements

In Figure 161, an array, ARRX, of six elements with 12 positions each, is loaded from a single record from file ARRFILE; a blank column appears between each two elements.



RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

Form X21-9091-2
Printed in U.S.A.

IBM International Business Machine Corporation

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

Extension Specifications

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
01	E					ARRX		6	12									
02	E																	
03	E																	

RPG INPUT SPECIFICATIONS

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

IBM International Business Machine Corporation

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

Line	Form Type	Filename	Sequence (O A / R N / D)	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	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
01	I	ARRFILE	AA			01															1	12		ARRX,1						
02	I																				14	25		ARRX,2						
03	I																				27	38		ARRX,3						
04	I																				40	51		ARRX,4						
05	I																				53	64		ARRX,5						
06	I																				66	77		ARRX,6						
07	I																													
08	I																													
09	I																													

Figure 161. Defining an Execution-Time Array with Scattered Elements

The following input specifications are required for loading an array from a single input record:

Column	Entry
6	I
7-42	Blank
43	P (packed), B (binary), or blank (unpacked)
44-47 and 48-51	Field location of either an entire array (consecutive elements) or individual field locations of single elements of the array
52	This column must be left blank
53-58	The name of the array or the name of a single element (array name with index). This array name must be the same name as that used on the extension specifications sheet
59-62	Blank
63-64	Field record relation indicator. See *Columns 63-64 (Field Record Relation)* under *Input Specifications* for information on this entry
65-70	Blank
71-74	Blank

Array Information in More Than One Record

If the array information is in two or more records, there are many methods that can be used to load the array into the system. The method you use is primarily based on the size of the array and whether the array entries are consecutive in the input records. Figure 162 shows the array that results by loading array information from certain input records. Each record identified by a 1 or 3 in column 1 contains 12 items of array information. Records identified by a 2 in column 1 do not contain array information, although they appear in the same input file. Examples of loading and storing array information are found in *Examples of Building and Using Arrays* in this section. Keep in mind that the RPG II program processes one record at a time. You cannot process the entire array until all of the records containing the array information are read and the information is moved into the array fields. It may, therefore, be necessary to suppress calculation and output operations until the entire array is read into the system.

SEARCHING TABLES AND ARRAYS

Tables and arrays can be searched using the LOKUP operation code. See *Look Up Operations* under *Operation Codes* in this section for a description of how to use LOKUP.

USING ARRAYS

Arrays can be used in input, output, or calculation specifications (see *Examples*). The elements in an array can be referenced individually, or the array can be referenced as a whole. Individual elements are referenced by an array name plus an index. The array name alone references the entire array.

Array Name and Index

The array name is specified beginning in column 27 or column 46 of the extension specifications sheet and must be a valid RPG II name.

The length of the array name depends on how the array is being used. The array name can be from one to six characters long. The array name is used by itself only when referencing the entire array.

If individual elements of the array are to be referenced, the array name requires an index. An index can be a numeric field with zero decimal positions or a literal. The array name and index must be separated by a comma. The array name with comma and index never occupies fewer than three character positions. The total length of an array name with comma and index entry is limited to either six or 10 positions. It is limited to six positions for input specifications, output specifications, or the result field of calculation specifications or 10 positions for factor 1 or factor 2 of calculation specifications. The index must not be zero, negative, or greater than the number of elements in the array.

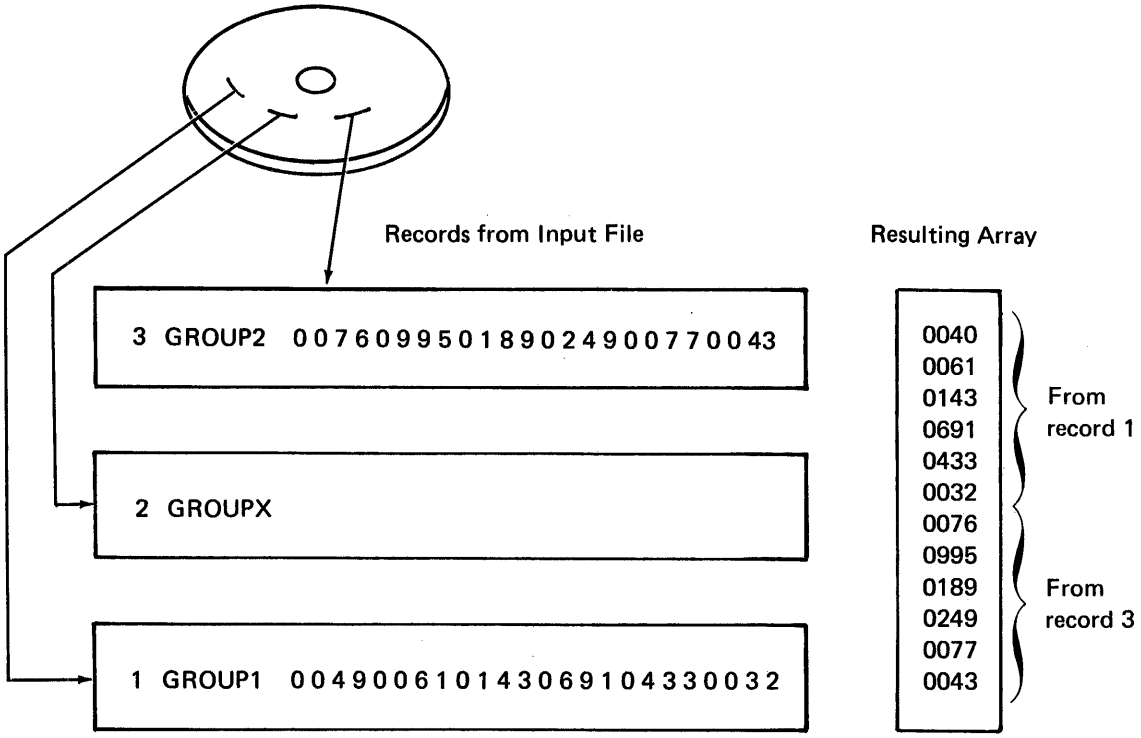


Figure 162. Loading an Array from Input Records

Some examples of array names with and without index are:

Valid		Invalid	
ARAYOL		BALANCE	(array name has more than six characters)
B		6TOTAL	(first character is not alphabetic)
AR,1	(the first element of array AR)	TOTAL-	(name contains special character)
X,YY2	(where YY2 is a field name)	CR TOT	(name contains blank)
		A1,A1	(array is used as index)
		BAL,XX1	(name including comma has more than six characters. This name is valid for factor 1 and factor 2 of the calculation specifications only)

Referencing an Array in Calculations

You can reference an entire array or individual elements in an array using calculation specifications. Process individual elements like normal fields. Remember, if an array field is to be used as a result field, the array name with comma and index entry cannot exceed six characters (see *Examples, Example 8*).

To reference an entire array, use only the array name. You can use it as factor 1, factor 2, or the result field. The following operations can be used with an array name: ADD, Z-ADD, SUB, Z-SUB, MULT, DIV, SQRT, MOVE, MOVEL, MOVEA, MLLZO, MLHZO, MHLZO, MHHZO, DEBUG, XFOOT, and LOKUP. Except when using XFOOT and LOKUP, factor 1 and factor 2 cannot be an array name unless the result field is also an array name.

There are also several operations that can be used with an array element only, but not the array name alone. These operations are COMP, TESTZ, BITON, BITOF, TESTB, KEY, SET, and MVR.

The following rules apply when using array names without an index in calculations:

- When the factors and the result field are arrays with the same number of elements, the operation is performed using the first element from every array, then the second element from every array, etc, until all elements in the arrays are processed. If the arrays do not have the same number of entries, the operation ends when the last element of the array with the fewest elements has been processed.

- When one of the factors in a field or constant, and the other factor is an array, and the result field is an array, the operation is performed once for every element in the shorter array. The same field or constant is used in all of the operations. Except for XFOOT, the result field must always be an array.

- If an operation code uses factor 2 only (such as Z-ADD, Z-SUB, or SQRT) and the result field is an array, the operation is performed once for every element in the array. The same field or constant is used in all of the operations.

- Resulting indicators (columns 54-59) cannot be used due to the number of operations being performed. Exceptions are XFOOT and LOKUP which allow resulting indicators.

MODIFYING CONTENTS OF TABLES AND ARRAYS

Tables and arrays can be temporarily changed during the execution of a job. This is done when the table or array name is used as a result field in an arithmetic or move operation. The appropriate entry in the table or array is modified for the duration of the job. The next time the job is executed, however, the table or array will have the original entries. Temporary changes can be made permanent by changing the table input records.

Figure 163 shows the specifications for modifying the contents of related tables TABFIL and TABLIT. The item in TABFIL which contains 25 is to be changed to 30. The corresponding item in TABLIT is to be changed to 500. The search word is the constant 25. When a match is found in the table TABFIL, the item from TABFIL and its corresponding item in TABLIT are placed in their respective storage areas. The number 500 is then moved into the storage area for TABLIT; the number 30 is moved into the storage area for TABFIL. The contents of the appropriate original table entry are now modified to agree with the new entry in the special storage areas.

Adding Entries to Short Tables or Arrays

Entries can be added to short tables and arrays before or during execution of the job. The simplest way to add entries to a table or array is to write additional entries on the table input records before program execution. However, entries can also be added during execution of a program. The entries added during execution can be created by calculation operations or read from an input record.

Figure 164 shows how entries are added to related numeric tables. The LOKUP operation is conditioned by indicator 01. Indicator 01 is on when a record is read which contains information in the fields NEWA and NEWB. These fields are to be added to the short tables TABA and TABB respectively. To get the entry in the correct place in the table, a search is made to find the first empty entry. Unfilled entries in short numeric tables are filled with zeros. Thus, the search word used is 000. When the first 000 entry is found (indicator 35 turns on), NEWA and NEWB are moved into the storage areas of the related tables TABA and TABB. They, in turn, become part of the tables.

These entries are temporary unless they are written in table input records. If these entries are to become a permanent part of the short table, they must be written in records and included with the other table file records.

RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 1 2 of __ Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / And — Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus / Minus / Zero; Compare: 1>2 / 1<2 / 1=2; Lookup(Factor 2)is: High / Low / Equal	Comments
0 1	C			25	LOKUPTABFIL		TABLIT				10	
0 2	C		10		MOVE	500	TABLIT					
0 3	C		10		MOVE	30	TABFIL					
0 4	C											
0 5	C											
0 6	C											
0 7	C											
0 8	C											

Figure 163. Changing Table Data with MOVE Operations

RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 1 2 of __ Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / And — Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus / Minus / Zero; Compare: 1>2 / 1<2 / 1=2; Lookup(Factor 2)is: High / Low / Equal	Comments
0 1	C			000	LOKUPTABA		TABB				35	
0 2	C		01		MOVE	NEWA	TABA					
0 3	C		35 01		MOVE	NEWB	TABB					
0 4	C		35 01									
0 5	C											
0 6	C											
0 7	C											
0 8	C											

Figure 164. Adding Entries to Short Tables

TABLE AND ARRAY OUTPUT

Entire tables and arrays can be written out under control of RPG II only at the end of the job (LR is on). To indicate entire table or array output, specify in columns 19-26 of the extension specifications sheet, the name of the output file to be used. This specification causes the program to write out the entire table or array with all modifications.

If an entire array is to be written on an output record (via output specifications), describe the array along with any normal fields for the record:

- Columns 32-37 of the output specifications sheet must contain the same array name used on the extension specifications sheet.
- Columns 40-43 of the output specifications sheet must contain the record position where the last field of the array is to end.

If an output record is to contain only certain fields from a table or array, describe the fields in the same way as you do normal fields, using either an array name with an index or a table name as the field name.

Editing Entire Arrays

When editing an entire array, any editing you specify applies equally to all fields in the array. If you require different editing for various elements, reference them individually.

When you specify an edit code for an entire array (column 38), note that two blanks are automatically inserted to the left of every field in the array. When you specify an edit word instead, the blanks are not inserted. The edit word must specify all the blanks you want inserted.

EXAMPLE OF USING TABLES

A payroll job requires two related tables. TABNUM is the search table containing employee numbers. TABRAT is the related table containing employee salary rates (Figure 165, insert A). After an employee's rate is found, rate is multiplied by the number of hours worked. The result is the amount earned.

TABNUM	TABRAT
12345	407
12346	593
12347	369
12348	390
12349	1379

Ⓐ

Figure 165 (Part 1 of 3). Related Tables Used in Payroll Job

Table input records are organized in alternating format. There are eight entries in each record. Each table has 500 entries. Therefore, 63 records are required. The first 62 contain characters in positions 1-72 (5-position element in TABA, plus 4-position element in TABB, times eight entries per record). The last record has only four entries and contains characters in positions 1-36. Figure 165, inserts B and C, shows the RPG II specifications needed to do the job. The following paragraphs explain the entries made.

File Description Specifications

The file containing input records is called TIMECARD. It is an input file (I in column 15) and a primary file (P in column 16). When this file reaches end of file, processing ends (E in column 17). This file is read in from the disk.

The file containing the related tables is called RATETABL. It is an input table file (I in column 15, T in column 16). The file is to be read in from disk prior to execution time. An E is required in column 39 to show that additional information about the file is on the extension specifications sheet.

File Description Specification

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Core Index	K	Option	Entry	A/U	R/U/N	File Condition U1-U8
02	F	TIMECARDIPE				F		96							DISK											
03	F	RATETABLIT				F		72							EDISK											
04	F																									
05	F																									
06	F																									

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

Form X21-9091-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program | Punching Instruction: Graphic / Punch | Card Electro Number | Page __ of __ | Program Identification

Programmer | Date

Extension Specifications

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
01	E			RATETABL		TABNUM	8	500	5				ØATABRAT	4		2		
02	E																	
03	E																	
04	E																	

RPG INPUT SPECIFICATIONS

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

IBM International Business Machine Corporation

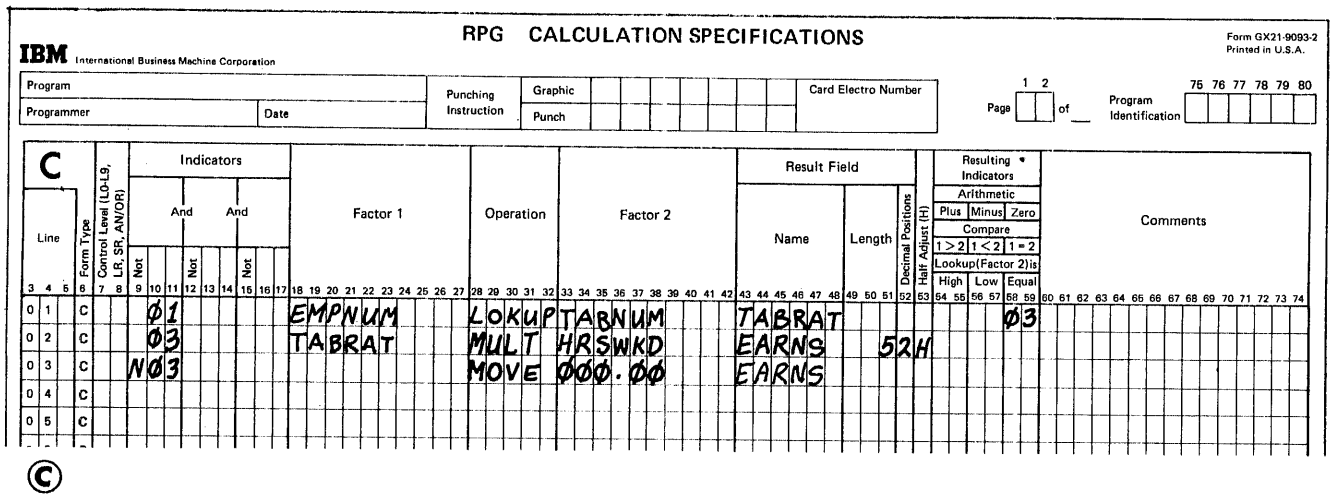
Program | Punching Instruction: Graphic / Punch | Card Electro Number | Page __ of __ | Program Identification

Programmer | Date

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	Field Location: P/B/L/R	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
			O R A N D																											
01	I	TIMECARDAA				Ø1																								
02	I																				1	5		ØEMPNUM						
03	I																				42	44	1	HRSWKD						
04	I																													
05	I																													
06	I																													

(B)

Figure 165 (Part 2 of 3). Related Tables Used in Payroll Job


RPG CALCULATION SPECIFICATIONS

IBM International Business Machine Corporation

Form GX21-9093-2 Printed in U.S.A.

Line	Form Type	Control Level (LO-L9, LR, SR, AN/OR)	Indicators: And Not	And Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus	Minus	Zero	Compare 1>2	1<2	1=2	Lookup(Factor 2)is High	Low	Equal	Comments
01	C		01		EMPNUM	LOKUP	TABNUM	TABRAT												03	
02	C		03		TABRAT	MULT	HRSWKD	EARNS		52	H										
03	C		N03			MOVE	000.00	EARNS													
04	C																				
05	C																				

Figure 165 (Part 3 of 3). Related Tables Used in Payroll Job

Extension Specifications

The extension specifications complete the definition of the file RATETABL. The table searched is called TABNUM (columns 27-32). There are eight entries in each record (columns 33-35) and 500 entries in the table (columns 36-39).

Each table entry is five positions long (columns 40-42) and contains no decimal positions (column 44). The table is organized in ascending sequence (column 45). The alternating table is called TABRAT (columns 46-51). Each entry is four positions long (columns 52-54) and contains two decimal positions (column 56).

Input Specifications

The input file (TIMECARD) is assigned a sequence of AA (columns 15-16). Record identifying indicator 01 is turned on whenever an input record is present for processing. No record identification codes are specified, because there is only one record type. Lines 02 and 03 describe the locations of the two input fields used by the program. The employee number (EMPNUM) is in positions 1-5 of the input record. The number of hours worked by the employee (HRSWKD) is in positions 42-44 of the input record.

Calculation Specifications

On line 01, factor 1 contains the search word EMPNUM (employee number). LOKUP in columns 28-32 causes the look up operations to be performed. Factor 2 contains the name of the table to be searched (TABNUM). The result field contains the name of the related table (TABRAT).

Line 01 causes the employee number (EMPNUM) to be used as the search word for the data contained in TABNUM. Indicator 03 is turned on when the program finds an entry in the search table that is equal to the search word.

Line 02 is performed when indicator 03 is on. The rate for the employee, taken from the related table TABRAT, is multiplied by the number of hours worked (HRSWKD). The result is stored in the field EARNS, which is five positions long with two decimal positions. The result is half-adjusted.

When an equal entry is not found in TABNUM (indicator 03 is not on), line 03 is performed. The literal 000.00 is then moved to the field EARNS, specifying that the employee does not have an entry in the table.

EXAMPLES OF BUILDING AND USING ARRAYS

Example 1: Figure 166 illustrates a method of loading an array using fields in input records as indexes. The example shows a 12-element array with element length five. The array can be made larger without additional input specifications by assigning different values to fields X1-X10 on each input record type 03 and to fields X1 and X2 on each 04 record type. Succeeding type 03 records then load 10 additional elements into array AR; each type 04 record loads two additional elements.

Blanks and other fields can appear on the input records since the array elements and their index are identified by the from and to entries.

This method requires a minimum of coding and no calculations to set up the array. Extra work, however, is required to set up the indexing scheme for the input records.

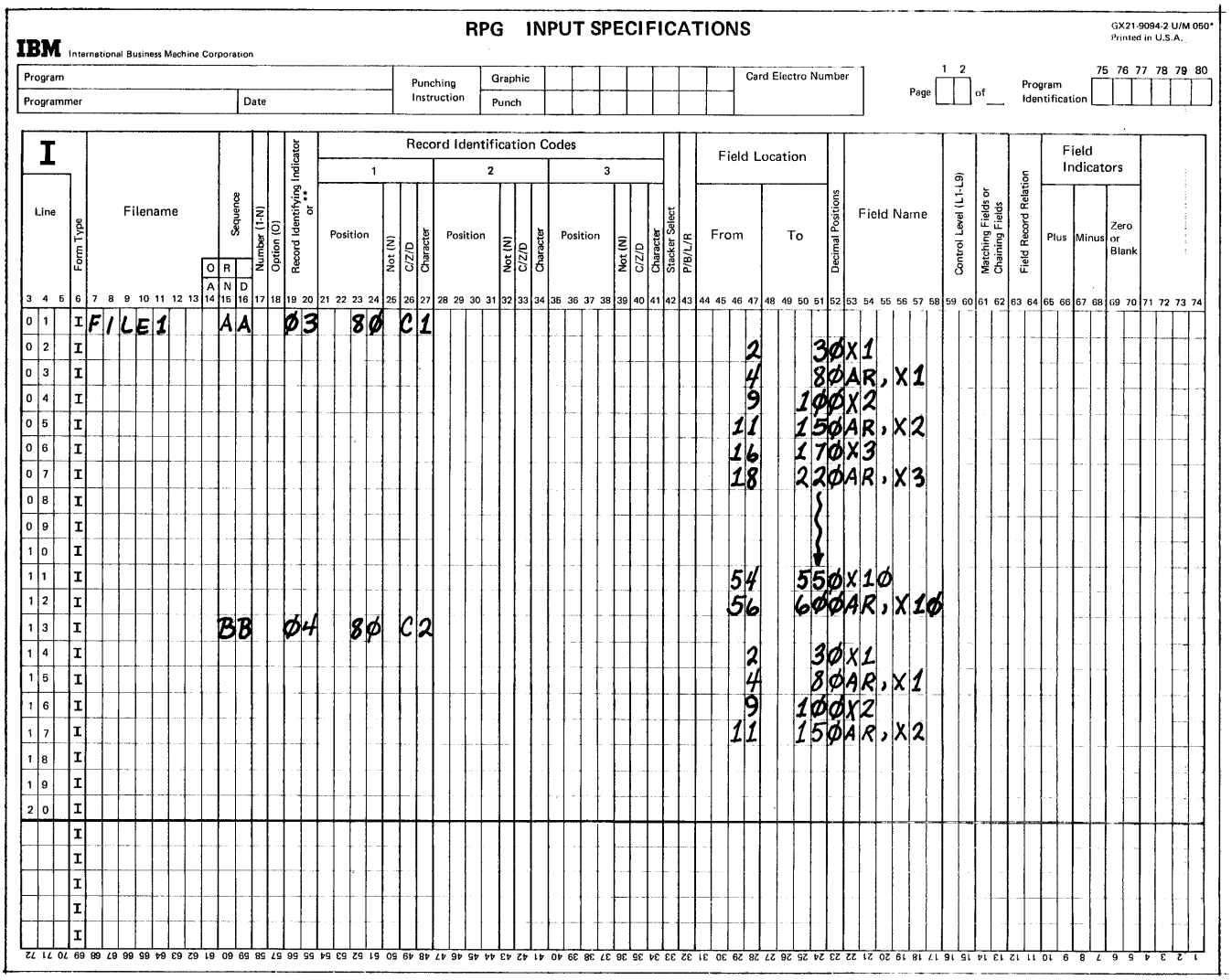


Figure 166. Building an Array Using Input Fields as Indexes

Example 2: In Figure 167, eighteen 5-position elements of array AR1 are loaded with only two specification lines. On succeeding input specifications, other elements of AR1 are loaded one after another until the array is full. Each additional element is coded on a separate line. Each new record requires a separate means of identification. For example, if another 03 record followed the first, the fields on the second record overlay the fields read in from the first record. This method illustrated in example 2 works well for small arrays.

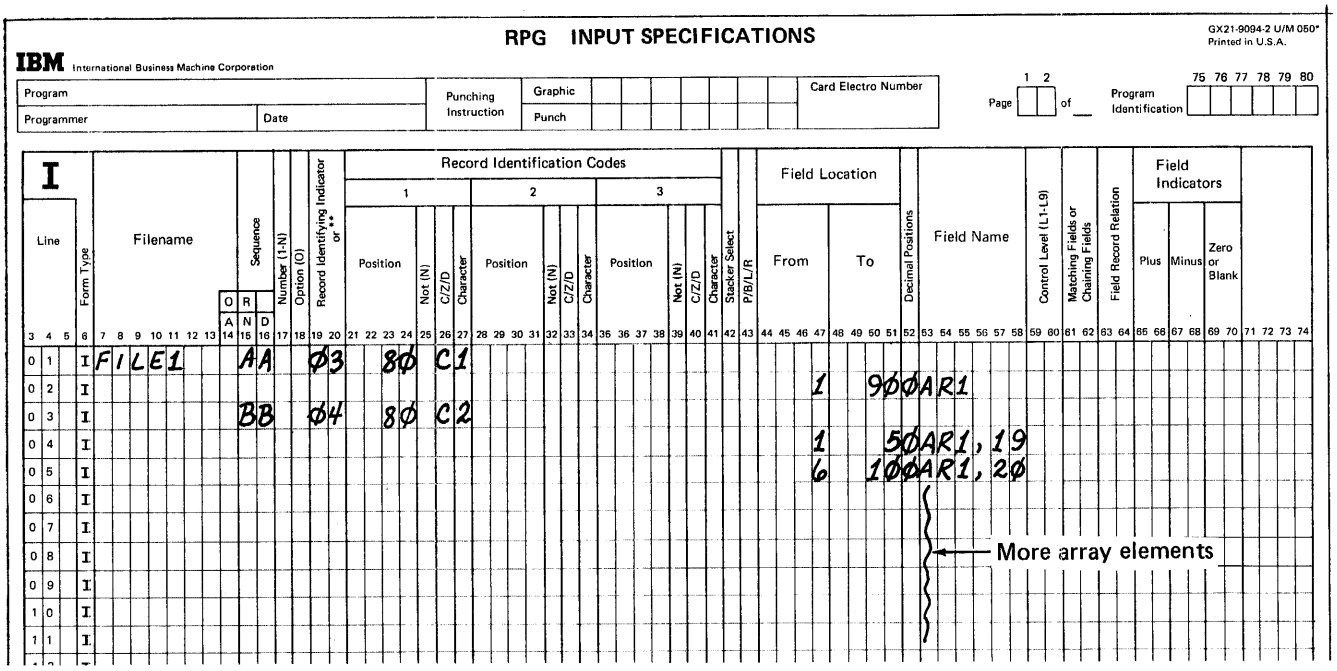


Figure 167. Building an Array Using Fixed Indexes

Example 3: The specifications in Figure 168 perform the function of tabulating three levels of totals. The fields FIELDA, FIELDB, FIELDC, and FIELDD are added as they are read from input records, to the first level totals L1A, L1B, L1C, and L1D. These first level totals are added at the time of an L1 control break to totals L2A, L2B, L2C, and L2D. Similarly, at an L2 control break the second level totals are added to third level totals L3A, L3B, L3C, and L3D. In addition, as control breaks occur, L1, L2, and L3 total output is performed; total fields are zeros after they are written on the output device.

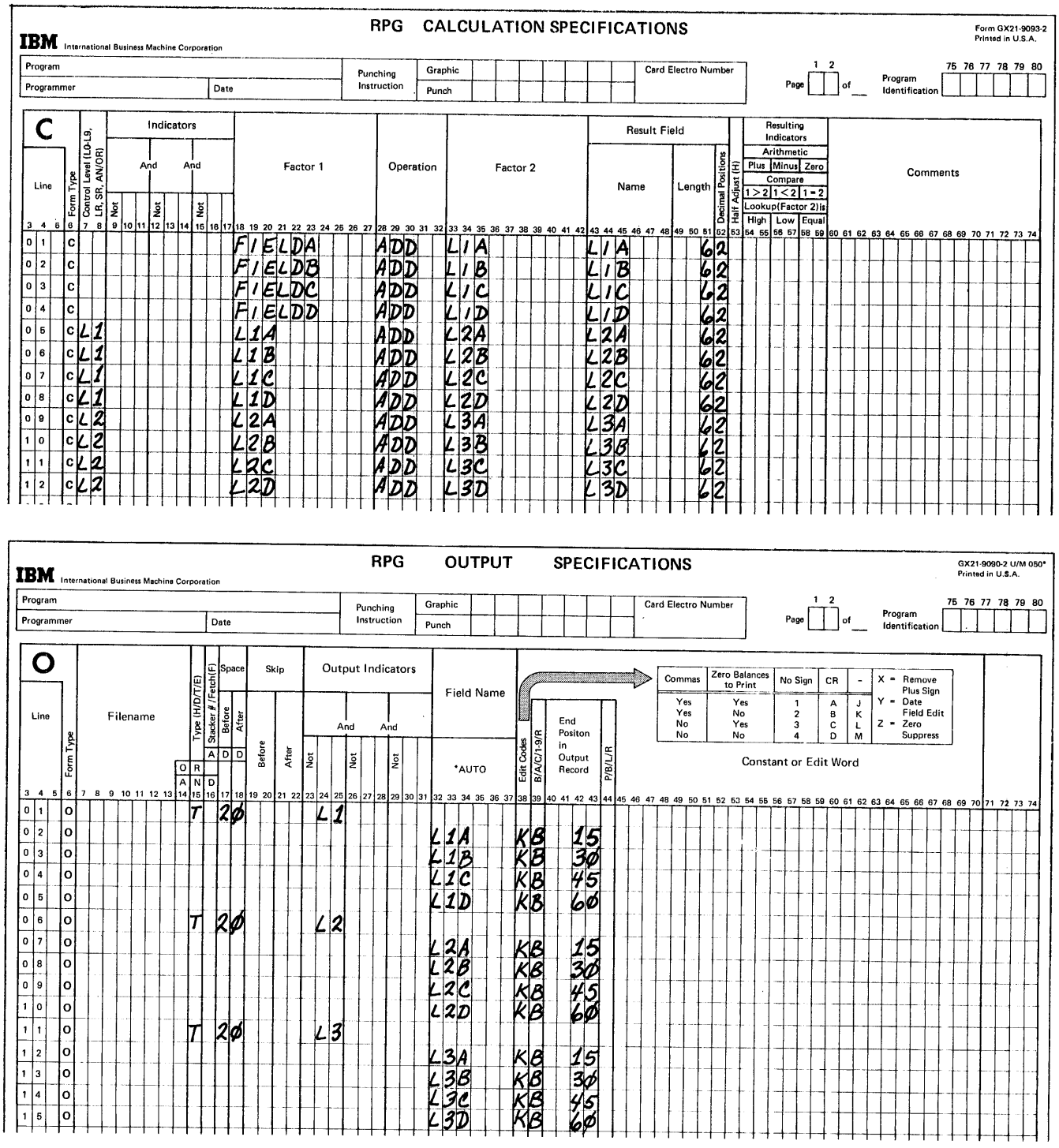


Figure 168. Calculating Totals Without Arrays

Figure 169 shows the same functions being performed using arrays. Note the reduction in coding required to specify the functions. For example, line 5 of the calculation specifications performs the same function as lines 5 through 8 of the calculation specifications of Figure 168. Similarly, the output specifications are reduced from 15 lines to 6. (Notice, however, that the method using arrays results in only two positions between array elements.)

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

IBM International Business Machine Corporation — Form X21-9091-2, Printed in U.S.A.

Program | Programmer | Date | Punching Instruction: Graphic / Punch | Card Electro Number | Page 1 2 of ___ | Program Identification 75 76 77 78 79 80

Extension Specifications

Line (3-5)	Form Type (6)	Record Sequence of the Chaining File (7-8)	Number of the Chaining Field (9-10)	From Filename (11-18)	To Filename (19-26)	Table or Array Name (27-32)	Number of Entries Per Record (33-35)	Number of Entries Per Table or Array (36-39)	Length of Entry (40-42)	P/B/L/R (43)	Decimal Positions (44)	Sequence (A/D) (45)	Table or Array Name (Alternating Format) (46-51)	Length of Entry (52-54)	P/B/L/R (55)	Decimal Positions (56)	Sequence (A/D) (57)	Comments (58-74)
01	E					SL1		4	6		2							
02	E					SL2		4	6		2							
03	E					SL3		4	6		2							

RPG CALCULATION SPECIFICATIONS

IBM International Business Machine Corporation — Form GX21-9093-2, Printed in U.S.A.

Program | Programmer | Date | Punching Instruction: Graphic / Punch | Card Electro Number | Page 1 2 of ___ | Program Identification 75 76 77 78 79 80

Line (3-5)	Form Type (6)	Control Level (L0-L9, LR, SR, AN/OR) (7-8)	Indicators (9-17)	Factor 1 (18-27)	Operation (28-32)	Factor 2 (33-42)	Result Field Name (43-48)	Length (49-51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators (54-59)	Comments (60-74)
01	C			FIELDA	ADD	SL1,1	SL1,1					
02	C			FIELDB	ADD	SL1,2	SL1,2					
03	C			FIELDC	ADD	SL1,3	SL1,3					
04	C			FIELDD	ADD	SL1,4	SL1,4					
05	C	L1		SL1	ADD	SL2	SL2					
06	C	L2		SL2	ADD	SL3	SL3					
07	C											

RPG OUTPUT SPECIFICATIONS

IBM International Business Machine Corporation — GX21-9090-2 U/M 050*, Printed in U.S.A.

Program | Programmer | Date | Punching Instruction: Graphic / Punch | Card Electro Number | Page 1 2 of ___ | Program Identification 75 76 77 78 79 80

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

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

Line (3-5)	Form Type (6)	Filename (7-14)	Type (H/D/T/E) (15)	Stacker #/Fetch (F) (16)	Space Before (17)	Space After (18)	Skip Before (19-20)	Skip After (21-22)	Output Indicators (23-31)	Field Name / *AUTO (32-37)	Edit Codes (38)	B/A/C/1-9/R (39)	End Position in Output Record (40-43)	P/B/L/R (44)	Constant or Edit Word (45-70)
01	O		T		2	ø			L1						
02	O									SL1	K	B	6ø		
03	O		T		2	ø			L2						
04	O									SL2	K	B	6ø		
05	O		T		2	ø			L3						
06	O									SL3	K	B	6ø		
07	O														

Figure 169. Calculating Total With Arrays

Example 4: This example illustrates the use of three arrays defined as follows (Figure 170).

Array Name	Number of Fields	Field Length
ARA	4	5
ARB	5	10
ARC	6	4

Array ARA is contained in the input records corresponding to indicator 01, ARB in the records corresponding to 02, and ARC in both types of records. Array ARC and the first field of array ARA are to be included together in an output record as are arrays ARC and a field (identified by field X1) of array ARB. Every field in array ARC is edited according to the edit word 0ø.øø&CR (where ø represents a blank).

Assume that the contents of the arrays in the first two input records are:

Record	Array	Array Contents
1	ARA	12345678901234567890
	ARC	012345678901234567890876N (note that N equals minus 5)
2	ARB	JOHNøDOEøøJOEøSMITHø LEEøMARXøøJIMøKNOTSø TIMøTYLERø
	ARC	(the same as in record 1)

In the first output record, the location and contents of the arrays are (ø represents a blank):

Array	Location	Contents
ARA (first field)	85-89	12345
ARC	37-84	ø1.23øøø45.67øøø 89.01øøø23.45øøø 67.89øøø87.65øCR

For the second output record assume that the contents of field X1 is 4; the locations and contents of the arrays are:

Array	Location	Contents
ARB	91-100	JIMøKNOTSø
(fourth field)		
ARC	37-84	ø1.23øøø45.67øøø 89.01øøø23.45øøø 67.89øøø87.65øCR

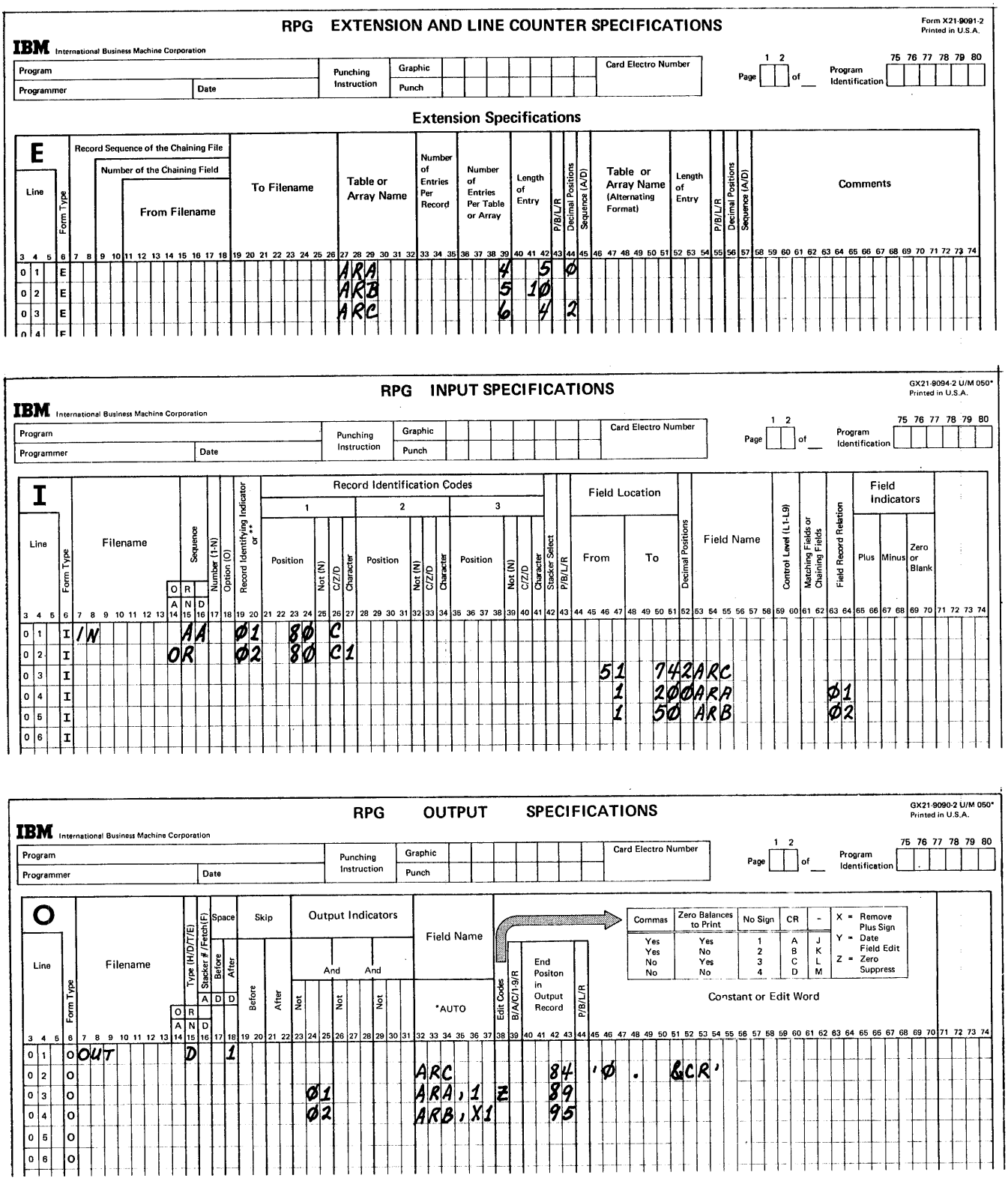
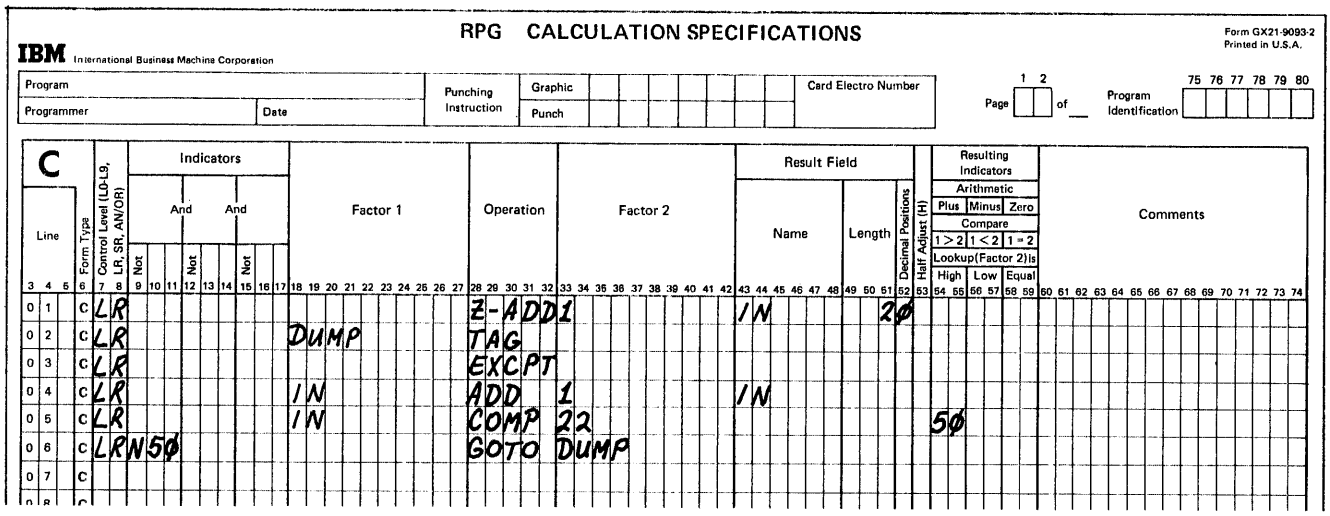


Figure 170. Using Arrays to Format Field Output

Example 5: Figure 171 shows a method of writing short arrays on the output device. The contents of one element of a 22-element array, AR2, is written to the output file ARFILE each time the specification in line 3 of the calculation specifications is performed.



RPG CALCULATION SPECIFICATIONS

IBM International Business Machine Corporation

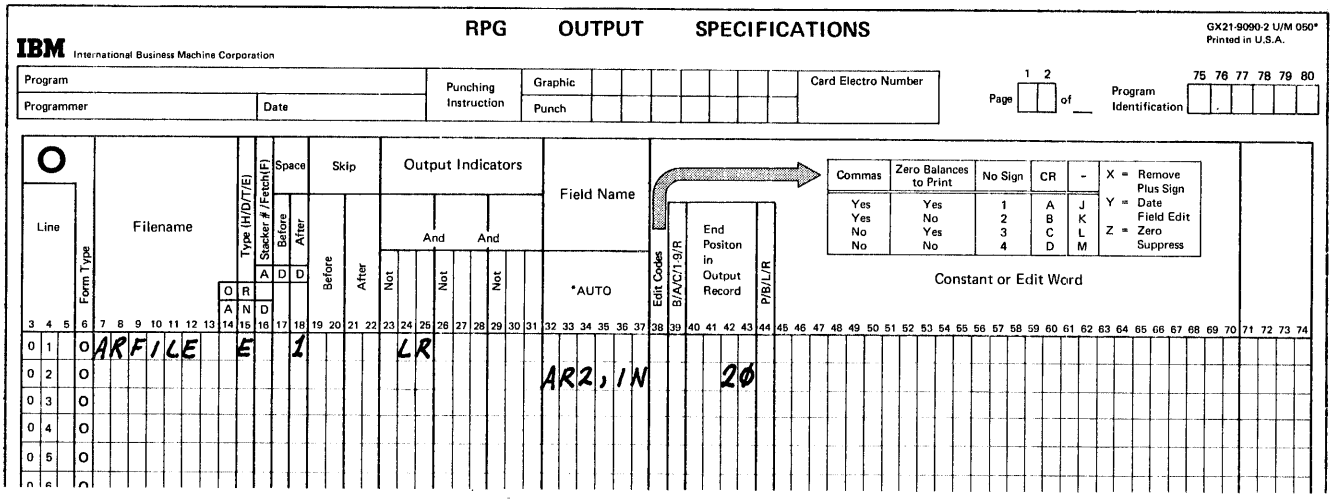
Form GX21-9093-2 Printed in U.S.A.

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 1 2 of __ Program Identification 75 76 77 78 79 80

C

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And	And	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus	Minus	Zero	Compare 1>2	1<2	1=2	Lookup(Factor 2)Is High	Low	Equal	Comments
01	C	LR				Z-ADD1		IN		2Ø											
02	C	LR			DUMP	TAG															
03	C	LR				EXCPT															
04	C	LR			IN	ADD 1		IN													
05	C	LR			IN	COMP 22												5Ø			
06	C	LRN5Ø				GOTO DUMP															
07	C																				



RPG OUTPUT SPECIFICATIONS

IBM International Business Machine Corporation

GX21-9090-2 U/M 050* Printed in U.S.A.

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 1 2 of __ Program Identification 75 76 77 78 79 80

O

Line	Form Type	Filename	Type (H/D/T/E) (O R / A N D)	Stacker # /Fetch(F)	Space: Before	Space: After	Skip: Before	Skip: After	Output Indicators: And Not	And Not	And Not	Field Name	Edit Codes	B/A/C/1-9/R	End Positon in Output Record	P/B/L/R
01	O	ARFILE	E		1				LR							
02	O											AR2,IN			2Ø	
03	O															
04	O															
05	O															
06	O															

*AUTO

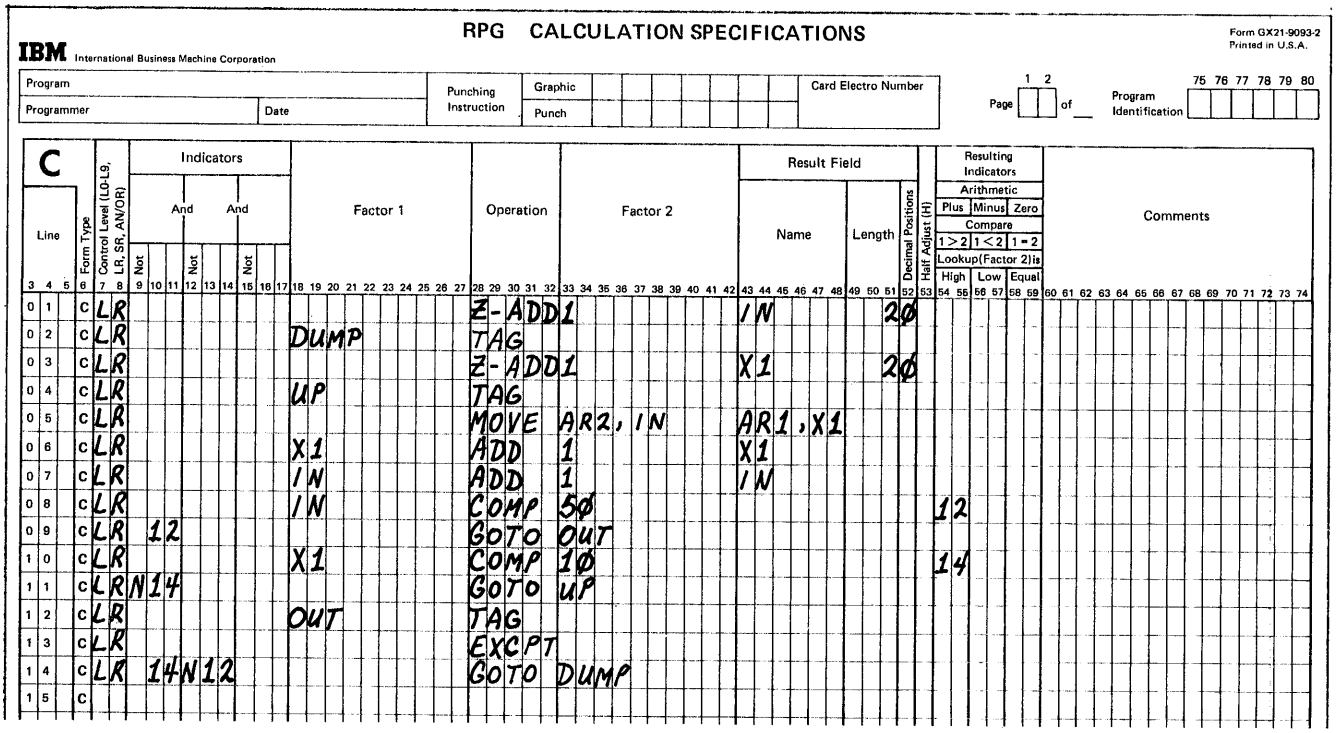
Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

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

Constant or Edit Word

Figure 171. Printing One Array Element Per Line

Example 6: Figure 172 shows a method of writing a large array on the output device. The number of fields printed on a line depends on the value assigned to the compare on line 10 of the calculation specifications sheet. If an edit code is used, each array field is separated by two spaces. These spaces must be considered when computing the end position in the output specifications.



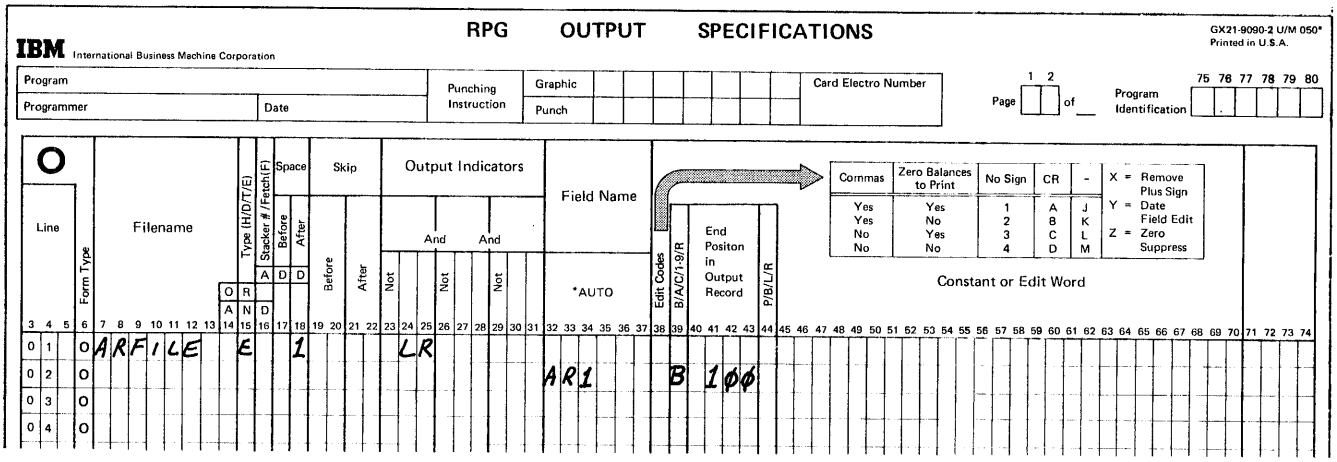


Figure 172. Printing More Than One Array Element Per Line

Interactive data entry (IDE) is a function which allows you to enter input to the executing RPG II program via the keyboard. IDE files must use the keyboard as an input file. This is accomplished by specifying CONSOLE in columns 40-46 on the file description specifications sheet. Only one CONSOLE (IDE) file is allowed per program. (For an example of showing data displayed on the display screen, see Chapter 1, *RPG II Sample Programs* in Part 3, *Supplementary Information*. As the operator enters data from the keyboard, the data is displayed on the display screen. The data is placed in a buffer, independent of the execution of the RPG II program. At input time of the RPG II cycle, the RPG II program retrieves the data from the buffer. Records can be entered any time during the RPG II cycle.

To use the interactive data entry function, certain coding is required on the file description and the input specifications sheets.

IDE FILE DESCRIPTION SPECIFICATIONS

Columns 1-2 (Page)

See *Common Entries.*

Columns 3-5 (Line)

See *Common Entries.*

Column 6 (Form Type)

Enter an F for file description specifications.

Columns 7-14 (Filename)

Use columns 7-14 to assign a unique filename to the console file.

Column 15 (File Type)

Enter an I in column 15.

Column 16 (File Designation)

Entry	Explanation
P	Primary
S	Secondary
D	Demand
R	Record address

Column 17 (End of File)

Entry	Explanation
E	All records from the file must be processed before the program can end. The operator can signify a normal end of file by pressing the CMD and / (slash) keys.
Blank	The program can end whether or not all records from the file have been processed. If column 17 is blank for all files, all records from every file must be processed before the program can end.

Column 18 (Sequence)

Entry	Explanation
A	Sequence checking is to be done. Records in the file are in ascending order.
D	Sequence checking is to be done. Records in the file are in descending order.
Blank	No sequence checking is to be done.

Column 19 (File Format)

Entry	Explanation
F	Fixed length records. This entry indicates that all records in the file are of the same length.
Blank	F is assumed.

Columns 20-23 (Block Length)

Entry	Explanation
6-4096	Enter the length of the buffer desired for the console file. This buffer gives the operator the capability to get ahead of the system by containing as many records at a time as is physically possible. The length must be at least two greater than the record length specified in columns 24-27. This length is the physical buffer length.

Columns 24-27 (Record Length)

Entry	Explanation
4-160	To determine the record length for a console file, do the following: 1. Take the value from the record type with the highest end position specified for the console file by the input specifications and add one. 2. Add one for each field specified except the record identification field, if specified. See Figure 173 for an example on how to calculate record length. *Note:* The record length cannot be less than four or greater than 160.

Column 28 (Mode of Processing)

Leave this column blank.

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

Entry	Explanation
Blank	These columns must be blank if column 16 contains a P, S, or D.
1-29	Length of record key or disk address.

Column 31 (Record Address Type)

Entry	Explanation
A	Indexed file with unpacked keys.
Blank	Sequential or direct file.

Columns 32-38

Leave these columns blank.

Column 39 (Extension Code)

Entry	Explanation
E	Indicates R in column 16 for record address file.
Blank	Must be blank for any other file designation.

Columns 40-46 (Device)

Entry	Explanation
Console	CONSOLE as the device name specifies that this is a console (IDE) file. This specification logically ties the keyboard and the display screen together as one device. There can be only one console file in a program.

Columns 47-70

Leave these columns blank.

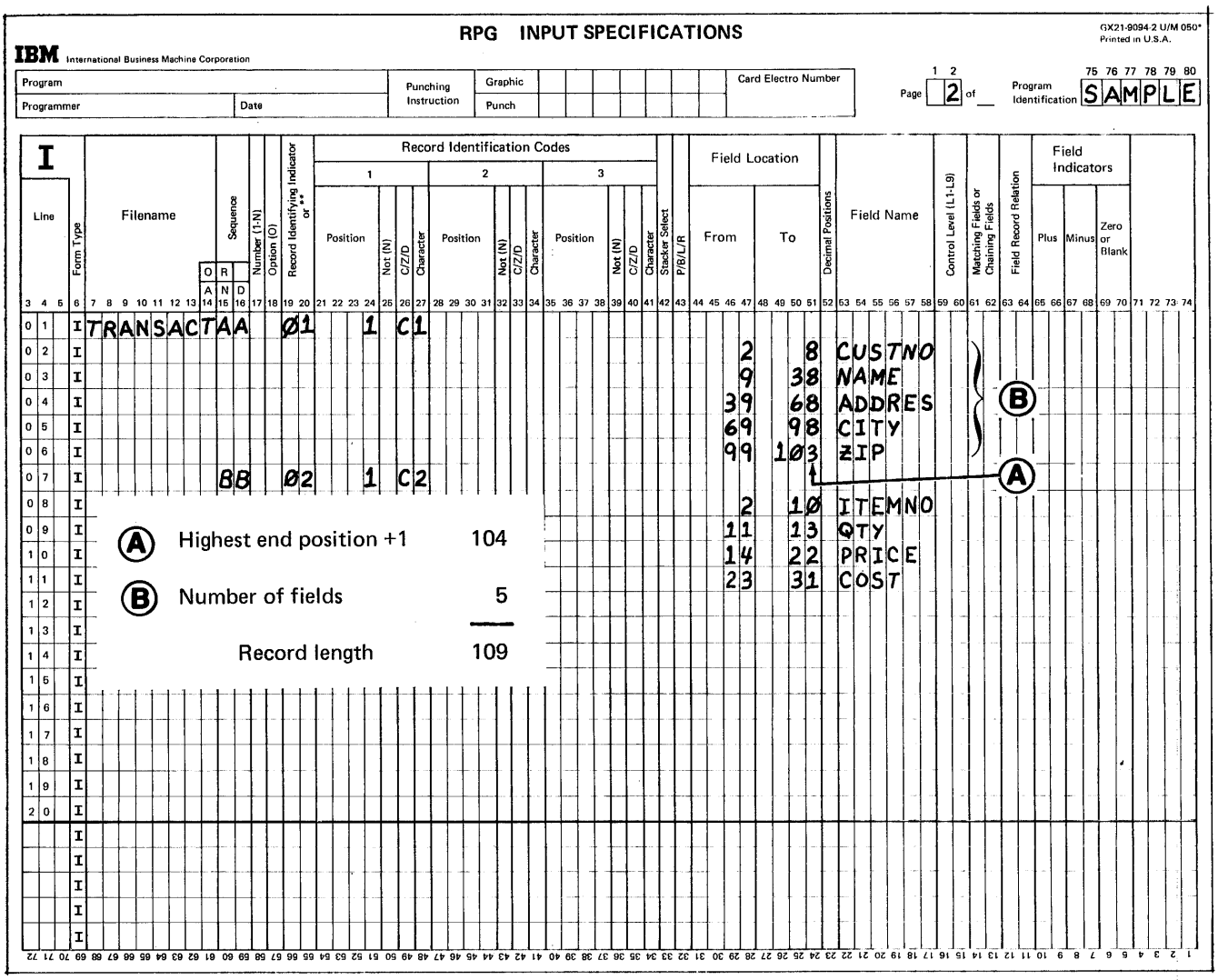


Figure 173. Calculating Record Length

Columns 71-72 (File Condition)

Entry	Explanation
U1-U8	The file is conditioned by the specified external indicator.
Blank	The file is not conditioned by an external indicator.

Columns 73-74

Leave these columns blank.

Columns 75-80 (Program Identification)

See *Common Entries.*

IDE INPUT SPECIFICATIONS

File and Record Identification Specifications

Column 6 (Form Type)

Enter an I for input specifications.

Columns 7-18

See *Input Specifications* in Part 1, *RPG II Programming Specifications.*

Columns 19-20 (Record Identifying Indicator)

Entry	Explanation
1-10	The indicator specified defines the command key the operator enters when selecting this record type. This indicator cannot be used to define more than one record type within the input specifications in a given program.

Columns 21-34 (Record Identification Codes)

Position 1 or positions 1 and 2 of each IDE record must contain record identification characters identifying which record was keyed. You must specify this 1-character or 2-character record identification code in columns 21-34. The record identification characters specified in columns 21-34 are automatically inserted into each new record when it is prompted. The rules for coding columns 21-34 follow.

Columns 21-24:

Entry	Explanation
1	Must contain the number 1, right-justified. Record position 1 must contain the record identification code for records entered from a console file.

Column 25: Leave this column blank.

Column 26:

Entry	Explanation
C	Character code

Column 27: Enter one of the alphabetic characters, special characters or digits, indicating the character that is present in position 1 of the IDE input record.

Columns 28-34: If a 1-character record identifier is used, leave these columns blank. If a 2-character record identifier is used, these columns are coded the same as columns 21-34 except columns 28-31 must contain the number 2 indicating record position 2.

Columns 35-74

Leave these columns blank.

Columns 75-80 (Program Identification)

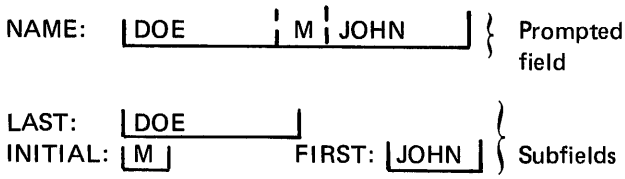
See *Common Entries.*

Field Specifications

Each field of an IDE record is described in columns 44-70 of the input specifications sheet. Each field is prompted for by the name specified in columns 53-58. You do not need to specify a field for the record identification characters; but if you do, the record identification field must be specified for position 1 or positions 1 and 2 of the record (Figure 174). (The record identification field is not prompted but is displayed automatically.) You can have a record identification with no other fields defined. However, you must define at least one field in the console file other than a record identification field.

Fields are prompted in the sequence they are specified on the input specifications sheet. The from field location entry (columns 44-47) must be one higher than the to field position entry (columns 48-51) for the previous field, unless subfields are specified. You cannot leave blanks between fields.

You can also specify subfields within your IDE fields. The from and to field locations for subfields overlap the from and to field locations for another field. Subfields are not prompted for, but are assigned values from the prompted field and can be used by your calculation and output specifications. In the following example, the name JOHN M. DOE is entered when the field NAME is prompted. LAST, INITAL, and FIRST are subfields within the NAME field. The values of the subfields are extracted from the prompted NAME field.



Column 6 (Form Type)

Enter an I for input specifications.

Columns 7-43

Leave these columns blank.

Note: AND lines are not allowed in input specifications for IDE records.

Columns 44-47 (From Field Location)

Enter the beginning position of the field specified in columns 53-58.

Columns 48-51 (To Field Location)

Enter the ending position of the field specified in columns 53-58.

Column 52 (Decimal Positions)

Entry	Explanation
0-9	Enter the number of decimal positions for numeric fields. Leave this column blank for alphameric fields.

Columns 53-58 (Field Name)

Entry	Explanation
1-6 alphameric characters	Enter the field name to be used with this data. Since this field name is used to prompt the operator for data, it should be fairly descriptive.

Columns 59-60 (Control Level)

Entry	Explanation
L1-L9	If this is a primary or secondary file, a control level indicator can be entered to indicate a control break occurs on the change in value of the field's contents.

Columns 61-62 (Matching Fields)

Entry	Explanation
M1-M9	If this is a primary or secondary file, a matching record's code can be entered to indicate a match field. Otherwise leave these columns blank.

Columns 63-64 (Field Record Relation)

Leave these columns blank.

Columns 65-70 (Field Indicators)

Entry	Explanation
01-99	Numeric indicator
H1-H9	Halt indicator

Columns 71-74

Leave these columns blank.

Columns 75-80 (Program Identification)

See *Common Entries.*

A 2-character record identification code consisting of the character 1 in position 1 and the character * in position 2 is specified in columns 24-34 of the input specifications sheet. This code is automatically entered in the input record and displayed on the display screen. The record identification code is also defined as a 2-character field named CODE. This field specification is optional and the field CODE is not prompted.

EXAMPLE

Figure 174 shows how a CONSOLE file is coded on the file description and input specifications sheets. The file TRANSACT is specified as a CONSOLE file by the file description specifications.

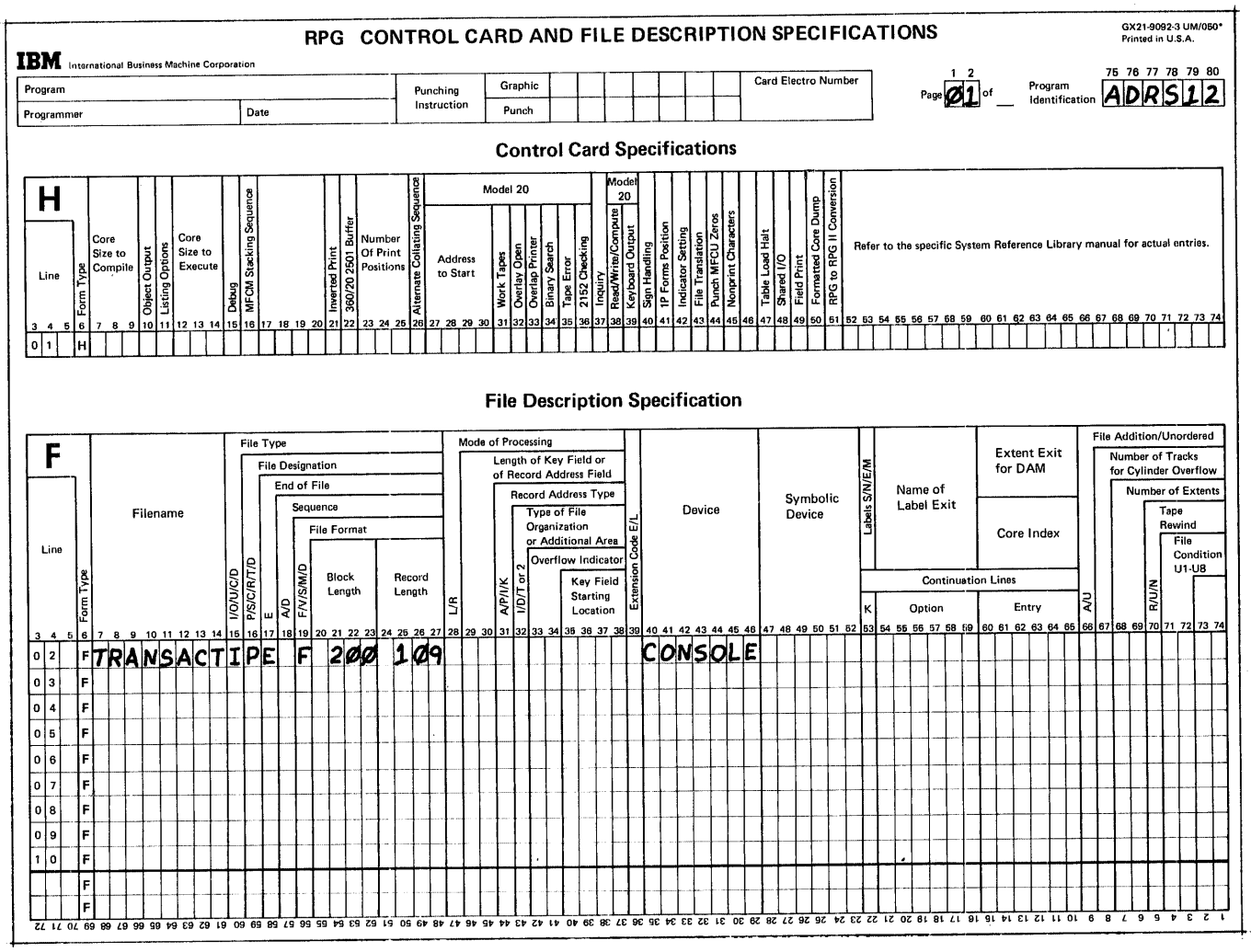


Figure 174 (Part 1 of 2). Coding a CONSOLE File

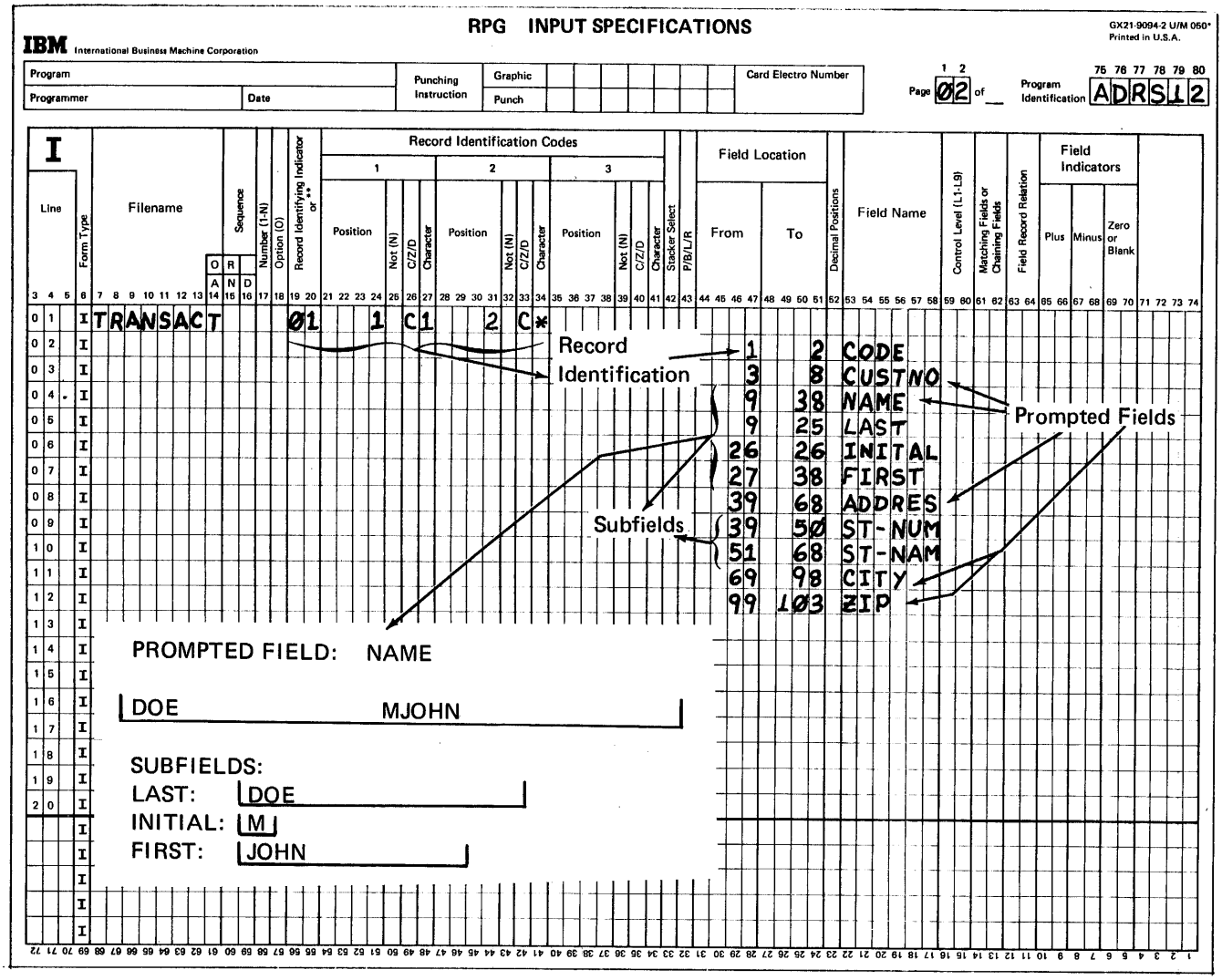


Figure 174 (Part 2 of 2). Coding a CONSOLE File

The fields CUSTNO, NAME, ADDRES, CITY, and ZIP are prompted. Notice that the from field location entry in columns 44-47 for each of these fields is one higher than the to position entry in columns 48-51 for the previous prompted field (for example, the from field location of 39 for ADDRES is one higher than the to field location of 38 for NAME). The field NAME is assigned the subfields LAST, INITAL, and FIRST and the field ADDRES is assigned the subfields ST-NUM and ST-NAM. These subfields are not prompted for but are assigned values extracted from the fields NAME and ADDRES. The insert on the bottom of Figure 174 shows the value of the subfields when DOE MJOHN is entered for NAME.

Part 3

Supplementary Information

This chapter contains five complete RPG II sample programs: SAMPL1 and SAMPL2, and EXMPL1, EXMPL2, and EXMPL3. These sample programs are designed to illustrate some of the functions of RPG II. Although they are relatively simple programs, they are complete jobs which can be run on any IBM System/32. The sample programs EXMPL1, EXMPL2, and EXMPL3 are included on the distribution diskette for the RPG II program product. For information on RPG II installation procedures, see *IBM System/32 System Control Programming Reference Manual*, GC21-7593.

SAMPLE PROGRAM 1 (SAMPL1)

SAMPL1 loads an indexed disk file which consists of 100 data records created by calculation operations. (Each record contains two fields: COUNT and RECNBR.) The program only requires you to enter a blank data record at the beginning of the job and press the CMD and / keys (console end of file) to end the job when the first prompt appears for the field EOF. SAMPL1 should be followed by SAMPL2 which prints the indexed file loaded in SAMPL1 to verify that the file was loaded properly. Figure 175 shows the specification sheets required for SAMPL1.

Specifications

The control specification (Figure 175, part 1) should be supplied for every job. It is the first record in the source program and identifies the program.

All files used in SAMPL1 are first described on the file description specifications sheet (Figure 175, part 1). The primary input file, INPUT, is assigned to the CONSOLE. The E in column 17 ensures that the program does not end until after the last record is read from INPUT. At the end of the job, the indexed output file, DISKOUT, consists of 128-position records with a 6-position key field starting in the first record position. Messages indicating that the job was completed successfully are written to the printer output file, OUTPUT, at the end of the job.

The input file, INPUT, is further described on the input specifications sheet (Figure 175, part 2). The INPUT record contains a 1-position blank field called NODATA (blank in position 1) describing the record identification code. The field NODATA is not prompted. A 1-position field that will be prompted called EOF is described for position 2 of the input record. When the prompt for EOF is made and the CMD and / keys are pressed (end of file on the console), the LR indicator turns on.

All calculations (Figure 175, part 3) are conditioned by the LR indicator; therefore, they are executed at LR calculation time. See Chapter 6, *RPG II Object Program Logic (Detailed)* in Part 2, *RPG II Programmer's Guide*. The record number field (RECNBR) keeps track of the number of records written to DISKOUT. The COUNT field accumulates in increments of five to provide a unique key field for each record written to DISKOUT. The records are written on disk using the EXCPT operation code and exception output (E in column 15 of the output specifications sheet). These calculations are part of the REPEAT loop and are executed 100 times, until COUNT equals 505 and 100 disk records have been created. At the end of the loop, one is subtracted from RECNBR to indicate the actual number of records that have been loaded.

The output specifications (Figure 175, part 4) conditioned by LR cause a message to be written to the OUTPUT file indicating:

- The job is finished.
- The number of records loaded.
- File and key field descriptions.
- Brief description of the function of program SAMPL2.

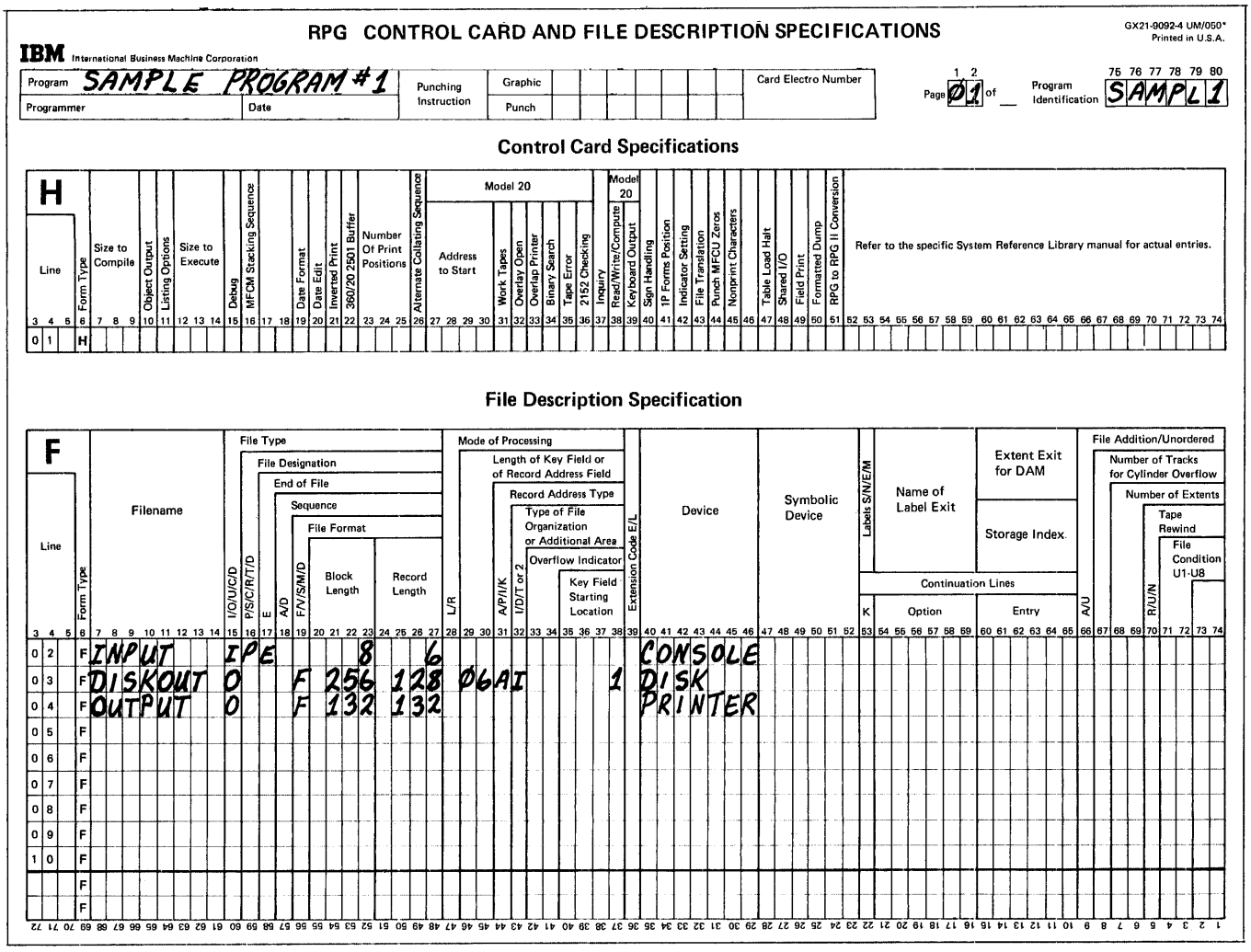


Figure 175 (Part 1 of 4). Sample Program 1 (SAMPL1)

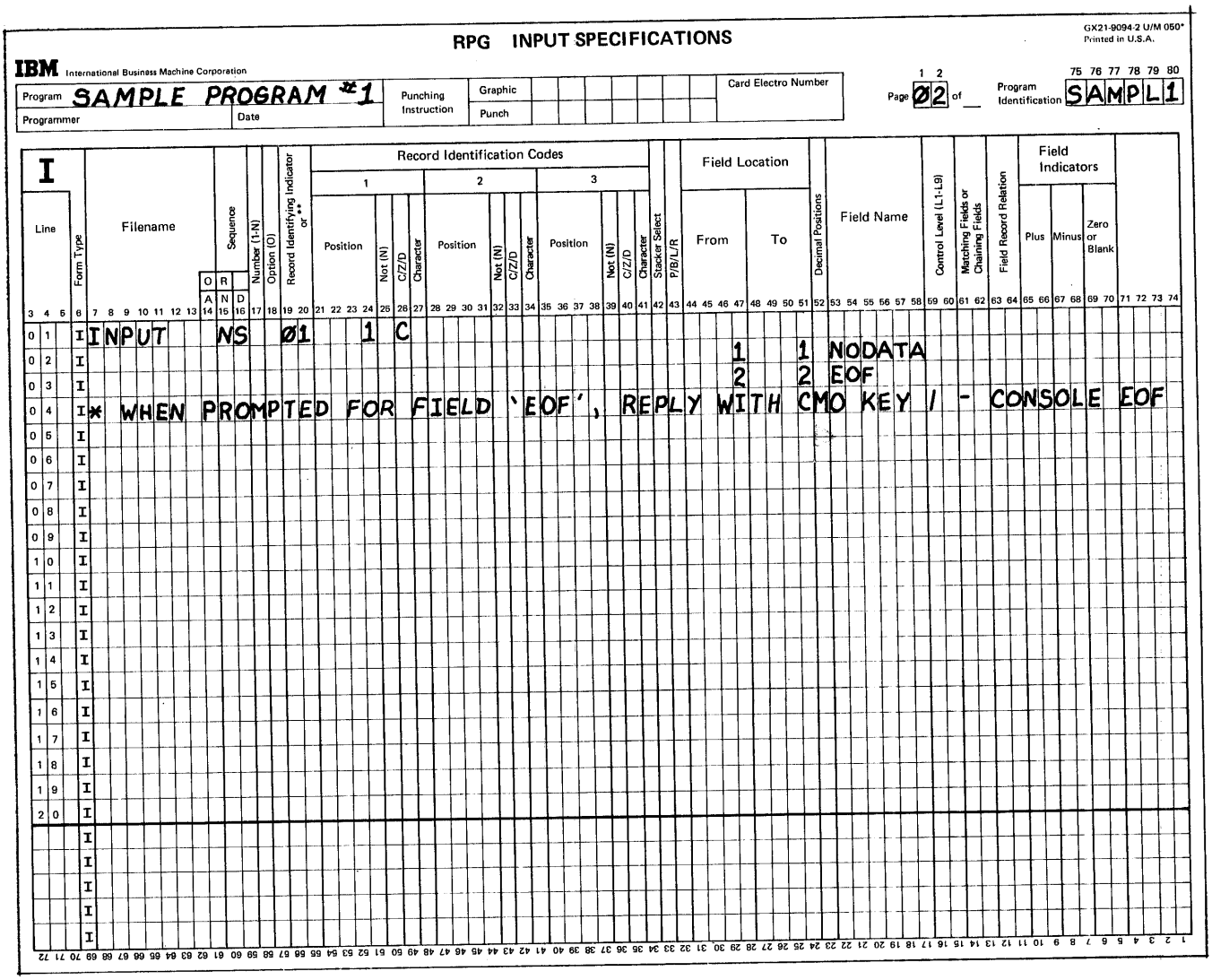
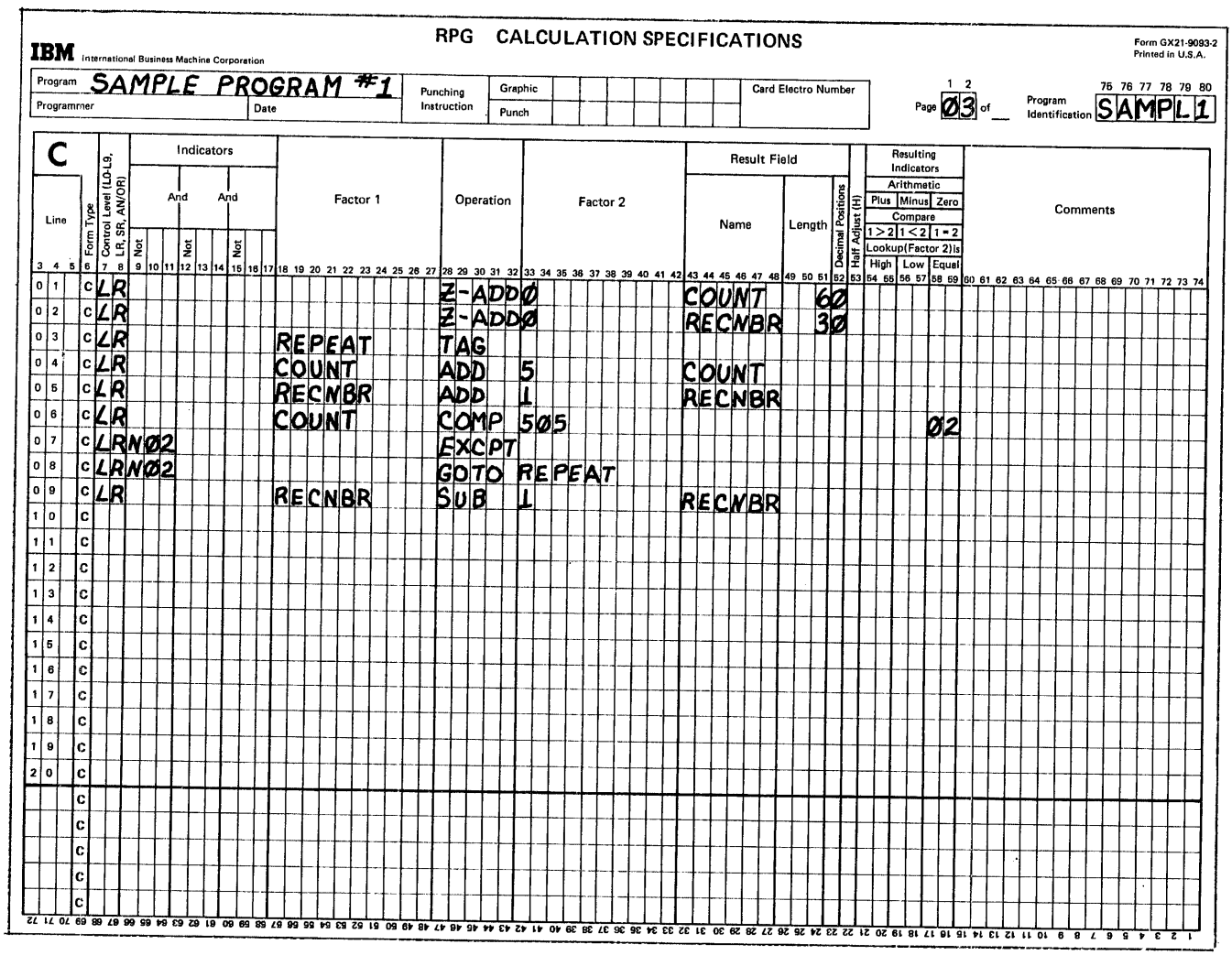


Figure 175 (Part 2 of 4). Sample Program 1 (SAMPL1)



IBM International Business Machine Corporation

RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

Program: SAMPLE PROGRAM #1 | Programmer | Date | Punching Instruction | Graphic | Punch | Card Electro Number

Page 03 of __ | Program Identification: SAMPL1

Line	Form Type	Control Level (LO-L9, LR, SR, AN/OR)	Indicators: And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus / Minus / Zero; Compare: 1>2 / 1<2 / 1=2; Lookup(Factor 2)is: High / Low / Equal	Comments
01	C	LR			Z-ADD	Ø	COUNT		60			
02	C	LR			Z-ADD	Ø	RECNBR		30			
03	C	LR		REPEAT	TAG							
04	C	LR		COUNT	ADD	5	COUNT					
05	C	LR		RECNBR	ADD	1	RECNBR					
06	C	LR		COUNT	COMP	505					Ø2	
07	C	LR	N02		EXCPT							
08	C	LR	N02		GOTO	REPEAT						
09	C	LR		RECNBR	SUB	1	RECNBR					
10	C											
11	C											
12	C											
13	C											
14	C											
15	C											
16	C											
17	C											
18	C											
19	C											
20	C											
	C											
	C											
	C											
	C											
	C											

Figure 175 (Part 3 of 4). Sample Program 1 (SAMPL1)

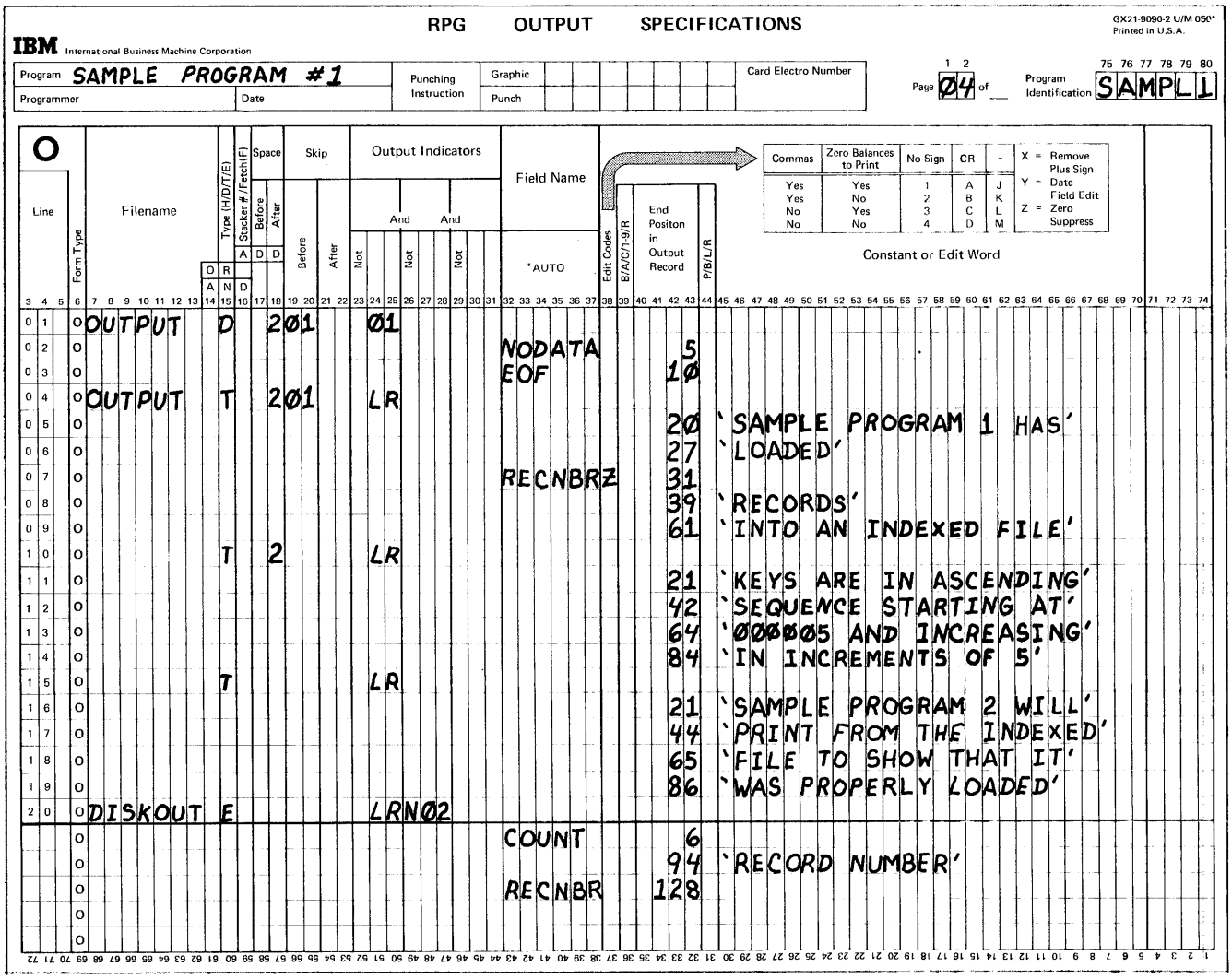


Figure 175 (Part 4 of 4). Sample Program 1 (SAMPL1)

SAMPLE PROGRAM 2 (SAMPL2)

SAMPL2 must be preceded by sample program 1 (SAMPL1). SAMPL2 reads the indexed file created by SAMPL1 and prints out fields from each record read. Thus, SAMPL2 allows you to verify that SAMPL1 loaded the indexed file properly. The program specifications required for SAMPL2 are shown in Figure 176.

Specifications

The control specifications (Figure 176, part 1) should be present in every job. It is the first record in the source program.

The files in SAMPL2 are described on the file description specifications sheet (Figure 176, part 1). The indexed file (DISKOUT) loaded in SAMPL1 is defined as the primary input file for SAMPL2. The E in column 17 ensures that SAMPL2 does not end until end of file is reached on DISKOUT. The records read from DISKOUT are printed on the output file, OUTPUT. An overflow indicator is specified in columns 33-34 so that subsequent operations can be conditioned on overflow.

The primary input file, DISKOUT, is further described on the input specifications sheet (Figure 176, part 4). DISKOUT records are 128 positions long identified by a zero in position 1. When an input record containing a zero in position 1 is read, indicator 01 is turned on.

RPG CONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS

GX21-9092-4 U/M 050*
Printed in U.S.A.

IBM International Business Machine Corporation

Program: SAMPLE PROGRAM 2 | Punching Instruction: Graphic / Punch | Card Electro Number

Programmer | Date

Page 01 of __ | Program Identification: SAMPL2

Control Card Specifications

H

Line	Form Type	Size to Compile	Object Output	Listing Options	Size to Execute	Debug	MFCM Stacking Sequence	Date Format	Date Edit	Inverted Print	360/20 2501 Buffer	Number Of Print Positions	Alternate Collating Sequence	Model 20: Address to Start	Model 20: Work Tapes	Model 20: Overlay Open	Model 20: Overlap Printer	Model 20: Binary Search	Model 20: Tape Error	Model 20: 2152 Checking	Model 20: Inquiry	Model 20: Read/Write/Compute	Model 20: Keyboard Output	Sign Handling	1P Forms Position	Indicator Setting	File Translation	Punch MFCU Zeros	Nu.nprint Characters	Table Load Halt	Shared I/O	Field Print	Formatted Dump	RPG to RPG II Conversion	Refer to the specific System Reference Library manual for actual entries.
3 4 5	6	7 8 9	10	11	12 13 14	15	16	17 18	19	20	21	22 23 24 25	26	27 28 29 30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
01	H																																		

File Description Specification

F

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	K	Option	Entry	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	R/U/N	File Condition U1-U8
3 4 5	6	7 8 9 10 11 12 13 14	15	16	17	18	19	20 21 22 23	24 25 26 27	28	29 30	31	32	33 34	35 36 37 38	39	40 41 42 43 44 45 46	47 48 49 50 51 52	53	54 55 56 57 58 59	60 61 62 63 64 65	66	67	68 69	70	71 72	73 74
02	F	DISKOUT	I	P	E	A	F	512	128		Ø6A	I				1	DISK														
03	F	OUTPUT	O				F	132	132				OF				PRINTER														
04	F																														
05	F																														
06	F																														
07	F																														
08	F																														
09	F																														
10	F																														
	F																														
	F																														

72 71 70 69 68 67 66 65 64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

RPG INPUT SPECIFICATIONS

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

IBM International Business Machine Corporation

Program: SAMPLE PROGRAM 2 | Punching Instruction: Graphic / Punch | Card Electro Number

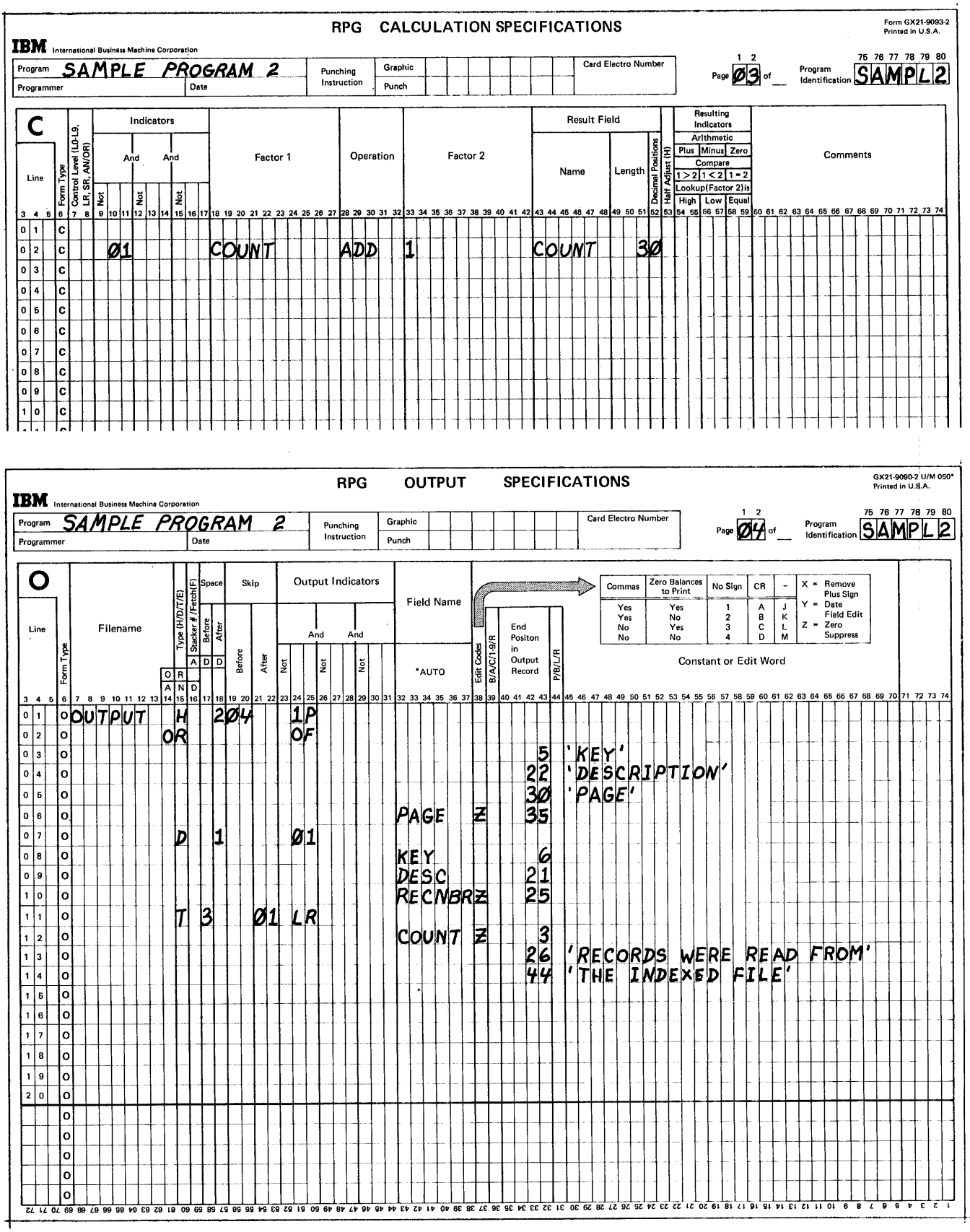
Programmer | Date

Page 02 of __ | Program Identification: SAMPL2

I

Line	Form Type	Filename	Sequence (AND/OR)	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	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
3 4 5	6	7 8 9 10 11 12 13	O A 14 / R N 15 / D 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
01	I	DISKOUT	NS			Ø1		1	C	Ø																				
02	I																				1	6		KEY						
03	I																				82	94		DESC						
04	I																				126	128	Ø	RECNBR						
05	I																													
06	I																													
07	I																													
08	I																													
09	I																													

● **Figure 176 (Part 1 of 2). Sample Program 2 (SAMPL2)**



RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2 Printed in U.S.A.

IBM International Business Machine Corporation

Program: SAMPLE PROGRAM 2 | Punching Instruction: Graphic / Punch | Card Electro Number

Programmer | Date

Page 03 of __ | Program Identification: SAMPL2

C

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / And: Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus / Minus / Zero; Compare: 1>2 / 1<2 / 1=2; Lookup(Factor 2)is: High / Low / Equal	Comments
01	C		01	COUNT	ADD	1	COUNT		30			

RPG OUTPUT SPECIFICATIONS

GX21-9090-2 U/M 050* Printed in U.S.A.

IBM International Business Machine Corporation

Program: SAMPLE PROGRAM 2 | Punching Instruction: Graphic / Punch | Card Electro Number

Programmer | Date

Page 04 of __ | Program Identification: SAMPL2

O

Line	Form Type	Filename	Type (H/D/T/E): O A / R N / D	Stacker # / Fetch(F)	Space: Before A / After D	Skip: Before D / After D	Output Indicators: And / And: Not	Field Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R	Constant or Edit Word
													5 'KEY'
													22 'DESCRIPTION'
													30 'PAGE'
								PAGE		N	35		
01	O	OUTPUT	H	2	04		1P						
	O		OR				OF						
	O		D	1			01						
	O							KEY			6		
	O							DESC			21		
	O							RECNBR		N	25		
	O		T	3		01	LR						
	O							COUNT		N	3		
	O										26		'RECORDS WERE READ FROM'
	O										44		'THE INDEXED FILE'

Commas | Zero Balances to Print | No Sign | CR | -

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

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

Figure 176 (Part 2 of 2). Sample Program 2 (SAMPL2)

The calculation specifications (Figure 176, part 2) adds one to COUNT when indicator 01 is on. The COUNT field is used to keep track of the number of records read from the DISKOUT file.

In output specifications (Figure 176, part 2), the 1P and OF indicators, specified in an OR relationship, cause a heading line to be printed on the first output page and on each succeeding page. Conditioned by indicator 01, the disk record just read is printed.

The next record is read from DISKOUT and the same calculation and output operations are repeated until there are no more records in the disk file. When end of file is reached on DISKOUT, the LR indicator is turned on.

Conditioned by LR, a total line is printed indicating how many records are read from DISKOUT. If the number printed (COUNT) is 100, SAMPL1 and SAMPL2 were executed properly.

EXAMPLE PROGRAMS

This example contains specifications sheets for the three complete RPG II programs: EXMPL1, EXMPL2, and EXMPL3. The programs are designed to be run in sequence. EXMPL1, EXMPL2, and EXMPL3 are included on the distribution diskette for the RPG II program product.

Example Program 1

EXMPL1 loads master records into an indexed file and creates a consecutive file of transactions. The transaction file is processed against the master file in EXMPL2. EXMPL2 should follow EXMPL1. Figure 177 shows the completed specifications sheets for EXMPL1.

Control Specifications

This record may be present in every job. It is the first record in the source program.

File Description Specifications

These specifications describe the files in the program. The input record file, INPUT, is read from CONSOLE. An E in column 17 indicates that the program ends when the last data record keyed in is processed. The indexed output file, MASTER, consists of 26-position records with a 5-position key field starting in the second record position.

A consecutive output file, TRANS, with a 10-position record length is also specified by the file description specifications. A printer output file, PRINT, with a record length of 78 is also defined by file description specifications.

Input Specifications

There are two types of records in the input console file, INPUT: master and transaction. These records are entered using interactive data entry. A character M in position 1 of the input records turns on record identifying indicator 01, indicating a master record. A character A, B, and C in position 1 of the input records turns on record identifying indicator 02, 03, or 04, indicating a transaction record type A, B, or C, respectively. No sequence checking occurs for either type of record (AA and AB in columns 15-16).

RPG CONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS

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

IBM International Business Machine Corporation

Program	EXAMPLE PROGRAM #1	Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 01 of __ Program Identification EXMPL1

Control Card Specifications

H

Line	Form Type	Core Size to Compile	Object Output	Listing Options	Core Size to Execute	Debug	MFCM Stacking Sequence	Inverted Print	360/20 2501 Buffer	Number Of Print Positions	Alternate Collating Sequence	Model 20: Address to Start	Model 20: Work Tapes	Model 20: Overlay Open	Model 20: Overlap Printer	Model 20: Binary Search	Model 20: Tape Error	Model 20: 2152 Checking	Model 20: Inquiry	Model 20: Read/Writes/Compute	Model 20: Keyboard Output	Sign Handling	1P Forms Position	Indicator Setting	File Translation	Punch MFCU Zeros	Nonprint Characters	Table Load Halt	Shared I/O	Field Print	Formatted Core Dump	RPG to RPG II Conversion	Refer to the specific System Reference Library manual for actual entries.
01	H				008																											

File Description Specification

F

Line	Form Type	Filename	File Type	File Designation	End of File	Sequence	File Format	Block Length	Record Length	Mode of Processing	Length of Key Field or of Record Address Field	Record Address Type	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Core Index	Continuation Lines: K	Continuation Lines: Option	Continuation Lines: Entry	A/U	File Addition/Unordered	Number of Tracks for Cylinder Overflow	Number of Extents	R/U/N	Tape Rewind	File Condition U1-U8
02	F																															
03	F																															
04	F	THIS PROGRAM																														
05	F																															
06	F	1. LOADS MASTER RECORDS INTO AN INDEXED																														
07	F	FILE.																														
08	F																															
09	F	2. CREATES A CONSECUTIVE FILE OF																														
10	F	TRANSACTIONS.																														
11	F																															
12	F	3. COUNTS THE TOTAL NUMBER OF RECORDS LOADED																														
13	F	INTO EACH FILE. THIS TOTAL IS PRINTED AT																														
14	F	THE END OF JOB.																														
15	F																															
16	F																															
17	F	INPUT		I	P	E	F	32	32								CONSOLE															
18	F	MASTER		O			F	52	26		05A					2	DISK															
19	F	TRANS		O			F	30	10								DISK															
20	F	PRINT		O			F	78	78								PRINTER															
	F																															
	F																															
	F																															
	F																															
	F																															

Figure 177 (Part 1 of 3). Specifications for EXMPL1

RPG INPUT SPECIFICATIONS

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

IBM International Business Machine Corporation

Program: EXAMPLE PROGRAM *1 | Programmer | Date | Punching Instruction | Graphic | Punch | Card Electro Number

Page 02 of __ | Program Identification: EXMPL1

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or *	Record Identification Codes: 1 Position	1 Not (N)	1 C/Z/D	1 Character	2 Position	2 Not (N)	2 C/Z/D	2 Character	3 Position	3 Not (N)	3 C/Z/D	3 Character	Stacker Select	P/B/L/R	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
01	I	*																												
02	I	* MASTER RECORD DESCRIPTION																												
03	I	*																												
04	I	INPUT	AA	01			1		CM																					
05	I																				1	1		ID						
06	I																				2	6		KEY						
07	I																				7	14		DESC						
08	I																				15	180		VALUEA						
09	I																				19	220		VALUEB						
10	I																				23	260		VALUEC						
11	I	*																												
12	I	* TRANSACTION RECORD DESCRIPTIONS																												
13	I	*																												
14	I		AB			02		1	CA																					
15	I		OR			03		1	CB																					
16	I		OR			04		1	CC																					
17	I																				1	1		ID						
18	I																				2	6		KEY						
19	I																				7	10		AMT						
20	I																													

RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program: EXAMPLE PROGRAM *1 | Programmer | Date | Punching Instruction | Graphic | Punch | Card Electro Number

Page 03 of __ | Program Identification: EXMPL1

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators And Not	And Not	And Not	Factor 1	Operation	Factor 2	Result Field Name	Result Field Length	Result Field Decimal Positions	Half Adjust (H)	Resulting Indicators Arithmetic Plus	Arithmetic Minus	Arithmetic Zero	Compare 1>2	Compare 1<2	Compare 1=2	Lookup(Factor 2)is High	Lookup(Factor 2)is Low	Lookup(Factor 2)is Equal	Comments
01	C		01			TOTMAS	ADD	1	TOTMAS	40			05									
02	C		02																			
03	C	OR	03																			
04	C	OR	04			TOTTRN	ADD	1	TOTTRN	40			06									
05	C																					
06	C																					

Figure 177 (Part 2 of 3). Specifications for EXMPL1

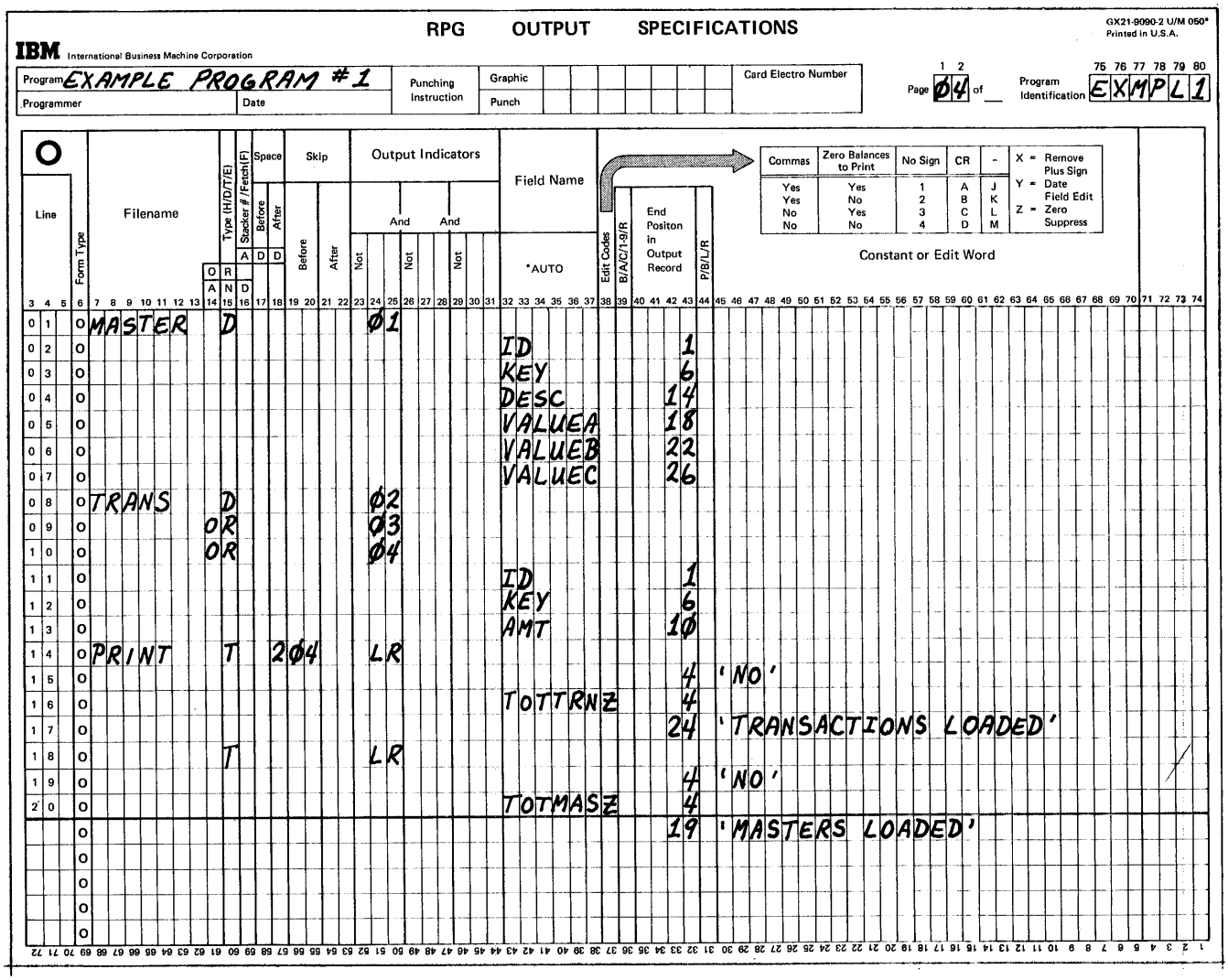


Figure 177 (Part 3 of 3). Specifications for EXMPL1

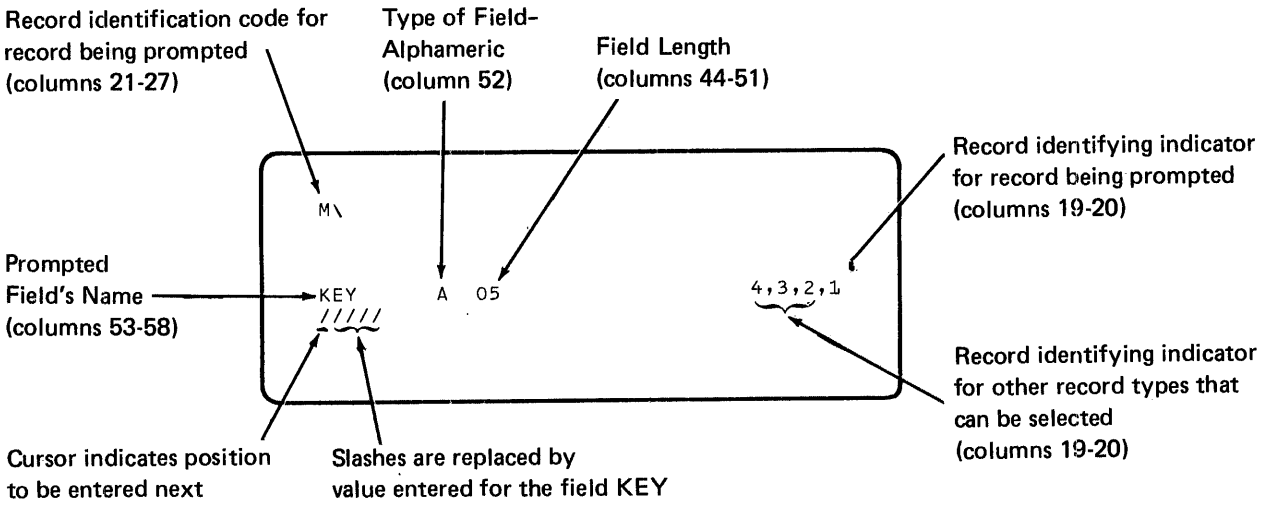


Figure 178. Display Screen Before Record Type M is Entered

Figure 178 shows the display screen for the record type M before any data has been entered. Notice that the record identification code M is automatically displayed at the top of the screen. Fields are prompted in the order they are specified on the input specifications sheet. The first field prompted is KEY. The screen also displays the format and the length of the field being prompted. As the field is entered, the slashes below the field name are replaced by the field's contents. The cursor indicates the position to be entered next.

On the right side of the screen, you can see a group of numbers displayed. These numbers correspond to the record identifying indicators for all record types that you can select to be processed next. You can select any of these record types to be prompted next by pressing the CMD key and the digit key corresponding to the record type you want. The last number indicates the record type currently being prompted. (The last number may not indicate the record currently being prompted if the record is part of an OR group.)

Figure 179 shows the display screen for the record type M after the first four fields have been entered. The last field in the record, VALUEC, is currently being prompted. Notice that all fields that have already been entered for this record are displayed at the top of the screen.

When you want to stop entering input data, press the CMD, /, and ENTER keys.

Calculation Specifications

The field name TOTMAS is incremented by one when record identifying indicator 01 is on. This maintains a running total of the master records which have been read from INPUT and transferred to disk. The field TOTTRN is incremented by one when record identifying indicator 02, 03, or 04 is on, maintaining a running total of the transaction records which are read from INPUT and transferred to disk.

Output Specifications

Four different output records are described in these specifications: one detail record for the master file (MASTER), one detail record for the transaction file (TRANS), and two total records for the printer file (PRINT).

The detail records for MASTER are conditioned by record identifying indicator 01. The detail records for TRANS are conditioned by record identifying indicators 02, 03, and 04. Both total lines for PRINT are printed when the last record identifying indicator is turned on (LR in columns 23-25). The first total line is for total transactions loaded. The printer skips to line 4 before the printing of the first total line and double spacing occurs before the printing of the second total line. The second total line is for total masters loaded.

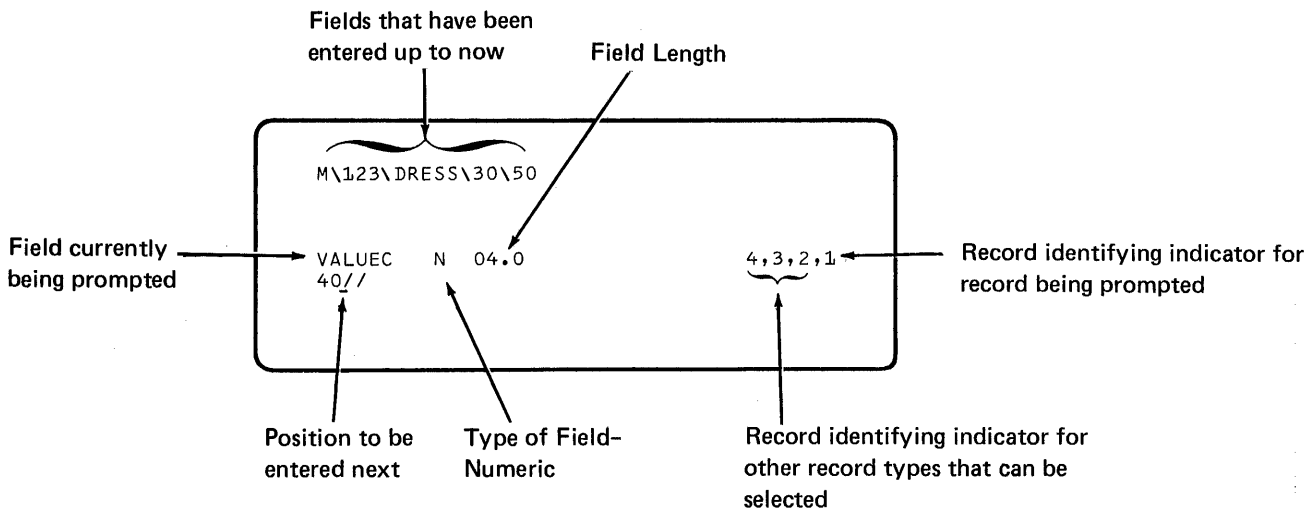


Figure 179. Display Screen When Record Type M is Partially Entered

Example Program 2

EXMPL2 must be preceded by EXMPL1. EXMPL2 reads from the transaction file, TRANS, created by EXMPL1 and accumulates totals for A, B, and C records. EXMPL2 also retrieves corresponding master records for transaction records and prints an error message if a corresponding master record is not found. Figure 180 shows the completed specifications sheets for EXMPL2.

Control Specifications

Control specifications may be present in every job. They are the first record in the source program.

File Description Specifications

The input file for EXMPL2, TRANS (the output transaction file for EXMPL1), is read from disk. An E in column 17 indicates that the program ends when the last data record in the input file is processed. The output file, PRINT, consists of 72-position records.

An overflow indicator (OF in columns 33-34) conditions printing of records in the file. The indexed file, MASTER, is a chained update file to be processed by keys. It consists of 26-position records with a 5-position key field starting in the second record position. It is a single volume file on disk.

Input Specifications

There are two types of files specified by the input specifications: transaction and master. A character A, B, or C in position 1 of the input records turns on record identifying indicator 01, 02, or 03, indicating a transaction record type A, B, or C respectively. A character M in position 1 of the update records turns on record identifying indicator 04, indicating an update record. No sequence checking occurs for either type (AA and AB in columns 15-16).

Calculation Specifications

When indicator 01, 02, or 03 is on, two operations occur:

1. A matching master record is retrieved for a transaction record (lines 01, 02, and 03 of the calculation specifications).

2. The AMT field of the transaction cards is added to the appropriate value (VALUEA, VALUEB, or VALUEC) on the master record depending on the type of record (record identifying indicator 01, 02, or 03).

If no matching record is found, indicator 10 turns on.

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

IBM International Business Machine Corporation

RPG CONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS

Program EXAMPLE PROGRAM # 2 | Punching Instruction: Graphic / Punch | Card Electro Number | Page 01 of __ | Program Identification (75–80): EXMPL2

Programmer | Date

Control Card Specifications

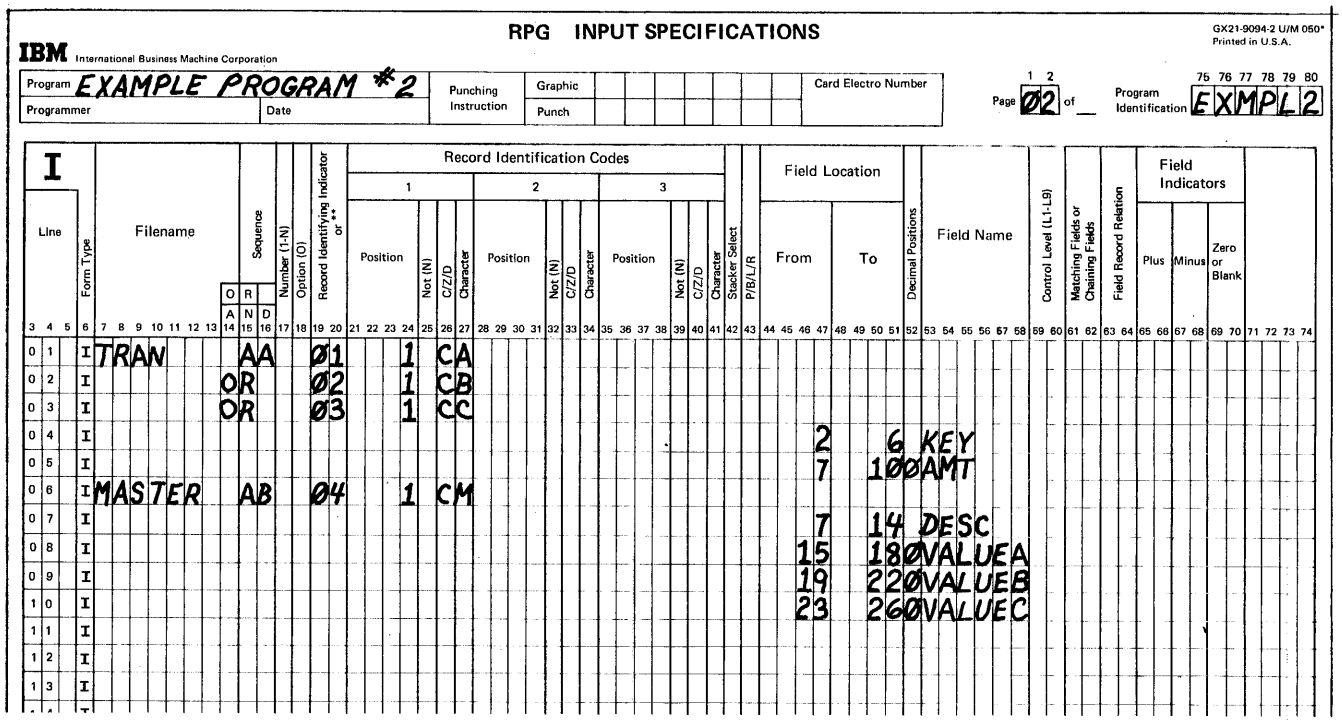
```
  01 H
```

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

File Description Specification

```
  02 F*************************************************
  03 F*                                               *
  04 F* THIS PROGRAM                                  *
  05 F*                                               *
  06 F*     1. READS EACH TRANSACTION RECORD AND      *
  07 F*        DETERMINES IF IT IS AN A, B, OR C RECORD.*
  08 F*                                               *
  09 F*     2. USES THE KEY FIELD OF EACH TRANSACTION *
  10 F*        TO DIRECTLY RETRIEVE THE MATCHING      *
     F*        MASTER RECORD.                         *
     F*                                               *
  02 F*     3. ADDS 'A' RECORD AMOUNTS TO VALUE A, 'B'*
  03 F*        RECORD AMOUNTS TO VALUE B, AND 'C' RECORD*
  04 F*        AMOUNTS TO VALUE C.                    *
  05 F*                                               *
  06 F*     4. PRINTS AN ERROR MESSAGE IF THERE IS ONE*
  07 F*        TRANSACTION RECORD FOR WHICH THERE IS  *
  08 F*        NO MASTER RECORD.                      *
  09 F*                                               *
  10 F*************************************************
     FTRANS   IPE F  30  10            DISK
     FPRINT   O   F  72  72     OF     PRINTER
  10 FMASTER  UC  F  52  26R05AI      2DISK
     F
     F
```

Figure 180 (Part 1 of 4). Specifications for EXMPL2



RPG INPUT SPECIFICATIONS

IBM International Business Machine Corporation

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

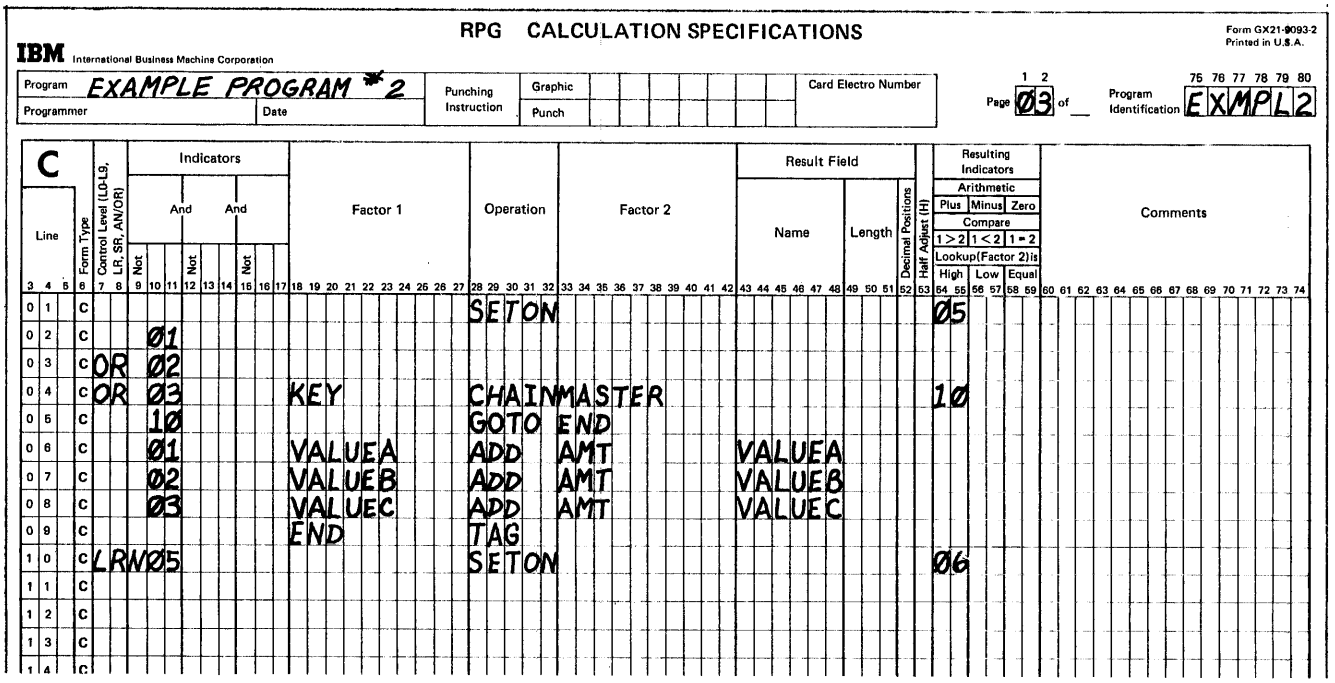
Program EXAMPLE PROGRAM *2 | Punching Instruction | Graphic / Punch | Card Electro Number

Programmer | Date

Page 02 of __ | Program Identification EXMPL2

I

Line	Form Type	Filename	Sequence (O A / R N / D)	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	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
01	I	TRAN	AA			01	1		CA																					
02	I		OR			02	1		CB																					
03	I		OR			03	1		CC																					
04	I																				2	6		KEY						
05	I																				7	100		AMT						
06	I	MASTER	AB			04	1		CM																					
07	I																				7	14		DESC						
08	I																				15	180		VALUEA						
09	I																				19	220		VALUEB						
10	I																				23	260		VALUEC						
11	I																													
12	I																													
13	I																													



RPG CALCULATION SPECIFICATIONS

IBM International Business Machine Corporation

Form GX21-9093-2
Printed in U.S.A.

Program EXAMPLE PROGRAM *2 | Punching Instruction | Graphic / Punch | Card Electro Number

Programmer | Date

Page 03 of __ | Program Identification EXMPL2

C

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And Not	And Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus	Minus	Zero; Compare 1>2	1<2	1=2; Lookup(Factor 2)is High	Low	Equal	Comments
01	C					SETON						05							
02	C		01																
03	C	OR	02																
04	C	OR	03		KEY	CHAIN	MASTER					10							
05	C		10			GOTO	END												
06	C		01		VALUEA	ADD	AMT	VALUEA											
07	C		02		VALUEB	ADD	AMT	VALUEB											
08	C		03		VALUEC	ADD	AMT	VALUEC											
09	C				END	TAG													
10	C	LR	W05			SETON						06							
11	C																		
12	C																		
13	C																		
14	C																		

Figure 180 (Part 2 of 4). Specifications for EXMPL2

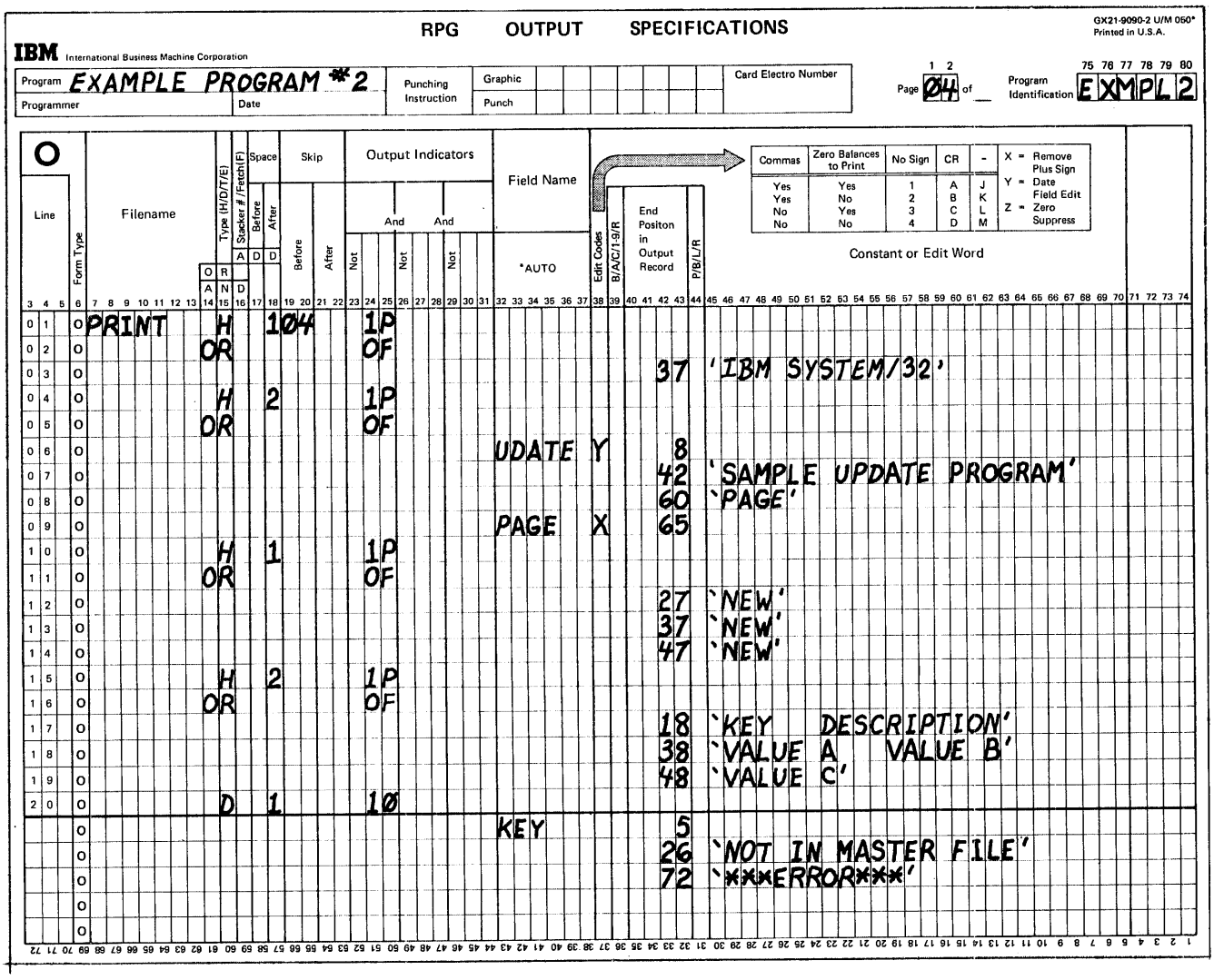


Figure 180 (Part 3 of 4). Specifications for EXMPL2

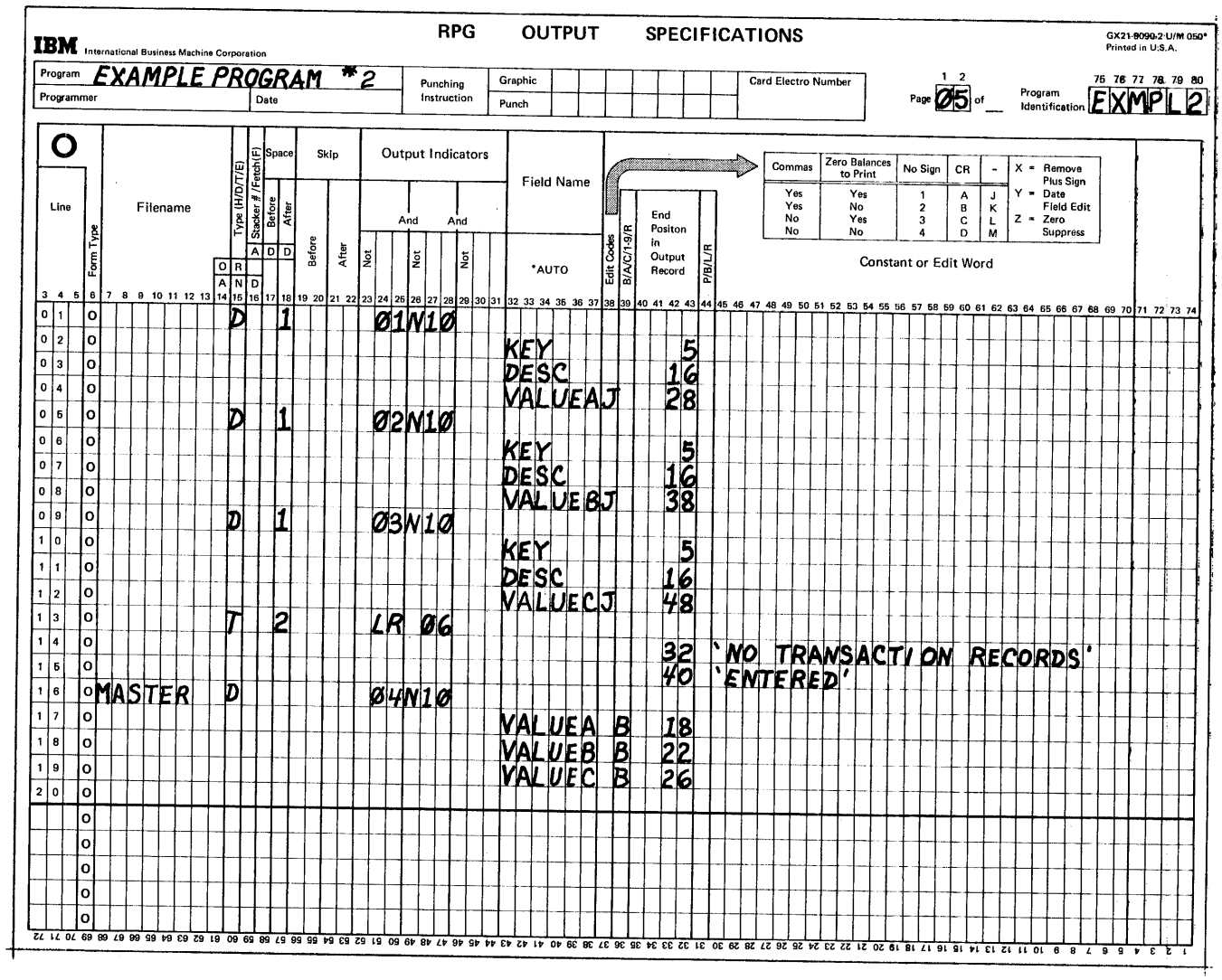


Figure 180 (Part 4 of 4). Specifications for EXMPL2

Output Specifications

Nine printer output lines are described in these specifications. Four header lines conditioned by the first page indicator (1P in columns 23-25) or an overflow indicator (OF in columns 23-25) are printed. They are printed at the top of each page of the listing.

Four detail lines are also printed. A detail line is printed for each transaction record with no matching master record. For each type of transaction record, A, B, or C, the accumulative value is printed (detail lines conditioned by indicators 01, 02, or 03, and not 10). These detail lines are single spaced.

A total line is printed if no transaction records were entered.

A detail record is written on disk for the indexed update file, MASTER. It is conditioned by two indicators—the record identifying indicator 04 and not 10, which is the record identifying indicator for no matching master record, a match between the master and transaction record.

Example Program 3

EXMPL3 must be preceded by EXMPL2. EXMPL3 reads from the indexed file, MASTER, and performs the following calculation: value A + value B - value C. If the result is negative, a message is printed. Figure 181 shows the completed specifications sheets for EXMPL3.

Control Specifications

These specifications should be present in every job. They are the first record in the source program.

File Description Specifications

The input file for EXMPL3, MASTER, is an indexed file (I in column 32). An E in column 17 indicates that the program ends when the last data record in the input file is processed. It consists of 26-position records with a 5-position key field starting in the second record position. A printer output file, PRINT, with a record length of 78 is also defined by the file description specifications.

Input Specifications

A character M in position 1 of the input records turns on record identifying indicator 01.

Calculation Specifications

The record identifying indicator 01 conditions all calculations. Values A, B, and C are accumulated (lines 03-05). The calculation, value A + value B - value C, is performed and accumulated (lines 01, 02, and 06). If the calculation is negative, the resulting indicator 22 is set on to condition the printing of a message.

Output Specifications

These specifications print four header lines, each conditioned by the first page indicator (1P) or an overflow indicator (OF).

One detail line is printed for each program cycle. One total line is also printed when the last record indicator, LR, is on.

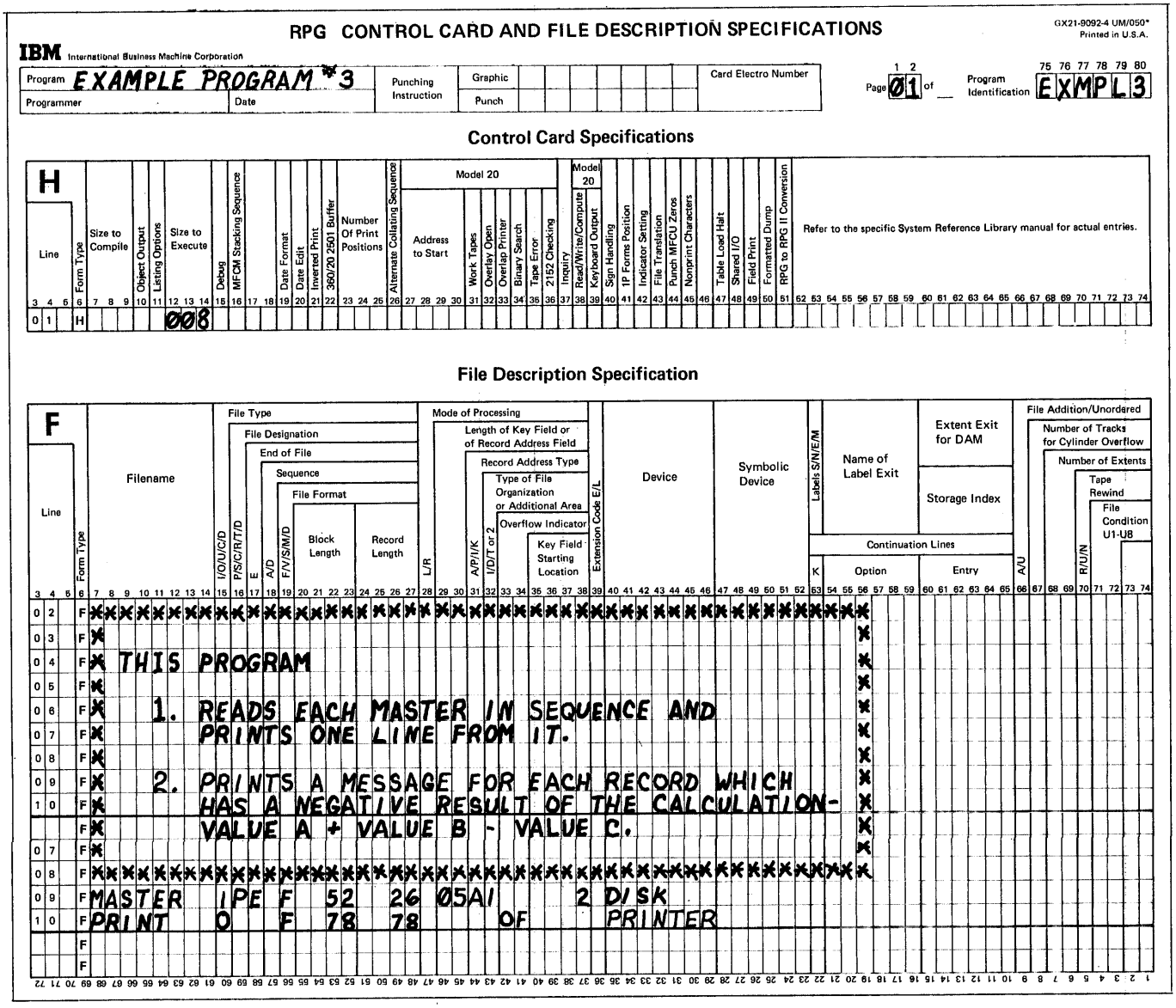


Figure 181 (Part 1 of 4). Specifications for EXMPL3

RPG INPUT SPECIFICATIONS

IBM International Business Machine Corporation

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

Program: EXAMPLE PROGRAM #3 | Programmer: | Date: | Punching Instruction: Graphic / Punch | Card Electro Number

Page (1 2): Ø2 of __ | Program Identification (75–80): EXMPL3

Line (3–5)	Form Type (6)	Filename (7–14)	Sequence (15–16)	Number (1-N) (17)	Option (O) (18)	Record Identifying Indicator or ** (19–20)	Record Identification Codes 1: Position (21–24)	Not (N) (25)	C/Z/D (26)	Character (27)	Field Location From (44–47)	Field Location To (48–51)	Decimal Positions (52)	Field Name (53–58)	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)
01	I	MASTER	AB			Ø1	1		C	M										
02	I										2	6		KEY						
03	I										7	14		DESC						
04	I										15	18	Ø	VALUEA						
05	I										19	22	Ø	VALUEB						
06	I										23	26	Ø	VALUEC						
07	I																			
08	I																			

RPG CALCULATION SPECIFICATIONS

IBM International Business Machine Corporation

Form GX21-9093-2
Printed in U.S.A.

Program: EXAMPLE PROGRAM #3 | Programmer: | Date: | Punching Instruction: Graphic / Punch | Card Electro Number

Page (1 2): Ø3 of __ | Program Identification (75–80): EXMPL3

Line (3–5)	Form Type (6)	Control Level (LO-L9, LR, SR, AN/OR) (7–8)	Indicators (9–17)	Factor 1 (18–27)	Operation (28–32)	Factor 2 (33–42)	Result Field Name (43–48)	Length (49–51)	Decimal Positions (52)	Half Adjust (H) (53)	Resulting Indicators Plus / 1>2 / High (54–55)	Minus / 1<2 / Low (56–57)	Zero / 1=2 / Equal (58–59)	Comments (60–74)
01	C		Ø1	VALUEA	ADD	VALUEB	CALC	5	Ø					
02	C		Ø1	CALC	SUB	VALUEC	CALC					22		
03	C		Ø1	TOTA	ADD	VALUEA	TOTA	5	Ø					
04	C		Ø1	TOTB	ADD	VALUEB	TOTB	5	Ø					
05	C		Ø1	TOTC	ADD	VALUEC	TOTC	5	Ø					
06	C		Ø1	TOTCAL	ADD	CALC	TOTCAL	5	Ø					
07	C													
08	C													

Figure 181 (Part 2 of 4). Specifications for EXMPL3

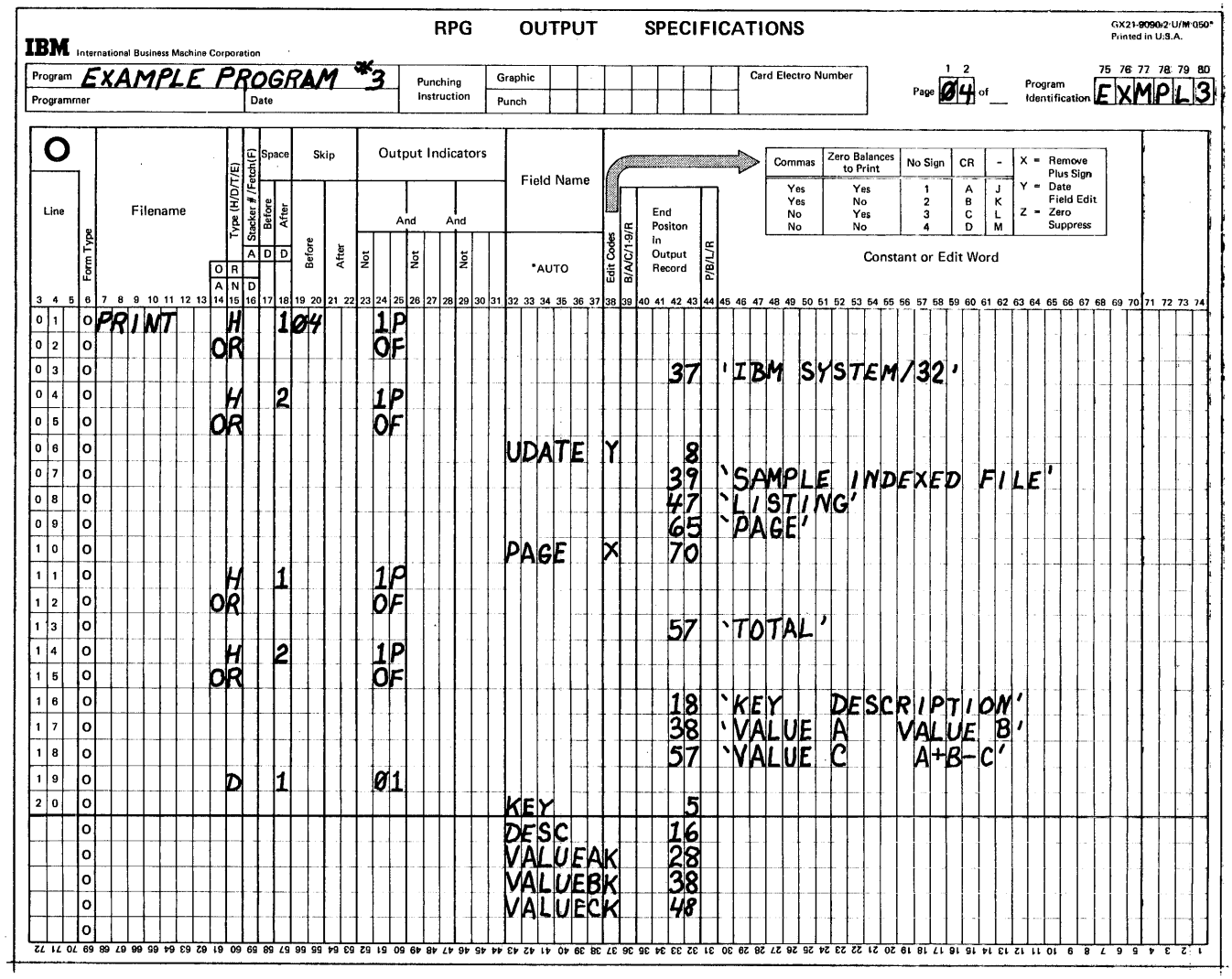


Figure 181 (Part 3 of 4). Specifications for EXMPL3

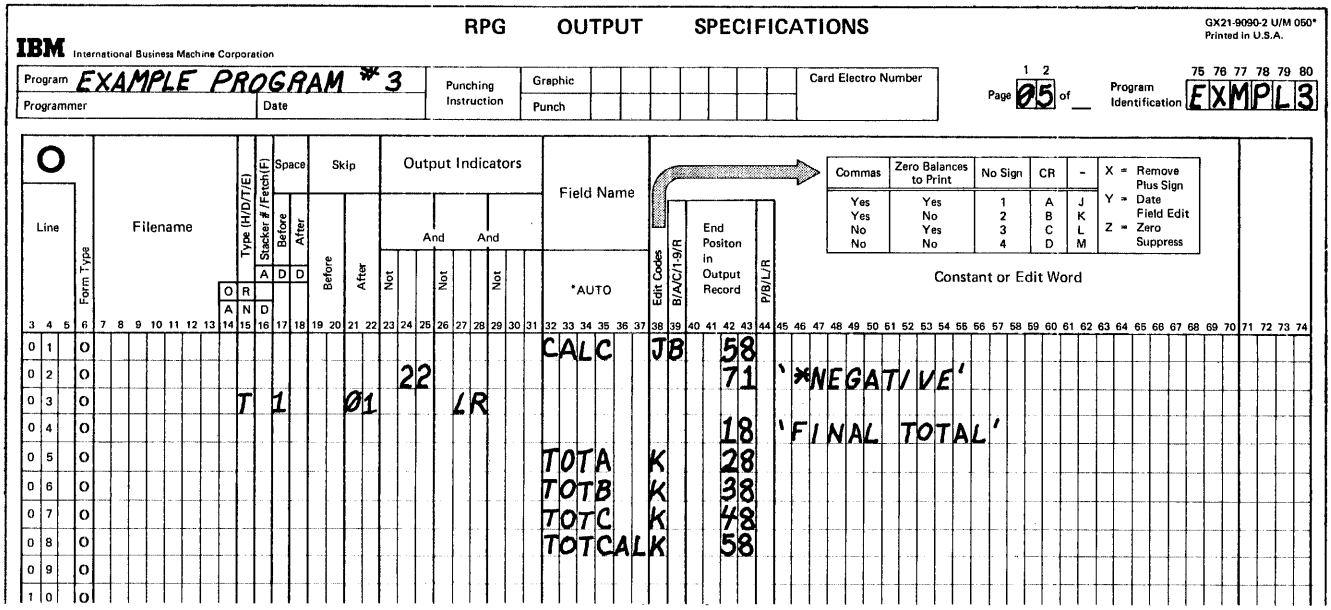


Figure 181 (Part 4 of 4). Specifications for EXMPL3

STORAGE SAVING TECHNIQUES

When your program is too large to fit into the execution storage size, you may want to use some storage saving techniques to help reduce the program size. Before you can use these techniques effectively, however, you need to understand: (1) how the RPG II Compiler creates overlays to make a program fit into the storage area available for execution and (2) how the compiler determines when a program is too large to fit into the storage available for execution. This section discusses the overlay process and then gives you some suggestions for saving storage.

Overlay Process

When your program exceeds the avallable storage for program execution, the RPG II Compiler places some RPG II object program routines on disk. These routines are then called into main storage as they are needed by your program. This is known as the overlay process.

When the overlay process is used, main storage is divided into two main parts: the *root segment* and the *overlay area*.

The root segment contains constants and data used more than once during the program execution. For this reason, the root segment always remains in main storage. The root segment can be used by routines in the overlay area. The root segment can call a routine in the overlay area by using a branch instruction.

The main overlay area contains the major routines of the RPG II object program. Routines in this area can be called by the root segment or by other routines in the same main overlay.

Some large programs require that storage be divided into two additional parts: the secondary root segment and the suboverlay area. The secondary root segment supplements the root segment. If the root segment and the overlay area fill main storage, the second root segment is not created. The suboverlay area, created by the RPG II Compiler, contains subroutines and other RPG II code needed to support a routine in the main overlay area. Figure 182 shows the location of the main storage areas.

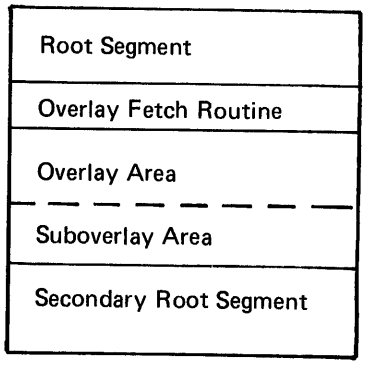


Figure 182. RPG II Storage Map

Creating the Overlays

To create overlays, the compiler must first determine which routines go into the main overlay areas and which routines go into the suboverlay areas. Then it calculates the size of the largest main overlay and the size of the largest suboverlay. These sizes are rounded off upwards in increments of 256 bytes (1 sector). The compiler then adds the lengths of the root segment, the largest main overlay, and the largest suboverlay. If the sum is larger than the available storage, your program is too large, and storage saving techniques must be used if the program is to be run.

Special Open/Close

Special open/close is used when the overlay requirements for open and close exceed the overlay requirements for the rest of the program. Special open/close can be easily identified because overlay #$$002 is the first overlay identified in the main storage usage map (Figure 183).

The initial load of the program brings in the code specified in the main storage usage map between and including the root segment and the overlay fetch area. Open is completely self-contained and does not need any of the nonoverlay code. When open is complete, overlay code is loaded.

MAIN STORAGE USAGE OF RPG II CODE

START ADDR	NAME IF OVERLAY	CODE LENGTH	NAME	TITLE
0800		1F81	RGROOT	ROOT
2826		00EF	RGSUBS	OVERLAY FETCH ROUTINE
2781		0075	#$BDMC	DATA MANAGEMENT INTERFACE
2915		0A00	RGSUBS	OVERLAY FETCH AREA
316E		00A0	RGMAIN	INPUT MAINLINE
3338		0008	RGSUBS	TRANSFER VECTOR
320E		0107	RGSUBS	RECORD IDENTIFICATION
3315		0026	RGSUBS	CONTROL FIELDS
3115		0059	RGSUBS	INPUT CONTROL ROUTINE
2915	$##002	0008	RGSUBS	INPUT HOOK
291D	$##002	017C	#$ISAD	INDEX SEQUENTIAL ADD
2A99	$##002	0098	#$SRBI	SYSTEM SUBROUTINE
2B31	$##002	0058	#$SRCL	SYSTEM SUBROUTINE
2B8C	$##002	001C	#$SRDF	SYSTEM SUBROUTINE
2BA8	$##002	0038	#$SRDI	SYSTEM SUBROUTINE
2BE0	$##002	006D	#$SRIC	SYSTEM SUBROUTINE
2C4D	$##002	0033	#$SRMI	SYSTEM SUBROUTINE
2C80	$##002	0081	#$SRMO	SYSTEM SUBROUTINE
2D01	$##002	006F	#$SRRC	SYSTEM SUBROUTINE
2D70	$##002	0029	#$SRRI	SYSTEM SUBROUTINE
2D99	$##002	001C	#$SRTC	SYSTEM SUBROUTINE
2DB5	$##002	0026	#$SRUA	SYSTEM SUBROUTINE
2DD8	$##002	0017	#$SRBP	SYSTEM SUBROUTINE
3346		007E	RGMAIN	INPUT FIELDS
37EF		03BC	RGMAIN	DETAIL CALCULATIONS
3BEE		0042	RGSUBS	TRANSFER VECTOR
3484		0109	RGSUBS	CONSTANTS
365D		0192	RGSUBS	CONSTANTS
33C4		00C0	RGSUBS	OUTPUT CONTROL ROUTINE
3BAB		0043	#$PGRI	RESET RESULTING INDICATOR
2921	$##003	0008	RGSUBS	INPUT HOOK
2B59	$##003	006A	#$PGMC	MULTIPLY
2935	$##003	0224	RGSUBS	EXCEPTION
2915	$##003	000C	RGSUBS	OUTPUT HOOK
2929	$##003	000C	RGSUBS	OUTPUT HOOK
2C30	$##003	017C	#$ISAD	INDEX SEQUENTIAL ADD
2DAC	$##003	0098	#$SRBI	SYSTEM SUBROUTINE
2E44	$##003	0058	#$SRCL	SYSTEM SUBROUTINE
2E9F	$##003	001C	#$SRDF	SYSTEM SUBROUTINE
2EB8	$##003	0038	#$SRDI	SYSTEM SUBROUTINE
2EF3	$##003	006D	#$SRIC	SYSTEM SUBROUTINE
2F60	$##003	0033	#$SRMI	SYSTEM SUBROUTINE
2F93	$##003	0081	#$SRMO	SYSTEM SUBROUTINE
3014	$##003	006F	#$SRRC	SYSTEM SUBROUTINE
3083	$##003	0029	#$SRRI	SYSTEM SUBROUTINE
30AC	$##003	001C	#$SRTC	SYSTEM SUBROUTINE
30C8	$##003	0026	#$SRUA	SYSTEM SUBROUTINE
30EE	$##003	0017	#$SRBP	SYSTEM SUBROUTINE
28C3	$##003	006D	#$PGIC	DIVIDE
2915	$##004	007F	#$PGCB	COMMAND KEY INDICATOR SET
2994	$##004	0174	#$PGAC	SQRT SQUARE ROOT
3C3C		0025	RGMAIN	DETAIL OUTPUT
3C61		0008	RGSUBS	TRANSFER VECTOR
3C30		000C	RGSUBS	CONSTANTS
2915	$##005	000C	RGSUBS	OUTPUT HOOK
2921	$##005	017C	#$ISAD	INDEX SEQUENTIAL ADD
2A9D	$##005	0098	#$SRBI	SYSTEM SUBROUTINE
2B35	$##005	0058	#$SRCL	SYSTEM SUBROUTINE
2B90	$##005	001C	#$SRDF	SYSTEM SUBROUTINE
2BAC	$##005	0038	#$SRDI	SYSTEM SUBROUTINE
2BE4	$##005	006D	#$SRIC	SYSTEM SUBROUTINE
2C51	$##005	0033	#$SRMI	SYSTEM SUBROUTINE
2C84	$##005	0081	#$SRMO	SYSTEM SUBROUTINE

Figure 183 (Part 1 of 2). Main Storage Usage Map

```
2D05   $##005   006F    #$SRRC SYSTEM SUBROUTINE
2D74   $##005   0029    #$SRRI SYSTEM SUBROUTINE
2D9D   $##005   001C    #$SRTC SYSTEM SUBROUTINE
2D89   $##005   0026    #$SRUA SYSTEM SUBROUTINE
2DDF   $##005   0017    #$SRBP SYSTEM SUBROUTINE
3C6C           000B    RGMAIN TOTAL OUTPUT
3C77           0024    RGMAIN LR & OVERFLOW PROCESSING
3C9B           004D    RGSUBS TRANSFER VECTOR
2921   $##006  0060    RGSUBS OVERFLOW SUBSEGMENT
29D4   $##006  001C    RGSUBS SUBSEG
2983   $##006  0021    RGSUBS SUBSEG
2915   $##006  000C    RGSUBS OUTPUT HOOK
29F0   $##006  0012    RGSUBS SUBSEG
2A02   $##006  001C    RGSUBS SUBSEG
2981   $##006  0018    RGSUBS SUBSEG
299C   $##006  0017    RGSUBS SUBSEG
2878   $##007  0121    RGMAIN OPEN MAINLINE
2D36   $##007  0021    RGSUBS TRANSFER VECTOR
2915   $##007  00C0    RGSUBS OUTPUT CONTROL ROUTINE
29E6   $##007  0192    RGSUBS CONSTANTS
29E1   $##007  0005    RGSUBS CONSTANTS
29D5   $##007  000C    RGSUBS OUTPUT HOOK
2CEC   $##007  001C    RGSUBS SUBSEG
2D08   $##007  0012    RGSUBS SUBSEG
2D1A   $##007  001C    RGSUBS SUBSEG
2CB4   $##007  0017    RGSUBS SUBSEG
2C99   $##007  0018    RGSUBS SUBSEG
2CCB   $##007  0021    RGSUBS SUBSEG
2B7E   $##008  002A    RGMAIN CLOSE MAINLINE
30A0   $##008  0016    RGSUBS TRANSFER VECTOR
29EC   $##008  0192    RGSUBS CONSTANTS
2915   $##008  00C0    RGSUBS OUTPUT CONTROL ROUTINE
29E1   $##008  0008    RGSUBS CONSTANTS
2BA8   $##008  0023    RGSUBS LR OUTPUT
29D5   $##008  000C    RGSUBS OUTPUT HOOK
2BCB   $##008  017C    #$ISAD INDEX SEQUENTIAL ADD
2D47   $##008  0098    #$SR3I SYSTEM SUBROUTINE
2DDF   $##008  005B    #$SRCL SYSTEM SUBROUTINE
2E3A   $##008  001C    #$SRDF SYSTEM SUBROUTINE
2E56   $##008  0038    #$SRDI SYSTEM SUBROUTINE
2E8E   $##008  006D    #$SRIC SYSTEM SUBROUTINE
2EF8   $##008  0033    #$SRMI SYSTEM SUBROUTINE
2F2E   $##008  0031    #$SRMO SYSTEM SUBROUTINE
2FAF   $##008  006F    #$SRRC SYSTEM SUBROUTINE
301E   $##008  0029    #$SRRI SYSTEM SUBROUTINE
3047   $##008  001C    #$SRTC SYSTEM SUBROUTINE
3063   $##008  0026    #$SRUA SYSTEM SUBROUTINE
3089   $##008  0017    #$SRBP SYSTEM SUBROUTINE

               13589    RPF113 MAIN STORAGE REQUIRED TO EXECUTE.
               14266    RPF113 MAIN STORAGE REQUIRED TO EXECUTE WITHOUT OVERLAYS.
```

OVERLAY NAME	RELATIVE START DISK ADDRESS	NUMBER OF TEXT SECTORS	STARTING CORE ADDRESS
$##001	0023	14	2915
$##002	0039	05	2915
$##003	003F	08	2915
$##004	0048	02	2915
$##005	004B	05	2915
$##006	0051	02	2915
$##007	0054	05	2915
$##008	005A	08	2915

Figure 183 (Part 2 of 2). Main Storage Usage Map

Overlay code consists of all code that is identified as nonoverlay and was not loaded during the first load. (Overlay code is also identified as overlay $##001 in the overlay map following the storage usage map. The other overlay numbers correspond to their respective overlay numbers as they appear in the storage usage map.) The program then executes as a normal overlay program until close is needed. At this time, close is brought into main storage starting at the overlay fetch area and using as much main storage as is needed. You can find the overlay fetch area size for the rest of the program by subtracting the start of the overlay fetch area from the lowest start address of the nonoverlay code that was not included in the first load; that is, input control routine starts at 3115 so 3115 minus 2915 equals 0800 — the overlay fetch area size.

Saving Storage

When the compiler finds that your program is too large, an error message is written. You can reduce the main storage needed for your program either by using some general storage saving techniques or by reducing the size of the overlays.

General Storage Saving Techniques

Some of the techniques you can use are:

- Divide your program into separate tasks, creating a separate program for each task. For example, suppose you want to update a file and print a listing of the updated file. You can save storage by updating the file with one program and printing the listing with another program.

- Eliminate unreferenced indicators. Eliminating unreferenced indicators can eliminate the instructions required to set the indicators on and off.

- Eliminate unnecessary conditioning indicators. Two possible forms of unnecessary indicator tests are:

  1. If only one type of input record is to be processed, the indicator associated with that record is always on except during the first detail output time. It is, therefore, not necessary for any calculation to be conditioned with this indicator.

  2. When two subsequent operations on the same result field are conditioned on opposite indicator conditions, one of the conditions is not necessary. For instance, the N09 conditioning is not required in this example:

```
N09     Z-ADD   FLD    FLDB

 09     Z-ADD   FLDC   FLDB
```

  Note: This technique may not work for certain operations if the same field is used as the result field and as factor 1 or factor 2.

- Reuse calculation work areas and temporary hold areas. Once the data stored in these areas is used for the last time in a given cycle, the area is available. Reusing these areas can eliminate the need for two or more additional areas to be defined.

  Note: Be sure you do not mix alphameric and numeric fields.

- Reuse input field name areas. In some instances, two or more input files have fields that contain identical information to use the same main storage area; these fields can have the same field name. You can also reuse input field areas by using the same names for fields in two files. This can be done only if both fields have the same attributes (length, alphameric/numeric, packed binary) and each field is used only in the cycle in which the record is processed. Both files cannot be used in the same cycle.

- Reduce calculation result field sizes. Be sure that no result field is defined any larger than is necessary. Reducing the result field size can cause a warning that the result field may not be large enough. If you know that the largest possible number fits into the result field specified, you can continue compiling the program.

- Include the necessary intervening blanks when describing alphameric fields and constants for output. This makes the fields adjacent. The output optimization phase moves all adjacent fields and constants with one instruction instead of using one instruction to move each line.

Not Optimized	**Optimized**
5 'DAILY'	18 'DAILY TRANSACTION'
17 'TRANSACTION'	26 'REGISTER'
26 'REGISTER'	

- Design files to contain record lengths that are an even multiple of 256 bytes or that divide into 256 bytes an even number of times.

- Design files so that match fields and control fields are assigned the same position within all record types.

- Do not designate a field as numeric unless the field is used in a numeric operation in the program. This saves on the amount of storage required to store the field and allows the input and output fields transfer routine to be optimized.

- Use only one type of file organization in a program (indexed, direct, or sequential). Also, use the same method of processing where possible. This can reduce the disk data management storage requirements.

- Use the shared input/output access method (SIAM) to process disk files. This may reduce the storage required even on programs with only one disk file.

  Note: Using SIAM may decrease program throughput.

- Group calculation statements together that are conditioned by the same indicators. When a large number of indicators are required, try to use GOTO or EXSR to reduce the number of indicator tests required on each statement.

- Use the actual bit pattern in factor 2 when using TESTB, BITON, or BITOF.

- Do not use half adjust unless absolutely necessary.

- Try to use either factor 1 or factor 2 as the result field whenever possible.

- Try to use numeric fields of the same length and with the same number of decimal positions. If the fields cannot be the same length, try to have the number of decimal positions the same.

- Do not sequence check your records unless absolutely necessary.

- Use OR lines rather than multiple record lines because OR lines require less code.

- Specify the fields in a record in ascending order by record position.

- Do not use halt indicators unless absolutely necessary.

Reduce the Overlay Size

To reduce the size of the overlay, you can reduce the size of the root segment or the overlay areas. First, however, you must identify the contents of the root segment and the largest overlays in main storage. Then you can determine if the contents of these areas can be reduced to fit into the storage available for execution.

Use the program listing to find the contents of the root segment, main overlay area, and suboverlay area. The root segment contains the data and routines which are not given an overlay name in the main storage usage of the RPG II code section of the program listing (Figure 184).

Two sections of the program listing determine the contents of the main overlay and suboverlay areas. The section shown in Figure 185 tells the:

- Overlay name

- Number of sectors in the overlay

- Start address of the overlay.

The start address separates main overlays and suboverlays. Two start addresses appear in the start address column. The lower address (183F) identifies a main overlay; the higher address (1A3F) identifies a suboverlay.

The text sectors column indicates the largest overlays. In Figure 185, overlay 004 is the largest suboverlay; overlays 001 and 009 are the largest main overlays.

Relate the name given in the overlay name column shown in Figure 185 to the main storage usage of RPG II code section shown in Figure 186. The name and title columns in this section identify the routines or subroutines in the overlay.

Note: If overlay 001 does not appear in the overlay name column, a special open/close overlay construction took place. When this occurs, overlay 001 is not treated as an overlay, but remains in main storage.

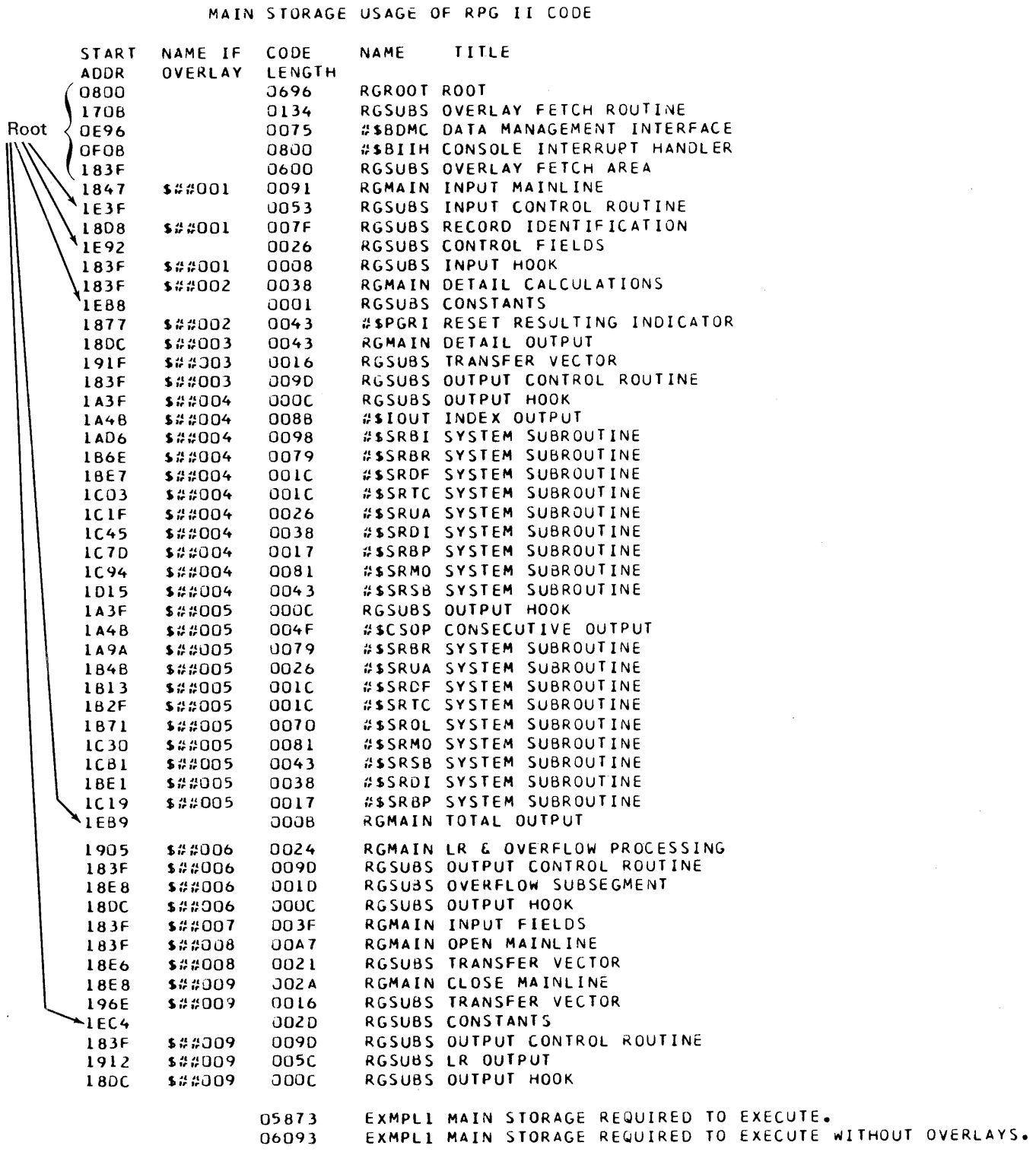


Figure 184. Overlay Usage Map

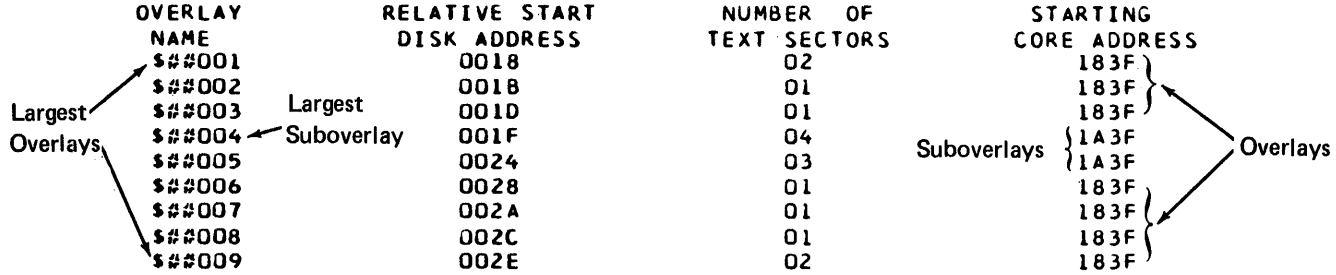


Figure 185. Overlay Identification Area

MAIN STORAGE USAGE OF RPG II CODE

	START ADDR	NAME IF OVERLAY	CODE LENGTH	NAME	TITLE
	0800		0696	RGROOT	ROOT
	1708		0134	RGSUBS	OVERLAY FETCH ROUTINE
	0E96		0075	#$BDMC	DATA MANAGEMENT INTERFACE
	0F08		0800	#$BIIH	CONSOLE INTERRUPT HANDLER
	183F		0600	RGSUBS	OVERLAY FETCH AREA
Overlay 001	1847	$##001	0091	RGMAIN	INPUT MAINLINE
	1E3F		0053	RGSUBS	INPUT CONTROL ROUTINE
	18D8	$##001	007F	RGSUBS	RECORD IDENTIFICATION
	1E92		0026	RGSUBS	CONTROL FIELDS
	183F	$##001	0008	RGSUBS	INPUT HOOK
Overlay 002	183F	$##002	0038	RGMAIN	DETAIL CALCULATIONS
	1E88		0001	RGSUBS	CONSTANTS
	1877	$##002	0043	#$PGRI	RESET RESULTING INDICATOR
Overlay 003	18DC	$##003	0043	RGMAIN	DETAIL OUTPUT
	191F	$##003	0016	RGSUBS	TRANSFER VECTOR
	183F	$##003	009D	RGSUBS	OUTPUT CONTROL ROUTINE
Overlay 004	1A3F	$##004	000C	RGSUBS	OUTPUT HOOK
	1A4B	$##004	0088	#$IOUT	INDEX OUTPUT
	1AD6	$##004	0098	#$SRBI	SYSTEM SUBROUTINE
	1B6E	$##004	0079	#$SRBR	SYSTEM SUBROUTINE
	1BE7	$##004	001C	#$SRDF	SYSTEM SUBROUTINE
	1C03	$##004	001C	#$SRTC	SYSTEM SUBROUTINE
	1C1F	$##004	0026	#$SRUA	SYSTEM SUBROUTINE
	1C45	$##004	0038	#$SRDI	SYSTEM SUBROUTINE
	1C7D	$##004	0017	#$SRBP	SYSTEM SUBROUTINE
	1C94	$##004	0081	#$SRMO	SYSTEM SUBROUTINE
	1D15	$##004	0043	#$SRSB	SYSTEM SUBROUTINE
Suboverlay 005	1A3F	$##005	000C	RGSUBS	OUTPUT HOOK
	1A4B	$##005	004F	#$CSOP	CONSECUTIVE OUTPUT
	1A9A	$##005	0079	#$SRBR	SYSTEM SUBROUTINE
	1B4B	$##005	0026	#$SRUA	SYSTEM SUBROUTINE
	1B13	$##005	001C	#$SRDF	SYSTEM SUBROUTINE
	1B2F	$##005	001C	#$SRTC	SYSTEM SUBROUTINE
	1B71	$##005	0070	#$SROL	SYSTEM SUBROUTINE
	1C30	$##005	0081	#$SRMO	SYSTEM SUBROUTINE
	1C81	$##005	0043	#$SRSB	SYSTEM SUBROUTINE
	1BE1	$##005	0038	#$SRDI	SYSTEM SUBROUTINE
	1C19	$##005	0017	#$SRBP	SYSTEM SUBROUTINE
	1EB9		000B	RGMAIN	TOTAL OUTPUT

Figure 186 (Part 1 of 2). Overlay Usage Map

```
                 1905    $##006    0024    RGMAIN LR & OVERFLOW PROCESSING
                 183F    $##006    009D    RGSUBS OUTPUT CONTROL ROUTINE
Suboverlay 006   18E8    $##006    001D    RGSUBS OVERFLOW SUBSEGMENT
                 18DC    $##006    000C    RGSUBS OUTPUT HOOK
                 183F    $##007    003F    RGMAIN INPUT FIELDS
                 183F    $##008    00A7    RGMAIN OPEN MAINLINE
                 18E6    $##008    0021    RGSUBS TRANSFER VECTOR
                 18E8    $##009    002A    RGMAIN CLOSE MAINLINE
                 196E    $##009    0016    RGSUBS TRANSFER VECTOR
                 1EC4              002D    RGSUBS CONSTANTS
                 183F    $##009    009D    RGSUBS OUTPUT CONTROL ROUTINE
                 1912    $##009    005C    RGSUBS LR OUTPUT
                 18DC    $##009    000C    RGSUBS OUTPUT HOOK

                                   05873    EXMPL1 MAIN STORAGE REQUIRED TO EXECUTE.
                                   06093    EXMPL1 MAIN STORAGE REQUIRED TO EXECUTE WITHOUT OVERLAYS.
```

Figure 186 (Part 2 of 2). Overlay Usage Map

PERFORMANCE IMPROVEMENT TECHNIQUES

Some relatively simple program changes can make significant improvements in your program's performance. However, these performance techniques do not improve performance in all programs. Therefore, study these techniques and determine if they can improve your program's performance before you use them. The performance improvement techniques are:

- Unblock all randomly processed indexed files. Blocking is not necessary since each record has its own index entry with the direct address of the record.
- Block all sequentially processed indexed files.
- Use the storage index. For a minimum cost in main storage, this allows the system to read the single track of indexes it needs rather than reading the entire index to look for an entry.
- Reduce or eliminate blocking of consecutive files and double the buffer instead. For example, instead of using a block of 1600 bytes with 80-byte records, use a block of 800 bytes and double buffer.

After identifying the root segment and the largest main overlays and suboverlays, you can determine whether they contain routines that can be manipulated to reduce the overlay size. The following routines can be controlled:

- Input records
- Detail calculations
- Total calculations
- Detail input
- Total output

Following are some storage saving techniques that can be used for these routines. These techniques may not necessarily work for all programs.

Input Records: One or more of the input or update files can be processed as a demand or chained file, using the READ or CHAIN operation code. With a demand or chained file, the instructions to read the file can be moved into the total or detail calculations routines.

Note: Total calculations are not done on the first cycle.

Detail or Total Calculations: Use the following techniques:

- Use subroutine calculations. In some instances this may increase, rather than decrease, the storage required due to the nature of the existing calculation routines. However, it may reduce the overall storage requirements.

  Note: If one subroutine calls another subroutine, both subroutines must be in storage at the same time. This may increase the size of the suboverlay area and the total storage required. To ensure the smallest requirement, do not call a subroutine from another subroutine.

- Eliminate exception output if possible. This moves the logic for those output operations to either total or detail output routines.

- Eliminate READ and/or CHAIN operations by using matching records and processing consecutively. This moves the logic to input records routine.

- Move part of the detail calculation logic to total calculations (or total calculation logic to detail calculations).

  Note: Total calculations are not done on the first cycle.

Detail or Total Output: Use the following techniques:

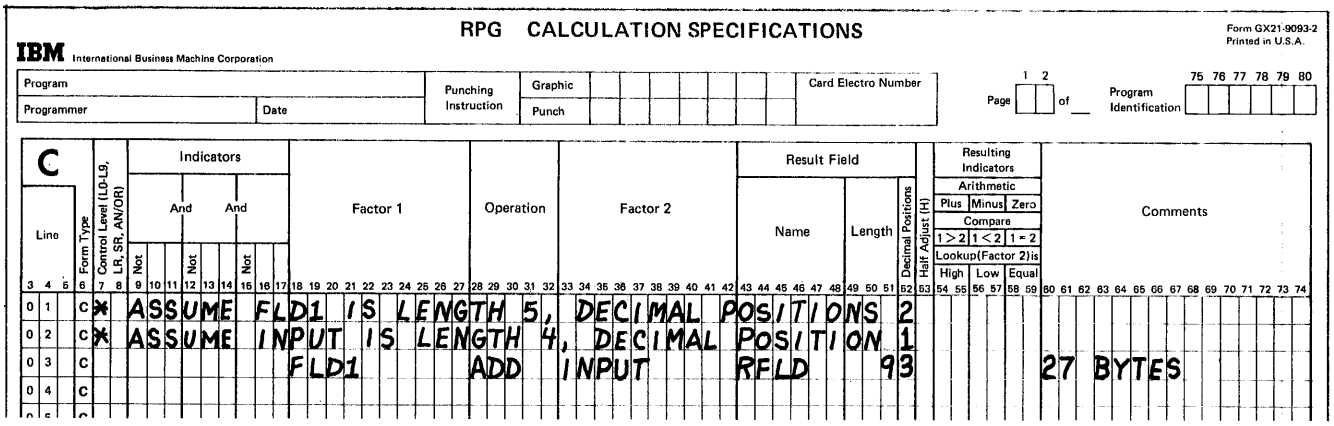
- Use exception output. This moves part of the output logic to detail or total calculation routines.

- Do some of the output at total (or detail) output time. This moves logic to the total (on detail) output routine.

- Do not specify blank after for fields; instead, clear them at the beginning of detail or total calculations.

This chapter contains the number of bytes of object code generated for RPG II operation codes. When used with Chapter 2, this information helps you determine the amount of storage that can be saved by using certain coding practices. For example, consider this storage saving technique: try to use numeric fields of the same length and with the same number of decimal positions. If the fields cannot be the same length, try to have the number of decimal positions the same.

If the decimal positions of factor 1, factor 2, and the result field are all different, an ADD operation generates 27 bytes. However, if all the fields are defined as having the same number of decimal positions, the same ADD operation generates only 15 bytes. Uniformity of fields not only saves main storage for ADD and SUB, but for most of the other arithmetic operations as well.



RPG CALCULATION SPECIFICATIONS

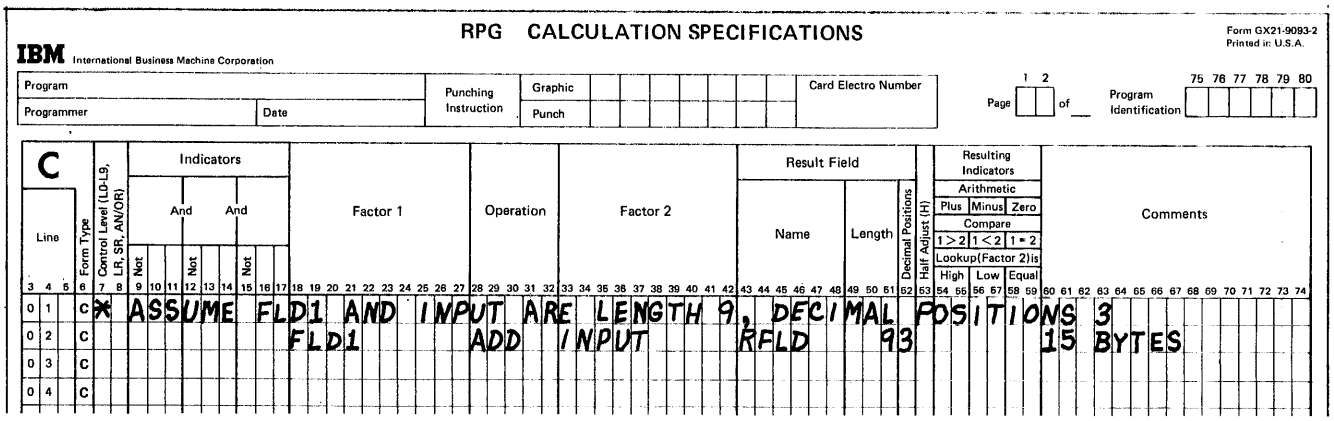
Form GX21-9093-2 Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page ___ of ___ Program Identification

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Result Field: Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus / Minus / Zero; Compare: 1>2 / 1<2 / 1=2; Lookup(Factor 2)is: High / Low / Equal	Comments
0 1	C	*		ASSUME FLD1 IS LENGTH 5, DECIMAL POSITIONS 2								
0 2	C	*		ASSUME INPUT IS LENGTH 4, DECIMAL POSITION 1								
0 3	C			FLD1	ADD	INPUT	RFLD	9	3			27 BYTES
0 4	C											



RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2 Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page ___ of ___ Program Identification

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Result Field: Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus / Minus / Zero; Compare: 1>2 / 1<2 / 1=2; Lookup(Factor 2)is: High / Low / Equal	Comments
0 1	C	*		ASSUME FLD1 AND INPUT ARE LENGTH 9, DECIMAL POSITIONS 3								
0 2	C			FLD1	ADD	INPUT	RFLD	9	3			15 BYTES
0 3	C											
0 4	C											

Abbreviations and symbols used in the following text:

F1	– Factor 1
F2	– Factor 2
RF	– Result field
L1	– Total length of factor 1
L2	– Total length of factor 2
LR	– Total length of result field
D1	– Number of decimal positions in factor 1
D2	– Number of decimal positions in factor 2
DR	– Number of decimal positions in result field
H/A	– Half adjust
RAF	– Record address file
=	– Equal
≠	– Not equal
-	– Minus
>	– Greater than
<	– Less than
+	– Plus

Operation	Bytes
SETON (each indicator set on)	3
SETOF (each indicator set off)	3
BITON	4
BITOF	4
TESTB	
Test bit off	10
Test bit mixed	17
Test bit on	10
Test bit off and mixed	23
Test bit off and on	23
Test bit mixed and on	23
Test bit off, mixed, and on	29
SUB	
F1 = RF and D1 = D2 = DR	6
F1 ≠ RF and D1 = D2 = DR	15
F1 ≠ RF and D2 = DR	23
F1 ≠ RF and D2 = DR H/A	27
All other combinations	31
All other combinations H/A	39
Z-SUB	
D2 = DR	14
D2 ≠ DR	18
D2 ≠ DR H/A	22
ADD	
F1 = RF and D1 = D2 = DR	6
F2 = RF and D1 = D2 = DR	6
F1 ≠ F2 ≠ RF and D1 = D2 = DR	15
F1 = RF and D2 > DR	14
F2 = RF and D1 > DR	14
F1 = RF and D2 > DR H/A	18
F2 = RF and D1 > DR H/A	18
F1 = RF and D2 < DR H/A	18
F2 = RF and D1 < DR H/A	18
D1 = D2 < DR	23
All other combinations	27
All other combinations H/A	35
Z-ADD	
D2 = DR	6
D2 > DR	14
D2 > DR H/A	18
D2 < DR	18
COMP	
F1 and F2 are numeric and D1 = D2	10
F1 and F2 are numeric and D1 ≠ D2	18
F1 and F2 are alphameric and L1 = L2	6
F1 and F2 are alphameric and F1 is a field	22
F1 and F2 are alphameric and F1 is a table	26
Alternate collating sequence (add these bytes to the appropriate COMP listed previously)	10

Operation	Bytes
TESTZ	
RF is a field	9
RF is a table	20
MULT	23
with H/A	27
DIV	
D1 - D2 = DR	23
D1 - D2 ≠ DR	27
D1 - D2 = DR + 1 H/A	31
D1 - D2 ≠ DR + 1 H/A	35
MVR	
D2 = DR	5
D2 ≠ DR	9
XFOOT	
D2 = DR	9
D2 ≠ DR	13
FORCE	
with external indicator	13 + 7 = 20
Conditioning indicators (does not apply to CHAIN, FORCE, LOKUP, and READ)	
each indicator	3
each AND type	3
Resulting indicators (does not apply to CHAIN, FORCE, LOKUP, and READ)	5
with each resulting indicator	3
CHAIN (base = 16)	
with external indicator	6
when F1 has a variable index	11
when key is not packed	14
when key is packed	23
when key is packed and F1 is a table element	6
when key is a record number	8
when key is a record number and F1 is a table element	6
when record-not-found indicator is given	1
when record-not-found indicator not given	16
READ (base = 29)	
with external indicator	6
with EOF indicator with BSCA	6
with EOF indicator without BSCA	12
with BSCA without EOF indicator	6
without BSCA without EOF indicator	19
with RAF limits	6
LOKUP (base = 15)	
when F1 is a table	6
when F1 is a variable	11
with each resulting indicator	12

Operation	Bytes
KEYnn (base = 37)	
when RF is a variably indexed array	6
when RF is numeric, and	10
a table element	6
with each resulting indicator	14
when RF is alphameric, and	0
with resulting indicator and field length > 1	23
length = 1	7
when F1 is displayed, and	12
F1 is an array with variable index	11
F1 is a table	6
SETnn	
with ERASE function	4
with message indicators and/or command keys	37
when F1 is displayed, and	12
F1 is an array with variable index	11
F1 is a table	6
SETnn/KEYnn combination (base = 37)	
See KEYnn operation for code in addition to base. If F1 code appears on both SET and KEY instructions, both counts should be included.	
SETLL (base = 18)	
when key is packed	12
EXSR	4
GOTO	4
MOVEA	14
MOVE, MOVEL, MHHZO, MHLZO, MLHZO, MLLZO	The number of bytes specified includes all array control code lengths. See table on page 3-38.

Note: If F1 is displayed in a KEYnn, SETnn, or SETnn/KEYnn combination operation, a 153-byte subroutine is used to do the display.

	MOVE Alphameric/Numeric	MOVEL LR < L2 Numeric	MOVEL LR > L2 Numeric	MOVEL LR = L2 Alphameric/Numeric	MOVEL LR < L2 Alphameric	MOVEL LR > L2 Alphameric	MLHZO	MHLZO	MHHZO	MLLZO
Field to Field	6	26	10	6	6	6	20	20	20	20
Array to Array	42	55	45	42	42	42	42	42	42	42
Field to Array	29	43	32	29	29	29	29	29	29	29
Table to Array	35	53	38	35	40	35	35	41	40	35
Array, Variable Index to Array	40	66	43	40	52	40	40	52	52	40
Array to Array, Variable Index	28	57	38	28	35	35	35	35	47	42
Field to Array, Variable Index	17	34	27	17	17	24	24	31	24	31
Table to Array, Variable Index	20	52	33	20	24	30	30	24	36	20
Array, Variable Index to Table	20	46	27	20	30	24	24	24	36	20
Field to Table	9	23	16	9	9	13	13	9	13	9
Table to Table	15	41	22	15	19	19	19	19	25	15
Array, Variable Index to Field	17	40	21	17	24	17	31	24	36	31
Table to Field	9	29	13	9	13	9	9	13	13	9

Array control code (initialization and processing) is generated for all calculations except LOKUP, CHAIN, READ, and FORCE.

Operation	Bytes
Array initialization	
F1 or F2 is an array	6
F1 or F2 is a table	4
F1 or F2 is an array with variable index	11
Array processing	
F1, F2, RF are arrays	28
F1 and RF, F2 and RF arrays	22
RF arrays	16

Suppose for example, that a SUB operation code was specified and has the following conditions:

F1 = RF
D1 = D2 = DR
F1 and RF = full array
F2 = table

The length of object code generated is as follows:

Array initialization	
F1 is an array	6 bytes
F2 is a table	4 bytes
RF is an array	6 bytes
SUB	6 bytes
Array processing	
F1 and RF are arrays	22 bytes

Thus, the total bytes of code generated for a SUB operation code is 44 bytes.

IN-LINE INQUIRY SUBROUTINE (SUBR95)

You can use SUBR95 to perform an inquiry type function without rolling your program out and rolling another program in (see *Control Specifications, Inquiry* for a discussion of rollout/rollin). You can check at any point in your calculations to determine if an inquiry request was made.

To use SUBR95, you must call it by specifying the linkage shown in Figure 187 whenever you want to check for an inquiry request. The indicator specified in columns 45-46 can be an RPG II indicator. For a detailed discussion of this linkage, see *Operation Codes, Branching to External Subroutines* under *RPG II Programmer's Guide*.

When SUBR95 is called, it checks if an inquiry request was made. If an inquiry request was made, the indicator specified in the RLABL operation is turned on and the inquiry request is reset. You can use this indicator to condition a GOTO or EXIT to another subroutine within your program. This subroutine can then perform whatever function you want.

You should not use SUBR95 in a B-type program (one that performs normal rollout/rollin runctions) as this can cause loss of interrupt requests.

RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2 Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page ___ of ___ Program Identification

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Result Field: Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus / Minus / Zero	Resulting Indicators: Compare: 1>2 / 1<2 / 1=2	Resulting Indicators: Lookup (Factor 2) is: High / Low / Equal	Comments
01	C				EXIT	SUBR95								
02	C				RLABL		INXX							
03	C													
04	C													
05	C													
06	C													
07	C													
08	C													

Figure 187. Linkage for SUBR95

RPG II HALT PROCEDURES

Error conditions found in your RPG II program result in a halt during execution or compilation of the program. Options available to the operator following each halt are also given. The options are:

0-Continue: Control is returned to the program, and processing continues.

1-Bypass: The remainder of the program cycle is bypassed, and the next record is read.

2-Controlled Cancel: End-of-job operations specified by your program are done, tables are dumped, and file labels are cataloged.

3-Immediate Cancel: The job is canceled without returning control to the RPG II program.

To select an option, the operator keys in the option (0, 1, 2, 3) and presses ENTER key.

A complete discussion of the halts, compilation errors, and the necessary operator procedures appears in the *IBM System/32 Messages Guide—RPG II,* SC21-7617.

OPERATION CONTROL LANGUAGE FOR RPG II

To compile an RPG II source program, the RPG II Compiler program must be loaded into main storage. Do this by evoking an IBM-supplied procedure named RPG (located in the library).

The command statement that executes the library procedure is:

RPG sourcename [,mm] [,nn]

where sourcename is the name of the source program to be compiled.

mm is the number of blocks (2560-byte segments) for the $SOURCE file. If this is not specified, the default is 20 blocks.

nn is the number of blocks for the $WORK file. If this is not specified, the default is 20 blocks.

The OCL statements included in the library procedure named RPG are shown in Figure 188.

Library procedures can be modified. OCL statements necessary to modify a library procedure are described in the *IBM System/32 System Control Programming Reference Manual,* GC21-7593.

```
*  THIS PROCEDURE EXECUTES THE RPG II COMPILER.
// * ' RPG PROCEDURE EXECUTING'
// LOAD #RPG
// FILE NAME-$SOURCE,RETAIN-S,
// IFF ?2?/ BLOCKS-?2?
// ELSE BLOCKS-20
// FILE NAME-$WORK,RETAIN-S,
// IFF ?3?/ BLOCKS-?3?
// ELSE BLOCKS-20
// MEMBER USER1-RPGCMPL1
// MEMBER PROGRAM1-RPGCMPL1
// MEMBER PROGRAM2-RPGCMPL2
// COMPILE SOURCE-?1R'1016'?
// RUN
```

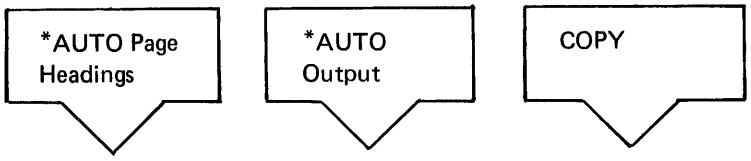
Figure 188. IBM-Supplied Library Procedure for Compiling an RPG II Source Program

Part 4

RPG II Auto Report Function

WHAT IS THE AUTO REPORT FUNCTION?

The RPG II auto report function is a program that operates prior to the RPG II Compiler. Auto report accepts special, simplified specifications and standard RPG II source specifications and uses them to generate a complete RPG II source program. The special auto report statements control the three separate functions of auto report:



Provides a simplified method of coding page headings.

Provides a simplified method of coding output specifications.

Provides a method of copying cataloged specifications from a source library to include them in an RPG II source program.

PURPOSE OF THE AUTO REPORT FUNCTION

The RPG II auto report function has two primary purposes:

- Enables beginning RPG II users to easily code a program to produce a simple report.
- Helps experienced RPG II programmers code programs faster and provides them with additional features not available in RPG II.

The auto report function reduces the time required to plan and code RPG II programs by freeing the programmer from many tasks, such as repetitive coding of specifications in different programs, planning the format of reports, and coding specifications to accumulate and print totals for numeric fields. By simplifying programming tasks so that the programmer makes fewer errors and by providing a set of diagnostics in addition to the RPG II diagnostics, auto report can reduce debugging time.

The RPG II auto report function includes three separate functions that can be used in any combination.

*AUTO Page Headings

Auto report simplifies the specification of page headings. The programmer does not have to specify conditioning indicators, spacing, and end positions. Auto report automatically centers the title and prints it at the top of each page. A date and page number are also printed unless the programmer specifies otherwise in the auto report option specifications.

*AUTO Output

Auto report simplifies the specifications for a report that includes columns of data with column headings and totals. On one output specifications line, the programmer can name a field, specify a column heading to appear above the field, and specify that several levels of totals be accumulated for the field. The programmer does not have to code separate RPG II output specifications to print the column headings, detail lines, or total lines, or calculation specifications to accumulate the totals. Auto report assumes edit codes if the programmer does not provide them and determines spacing and end positions to produce a report with a neatly prepared format.

Copy

Any RPG II specifications, including tables and arrays, and specifications for the *AUTO page headings and *AUTO output functions can be cataloged in the library and then be copied by auto report and included in the RPG II program generated by auto report. Use the copy function to include frequently used specifications that are common to different programs. You can modify copied file description and input specifications to suit each particular program.

HOW AUTO REPORT WORKS

Auto report generates a complete RPG II source program that is ready to be compiled from the following kinds of input:

- Auto report option specifications (Figure 189).
- *AUTO page headings and *AUTO output specifications in the source programs.
- Standard RPG II specifications in the source program.
- Auto report copy statements in the source program, with or without modifier statements.
- Standard RPG II specifications, including tables and arrays, and *AUTO specifications copied from the library by the auto report copy function.

If you use the auto report copy function, auto report copies the designated specifications from the library. Then, if you provide specifications for the *AUTO page headings and *AUTO output functions, auto report generates normal RPG II specifications (Figure 190) for the final RPG II source program. When a complete RPG II source program is generated and no terminal auto report messages have been issued, auto report calls the RPG II Compiler. Figure 191 shows the general method of operation of the auto report function.

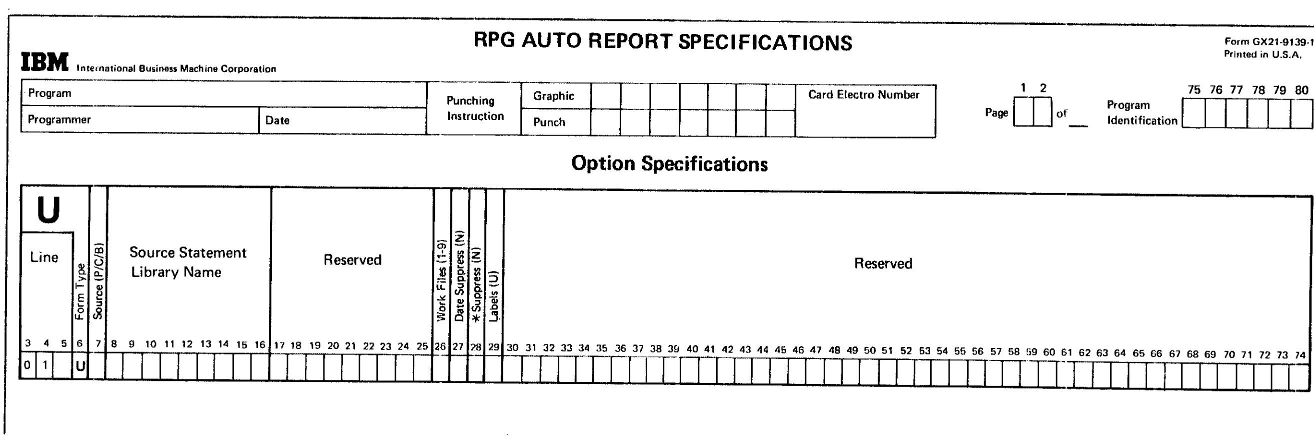


Figure 189. Auto Report Option Specifications

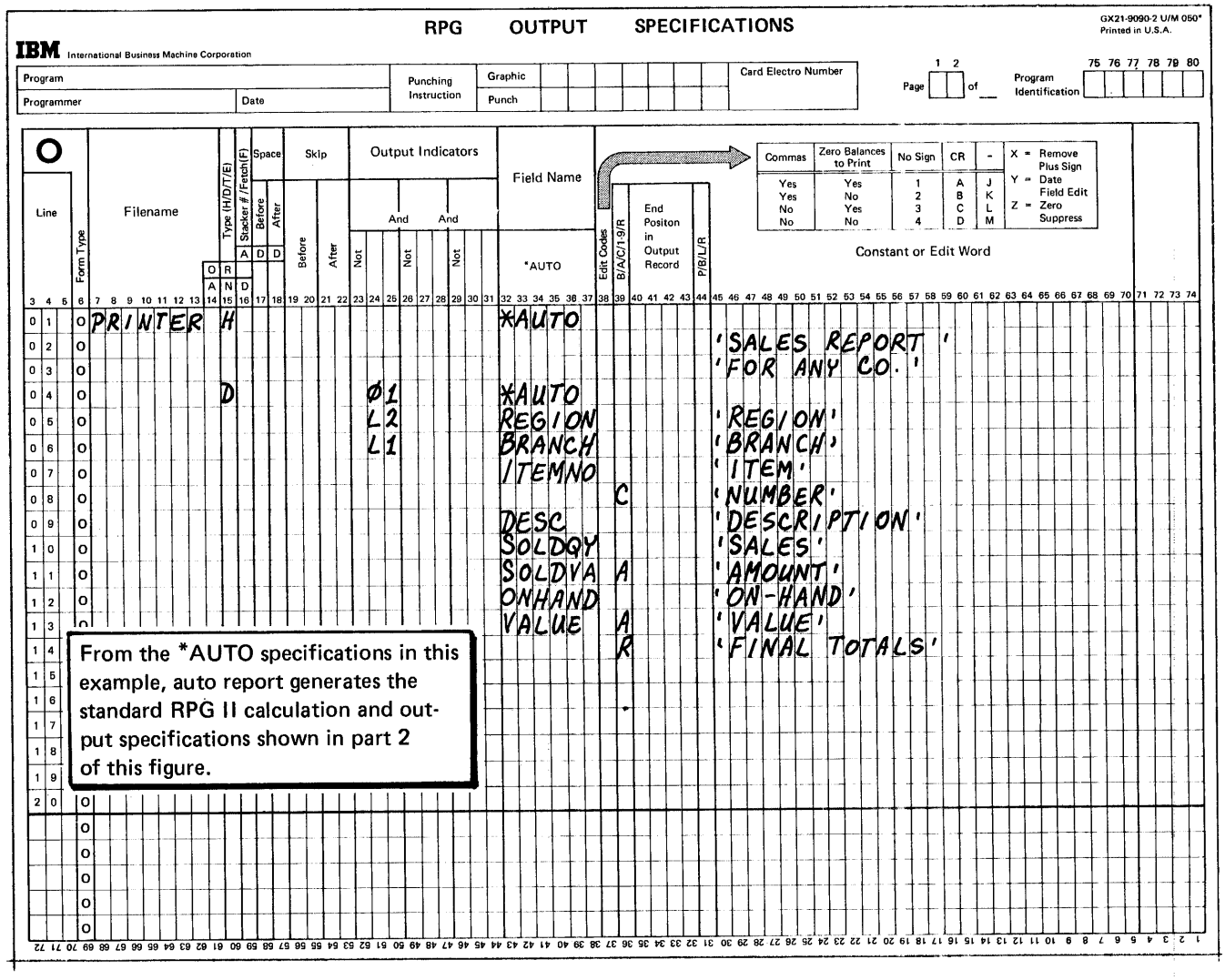


Figure 190 (Part 1 of 2). Using *AUTO Specifications, Auto Report Generates Standard RPG II Specifications

Generated Specifications

```
0012  0140EC   01                    EXSR A$$SUM
0013  0150EC L1          SOLDV2      ADD  SOLDV1    SOLDV2  92
0014  0160EC L1          VALUE2      ADD  VALUE1    VALUE2  92
0015  0170EC L2          SOLDVR      ADD  SOLDV2    SOLDVR  92
0016  0180EC L2          VALUER      ADD  VALUE2    VALUER  92
0017  0190EC SR          A$$SUM      BEGSR
0018  0200EC SR          SOLDV1      ADD  SOLDVA    SOLDV1  92
0019  0210EC SR          VALUE1      ADD  VALUE     VALUE1  92
0020  0220EC SR                      ENDSR

0021  0230EUPRINTER H  206    1P
0022  0240EU       OR        OA
0023  0250EU                                    45 'SALES REPORT '
0024  0260EU                                    56 'FOR ANY CO.'
0025  0270EU                          UDATE Y    8
0026  0280EU                          PAGE  Z   89
0027  0290EU                                    85 'PAGE '
0028  0300EUPRINTER H  1      1P
0029  0310EU       OR        OA
0030  0320EU                                     6 'REGION'
0031  0330EU                                    14 'BRANCH'
0032  0340EU                                    21 'ITEM'
0033  0350EU                                    36 'DESCRIPTION'
0034  0360EU                                    47 'SALES'
0035  0370EU                                    62 'AMOUNT'
0036  0380EU                                    71 'ON-HAND'
0037  0390EU                                    86 'VALUE'
0038  0400EUPRINTER H  2      1P
0039  0410EU       OR        OA
0040  0420EU                                    22 'NUMBER'
0041  0430EUPRINTER D  1      01
0042  0440EU                 L2       REGION     3

0043  0450EU                 L1       BRANCH     12
0044  0460EU                          ITEMNO     23
0045  0470EU                          DESC       40
0046  0480EU                          SOLDQYK    46
0047  0490EU                          SOLDVAKB   62
0048  0500EU                          ONHANDK    69
0049  0510EU                          VALUE KB   86
0050  0520EUPRINTER T 12     L1
0051  0530EU                          SOLDV1KB   62
0052  0540EU                          VALUE1KB   86
0053  0550EU                                    87 '*'
0054  0560EUPRINTER T  2     L2
0055  0570EU                          SOLDV2KB   62
0056  0580EU                          VALUE2KB   86
0057  0590EU                                    88 '**'
0058  0600EUPRINTER T 12     LR
0059  0610EU                          SOLDVRKB   62
0060  0620EU                          VALUERKB   86
0061  0630EU                                    47 'FINAL TOTALS'
0062  0640EU                                    89 '***'
```

Calculations to roll totals for SOLDVA and VALUE fields

Page heading (includes date and page number)

Column headings

Detail output specifications

Total output specifications

Figure 190 (Part 2 of 2). Using *AUTO Specifications, Auto Report Generates Standard RPG II Specifications

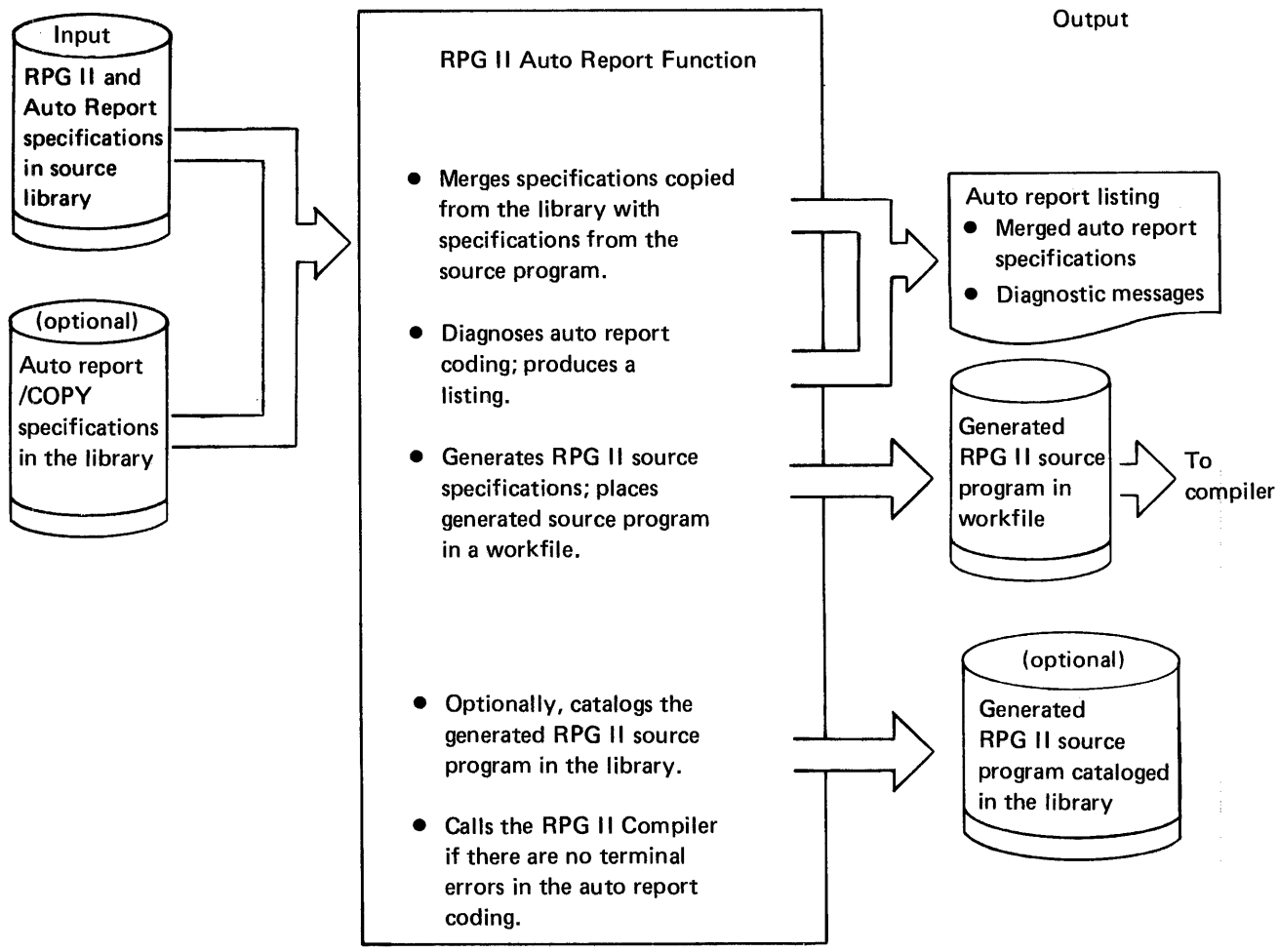


Figure 191. Operations of the Auto Report Function

*AUTO PAGE HEADINGS AND *AUTO OUTPUT

Examples 1 through 4 explain how auto report is used in generating report page headings and such output specifications as: column headings, detail lines, and total lines.

EXAMPLE 1

*AUTO Page Headings | *AUTO Output

Problem

Produce the sales report shown below using the *AUTO page headings and *AUTO output functions of auto report.

Procedure

1. Code normal RPG II file description and input specifications for the job.
2. Code *AUTO page headings to produce a one-line page heading that includes date and page number.
3. Code *AUTO output to produce one-line column headings, detail report lines, and final totals.

Letters refer to fields on the opposite page.

10/26/71 SALES REPORT FOR ANY CO. PAGE 1

C REGION	B BRANCH	A ITEM	D DESCRIPTION	E SALES	F AMOUNT	G ON-HAND	H VALUE
1	17	AG7701T	2-TON TRUCK	5	25,000.00	2	10,000.00
1	17	AG7705S	PICK-UP	10	20,000.00	1	2,000.00
1	17	AP6545B	CAMPER	2	8,000.00		
1	22	AG7701T	2-TON TRUCK	2	10,000.00	1	5,000.00
1	22	AG7705S	PICK-UP	4	8,000.00	1	2,000.00
3	25	AG6545B	CAMPER	10	40,000.00	5	20,000.00
3	25	AP6549P	1/4 TON TRUCK	20	30,000.00	6	9,000.00
					141,000.00		48,000.00 *

1 Code RPG II file description and input specifications.

File Description Specification

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/O/T or 2	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	File Addition/Unordered	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	File Condition U1-U8	R/U/N
02	F	SALES	I	P			F	473	473								DISK										
03	F	PRINTER	O				F	120	120								PRINTER										
04	F																										

Continuation Lines: K | Option | Entry | A/U

RPG INPUT SPECIFICATIONS

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

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic	Card Electro Number
Programmer	Date		Punch	

Page ___ of ___ Program Identification

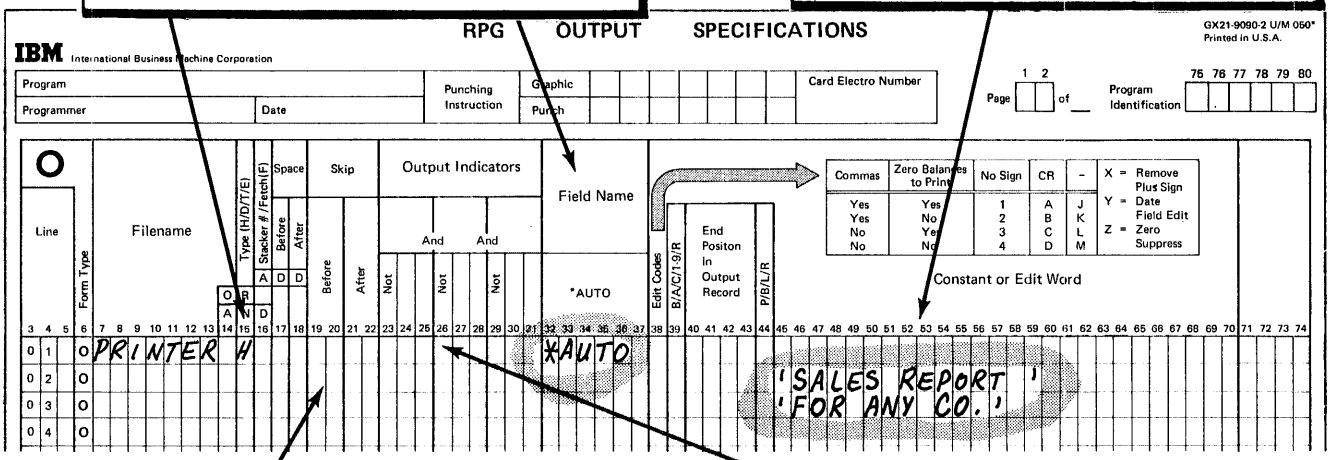
Line	Form Type	Filename	Sequence (OR AND)	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location 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
01	I	SALES	AA			01																								
02	I																				1	7		ITEMNO						
03	I																				8	9		BRANCH						
04	I																				10	10		REGION						
05	I																				11	25		DESC						
06	I																				26	27	0	SOLDQY						
07	I																				28	34	2	SOLDVA						
08	I																				35	36	0	ONHAND						
09	I																				37	43	2	VALUE						
10	I																													

	Field Name	Contents
A	ITEMNO	Item number.
B	BRANCH	Number of the branch office where the item was sold.
C	REGION	Sales region in which the branch office is located.
D	DESC	Description of the sales item.
E	SOLDQY	Quantity of the item sold.
F	SOLDVA	Total value of the items sold.
G	ONHAND	Quantity of the item remaining on hand.
H	VALUE	Total value of the items remaining on hand.

2 Code *AUTO page heading specifications.

A Enter an H in position 15 and *AUTO in 32-36 to request an auto report page heading. Up to five page heading lines can be described. The system date is printed on the left and the page number on the right of the first heading line on each page. The date and page can be suppressed by placing an N in position 27 of the auto report option specifications sheet.

B The title information is centered by auto report; do not enter end positions in 40-43. Fields and table/array elements can also be used.



C When space and skip entries (positions 17-22) are left blank, skip to line 06 is assumed for the first heading line; single spacing is done between heading lines; double spacing after the last heading line. (See *Example 4* for an example of multiple page heading lines.)

D When output indicators (positions 23-31) are left blank, auto report page headings are printed on each page (conditioned by 1P or overflow). If no overflow indicator is defined for the printer file, auto report assigns an unused overflow indicator to the printer file.

C Line 06

Blank line

A **B** **A**

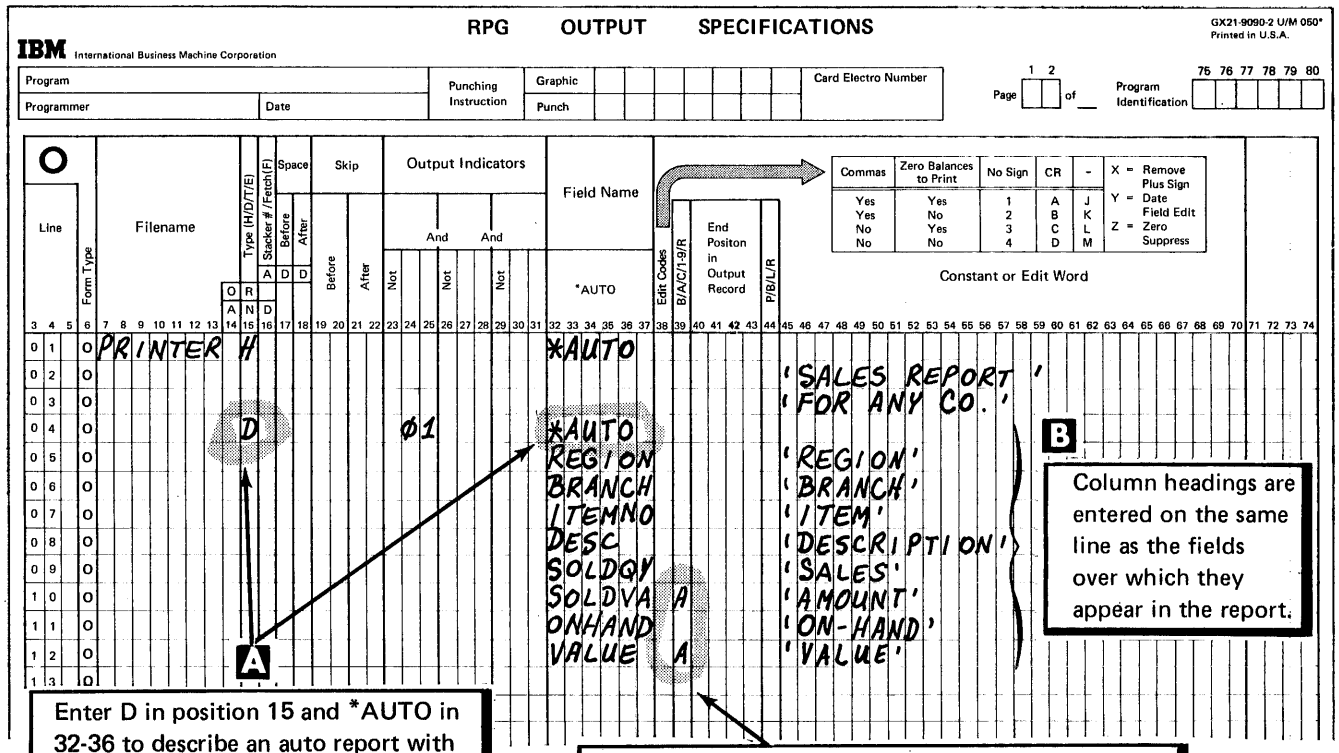
```
10/26/71                    SALES REPORT FOR ANY CO.                    PAGE    1

REGION  BRANCH  ITEM      DESCRIPTION      SALES      AMOUNT    ON-HAND      VALUE

  1        17   AG7701T   2-TON TRUCK          5    25,000.00         2   10,000.00
  1        17   AG7705S   PICK-UP             10    20,000.00         1    2,000.00
  1        17   AP6545B   CAMPER               2     8,000.00
  1        22   AG7701T   2-TON TRUCK          2    10,000.00         1    5,000.00
  1        22   AG7705S   PICK-UP              4     8,000.00         1    2,000.00
  3        25   AG6545B   CAMPER              10    40,000.00         5   20,000.00
  3        25   AP6549P   1/4 TON TRUCK       20    30,000.00         6    9,000.00

                                            141,000.00              48,000.00 *
```

3 Code *AUTO output specifications to produce:

A Detail report lines

B Column headings

C Final totals



10/26/71 SALES REPORT FOR ANY CO. PAGE 1

REGION	BRANCH	ITEM	DESCRIPTION	SALES	AMOUNT	ON-HAND	VALUE
1	17	AG7701T	2-TON TRUCK	5	25,000.00	2	10,000.00
1	17	AG7705S	PICK-UP	10	20,000.00	1	2,000.00
1	17	AP6545B	CAMPER	2	8,000.00		
1	22	AG7701T	2-TON TRUCK	2	10,000.00	1	5,000.00
1	22	AG7705S	PICK-UP	4	8,000.00	1	2,000.00
3	25	AG6545B	CAMPER	10	40,000.00	5	20,000.00
3	25	AP6549P	1/4 TON TRUCK	20	30,000.00	6	9,000.00
					141,000.00		48,000.00 *

A

C

Auto report formats the report so that column headings and data are neatly spaced and centered on each other.

All numeric fields for which a blank, B, or A is specified in position 39 are edited by the K edit code unless a different edit code is specified.

EXAMPLE 2

***AUTO Output**

Problem

Expand sales report from *Example 1* to include three levels of totals:

1. Total for each branch
2. Total for each region
3. Final total

Procedure

1 Code file description and *AUTO specifications as in *Example 1*.

2 Add control level indicators to the input fields BRANCH and REGION.

Note: The *AUTO output function can also be used to produce a group printed report. See index entry: *group printing*, for a discussion and examples of group printing.

RPG INPUT SPECIFICATIONS

IBM International Business Machine Corporation

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

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

Programmer | Date

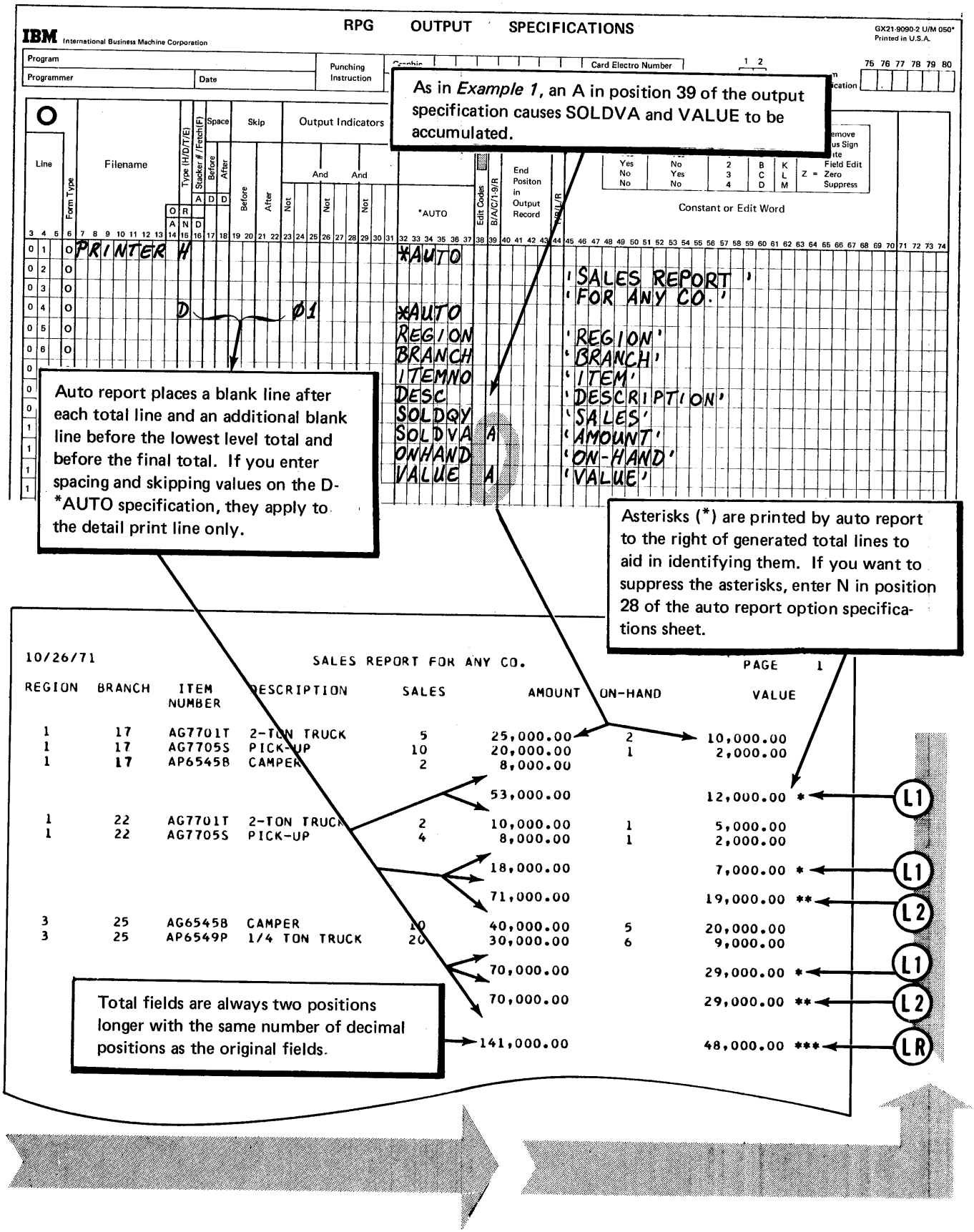
Page ___ of ___ | Program Identification

I

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location: From	To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank
01	I	SALES	AA			01															1	7		ITEMNO						
02	I																				8	9		BRANCHL1						
03	I																				10	10		REGIONL2						
04	I																				11	25		DESC						
05	I																				26	270		SOLDQY						
06	I																				28	342		SOLDVA						
07	I																				35	360		ONHAND						
08	I																				37	432		VALUE						
09	I																													
10	I																													
11	I																													

2

Since two control levels are defined, SOLDVA and VALUE fields (see opposite page) are accumulated to two levels of totals (branch and region) and a final total (LR).


RPG OUTPUT SPECIFICATIONS
IBM International Business Machine Corporation
GX21-9090-2 U/M 050*
Printed in U.S.A.
Program
Programmer
Date
Punching Instruction
Card Electro Number
Line
Filename
Form Type
Type (H/D/T/E)
Space
Skip
Output Indicators
Edit Codes
End Positon in Output Record
Constant or Edit Word
PRINTER H
*AUTO
'SALES REPORT '
'FOR ANY CO.'
D 01
*AUTO
REGION 'REGION'
BRANCH 'BRANCH'
ITEMNO 'ITEM'
DESC 'DESCRIPTION'
SOLDQY 'SALES'
SOLDVA A 'AMOUNT'
ONHAND 'ON-HAND'
VALUE A 'VALUE'
As in Example 1, an A in position 39 of the output specification causes SOLDVA and VALUE to be accumulated.
Auto report places a blank line after each total line and an additional blank line before the lowest level total and before the final total. If you enter spacing and skipping values on the D-*AUTO specification, they apply to the detail print line only.
Asterisks (*) are printed by auto report to the right of generated total lines to aid in identifying them. If you want to suppress the asterisks, enter N in position 28 of the auto report option specifications sheet.
10/26/71 SALES REPORT FOR ANY CO. PAGE 1
REGION BRANCH ITEM NUMBER DESCRIPTION SALES AMOUNT ON-HAND VALUE
1 17 AG7701T 2-TON TRUCK 5 25,000.00 2 10,000.00
1 17 AG7705S PICK-UP 10 20,000.00 1 2,000.00
1 17 AP6545B CAMPER 2 8,000.00
53,000.00 12,000.00 * L1
1 22 AG7701T 2-TON TRUCK 2 10,000.00 1 5,000.00
1 22 AG7705S PICK-UP 4 8,000.00 1 2,000.00
18,000.00 7,000.00 * L1
71,000.00 19,000.00 ** L2
3 25 AG6545B CAMPER 10 40,000.00 5 20,000.00
3 25 AP6549P 1/4 TON TRUCK 20 30,000.00 6 9,000.00
70,000.00 29,000.00 * L1
70,000.00 29,000.00 ** L2
141,000.00 48,000.00 *** LR
Total fields are always two positions longer with the same number of decimal positions as the original fields.

EXAMPLE 3

*AUTO Output

Problem

Expand the sales report from *Examples 1* and *2* to contain:

A Group indication for REGION and BRANCH fields

B Second column heading line

C Literal (constant) on the final total line

Procedure

1 Code file description and input specifications as for *Example 2.*

2 Code *AUTO output with:

A Output indicator on field description specifications

B C in position 39 and a literal in 45-70

C R in position 39 and a literal in 45-70

1

File Description Specification

Line	Form Type	Filename	File Type (I/O/U/C/D)	File Designation (P/S/C/R/T/D)	End of File (E)	Sequence (A/D)	File Format (F/V/S/M/D)	Block Length	Record Length	Device
02	F	SALES	I	P			F	473	43	DISK
03	F	PRINTER	O				F	120	120	PRINTER
04	F									

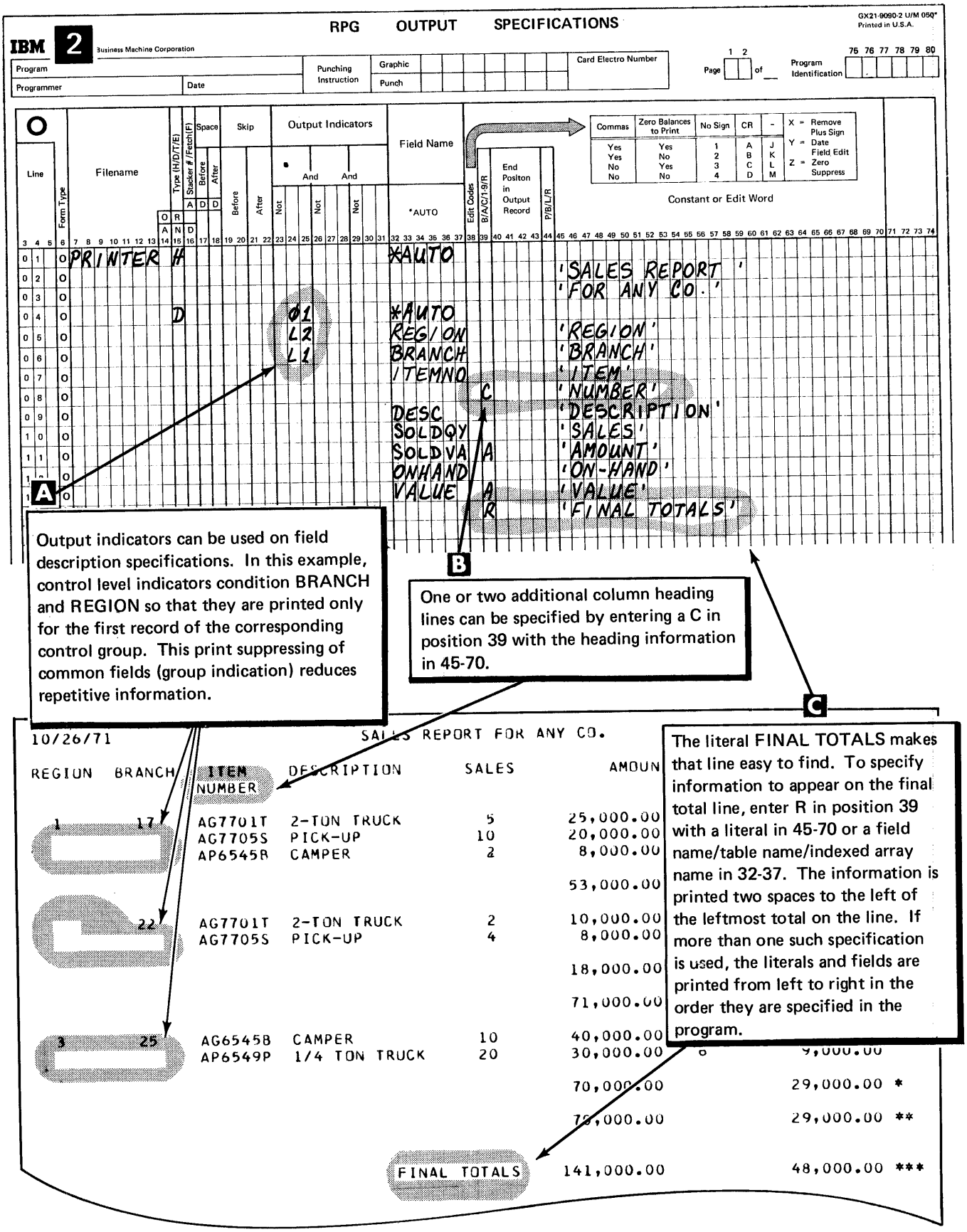
RPG INPUT SPECIFICATIONS

IBM International Business Machine Corporation

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

Program | Programmer | Date | Punching Instruction: Graphic, Punch | Card Electro Number | Page 1 2 of ___ | Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Record Identifying Indicator	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)
01	I	SALES	AA	Ø1					
02	I				1	7		ITEMNO	
03	I				8	9		BRANCH	L1
04	I				1Ø	1Ø		REGION	L2
05	I				11	25		DESC	
06	I				26	27		SOLDQY	
07	I				28	34	2	SOLDVA	
08	I				35	36	Ø	ONHAND	
09	I				37	43	2	VALUE	
10	I								


RPG OUTPUT SPECIFICATIONS
IBM 2
Program
Programmer
Date
Punching Instruction
Graphic
Punch
Card Electro Number
Page
Program Identification
Line
Form Type
Filename
Type (H/D/T/E)
Stacker # /Fetch(F)
Space
Skip
Output Indicators
Field Name
Edit Codes
End Position in Output Record
P/B/L/R
Commas
Zero Balances to Print
No Sign
CR
X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress
Constant or Edit Word
PRINTER
*AUTO
'SALES REPORT '
'FOR ANY CO.'
01
L2
L1
*AUTO
REGION
'REGION'
BRANCH
'BRANCH'
ITEMNO
'ITEM'
'NUMBER'
DESC
'DESCRIPTION'
SOLDQY
'SALES'
SOLDVA
'AMOUNT'
ONHAND
'ON-HAND'
VALUE
'VALUE'
'FINAL TOTALS'
A
B
C
Output indicators can be used on field description specifications. In this example, control level indicators condition BRANCH and REGION so that they are printed only for the first record of the corresponding control group. This print suppressing of common fields (group indication) reduces repetitive information.
One or two additional column heading lines can be specified by entering a C in position 39 with the heading information in 45-70.
The literal FINAL TOTALS makes that line easy to find. To specify information to appear on the final total line, enter R in position 39 with a literal in 45-70 or a field name/table name/indexed array name in 32-37. The information is printed two spaces to the left of the leftmost total on the line. If more than one such specification is used, the literals and fields are printed from left to right in the order they are specified in the program.
10/26/71
SALES REPORT FOR ANY CO.
REGION BRANCH ITEM NUMBER DESCRIPTION SALES AMOUN
1 17 AG7701T 2-TON TRUCK 5 25,000.00
AG7705S PICK-UP 10 20,000.00
AP6545B CAMPER 2 8,000.00
53,000.00
22 AG7701T 2-TON TRUCK 2 10,000.00
AG7705S PICK-UP 4 8,000.00
18,000.00
71,000.00
3 25 AG6545B CAMPER 10 40,000.00
AP6549P 1/4 TON TRUCK 20 30,000.00
70,000.00 29,000.00 *
29,000.00 **
FINAL TOTALS 141,000.00 48,000.00 ***

EXAMPLE 4

*AUTO Page Headings | *AUTO Output

Problem

Expand the sales report from *Examples 1-3* to include a cross-totals column and:

- **A** A new report page for each region
- **B** Two heading lines on each page
- **C** A field in a page heading line
- **D** Identification of branch and region totals

Procedure

1. Code file description and input specifications as in *Example 3*; add an overflow indicator to the printer file.
2. Code RPG II calculation specifications for cross-total.
3. Code *AUTO specifications:
   - **A** Output indicators on page heading specifications
   - **B** Two heading lines per page
   - **C** Use of a field in an *AUTO page heading specification
   - **D** Fields and literals on L1-L9 total lines (1-9 in position 39)

11/18/71 — SALES REPORT FOR ANY CO. — REGION 1 (B → REGION 1 ← C) — 2 PAGE 1

BRANCH	ITEM NUMBER	DESCRIPTION	SALES QUANTITY	SALES VALUE	ON HAND	ON-HAND VALUE	TOTAL
17	AG7701T	2-TON TRUCK	5	25,000.00	2	10,000.00	35,000.00
	AG7705S	PICK-UP	10	20,000.00	1	2,000.00	22,000.00
	AP6545B	CAMPER	2	8,000.00			8,000.00
		D BRANCH 17 TOTALS		53,000.00		12,000.00	65,000.00 *
22	AG7701T	2-TON TRUCK	2	10,000.00	1	5,000.00	15,000.00
	AG7705S	PICK-UP	4	8,000.00	1	2,000.00	10,000.00
		BRANCH 22 TOTALS		18,000.00		7,000.00	25,000.00 *
		D REGION 1 TOTALS		71,000.00		19,000.00	90,000.00 **

11/18/71 — SALES REPORT FOR ANY CO. — REGION 3 — A PAGE 2

BRANCH	ITEM NUMBER	DESCRIPTION	SALES QUANTITY	SALES VALUE	ON HAND	ON-HAND VALUE	TOTAL
25	AG6545B	CAMPER	10	40,000.00	5	20,000.00	60,000.00
	AP6549P	1/4 TON TRUCK	20	30,000.00	6	9,000.00	39,000.00
		BRANCH 25 TOTALS		70,000.00		29,000.00	99,000.00 *
		REGION 3 TOTALS		70,000.00		29,000.00	99,000.00 **
		COMPANY TOTALS		141,000.00		48,000.00	189,000.00 ***

Note: Compare matching letters (**B**) on this and the opposite pages to see the auto report coding to obtain this report.

RPG CALCULATION SPECIFICATIONS

IBM International Business Machine Corporation

Program | Programmer | Date | Punching Instruction: Graphic / Punch | Card Electro Number **2**

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators	Factor 1	Operation	Factor 2	Result Field Name	Length	Decimal Positions	Half Adjust (H)
01	C		Ø1	SOLDVA	ADD	VALUE	TOTVAL	8	2	
02	C									

RPG II calculations can be among the input statements for auto report. This specification calculates a cross-total of the sales and on-hand values. (The placement of the calculation in relation to calculations generated by auto report is described under the index entry *generated RPG II program*.)

RPG OUTPUT SPECIFICATIONS

IBM International Business Machine Corporation

GX21-9090-2 U/M 050*
Printed in U.S.A.

Punching Instruction: Graphic / Punch | Card Electro Number **3** | Page __ of __ | Program Identification

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

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

Line	Form Type	Filename	Type (H/D/T/E)	Output Indicators	Field Name (*AUTO)	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O	PRINTER	H	L2	*AUTO					
02	O		OR	OFNL2						
03	O									'SALES REPORT '
04	O									'FOR ANY CO.'
05	O		H		*AUTO					
06	O									'REGION'
07	O				REGION					
08	O		D	Ø1	*AUTO					
09	O			L1	BRANCH					'BRANCH'
					ITEMNO					'ITEM'
							C			'NUMBER'
					DESC					'DESCRIPTION'
					SOLDQY					'SALES'
							C			'QUANTITY'
					SOLDVA		A			'SALES VALUE'
					ONHAND					'ON'
							C			'HAND'
					VALUE		A			'ON-HAND VALUE'
					TOTVAL		A			'TOTAL'
20	O						1			'BRANCH'
21	O				BRANCH		1			
22	O						1			'TOTALS'
23	O						2			'REGION'
24	O				REGION		2			
25	O						2			'TOTALS'
26	O						R			'COMPANY TOTALS'

A The headings are printed on a new page when the region number changes (L2) or when overflow occurs (OF). (OF must be defined for the printer file in file description specifications.)

B A second auto report page heading is specified. Since spacing is not specified, space one is done after the first and space two after the second. Since no output indicators are specified, the second heading is conditioned like the first.

C The contents of the REGION field are printed on the second page heading.

D Fields and literals can be printed on generated total lines by entering the number of the control level in position 39.

COPY

Examples 5 and 6 illustrate use of the auto report copy function to copy specifications from the library and to override copied specifications for a particular job.

EXAMPLE 5

COPY

Problem

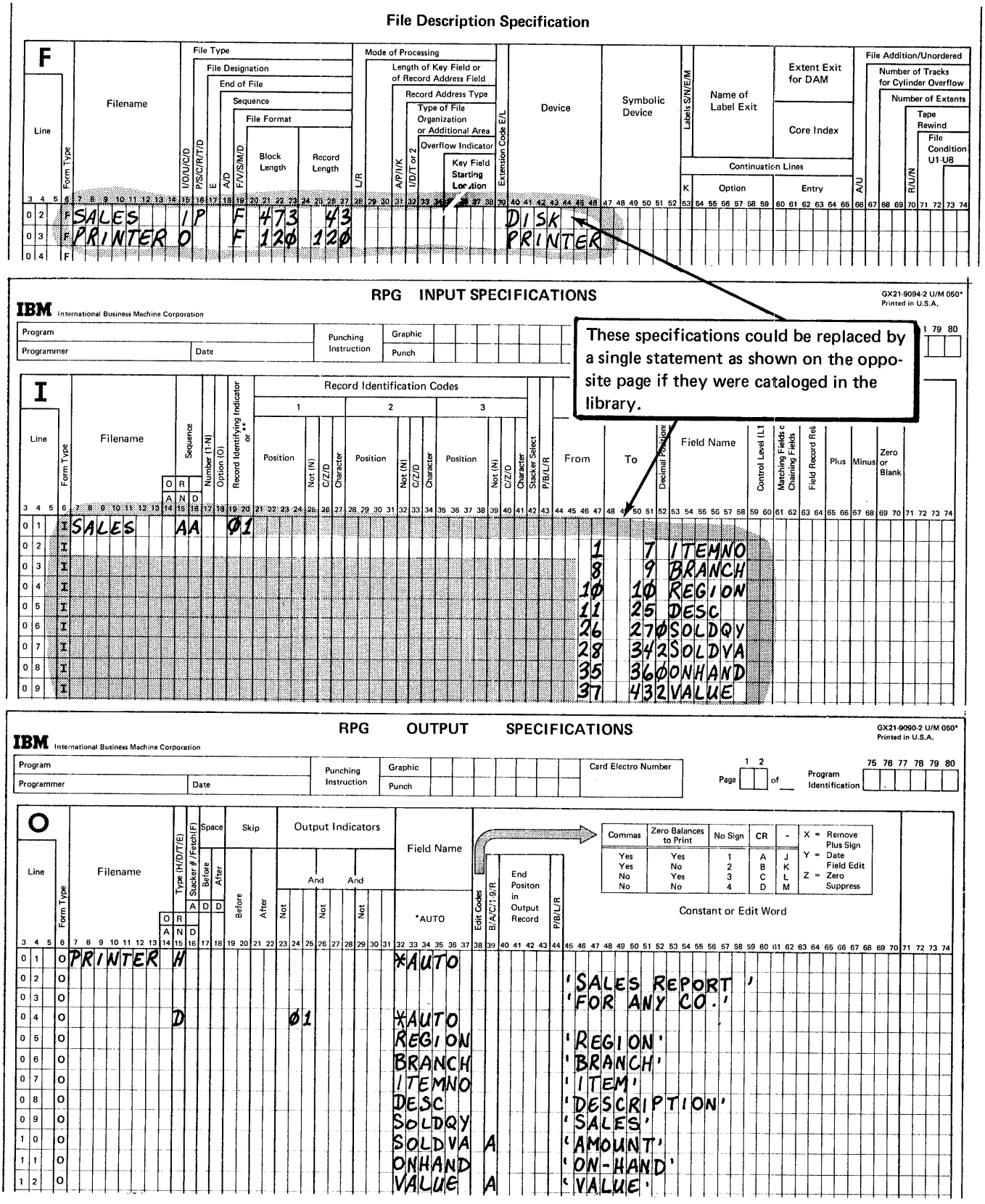
Use the copy function to obtain specifications for the sales report below (same as in *Example 1*).

Procedure

1. Catalog the file description and input specifications for the SALES file in the library.
2. Code the /COPY statement in the specifications for auto report.

```
10/26/71                        SALES REPORT FOR ANY CO.                        PAGE    1

REGION  BRANCH  ITEM     DESCRIPTION      SALES      AMOUNT   ON-HAND      VALUE

  1       17    AG7701T  2-TON TRUCK        5    25,000.00       2     10,000.00
  1       17    AG7705S  PICK-UP           10    20,000.00       1      2,000.00
  1       17    AP6545B  CAMPER             4     8,000.00
  1       22    AG7701T  2-TON TRUCK        2    10,000.00       1      5,000.00
  1       22    AG7705S  PICK-UP            4     8,000.00       1      2,000.00
  3       25    AG6545B  CAMPER            10    40,000.00       5     20,000.00
  3       25    AP6549P  1/4 TON TRUCK     20    30,000.00       6      9,000.00

                                          141,000.00             48,000.00 *
```

1 Catalog specifications for the SALES file in the library using the library maintenance utility program.



2 Code the /COPY statement to include the file description and input specifications. (For a detailed description of the copy function, see index entry: *auto report copy specifications*.)

File Description Specification

```
     F
Line Form Filename  I/O/U/C/D ... F/V/S/M/D  Block Length  Record Length ... Device
0 2  F    PRINTER   O              F          120           120               PRINTER
0 3  F
```

RPG INPUT SPECIFICATIONS

IBM International Business Machine Corporation

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

Program | Programmer | Date | Punching Instruction: Graphic, Punch | Card Electro Number | Page 1 2 of | Program Identification 75 76 77 78 79 80

```
     I
Line Form
0 1  I    /COPY F1,SALETR
0 2  I
```

The /COPY statement copies file description and input specifications for the SALES file from the source library member named SALETR.

The source library is on disk unit F1.

Position 6 of a /COPY statement must not contain a U or an H.

The /COPY statement can appear anywhere among the auto report specifications following the auto report option statement and preceding table and array input records. It is convenient to code the /COPY on the input sheet when you want to override copied input specifications, as in *Example 6*. After specifications are copied, all specifications are sorted into the order required by the RPG II Compiler.

RPG OUTPUT SPECIFICA

IBM International Business Machine Corporation

Program | Programmer | Date | Punching Instruction: Graphic, Punch

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

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

```
     O
Line Form Filename  Type  Output Ind.  Field Name  B/A/C/1-9/R  Constant or Edit Word
0 1  O    PRINTER   H                  *AUTO
0 2  O                                                          'SALES REPORT '
0 3  O                                                          'FOR ANY CO. '
0 4  O              D     01           *AUTO
0 5  O                                 REGION                   'REGION'
0 6  O                                 BRANCH                   'BRANCH'
0 7  O                                 ITEMNO                   'ITEM'
0 8  O                                 DESC                     'DESCRIPTION'
0 9  O                                 SOLDQY                   'SALES'
1 0  O                                 SOLDVA      A            'AMOUNT'
1 1  O                                 ONHAND                   'ON-HAND'
1 2  O                                 VALUE       A            'VALUE'
1 3  O
```

EXAMPLE 6

COPY

Problem

Override copied input specifications to produce a report (below) that includes subtotals for branch and region.

Procedure

1. Catalog specifications for the SALES file, as in *Example 5.*
2. Code the /COPY statement.
3. Code /COPY modifier statements to add control level indicators to BRANCH and REGION fields on copied specifications.

```
10/26/71                                SALES REPORT FOR ANY CO.                                  PAGE     1

REGION   BRANCH    ITEM     DESCRIPTION        SALES            AMOUNT   ON-HAND             VALUE
                   NUMBER

  1        17      AG7701T  2-TON TRUCK          5           25,000.00      2            10,000.00
                   AG7705S  PICK-UP             10           20,000.00      1             2,000.00
                   AP6545B  CAMPER               2            8,000.00

                                                             53,000.00                   12,000.00 *

           22      AG7701T  2-TON TRUCK          2           10,000.00      1             5,000.00
                   AG7705S  PICK-UP              4            8,000.00      1             2,000.00

                                                             18,000.00                    7,000.00 *

                                                             71,000.00                   19,000.00 **

  3        25      AG6545B  CAMPER              10           40,000.00      5            20,000.00
                   AP6549P  1/4 TON TRUCK       20           30,000.00      6             9,000.00
```

IBM International Business Machine Corporation

RPG INPUT SPECIFICATIONS

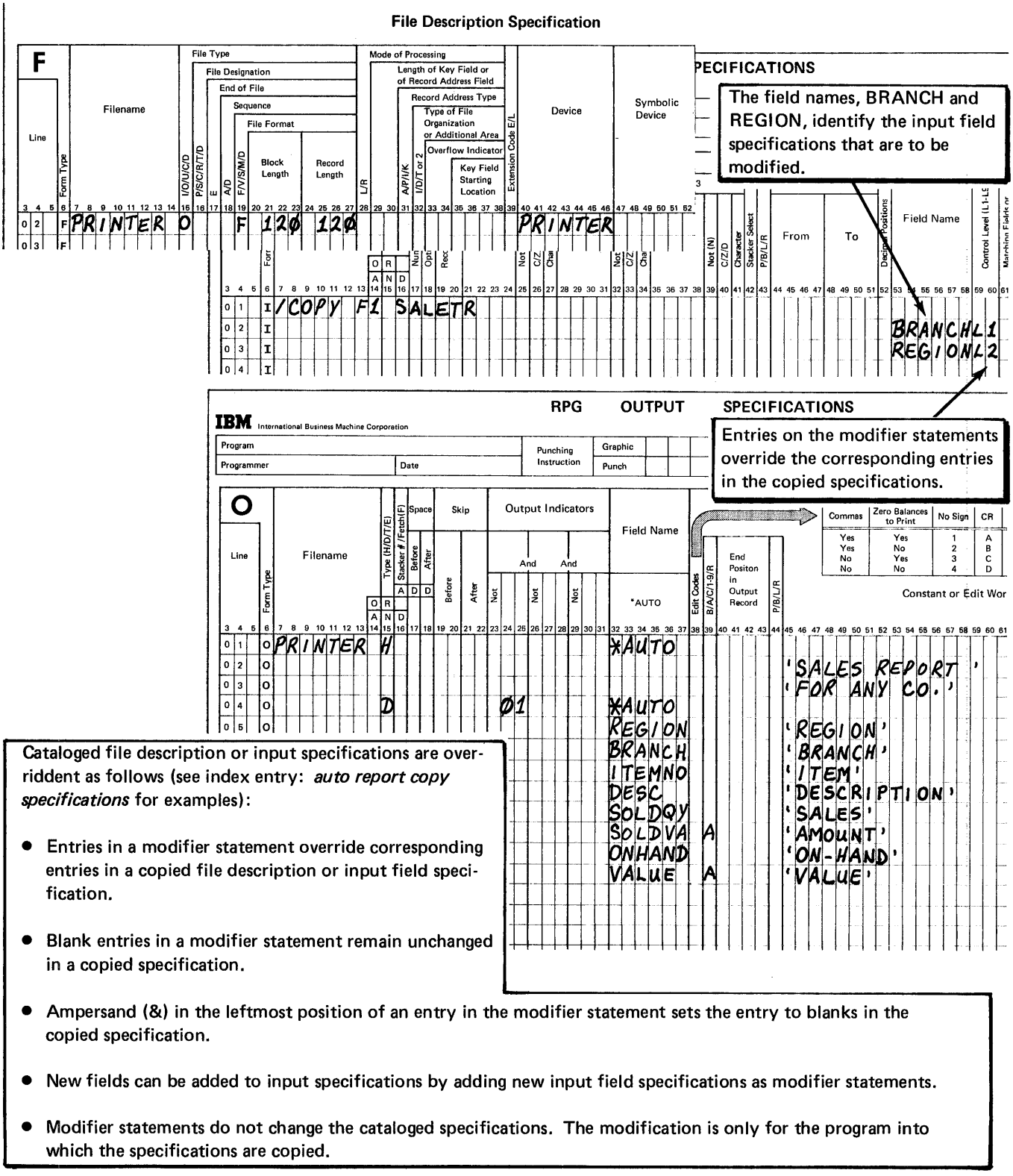
GX21-9094-2 U/M 050* Printed in U.S.A.

Cataloged input specifications for the SALES file.

Line	Type	Filename	Sequence	Resulting Indicator	From	To	Decimal Positions	Field Name
01	I	SALES	AA	Ø1				
02	I				1	7		ITEMNO
03	I				8	9		BRANCH
04	I				1Ø	1Ø		REGION
05	I				11	25		DESC
06	I				26	27	Ø	SOLDQY
07	I				28	34	2	SOLDVA
08	I				35	36	Ø	ONHAND
09	I				37	43	2	VALUE
10	I							
11	I							

To produce a report that has subtotals for branch and region, L1 must be assigned to BRANCH and L2 to REGION as the specifications are copied from the library.

2 and **3** Code /COPY and modifier statements. As a result of the modifier statements three levels of totals are accumulated for the SOLDVA and VALUE fields (L1, L2, and LR).



Cataloged file description or input specifications are overriddent as follows (see index entry: *auto report copy specifications* for examples):

- Entries in a modifier statement override corresponding entries in a copied file description or input field specification.
- Blank entries in a modifier statement remain unchanged in a copied specification.
- Ampersand (&) in the leftmost position of an entry in the modifier statement sets the entry to blanks in the copied specification.
- New fields can be added to input specifications by adding new input field specifications as modifier statements.
- Modifier statements do not change the cataloged specifications. The modification is only for the program into which the specifications are copied.

These sample programs on the distribution diskette illustrate the use of all auto report functions: *AUTO page headings, *AUTO output, and copy. The auto report specifications for the job are explained. The auto report listing, the RPG II listing, and the final report are shown.

JOB DESCRIPTION

This job prepares a cash receipts register using RPG II with the auto report function. The *AUTO page headings function and the *AUTO output function generate the RPG II output specifications for the report and the calculation specifications to accumulate final totals for several fields on the report. RPG II calculations specifications that cannot be generated by auto report are included in the auto report program to verify the discount taken by each customer and to calculate the balance due.

The file description specifications for the cash receipts register printer file, CSHRECRG, and the file description and input specifications for the input file, CSHREC (Figure 192) are cataloged as separate members in the library on disk unit F1. The cataloged specifications are included in the program by the auto report copy function.

AUTO REPORT CODING

Figure 193 shows the RPG II and auto report specifications that must be included in the auto report program EXAUT2 to produce the cash receipts register. The input data for the file CSHREC in EXAUT2 is generated by the program EXAUT1 (Figure 194). Figure 195 shows the input data.

RPG II Control Specifications

The control specifications shown in Figure 193, insert A, should be included in the auto report program, since it is not present among the cataloged specifications (Figure 192). None of the control specification options are required in this program, so the specification need contain only an H in position 6 and the program identification entry, EXAUT2, in positions 75-80. The program identification characters from positions 75-80 of the H specifications are placed in positions 75-80 of all specifications in the generated RPG II source program.

/COPY Statements

The /COPY statements shown in Figure 193, insert B, copy the file description and input specifications for the job from the library on disk unit F1. The first statement copies the file description specifications for the printer file from the library member named EXAUT3. The second statement copies the file description and input specifications for the disk file, CASHRC, from the library member named EXAUT4. A modifier statement adds an input field definition for the REGION field. As a result of these /COPY statements, the file description and input specifications shown in Figure 192 are included in the RPG II source program generated by auto report.

Calculation Specifications

The calculation specifications shown in Figure 193, insert C, are included in the auto report program to perform special operations that cannot be generated by auto report. First, the discount allowed for each customer is subtracted from the discount taken by each customer. Indicator 10 is turned on if the difference is greater than or equal to $1.00. The remaining calculations subtract the discount taken and the amount paid from the amount owed.

The order in which these calculations are placed in relation to the calculations generated by auto report is shown in the auto report listing of the generated RPG II source program (Figure 196).

*AUTO Specifications

The coding for the *AUTO page headings and the *AUTO output functions is shown in Figure 193, insert D. Notice that the Y edit code is used for the date fields (lines 10 and 12). Auto report generates a K edit code for numeric fields when an edit code is not specified. No edit code is generated for numeric fields when they are described with a 1-9 or R in position 39. The edit code 3 is specified for the INVNO field to suppress the printing of the comma edit character.

DIFF is printed on the detail line only if it is $1.00 or more. Remember, output indicator 10 only conditions the printing of the field on the detail line; it does not affect the printing of the generated field on the total line.

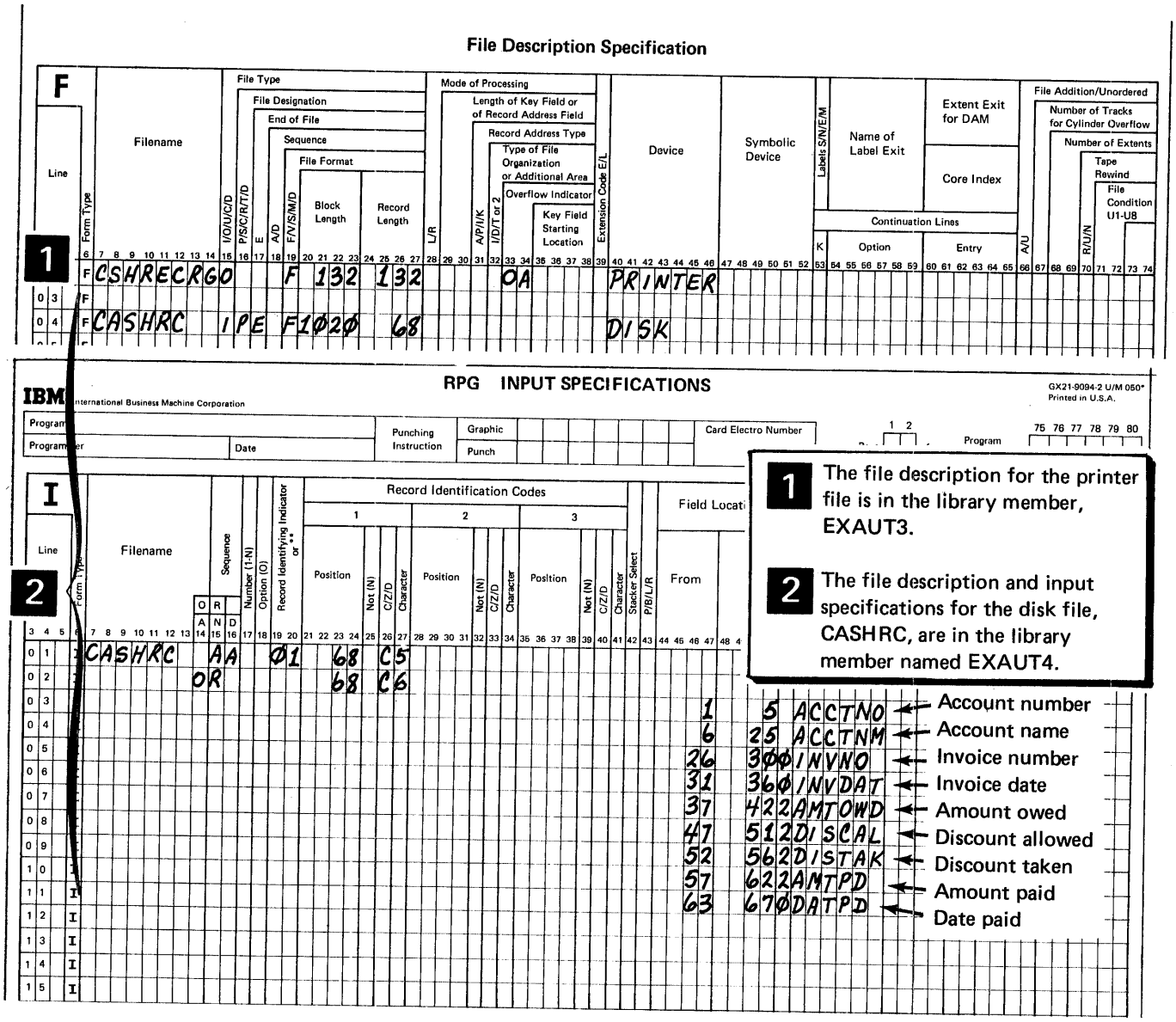
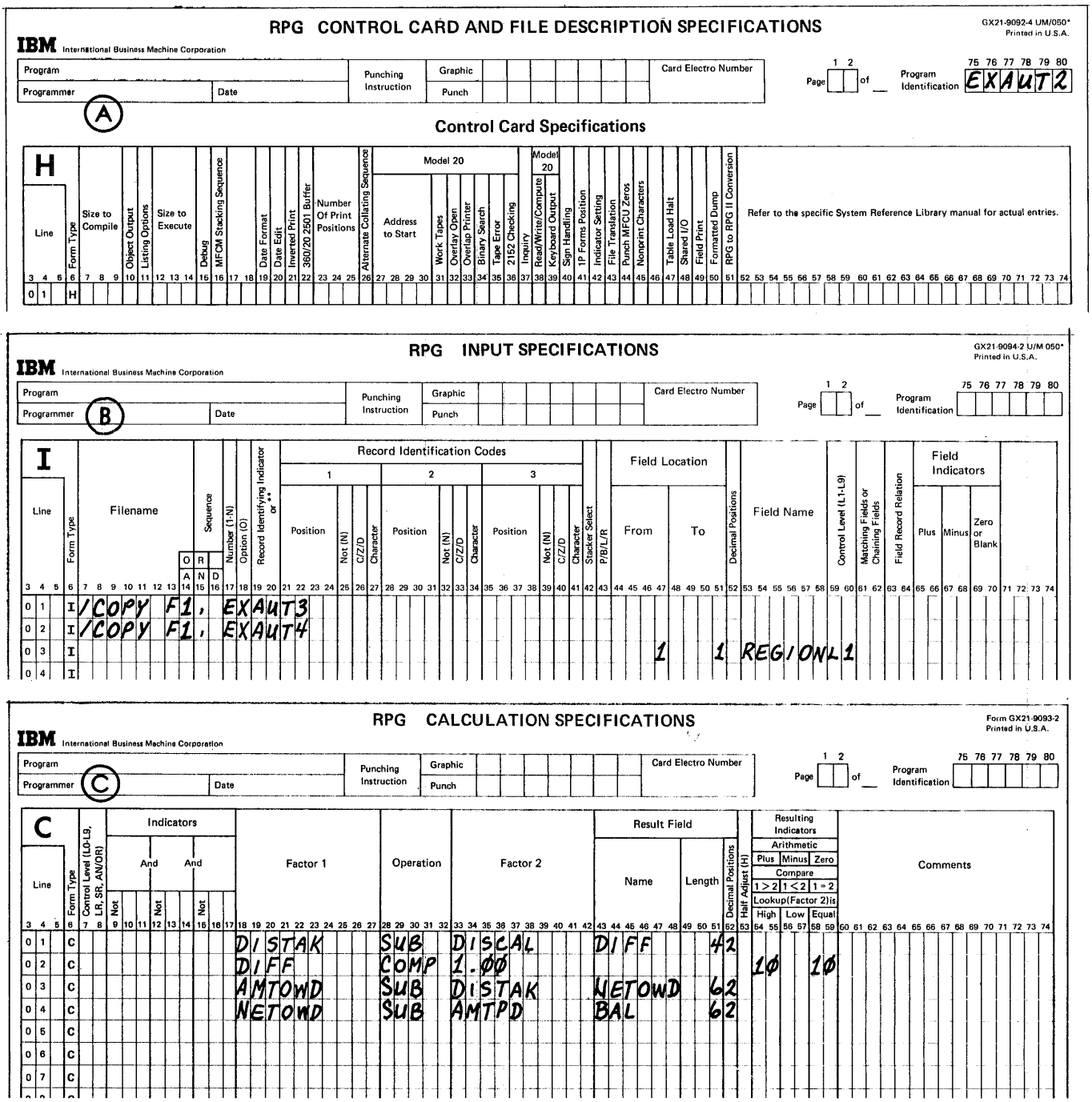


Figure 192. File Description and Input Specifications that are Cataloged in the Library Members EXAUT3 and EXAUT4



RPG CONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS

GX21-9092-4 U/M/050*
Printed in U.S.A.

IBM International Business Machine Corporation

Program | Programmer | Date | Punching Instruction | Graphic Punch | Card Electro Number | Page 1 2 of __ | Program Identification 75 76 77 78 79 80 EXAUT2

(A)

Control Card Specifications

H

Line	Form Type	Size to Compile	Object Output	Listing Options	Size to Execute	Debug	MFCM Stacking Sequence	Date Format	Date Edit	Inverted Print	360/20 2501 Buffer	Number Of Print Positions	Alternate Collating Sequence	Model 20: Address to Start	Work Tapes	Overlay Open	Overlap Printer	Binary Search	Tape Error	2152 Checking	Inquiry	Model 20: Read/Write/Compute	Keyboard Output	Sign Handling	1P Forms Position	Indicator Setting	File Translation	Punch MFCU Zeros	Nonprint Characters	Table Load Halt	Shared I/O	Field Print	Formatted Dump	RPG to RPG II Conversion	Refer to the specific System Reference Library manual for actual entries.
3 4 5	6	7 8 9	10 11	12 13 14	15	16	17 18	19	20	21	22	23 24 25	26	27 28 29 30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
0 1	H																																		

RPG INPUT SPECIFICATIONS

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

IBM International Business Machine Corporation

Program | Programmer | Date | Punching Instruction | Graphic Punch | Card Electro Number | Page 1 2 of __ | Program Identification 75 76 77 78 79 80

(B)

I

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank
			O A 14 / R N 15 / D 16																											
3 4 5	6	7 8 9 10 11 12 13		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
0 1	I	/COPY F1,		EXAUT3																										
0 2	I	/COPY F1,		EXAUT4																										
0 3	I																				1		1	REGIONL1						
0 4	I																													

RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2
Printed in U.S.A.

IBM International Business Machine Corporation

Program | Programmer | Date | Punching Instruction | Graphic Punch | Card Electro Number | Page 1 2 of __ | Program Identification 75 76 77 78 79 80

(C)

C

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And Not	And Not	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Result Field: Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus	Arithmetic Minus	Arithmetic Zero	Compare 1>2	Compare 1<2	Compare 1=2	Lookup(Factor 2)is High	Lookup(Factor 2)is Low	Lookup(Factor 2)is Equal	Comments
3 4 5	6	7 8	9 10 11 12 13 14 15 16		18 19 20 21 22 23 24 25 26 27	28 29 30 31 32	33 34 35 36 37 38 39 40 41 42	43 44 45 46 47 48	49 50 51	52	53	54	55	56	57	58	59	60	61	62	63 64 65 66 67 68 69 70 71 72 73 74
0 1	C				DISTAK	SUB	DISCAL	DIFF		42									10		10	
0 2	C				DIFF	COMP	1.00															
0 3	C				AMTOWD	SUB	DISTAK	NETOWD		62												
0 4	C				NETOWD	SUB	AMTPD	BAL		62												
0 5	C																					
0 6	C																					
0 7	C																					

Figure 193 (Part 1 of 2). RPG II and Auto Report Specifications to Produce the Cash Receipts Register

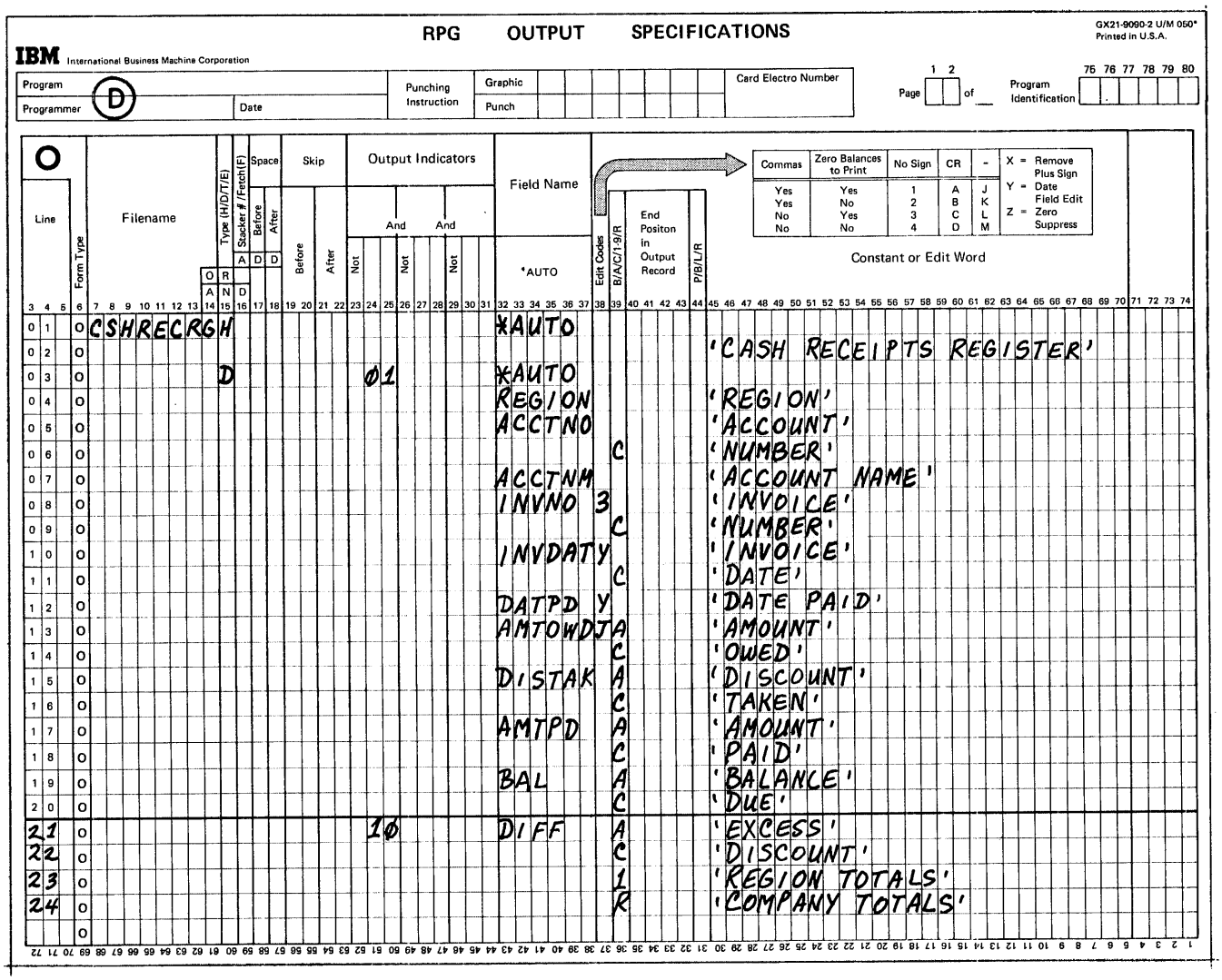


Figure 193 (Part 2 of 2). RPG II and Auto Report Specifications to Produce the Cash Receipts Register

The J edit code allows zero balance to print for the AMTOWD field.

Totals are accumulated and printed by auto report for five fields, as indicated by A-entries in position 39. Since an L1 control level is defined in the input field specifications for REGION which is added to the input specifications for the CASHRC file (Figure 193, insert B), regional and final totals are accumulated for each field which has an A in position 39. The total lines are identified by the literals shown in lines 23 and 24 of the *AUTO specifications (Figure 193, insert D).

RUNNING THE SAMPLE PROGRAM

The sample auto report programs and the procedures to compile and execute them are on the distribution diskettes. The sample programs should be run after the diskettes are copied to the disk. To run the sample test cases, key in the word RPGSAMPL. RPGSAMPL is the name of the procedure that compiles and executes the RPG II program product verification programs, of which programs EXAUT1 and EXAUT2 are a part. Successful execution of the sample programs indicates that the auto report function is working correctly. Upon successful completion, the program product verification programs are deleted from the library. Figure 196 shows the output listings for the auto report sample program EXAUT2. Figure 197 shows the output data produced by EXAUT2.

```
      01010H     008                                                                    EXAUT1
0001  0102 FKEYIN   IP  F 100 100            KEYBORD
0002  0103 FCASHRC  O   F1020  68            DISK
0003  0104 FPRINTER O   F 120 120            PRINTER
0004  0203 C                        SETOF                    02
0005  0204 C                        SETON                    LR
0006  0205 C   02                   KEY01          DUMMY   1
0007  0301 OPRINTER T   301   LR
0008  0302 O                                      54 'DATA FOR SAMPLE PROGRAM'
0009  030210                          02DUMMY      56
0010  0303 O         T  2     LR
0011  0304 O                                      24 '11243JONES HARDWARE      '
0012  0305 O                                      48 ' 27541021175  2375CASH  '
0013  0306 O                                      68 ' 47   47  2328022175'
0014  0307 O         T  2     LR
0015  0308 O                                      24 '11352NU-STYLE CLOTHIERS '
0016  0309 O                                      48 ' 27987021475  8707CASH  '
0017  0310 O                                      68 '174        4000022675'
0018  0311 O         T  2     LR
0019  0312 O                                      24 '11886MIDI FASHIONS INC   '
0020  0313 O                                      48 ' 15771020475 10722CASH  '
0021  0314 O                                      68 '214  214 10508021475'
0022  0401 O         T  2     LR
0023  0402 O                                      24 '12874ULOOK INTERIORS     '
0024  0403 O                                      48 ' 25622020975  6795CASH  '
0025  0404 O                                      68 '136        6795022375'
0026  0405 O         T  2     LR
0027  0406 O                                      24 '18274STREAMLINE PAPER IN'
0028  0407 O                                      48 'C29703022175 27403       '
0029  0408 O                                      68 '548  238 17055023075'
0030  0409 O         T  2     LR
0031  0410 O                                      24 '23347RITE-BEST PENS CO   '
0032  0411 O                                      48 ' 20842021875  1580       '
0033  0412 O                                      68 '31         1000022075'
0034  0413 O         T  2     LR
0035  0414 O                                      24 '25521IMPORTS OF NM       '
0036  0415 O                                      48 ' 29273022075 79740      1'
0037  0416 O                                      68 '593 1193 58547022775'
0038  0501 O         T  2     LR
0039  0502 O                                      24 '26723ALRIGHT CLEANERS    '
0040  0503 O                                      48 ' 19473020775 46200CASH  '
0041  0504 O                                      68 '924       46200022375'
0042  0505 O         T  2     LR
0043  0506 O                                      24 '28622NORTH CENTRAL SUPPL'
0044  0507 O                                      48 'Y17816020575  7597CASH  '
0045  0508 O                                      68 '152        7597022275'
0046  0509 O         T  2     LR
0047  0510 O                                      24 '29871FERGUSON DEALERS    '
0048  0511 O                                      48 ' 27229021075  6191CASH  '
0049  0512 O                                      68 '124        6191022275'
0050  0513 O         T  2     LR
0051  0514 O                                      24 '30755FASTWAY AIRLINES    '
0052  0515 O                                      48 ' 26158020675 74272CASH 1'
0053  0516 O                                      68 '495 1685 72587021975'
0054  0517 O         T  2     LR
0055  0518 O                                      24 '31275ENVIRONMENT CONCERN'
```

Figure 194 (Part 1 of 3). EXAUT1 Program

```
0056  0519 0                                                  48 'S20451020675  2943      '
0057  0520 0                                                  68 ' 59        1500023075'
0058  0601 0          T  2       LR
0059  0602 0                                                  24 '32457B SOLE SILOS       '
0060  0603 0                                                  48 ' 27425021075 11005CASH  '
0061  0604 0                                                  68 '220       11005022075'
0062  0605 0          T  2       LR
0063  0606 0                                                  24 '37945HOFFTA BREAKS INC  '
0064  0607 0                                                  48 ' 18276020675  4723CASH  '
0065  0608 0                                                  68 ' 94        4723022375'
0066  0609 0          T  2       LR
0067  0610 0                                                  24 '42622EASTLAKE GRAVEL CO '
0068  0611 0                                                  48 ' 16429020575  2937CASH  '
0069  0612 0                                                  68 ' 58        2937022375'
0070  0613 0CASHRC    T          LR
0071  0704 0                                                  24 '11243JONES HARDWARE     '
0072  0705 0                                                  48 ' 27541021175  2375CASH  '
0073  0706 0                                                  68 ' 47   47  2328022175'
0074  0707 0          T          LR
0075  0708 0                                                  24 '11352NU-STYLE CLOTHIERS '
0076  0709 0                                                  48 ' 27987021475  8707CASH  '
0077  0710 0                                                  68 '174        4000022675'
0078  0711 0          T          LR
0079  0712 0                                                  24 '11886MIDI FASHIONS INC  '
0080  0713 0                                                  48 ' 15771020475 10722CASH  '
0081  0714 0                                                  68 '214  214 10508021475'
0082  0801 0          T          LR
0083  0802 0                                                  24 '12874ULOOK INTERIORS    '
0084  0803 0                                                  48 ' 25622020975  6795CASH  '
0085  0804 0                                                  68 '136        6795022375'
0086  0805 0          T          LR
0087  0806 0                                                  24 '18274STREAMLINE PAPER IN'
0088  0807 0                                                  48 'C29703022175 27403      '
0089  0808 0                                                  68 '548  238 17055023075'
0090  0809 0          T          LR
0091  0810 0                                                  24 '23347RITE-BEST PENS CO  '
0092  0811 0                                                  48 ' 20842021875  1580      '
0093  0812 0                                                  68 '31         1000022075'
0094  0813 0          T          LR
0095  0814 0                                                  24 '25521IMPORTS OF NM      '
0096  0815 0                                                  48 ' 29273022075 79740     1'
0097  0816 0                                                  68 '593 1193 58547022775'
0098  0901 0          T          LR
0099  0902 0                                                  24 '26723ALRIGHT CLEANERS   '
0100  0903 0                                                  48 ' 19473020775 46200CASH  '
0101  0904 0                                                  68 '924       46200022375'
0102  0905 0          T          LR
0103  0906 0                                                  24 '28622NORTH CENTRAL SUPPL'
0104  0907 0                                                  48 'Y17816020575  7597CASH  '
0105  0908 0                                                  68 '152        7597022275'
0106  0909 0          T          LR
0107  0910 0                                                  24 '29871FERGUSON DEALERS   '
0108  0911 0                                                  48 ' 27229021075  6191CASH  '
0109  0912 0                                                  68 '124        6191022275'
0110  0913 0          T          LR
0111  0914 0                                                  24 '30755FASTWAY AIRLINES   '
0112  0915 0                                                  48 ' 26158020675 74272CASH 1'
0113  0916 0                                                  68 '495 1685 72587021975'
0114  0917 0          T          LR
0115  0918 0                                                  24 '31275ENVIRONMENT CONCERN'
```

Figure 194 (Part 2 of 3). EXAUT1 Program

```
0116  0919 0                                          48 'S20451020675  2943      '
0117  0920 0                                          68 ' 59       1500023075'
0118  1001 0          T         LR
0119  1002 0                                          24 '324578 SOLE SILOS       '
0120  1003 0                                          48 ' 27425021075 11005CASH  '
0121  1004 0                                          68 '220       11005022075'
0122  1005 0          T         LR
0123  1006 0                                          24 '37945HOFFTA BREAKS INC  '
0124  1007 0                                          48 ' 18276020675  4723CASH  '
0125  1008 0                                          68 ' 94       4723022375'
0126  1009 0          T         LR
0127  1010 0                                          24 '42622EASTLAKE GRAVEL CO '
0128  1011 0                                          48 ' 16429020575  2937CASH  '
0129  1012 0                                          68 ' 58       2937022375'
```

Figure 194 (Part 3 of 3). EXAUT1 Program

```
                                DATA FOR SAMPLE PROGRAM

11243JONES HARDWARE          27541021175  2375CASH   47   47  2328022175
11352NU-STYLE CLOTHIERS      27987021475  8707CASH  174       4000022675
11886MIDI FASHIONS INC       15771020475 10722CASH  214  214 10508021475
12874ULOOK INTERIORS         25622020975  6795CASH  136       6795022375
18274STREAMLINE PAPER INC    29703022175 27403       548  238 17055023075
23347RITE-BEST PENS CO       20842021875  1580       31        1000022075
25521IMPORTS OF NM           29273022075 79740      1593 1193 58547022775
26723ALRIGHT CLEANERS        19473020775 46200CASH  924       46200022375
28622NORTH CENTRAL SUPPLY    17816020575  7597CASH  152        7597022275
29871FERGUSON DEALERS        27229021075  6191CASH  124        6191022275
30755FASTWAY AIRLINES        26158020675 74272CASH 1495 1685 72587021975
31275ENVIRONMENT CONCERNS    20451020675  2943       59        1500023075
32457B SOLE SILOS            27425021075 11005CASH  220       11005022075
37945HOFFTA BREAKS INC       18276020675  4723CASH   94        4723022375
42622EASTLAKE GRAVEL CO      16429020575  2937CASH   58        2937022375
```

Figure 195. Input Data Generated by EXAUT1 for Auto Report Sample Program EXAUT2

```
 IBM SYSTEM/32   RPG II AUTO REPORT                VER-01 MOD-00

0001        U                          N
0002        H      012                                                    EXAUT2
0003  0101  I/COPY F1,EXAUT3
0004C       FCSHRECRGO   F 132 132      OA      PRINTER
0005  0102  I/COPY F1,EXAUT4
0006C     02 FCASHRC   IPE F1020  68             DISK
0007C     01 ICASHRC   AA  01  68 C5
0008C     02 I        OR       68 CP
0009C     03 I                                          1   5 ACCTNO
0010C     04 I                                          6  25 ACCTNM
0011C     05 I                                         26  300INVNO
0012C     06 I                                         31  360INVDAT
0013C     07 I                                         37  422AMTOWD
0014C     09 I                                         47  512DISCAL
0015C     10 I                                         52  562DISTAK
0016C     11 I                                         57  622AMTPD
0017C     12 I                                         63  680DATPD
0018      01 I                                          1   1 REGIONL1
0019      01 C             DISTAK     SUB  DISCAL     DIFF     62
0020      02 C             DIFF       COMP 1.00                   10  10
0021      03 C             AMTOWD     SUB  DISTAK     NETOWD   62
0022  0204  C             NETOWD     SUB  AMTPD      BAL      62
0023  0301  OCSHRECRGH                    *AUTO
0024  0302  O                                        'CASH RECEIPTS REGISTER'
0025  0303  O          D         01       *AUTO
0026     0350                              REGION       'REGION'
0027  0304  O                             ACCTNO       'ACCOUNT'
0028  0305  O                                   C      'NUMBER'
0029  0306  O                             ACCTNM       'ACCOUNT NAME'
0030  0310  O                             INVNO 3      'INVOICE'
0031  0311  O                                   C      'NUMBER'
0032  0312  O                             INVDATY      'INVOICE'
0033  0313  O                                   C      'DATE'
0034  0314  O                             DATPD Y      'DATE PAID'
0035     14 O                             AMTOWDJA     'AMOUNT'
0036  0402  O                                   C      'OWED'
0037        O                             DISTAK A     'DISCOUNT'
0038  0404  O                                   C      'TAKEN'
0039  0405  O                             AMTPD  A     'AMOUNT'
0040  0406  O                                   C      'PAID'
0041        O                             BAL    A     'BALANCE'
0042  0408  O                                   C      'DUE'
0043  0409  O                       10    DIFF   A     'EXCESS'
0044  0410  O                                   C      'DISCOUNT'
0045  0411  O                                   1      'REGION TOTALS'
0046  0412  O                                   R      'COMPANY TOTALS'

 E N D   O F   S O U R C E

NO  ERRORS IN PROGRAM

 END OF AUTO REPORT PROGRAM
```

Figure 196 (Part 1 of 3). Auto Report Sample Program (EXAUT2)

```
      0010 H      012                                                                                    EXAUT2
0001  0020CFCSHRECRGO    F 132 132      OA      PRINTER                                                  EXAUT2
0002  0030CFCASHRC   IPE F1020  68              DISK                                                     EXAUT2
      0040 I*/COPY F1,EXAUT3                                                                           EXAUT2
      0050 I*/COPY F1,EXAUT4                                                                           EXAUT2
0003  0060CICASHRC   AA  01  68 C5                                                                     EXAUT2
0004  0070CI         OR      68 CP                                                                     EXAUT2
0005  0080CI                                          1   5 ACCTNO                                     EXAUT2
0006  0090CI                                          6  25 ACCTNM                                     EXAUT2
0007  0100CI                                         26  30 0INVNO                                     EXAUT2
0008  0110CI                                         31  36 0INVDAT                                    EXAUT2
0009  0120CI                                         37  42 2AMTOWD                                    EXAUT2
0010  0130CI                                         47  51 2DISCAL                                    EXAUT2
0011  0140CI                                         52  56 2DISTAK                                    EXAUT2
0012  0150CI                                         57  62 2AMTPD                                     EXAUT2
0013  0160CI                                         63  68 0DATPD                                     EXAUT2
0014  0170 I                                          1   1 REGIONL1                                   EXAUT2
0015  0180 C              DISTAK    SUB   DISCAL    DIFF    62                                          EXAUT2
0016  0190 C              DIFF      COMP  1.00                 10  10                                   EXAUT2
0017  0200 C              AMTOWD    SUB   DISTAK    NETOWD  62                                          EXAUT2
0018  0210 C              NETOWD    SUB   AMTPD     BAL     62                                          EXAUT2
0019  0220EC    01                EXSR  A$$SUM                                                         EXAUT2
0020  0230ECL1            AMTOWR    ADD   AMTOW1    AMTOWR  82                                          EXAUT2
0021  0240ECL1            DISTAR    ADD   DISTA1    DISTAR  72                                          EXAUT2
0022  0250ECL1            AMTPDR    ADD   AMTPD1    AMTPDR  82                                          EXAUT2
0023  0260ECL1            BALR      ADD   BAL1      BALR    82                                          EXAUT2
0024  0270ECL1            DIFFR     ADD   DIFF1     DIFFR   82                                          EXAUT2
0025  0280ECSR            A$$SUM    BEGSR                                                              EXAUT2
0026  0290ECSR            AMTOW1    ADD   AMTOWD    AMTOW1  82                                          EXAUT2
0027  0300ECSR            DISTA1    ADD   DISTAK    DISTA1  72                                          EXAUT2
0028  0310ECSR            AMTPD1    ADD   AMTPD     AMTPD1  82                                          EXAUT2
0029  0320ECSR            BAL1      ADD   BAL       BAL1    82                                          EXAUT2
0030  0330ECSR 10         DIFF1     ADD   DIFF      DIFF1   82                                          EXAUT2
0031  0340ECSR                      ENDSR                                                              EXAUT2
0032  0350E0CSHRECRGH  206    1P                                                                       EXAUT2
0033  0360E0         OR       0A                                                                       EXAUT2
0034  0370E0                                         76 'CASH RECEIPTS REGISTER'                      EXAUT2
0035  0380E0                          UDATE Y         8                                               EXAUT2
0036  0390E0                          PAGE  Z       131                                               EXAUT2
0037  0400E0                                       127 'PAGE '                                         EXAUT2
0038  0410E0CSHRECRGH  1      1P                                                                       EXAUT2
0039  0420E0         OR       0A                                                                       EXAUT2
0040  0430E0                                         6 'REGION'                                        EXAUT2
0041  0440E0                                        15 'ACCOUNT'                                       EXAUT2
0042  0450E0                                        29 'ACCOUNT NAME'                                  EXAUT2
0043  0460E0                                        46 'INVOICE'                                       EXAUT2
0044  0470E0                                        56 'INVOICE'                                       EXAUT2
0045  0480E0                                        67 'DATE PAID'                                     EXAUT2
0046  0490E0                                        80 'AMOUNT'                                        EXAUT2
0047  0500E0                                        92 'DISCOUNT'                                      EXAUT2
0048  0510E0                                       105 'AMOUNT'                                        EXAUT2
0049  0520E0                                       118 'BALANCE'                                       EXAUT2
0050  0530E0                                       130 'EXCESS'                                        EXAUT2
0051  0540E0CSHRECRGH  2      1P                                                                       EXAUT2
0052  0550E0         OR       0A                                                                       EXAUT2
0053  0560E0                                        14 'NUMBER'                                        EXAUT2
```

Figure 196 (Part 2 of 3). Auto Report Sample Program (EXAUT2)

```
0054  0570E0                                          45 'NUMBER'                 EXAUT2
0055  0580E0                                          54 'DATE'                   EXAUT2
0056  0590E0                                          79 'OWED'                   EXAUT2
0057  0600E0                                          90 'TAKEN'                  EXAUT2
0058  0610E0                                         104 'PAID'                   EXAUT2
0059  0620E0                                         116 'DUE'                    EXAUT2
0060  0630E0                                         131 'DISCOUNT'               EXAUT2
0061  0640E0CSHRECRGD  1      01                                                  EXAUT2
0062  0650E0                        REGION     3                                 EXAUT2
0063  0660E0                        ACCTNO    14                                 EXAUT2
0064  0670E0                        ACCTNM    37                                 EXAUT2
0065  0680E0                        INVNO 3   45                                 EXAUT2
0066  0690E0                        INVDATY   56                                 EXAUT2
0067  0700E0                        DATPD Y   66                                 EXAUT2
0068  0710E0                        AMTOWDJB  80                                 EXAUT2
0069  0720E0                        DISTAKKB  92                                 EXAUT2
0070  0730E0                        AMTPD KB 105                                 EXAUT2
0071  0740E0                        BAL   KB 118                                 EXAUT2
0072  0750E0                  10    DIFF  KB 131                                 EXAUT2
0073  0760E0CSHRECRGT 12      L1                                                  EXAUT2
0074  0770E0                        AMTOW1JB  80                                 EXAUT2
0075  0780E0                        DISTA1KB  92                                 EXAUT2
0076  0790E0                        AMTPD1KB 105                                 EXAUT2
0077  0800E0                        BAL1  KB 118                                 EXAUT2
0078  0810E0                        DIFF1 KB 131                                 EXAUT2
0079  0820E0                                  67 'REGION TOTALS'                 EXAUT2
0080  0830E0CSHRECRGT 12      LR                                                  EXAUT2
0081  0840E0                        AMTOWRJB  80                                 EXAUT2
0082  0850E0                        DISTARKB  92                                 EXAUT2
0083  0860E0                        AMTPDRKB 105                                 EXAUT2
0084  0870E0                        BALR  KB 118                                 EXAUT2
0085  0880E0                        DIFFR KB 131                                 EXAUT2
0086  0890E0                                  67 'COMPANY TOTALS'                EXAUT2
```

Figure 196 (Part 3 of 3). Auto Report Sample Program (EXAUT2)

EXAMPLE PROGRAM EXAUT2 EXECUTING

CASH RECEIPTS REGISTER PAGE 1

REGION	ACCOUNT NUMBER	ACCOUNT NAME	INVOICE NUMBER	INVOICE DATE	DATE PAID	AMOUNT OWED	DISCOUNT TAKEN	AMOUNT PAID	BALANCE DUE	EXCESS DISCOUNT
1	11243	JONES HARDWARE	27541	2/11/75	2/21/75	23.75	.47	23.28		
1	11352	NU-STYLE CLOTHIERS	27987	2/14/75	2/26/75	87.07		40.00	47.07	
1	11886	MIDI FASHIONS INC	15771	2/04/75	2/14/75	107.22	2.14	105.08		
1	12874	ULOOK INTERIORS	25622	2/09/75	2/23/75	67.95		67.95		
1	18274	STREAMLINE PAPER INC	29703	2/21/75	2/30/75	274.03	2.38	170.55	101.10	
					REGION TOTALS	560.02	4.99	406.86	148.17	
2	23347	RITE-BEST PENS CO	20842	2/18/75	2/20/75	15.80		10.00	5.80	
2	25521	IMPORTS OF NM	29273	2/20/75	2/27/75	797.40	11.93	585.47	200.00	
2	26723	ALRIGHT CLEANERS	19473	2/07/75	2/23/75	462.00		462.00		
2	28622	NORTH CENTRAL SUPPLY	17816	2/05/75	2/22/75	75.97		75.97		
2	29871	FERGUSON DEALERS	27229	2/10/75	2/22/75	61.91		61.91		
					REGION TOTALS	1,413.08	11.93	1,195.35	205.80	
3	30755	FASTWAY AIRLINES	26158	2/06/75	2/19/75	742.72	16.85	725.87		1.90
3	31275	ENVIRONMENT CONCERNS	20451	2/06/75	2/30/75	29.43		15.00	14.43	
3	32457	B SOLE SILOS	27425	2/10/75	2/20/75	110.05		110.05		
3	37945	HOFFTA BREAKS INC	18276	2/06/75	2/23/75	47.23		47.23		
					REGION TOTALS	929.43	16.85	898.15	14.43	1.90
4	42622	EASTLAKE GRAVEL CO	16429	2/05/75	2/23/75	29.37		29.37		
					REGION TOTALS	29.37		29.37		
					COMPANY TOTALS	2,931.90	33.77	2,529.73	368.40	1.90

Figure 197. Output from Auto Report Sample Program (EXAUT2)

Use the auto report option specifications sheet (Figure 198) to select certain options regarding output from auto report. Option specifications are not required in the auto report program. If present, they must appear as the first specifications in the program. If option specifications are not present, auto report assumes all of the options that correspond to blank entries on the specifications sheet (see individual entries for the meanings of the blank entries). Option specifications cannot be within a library member that is copied by a /COPY statement, but they can be cataloged with statements that are compiled using the COMPILE OCL statement. Do not confuse the option specifications (U in position 6) with RPG II control specifications (H in position 6). If RPG II control specifications are not present, either in the auto report source program or within a copied library member (see Part 4, Chapter 6, *Auto Report Copy Specifications*), auto report generates control specifications with blank entries.

SPECIFICATIONS

The following entries on the option specifications sheet are used in the same way as corresponding entries on standard RPG specifications sheets. See Part 1, Chapter 1, *Common Entries* for descriptions of these common entries:

- Page (positions 1-2)
- Line (positions 3-5)
- Program identification (positions 75-80)

Form Type (6)

Enter a U in position 6 of the statement to identify it as an auto report option specification.

Source (7)

Entry	Explanation
C	The generated source program is to be cataloged in the library on disk. (Enter the location and name of the cataloged member in positions 8-18.)
Blank	The generated source program is not to be cataloged.

Use position 7 to request additional output from auto report. You can specify that the source program be cataloged in the library, or not cataloged. In both cases, the generated source program is also written on a disk work file from which it is immediately compiled.

Generated source programs that are cataloged become permanent library members (RETAIN-P). That is, they are not deleted unless you use the DELETE function of the library maintenance program to delete individual members, or the ALLOCATE function to delete the entire library. A library member cataloged by auto report, however, is replaced by another member cataloged under the same name.

As library members, source programs can be compiled using the COMPILE OCL statement or they can be written using the COPY function of the library maintenance program. The generated source program is not cataloged when terminal errors exist in the auto report statements.

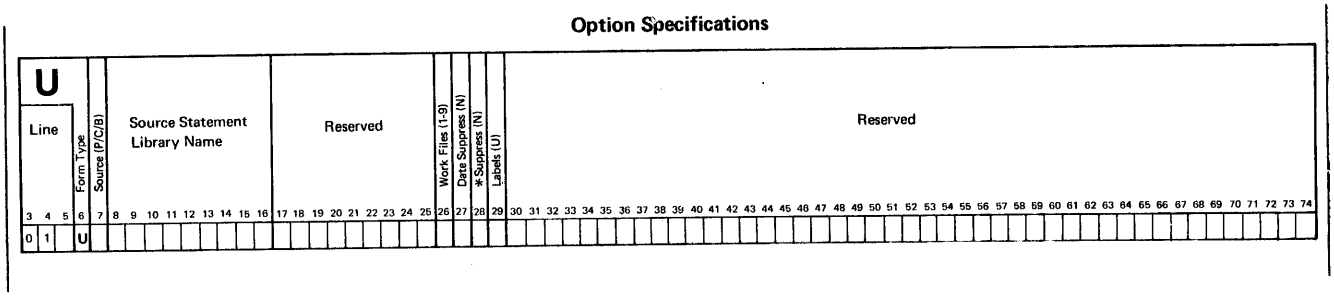


Figure 198. Auto Report Option Specifications

Source Statement Library Name (8-18)

Entry	Explanation
F1, name	The name of the disk on which the library resides followed by the library name of the cataloged source program.

Make an entry in positions 8-18 if the generated source program is to be cataloged in the library (C entry in position 7). Positions 8-9 must contain F1. Position 10 must contain a comma. Positions 11-18 contain the name under which the generated source program is to be cataloged. The name can consist of one to eight characters; the first character must be in position 11 and must be alphabetic (any of the letters A-Z or one of the three special characters #, $, or @. The remaining characters can be alphabetic or numeric.

If the name used to catalog the generated source program is the same as the name of an existing permanent member in the library, the old member is replaced by the new member.

Positions 19-26

Positions 19-26 are not used. Leave them blank.

Date Suppress (27)

Entry	Explanation
N	Suppresses the date and the page number on the first *AUTO page heading line.
Blank	Page number and date are included on the first *AUTO page heading line.

If you do not want the first *AUTO heading line to have the generated date on the left and page number on the right, enter an N in position 27. When these fields are suppressed, the page title and any other fields you specify can occupy the entire line. See **AUTO Page Headings Specifications* for further information on the generated date and page numbers.

***Suppress (28)**

Entry	Explanation
N	Suppresses the asterisk indication from generated total output lines.
Blank	Asterisks are generated for total output lines.

If you do not want asterisks to print beside generated totals, enter an N in position 28. See *Asterisk Indication* in Part 4, Chapter 5, **AUTO Specifications* for rules used in generating asterisk indication.

Positions 29-74

Positions 29-74 are not used. Leave them blank.

The *AUTO page headings function and the *AUTO output function provide simplified methods of describing printed output. These functions of auto report are requested by entering the characters *AUTO in positions 32-37 of a record description specification on the standard RPG output specifications sheet. *AUTO can be entered on a heading, detail, or total specification (H, D, or T in position 15), but not on an exception output specification (E in position 15). Use *AUTO with only one file in a program. That file must be printer file.

In RPG II, output specifications are divided into two general types (Figure 199):

- *Record description specifications* (positions 7-31) describe when and where the output line is to be printed. One record description specification is required for each different type of line to be printed. Only the first record description for a file need contain a filename in positions 7-14.
- *Field description specifications* (positions 23-74) following a record description specification tell when, where, and how each item of data (field or literal) is to be printed on the output record. There can be several field description specifications following a record description specification.

Auto report page headings and auto report output specifications are also divided into the two general categories: record description specifications and field description specifications. However, the entries on these specifications are used differently than in standard RPG II.

The following RPG II output entries are not changed when they are used with *AUTO. See Part 1, Chapter 1, *Common Entries* for descriptions of these entries:

- Page (positions 1-2)
- Line (positions 3-5)
- Form type (position 6)
- Program identification (positions 75-80)

Positions 71-74 must always be blank on auto report output specifications.

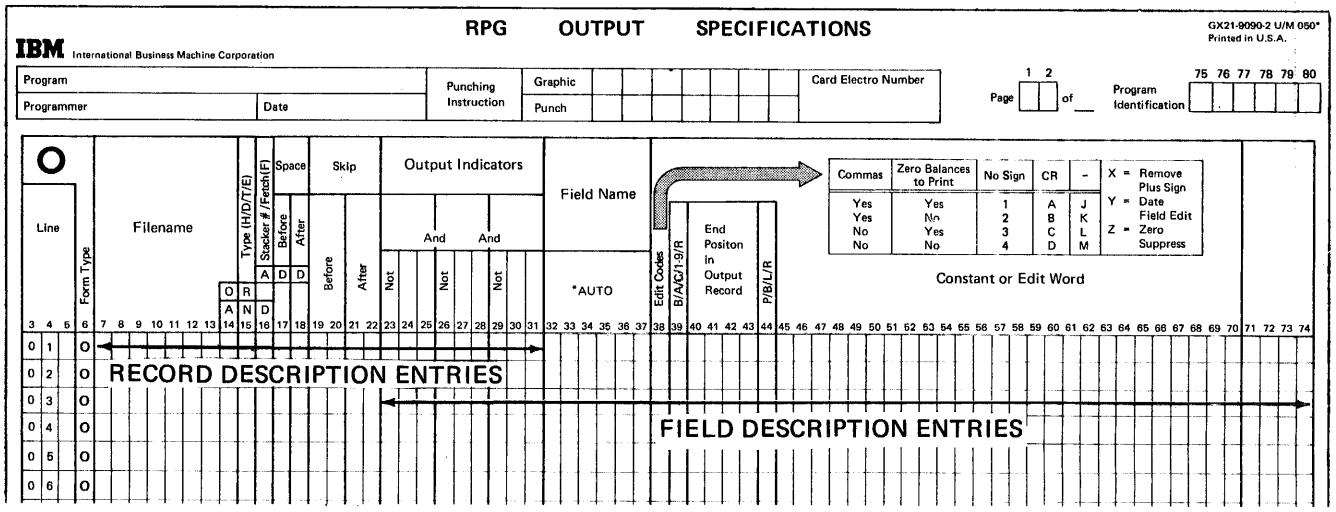


Figure 199. Two Categories of Output Specifications

*AUTO PAGE HEADINGS SPECIFICATIONS

The *AUTO page headings function provides an easy way to produce a page heading at the top of every page of a printed report (Figure 200). Up to five H-*AUTO specifications can be used if a multiple-line page heading is desired. If both normal RPG II heading lines and H-*AUTO lines are specified in combination for a file, they are printed in the order specified by the output specifications. The *AUTO page headings function can be used with only one file per program.

The heading line generated by the first H-*AUTO specification contains a date and page number. (The first heading line can also contain a title. See *Field Description Specifications* for entering a title.) The generated date is left-justified and prints with slashes as follows: mm/dd/yy (unless the format is altered by the RPG II date or inverted print option, positions 19-21 of the control specifications). The generated page number is right-justified and is preceded by the word PAGE. The page number field is four digits long and is zero suppressed. If you do not want the date and page number to print on the first heading line, you can suppress them by entering an N in position 27 of the auto report option specifications sheet.

Note: The auto report function uses one of the unused RPG II PAGE fields (PAGE, PAGE1, PAGE2) for page numbering. If all PAGE fields are used in the program, auto report does not number pages.

Record Description Specifications

Each *AUTO heading (H-*AUTO) record description defines a separate heading line. The record description entries allow the programmer to enter spacing and skipping information and to specify under what conditions the line is printed.

Filename (Positions 7-14)

Enter the name of the printer file on which the heading is to be printed. The filename must correspond to the rules for filenames given in Part 1, Chapter 3.

Type (Position 15)

Enter an H in position 15 on each record description specification line which defines a page heading line. This entry, with the entry *AUTO in positions 32-36, defines this as an H-*AUTO heading specification (Figure 200). Up to five H-*AUTO specifications are allowed.

Position 16

Position 16 is not used in H-*AUTO specifications. Leave it blank.

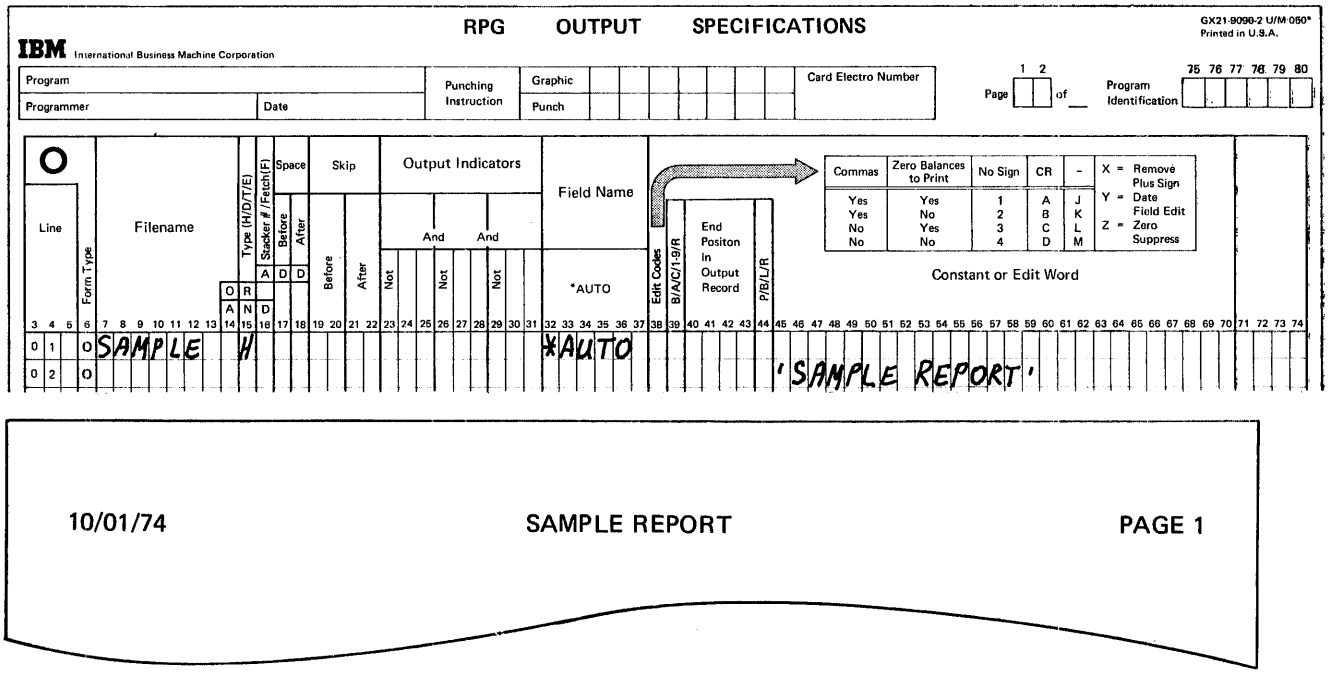


Figure 200. Specifications and Results—*AUTO Heading Line

Space/Skip (Positions 17-22)

Enter your own spacing and skipping values in these positions, according to the rules given in Part 1, Chapter 8. If you do not enter spacing and skipping values in these positions, auto report skips to line 06 before the first line is printed and spaces two after the last H-*AUTO lines is printed. If multiple H-*AUTO lines are used, auto report spaces one after each line except the last. For additional information on generated spacing and skipping values, see *Report Format*.

Output Indicators (Positions 23-31)

On the first H-*AUTO specification, you can either leave positions 23-31 blank or enter your own output indicators according to the rules given in Part 1, Chapter 8.

If you leave these positions blank, auto report causes the corresponding output line to be printed at first page (1P) time in the program cycle when overflow occurs. Thus the heading is printed at the top of each page of the printed report. You can assign indicators to subsequent H-*AUTO specifications. If positions 23-31 are blank on any H-*AUTO specification after the first, that specification is assigned the same indicators as the first.

If an overflow indicator is defined by the file description specifications for the printer file, that indicator conditions the generated heading specifications. Otherwise, an unused overflow indicator is defined for the file on the file description specifications by auto report and conditions the line.

You can use AND and OR specifications with H-*AUTO output indicators if you enter an output indicator on the first specification. Normal RPG II rules for AND and OR lines apply.

**AUTO (Positions 32-36)*

Enter *AUTO in positions 32-36. This entry and H in position 15 of the output specifications sheet (Figure 200) indicate that you are requesting an auto report heading line.

Positions 38-70

Positions 38-70 are not used on the record description line. Leave them blank.

Field Description Specifications

Each H-*AUTO record description specification can be followed by one or more field descriptions. The field description specifications specify the title to be printed on the heading line and describe any other fields and literals to be printed on the line.

Positions 7-31

Positions 7-31 are not used on field descriptions. Leave them blank. (Note that output indicators, positions 23-31, cannot condition a field on an H-*AUTO specification.)

Field Name (Positions 32-37)

Enter	Explanation
Blank	A constant (enclosed in apostrophes) must be entered in positions 45-70. The constant is printed on the heading line.
Field name	Field defined in the program is printed on the heading line.
Table name	A table element is printed on the heading line.
Indexed array name	An array element is printed on the heading line.

Use positions 32-37 to enter a field name, table name, or indexed array name defined elsewhere in the program that you want to print on the heading line. If a name is entered, an edit word, not a constant, can be entered in positions 45-70. A constant must be entered in positions 45-70 if positions 32-37 are blank.

If output indicators (positions 23-31) are left blank on the record description specification, auto report conditions all fields and table/array elements included on the heading line with N1P in positions 23-25. Therefore, the field or table/array element does not print on the first page. (If printed on the first page, the field may not contain meaningful data, since the first record is not read.) N1P is not generated for the following RPG II reserved words: PAGE, PAGE1, PAGE2, UDATE, UDAY, UMONTH, UYEAR.

For information on formatting and centering *AUTO heading lines, see *Report Format.*

Edit Codes (Position 38)

You can enter an edit code in position 38 if a numeric field, array element, or table element is named in columns 32-37. If you use an edit code, positions 45-70 must be blank unless asterisk protection or a floating dollar sign is specified. If position 38 is blank, no editing is done by auto report unless an edit word is used.

Blank After (Position 39)

Enter a B in position 39 if you want a numeric field reset to zeros after it is printed, or if you want an alphameric field reset to blanks after it is printed on the heading line.

Positions 40-44

Positions 40-44 are not used with *AUTO heading specifications. Leave them blank. For information on the positioning of fields and literals in the title line and centering of heading lines in relation to the body of the report, see *Report Format.*

Constant or Edit Word (Positions 45-70)

Entry	Explanation
Constant	Title or other constant (enclosed in apostrophes) that is to appear on the printed line.
Edit word	The edit pattern used to edit the numeric field named in positions 32-37 of the same field description line.
Blank	Positions 32-37 contain the name of a field which either is not edited or is edited by an edit code.

Use positions 45-70 to specify the title and other information that is to appear on the output line and to edit numeric fields that are to appear on the line. Rules for specifying constants and edit words are identical to those given in Part 1, Chapter 8, except that no end positions can be specified.

For information on the positioning of fields and constants in the title line and centering of heading lines in relation to the body of the report, see *Report Format.*

*AUTO OUTPUT SPECIFICATIONS

Detail reports (where a line is printed for each individual record that is read) and group printed reports (where only totals are printed) can be specified using the *AUTO output function alone or in combination with standard RPG II specifications. The *AUTO output function generates totals and formats columns and column headings.

A single detail or total *AUTO record description (D/R-*AUTO) specification and its associated field description specifications can specify:

- Up to three lines of column headings to appear above a field.
- Accumulation of several levels of totals, including a final total (known as *total rolling*).
- Generation by auto report of end positions for column headings and fields.
- Generation by auto report of the K edit code for numeric fields.
- Fields or constants to be printed next to generated totals.

This section describes the *AUTO output record description specifications and the four types of field description specifications that can be associated with it. The four types are distinguished by entries in position 39. The remaining entries on a field description specification have different meanings depending on the entry in position 39.

The valid entries in position 39 of the field description specifications and their meanings are:

- *Blank or B:* Indicates the associated field or constant appears on the detail line.
- *A:* Indicates the associated numeric field is printed on the detail line and accumulated. A total is printed for each control level defined in positions 59-60 of the input specifications for the program. A final total is also printed (LR).
- *C:* Indicates the associated constant is printed on the second or third line of column headings.
- *1, 2, 3, 4, 5, 6, 7, 8, 9, R:* Indicates the associated field or constant appears on the total line generated for the respective control level indicator (L1-L9, LR).

See *Group Printing* for the effect of these entries in a group printed report.

Note: Examples of four types of auto report field description specifications are found in Part 4, Chapter 2, *How To Use RPG II Auto Report*; Part 4, Chapter 3, *Sample Program* and under *Group Printing* later in this chapter.

Record Description Specifications

An auto report record description specification must contain the entry *AUTO in positions 32-36. *AUTO can appear only on a record description specification. This entry indicates that the record description and the following field descriptions are redefined according to their use by auto report.

Filename (Positions 7-14)

Enter the name of the printer file on which the report is to be printed. This must be the same file named on H-*AUTO specifications, if any. The filename must correspond to the rules for filenames given in Part 1, Chapter 3.

Type (Position 15)

Entry	Explanation
D	The auto report specifications describe a report containing detail lines.
T	The auto report specifications describe a report containing total lines, but no detail lines (group printed report).

Enter a D in position 15 and *AUTO in positions 32-36 if you want auto report to generate a report that contains detail lines. The field description specifications associated with the D-*AUTO record description specify:

- Fields to appear on the detail line
- Column headings
- Total rolling
- Constants to appear on total lines

Examples of D-*AUTO specifications and reports are found in Part 4, Chapter 2, *How To Use RPG II Auto Report.*

Enter a T in position 15 and *AUTO in positions 32-36 if you want auto report to generate a group printed report (see *Group Printing*).

Only one detail or total *AUTO (D/T-*AUTO) record description specification can be used in a program.

Fetch Overflow (Position 16)

Enter an F in position 16 if you want to specify fetch overflow. The normal rules for fetch overflow apply. See Part 1, Chapter 8.

When used with the *AUTO output function, fetch overflow applies only to the detail line. If group printing is specified (T in position 15), fetch overflow applies to the lowest level total line to be printed.

Space/Skip (Positions 17-22)

Enter spacing and skipping values in positions 17-22 according to the normal RPG II rules. Entries specified apply only to the detail line generated by a D-*AUTO specification or the first total line generated by a T-*AUTO specification.

Leave positions 17-22 blank if you want single spacing to be done after each detail line printed or, if group printing is specified, after the first total line printed. For information on spacing and skipping for generated column heading and total lines, see *Report Format.*

Output Indicators (Positions 23-31)

Enter any valid output indicators in positions 23-31 to condition the detail or group print line generated by this *AUTO specification. If these positions are left blank on a D-*AUTO specification, the generated detail line is conditioned by N1P. Therefore, it is not printed at first page (1P) time in the RPG II program cycle. If these positions are left blank for a T-*AUTO specification, the first generated total line is conditioned by the lowest control level indicator defined in the program. (See *Group Printing* for additional information about the use of this entry with a T-*AUTO specification.) You can use AND and OR specifications with *AUTO output indicators if you enter an output indicator on the first record description specification. Normal RPG II rules for AND and OR lines apply.

Indicators you enter in positions 23-31 of the record description specification (and its associated AND/OR lines) apply only to the detail line generated by a D-*AUTO specification or the group print line (lowest level total specification) generated by a T-*AUTO specification.

If column headings are specified in the field description specifications that follow this *AUTO record description, they are conditioned by one of the following:

- The same indicators that were specified for the first H-*AUTO specification.
- The first page (1P) indicator in an OR relationship with the overflow indicator specified for the file on the file description specifications. If no overflow indicator is specified, auto report defines an unused overflow indicator and uses it to condition the lines.

Restriction: If you specify N1P on a D-*AUTO record description specification which is followed by field description specifications for totaling fields (A in position 39), the calculations generated for the totaling fields are also conditioned by N1P. This causes a terminal diagnostic in the RPG II Compiler.

**AUTO (Positions 32-36)*

To indicate that you are using auto report, enter *AUTO in positions 32-36 on the record description line. Position 15 must contain a D or a T in this case to indicate a detail or total *AUTO specification. Only one D/R-*AUTO specification can be used in a program.

Positions 38-70

Positions 38-70 are not used on a D/T-*AUTO record description specification. Leave them blank.

Field Description (Blank or B in Position 39)

D-*AUTO and T-*AUTO field description specifications containing a blank or B in position 39 describe:

- An alphameric field such as an item description.
- A numeric field that is not totaled.
- A constant.
- A field with a literal to be used as a column heading (Figure 201).

A field named on the line (or a constant, when no field is named) following a D-*AUTO record description specification is printed only on the detail report line. If the field (or constant, when no field is named) on the line follows a T-*AUTO record description, it appears only on the first total line generated.

Positions 7-22

Positions 7-22 are not used on the field description lines. Leave them blank.

Output Indicators (Positions 23-31)

Enter any valid RPG II output indicators in positions 23-31 or leave them blank. If these positions are left blank, the field (or constant, when no field is named on the line) is printed on each detail line conditioned by the indicator. When group printing is specified (T-*AUTO specification), the field (or constant, when no field is named on the line) is printed each time the lowest level total line is printed. If a column heading is specified in positions 45-70 to appear over a field named in positions 32-37, the column heading is not affected by output indicators entered in positions 23-31.

Field Name (Positions 32-37)

Enter a field name, indexed array name, table name, or blanks in positions 32-37. If you leave positions 32-37 blank, you must enter a constant in positions 45-70 of the same field description specification. If a field name, indexed array name, or table name is entered, the value of the field or element is printed on the detail line (on the first total line, if group printing is specified).

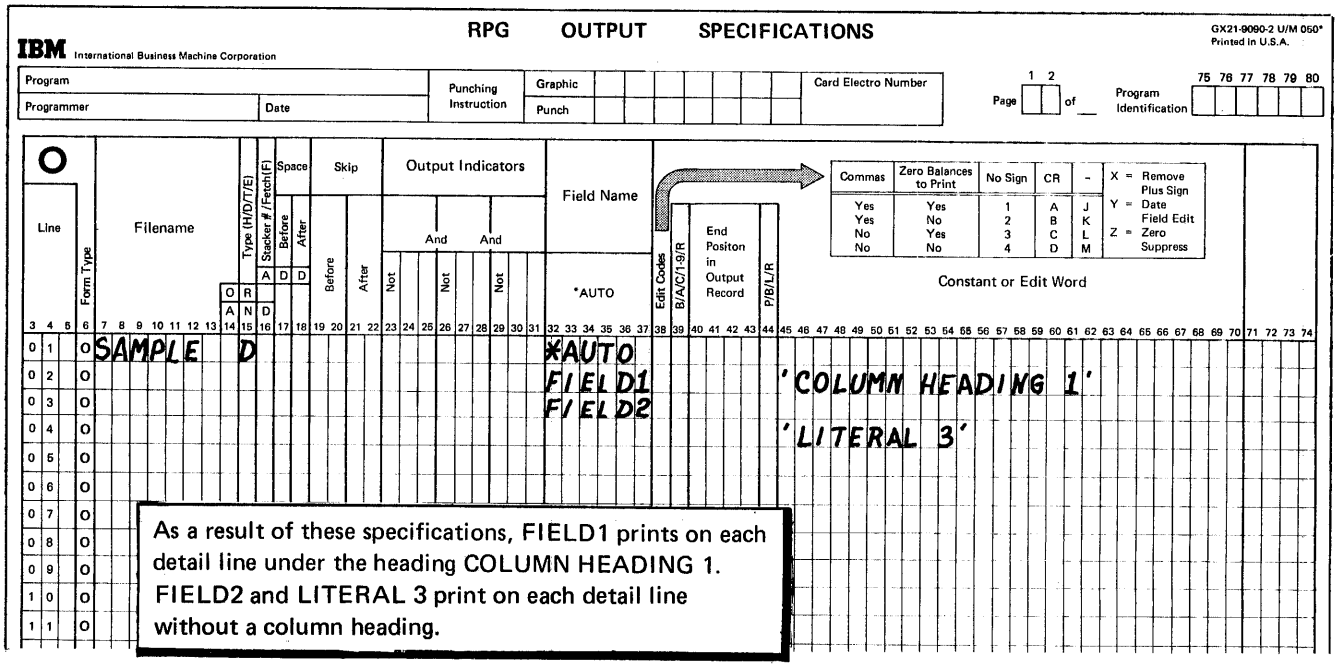


Figure 201. Auto Report Field Description Specifications (Blank in Position 39)

Edit Codes (Position 38)

You can enter a valid RPG II edit code in position 38 if positions 32-37 contain the name of a numeric field, a numeric array element, or a numeric table. This position must be blank for alphameric fields and table/array elements, and for literals. If position 38 is left blank on a field description line for a numeric field or table/array element, a K edit code is provided by the auto report program. The K edit code causes a numeric field or element to be printed with commas and a decimal point, such as 3,489.13. It also causes zero suppression; zero balances are not printed and negative balances are printed with a minus sign on the right.

Blank After (Position 39)

Entry	Explanation
Blank	Field is not to be reset to zeros or blanks after printing.
B	Numeric field is reset to zeros after it is printed; alphameric field is reset to blanks.

Enter a B in position 39 when alphameric or numeric fields, array elements, or table elements are to be reset to zeros or blanks after they are printed. Blank after cannot be used for constants. This entry applies only to the detail line (or the first total line, if group printing is specified).

End Position in Output Record (Positions 40-43)

Either leave positions 40-43 blank or enter the print position of the rightmost character of the field (constant, if no field is named in positions 32-37) to be printed. If this entry is blank, auto report generates end positions for fields, constants, and column headings. See *Report Format* for additional information and considerations.

Position 44

Position 44 is not used, since packed and binary data cannot be specified. Leave this position blank.

Constant (Positions 45-70)

Enter a constant or blanks in positions 45-70 when position 39 contains blank. Constants are enclosed in apostrophes according to the normal RPG II rules for coding constants.

If these positions are left blank, a field name, indexed array name, or table name must be entered in positions 32-37. Column heading continuation lines may follow this field description line, but the first line of the printed column heading will be blank. See *Field Description (C in Position 39)*.

If a constant is entered in these positions along with a field name in positions 32-37, the constant is printed on the first column heading line over the field value. When a column heading is used, the length used to space the column on the report is the greater of the longest column heading length or the field length, adjusted for editing. See *Report Format* for additional information on how columns and fields are centered and spaced by auto report.

If a constant is entered in positions 45-70 and field name (positions 32-37) is blank, the constant is printed each time the detail report line is printed. In group printing, the constant is printed each time the first generated total line is printed.

Field Description (A in Position 39)

Enter an A in position 39 of a field description specification following a D/T-*AUTO specification if you want auto report to accumulate and print totals for the field named in positions 32-37 (Figure 202). As many levels of totals are printed as you have defined in the control level entry (positions 59-60) on input specifications. A final total (LR) is also printed. (This process is called *total rolling*.)

If group printing is specified and a control level indicator higher than the lowest defined control level is specified in positions 23-31 on the record description specification, totals are generated for the indicator entered, all higher defined indicators, and LR.

The total output record generated by auto report as a result of entering an A in position 39 of a field description specification are conditioned by the associated control level indicator defined in the input specifications. One total output record is generated for each control level indicator defined in the program.

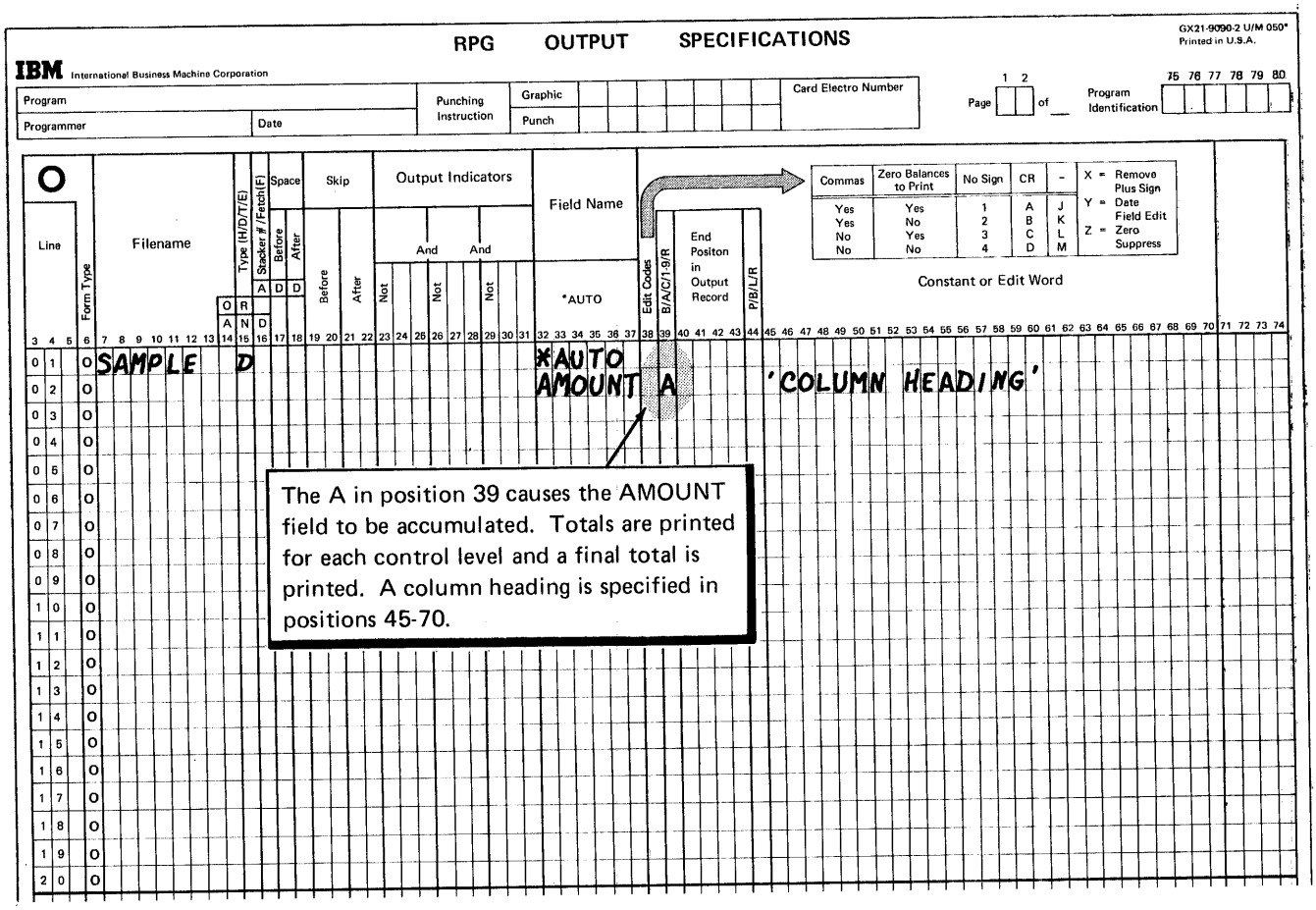


Figure 202. Describing a Field That is to be Accumulated

Generated Total Fields

When an A is specified in position 39 of a detail or total *AUTO field description specification, auto report generates and names total fields to be used in accumulating the required levels of totals. Field names for the total fields are generated by auto report based on the name in positions 32-37 of the A-type field description. Names are generated in the following way:

- If the specified field name has fewer than six characters, one character is added to the name to create a name for the total field. The added character is 1-9 or R, corresponding to the total indicators L1 through L9 and LR, respectively. For example, if ITEM is the specified field name and all nine control levels are defined, the generated field names are: ITEM1, ITEM2, . . . ITEM9, and ITEMR
- If the specified field name has six characters, the last character is replaced by one of the characters 1 through 9, or R. For example, if AMOUNT is the specified field name and all nine control levels are defined, the generated field names are: AMOUN1, AMOUN2, . . . AMOUN9, and AMOUNR

Total fields are generated and named for all control level indicators defined in the program and for LR. (For an exception to this rule, see *Group Printing, Example 1*.) For example, if L1 and L3 are assigned to control fields on the input specifications and the field QTY is specified, three total fields, QTY1, QTY3, and QTYR are generated and named by auto report. All total fields generated for the same level, such as QTY1 and AMOUN1, are printed on the same total line, and that line is conditioned by the corresponding control level indicator.

Generated total fields are two digits longer than the original field. For example, if the field QTY is defined with a length of three, QTY1, QTY3, and QTYR all have lengths of five. The number of decimal positions remains the same in the generated fields. You can define a field name previously in a program which is the same as a generated field name, giving that field whatever length and number of decimal positions you want. If you do this, the generated field is assigned the previously defined length and number of decimal positions (if the previous field is numeric).

Considerations

Generated field names can be referenced in RPG II specifications that are included in the program. The programmer must be aware, however, that the use of generated fields in this way may interfere with the automatic accumulation of totals performed by auto report.

Field names ending in 1-9 or R should not be used in an auto report program that accumulates totals, since auto report generates total fields ending in those characters. This is especially important for 6-character field names, since total fields are formed by replacing the last character with 1-9 or R. No field name can be used more than once with an A in position 39. Also, if a 5 or 6-character field name is specified with an A in position 39, a second 5 or 6-character field name in which the first five characters are identical cannot be specified with an A in position 39. For example, if the following four field names are specified with an A in position 39 in an auto report specification, all but the first are invalid:

FIELD

FIELDX — Invalid because the first five characters duplicate the first five characters of the first field.

FIELDY — Invalid for the same reason as FIELDX.

FIELD — Invalid since it is a duplicate of the first field.

Positions 7-22

Positions 7-22 must remain blank on the field description lines.

Output Indicators (Positions 23-31)

Enter any valid RPG II output indicators in positions 23-31 or leave them blank. If you leave these positions blank, the field described is printed on each detail line. If you enter indicators in positions 23-31, the field is printed only when the conditions represented by those indicators are met. Leave these positions blank for group printing.

If a column heading is specified in positions 45-70 to appear over a field named in positions 32-37, the column heading is not affected by output indicators entered in these positions. Also, output indicators specified when position 39 contains an A do not affect field description specifications generated for totals.

Output indicators specified on an A-type field description specification following a D-*AUTO specification condition the calculations generated for the field. If the A-type field description follows a T-*AUTO specification, however, a specified indicator does not condition calculations generated for the field.

Field Name (Positions 32-37)

When you enter an A in position 39, you must enter the name of a numeric field that is to be accumulated in positions 32-37. These positions cannot identify an array, array element, or table. The field named is printed on each element, or table. The field named is printed on each detail line of the report. If group printing is specified, the total field for the lowest control level indicator defined (L1, L2, . . . L9, LR, in that order) is printed on the generated total line. (For an exception to this rule, see *Group Printing, Example 1*.) Totaling for any particular field by entering an A in position 39 can be specified only once in each program.

To generate calculation and output specifications that accumulate and print the various levels of totals required, auto report creates and names additional totaling fields. Names for the fields are constructed based on the field name specified in these positions according to a set of rules (see *Generated Calculations*).

Edit Codes (Position 38)

Enter an edit code in position 38 or leave it blank. If you leave this position blank, a K edit code is generated for the field named in positions 32-37. This causes the field to be edited with commas and a decimal point, such as 1,234,567.89. The field is also zero suppressed. Zero balances are not printed; negative balances are printed with a minus sign on the right. The edit code you enter, or the generated K edit code, applies to all generated total fields as well as to the field named in columns 32-37.

Position 39

Enter an A to indicate that totals are to be accumulated for the field named in columns 32-37 of this field description. A total is printed for every control level indicator defined in the input specifications and for LR. When position 39 contains an A, positions 32-37 must contain the name of a numeric field. Positions 45-70 can contain a constant to be used as the first line of a column heading. (See *Generated Specifications* for additional information.)

Note: When the lowest defined control level indicator used to condition a T-*AUTO specification is higher than the lowest control level indicator defined in the input specifications, only the total lines corresponding to the lowest defined control level indicator used to condition the T-*AUTO specification, the higher defined control levels, and LR are generated. See *Group Printing.*

Resetting Total Fields to Zero: When position 39 contains an A, the auto report program generates a B (blank after) in position 39 of all the detail and total field description specifications generated from the field name specified. Thus, the field value for the specified name and any generated field name are reset to zero after the field value is printed.

If group printing is specified, auto report generates a calculation to reset the specified field to zero on each cycle. This prevents the same value from being accumulated more than once. An unconditioned total calculation operation (Z-ADD) sets the field value to zero. This calculation is the first total calculation in the generated RPG II source program.

Asterisk Indication: To indicate that a printed line is a generated total line, asterisks are printed on the line to the right of the highest end position generated from the D/T-*AUTO specification. One asterisk is printed to the right on the lowest level total line generated. One additional asterisk is printed on each higher level line including the final total.

For example, suppose L1 and L3 are defined control level indicators in a program. One asterisk is printed to the right of the L1 line, two asterisks are printed on the L3 line, and three are printed on the LR line. As many as 10 asterisks are printed on the LR line if all nine control level indicators are defined in the program.

If you do not want asterisk indication on your report, you can suppress the generation of asterisks on total lines by entering an N in position 28 of the auto report option specifications sheet.

End Position in Output Record (Positions 40-43)

Enter the print position of the rightmost character of the field to be printed or leave these positions blank. If this entry is blank, auto report generates end positions for fields and column headings. See *Report Format* for additional information and considerations.

Position 44

Position 44 is not used with auto report, since packed and binary data cannot be used. Leave these positions blank.

Constant (Positions 45-70)

Either leave positions 45-70 blank or enter a literal. Do not enter an edit word; editing is accomplished by an edit code. If a literal is entered when position 39 contains an A, the literal becomes the first line of the column heading over the accumulated field.

If these positions are left blank, the first line of the column heading is blank, but column heading continuation lines can specify the second and third line of the column heading. See *Field Description (C in Position 39).* See *Report Format* for information on how column headings and fields are centered and spaced by auto report.

Field Description (C in Position 39)

*AUTO field descriptions can specify a second and third column heading line by entering a C in position 39. At times you may want more information in a column heading than can be contained on one line. Auto report enables you to specify the second and third lines of column headings by simply specifying the literals to appear on those lines. No additional heading output lines need be coded; no end positions need be calculated. The special field description specification that allows you to do this is identified by a C in position 39 (Figure 203).

Positions 7-38

Positions 7-38 must be blank on a field description that has a C in position 39.

Position 39

Enter a C. One or two C-type specifications can follow a field description specifications that has an A, B, or blank in position 39 and an entry in positions 32-37. The first C-type specification causes a second column heading line to be generated. The second C-type specification causes a third column heading line to be generated (Figure 203).

Positions 40-44

Positions 40-44 must be blank on a C-type field description specification.

Constant (Positions 45-70)

Enter a constant, up to 24 positions including blanks, enclosed in apostrophes. The constant becomes the second or third line of column headings, depending on whether it is on the first or second C-type specification. If two or three column heading lines are specified, the shorter literals are centered on the longest.

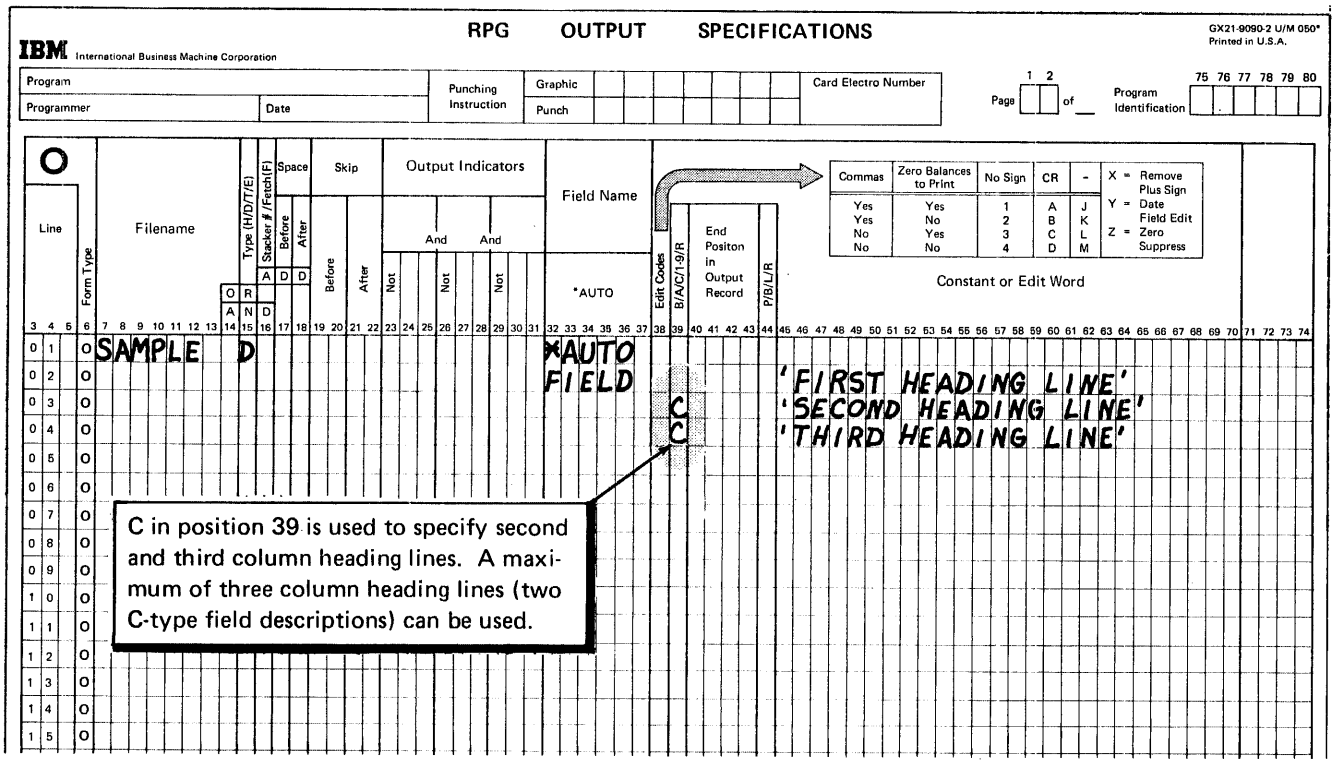


Figure 203. Specifying Second and Third Column Heading Lines

Field Description (1-9 or R in Position 39)

Enter 1, 2, 3, 4, 5, 6, 7, 8, 9, or R in position 39 of a field description to specify a field or constant to be printed on a specific total line.

Auto report allows you to print other information on generated total lines in addition to the generated totals resulting from A-type field descriptions. The value entered in position 39 corresponds to the level of the total line on which the information is to be printed (the corresponding control level must be defined in positions 59-60 in the input specifications). For example, a 3 in position 39 indicates the information is printed on the L3 total line; an R indicates the information appears on the final total, or LR line (Figure 204). Fields and constants specified in this way are printed to the left of the leftmost generated total on the line. See *Report Format* for exact placement.

This type of field description can print information such as DISTRICT TOTAL, GRAND TOTAL, or other literal information. It can also print a field and specify an edit word, floating dollar sign, or asterisk protection for the field.

If none of the *AUTO output fields are defined with an A in position 39, then 1-9 or R cannot be used in position 39. In group printing, only specify numbers that are higher than the lowest control level indicator used to condition the T-*AUTO specification. If the T-*AUTO specification is not conditioned by a control level indicator, use only numbers that are higher than the lowest control level defined in positions 59-60 on the input specifications.

Positions 7-31

Positions 7-31 must be blank on a field description line with 1-9 or R in position 39.

Field Name (Positions 32-37)

Enter the name of a field, an indexed array name, or a table name. The corresponding field or element value prints on the total line indicated by the entry in position 39. If you leave positions 32-37 blank, you must enter a constant in positions 45-70.

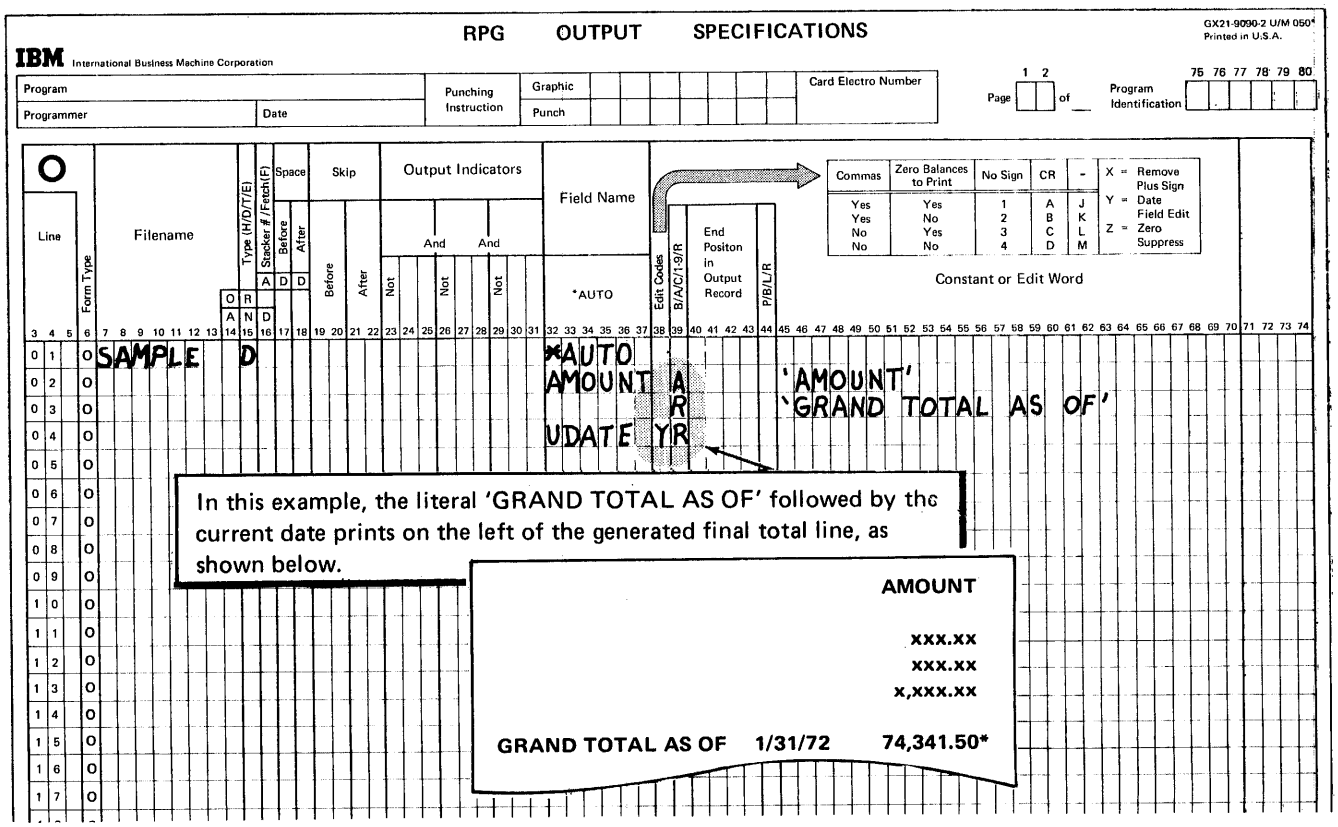


Figure 204. Specifying a Literal and a Field to Print on a Generated Total Line

Edit Code (Position 38)

Enter an edit code in position 38 to edit a numeric field named in field name (positions 32-37) or leave position 38 blank. If position 38 is left blank, an edit word can be entered in positions 45-70. If position 38 is blank, no edit code is assumed by auto report.

Position 39

Enter 1, 2, 3, 4, 5, 6, 7, 8, 9, or R. These entries correspond to the indicators L1, L2, . . . L9, and LR. The entry identifies a specific total line on which the field or literal described is to be printed. The entry in position 39 must correspond to a control level that is defined by the input specifications. In group printing, the entry in this position must be higher than the control level of the first total line generated.

End Position in Output Record (Positions 40-43)

Do not make an entry in positions 40-43 on field description specifications with 1-9 or R in position 39. See *Report Format* for additional information and considerations.

Position 44

Leave position 44 blank.

Constant or Edit Word (Positions 45-70)

Leave positions 45-70 blank, or enter a constant or edit word.

If field name (positions 32-37) on this specification line contains an entry, then positions 45-70 can contain any of the following:

- Blanks, if no editing is needed for the field or if the field is already edited by an edit code in position 38.
- Edit word, if special editing is desired.
- Floating dollar sign or asterisk protection entry used with an edit code.

Positions 45-70 cannot contain a constant when field name contains an entry. However, when field name is blank, positions 45-70 must contain a constant.

GROUP PRINTING

In group printing, data is summarized for a group of input records and only totals are printed on the report. Totals can have subtotals with a final total, or only a final total.

Specifications

To specify group printing using auto report, enter a T under type (position 15) with *AUTO in positions 32-36. You can enter a control level indicator under output indicators (positions 23-31).

When a T-*AUTO specification is used, a line is not printed for each individual record that is read, but only after a complete control group is read. This is shown in the two group printing examples which follow.

In example 1, a total line is generated for the lowest control level indicator specified in positions 23-31 and for all higher control level indicators defined and LR.

In example 2, however, no control level indicators are specified in positions 23-31. In this case a total line is generated for the lowest control level indicator defined by input specifications (positions 59-60) in the program and for all higher control level indicators defined and LR.

Fields and literals defined by field description specifications which have a blank or B in position 39 and follow a T-*AUTO record description are printed on the lowest level total line. Fields defined with an A in position 39 are not printed on the total lines, but the total fields created by auto report. *Generated Calculations* are printed on their associated total lines. Continued column headings (C in position 39) and total-indicated fields (1-9 or R in position 39) can also be specified by field descriptions following a T-*AUTO record description.

Output indicators can be entered in positions 23-31 of a field description specification following a T-*AUTO record description if position 39 of the field description specifications contains a blank or a B. If output indicators are used in a field description which has an A in position 39 following a T-*AUTO specification, those indicators are ignored by auto report. Output indicators cannot be used in a field description that contains C, 1-9, or R in position 39.

Example 1

In this example, a group printed report is prepared to show sales totals for any company. The report is similar to those prepared in Part 4, Chapter 2, *How To Use RPG II Auto Report*. However, in this example, only the totals for each region and the entire company are shown; individual items (detail lines) are not listed.

A disk summary file, DISKSUM, is also produced by this program. The summary file contains a summary record of the sales data for each branch.

Figure 205 shows the file description and input specifications for the program. BRANCH and REGION are defined as control fields.

Figure 206 shows the output specifications and the group printed report. Since the T-*AUTO specification is conditioned by L2, only the totals for REGION (L2) and for the entire company (LR) are printed on the report. The totals for BRANCH (L1) are not printed.

The output specifications for DISKSUM (Figure 206) illustrate use of standard RPG II output specifications in the same program with *AUTO specifications. The output record described is written on the disk file, DISKSUM, when there is an L1 control break (BRANCH field changes).

Since the T-*AUTO specification is conditioned by L2, auto report does not generate fields for the L1 control level, although L1 is defined by the input specifications. Therefore, standard RPG II calculation specifications must be used to calculate the L1 totals. The L1 total fields that are written on the DISKSUM file (SOLDQ1, SOLDV1, and VALUE1) must be defined in these calculations.

File Description Specification

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/I/K	I/D/T or 2	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	File Addition/Unordered	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	File Condition U1-U8	A/U	R/U/N
02	F	SALES	I	P			F	473	43							DISK											
03	F	PRINTER	O				F	120	120							PRINTER											
04	F	DISKSUM	O				F	250	25							DISK											

Continuation Lines: K, Option, Entry

RPG INPUT SPECIFICATIONS

IBM International Business Machine Corporation

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

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

Line	Form Type	Filename	Sequence (AND)	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Field
01	I	SALES	AA			01																				
02	I																				1	7		ITEMNO		
03	I																				8	9		BRANCHL1		
04	I																				10	10		REGIONL2		
05	I																				11	25		DESC		
06	I																				26	270		SOLDQV		
07	I																				28	342		SOLDVA		
08	I																				35	360		ONHAND		
09	I																				37	432		VALUE		

L1 and L2 are the defined control levels.

Figure 205. File Description and Input Specifications For the Group Printed Reports in Example 1 and Example 2

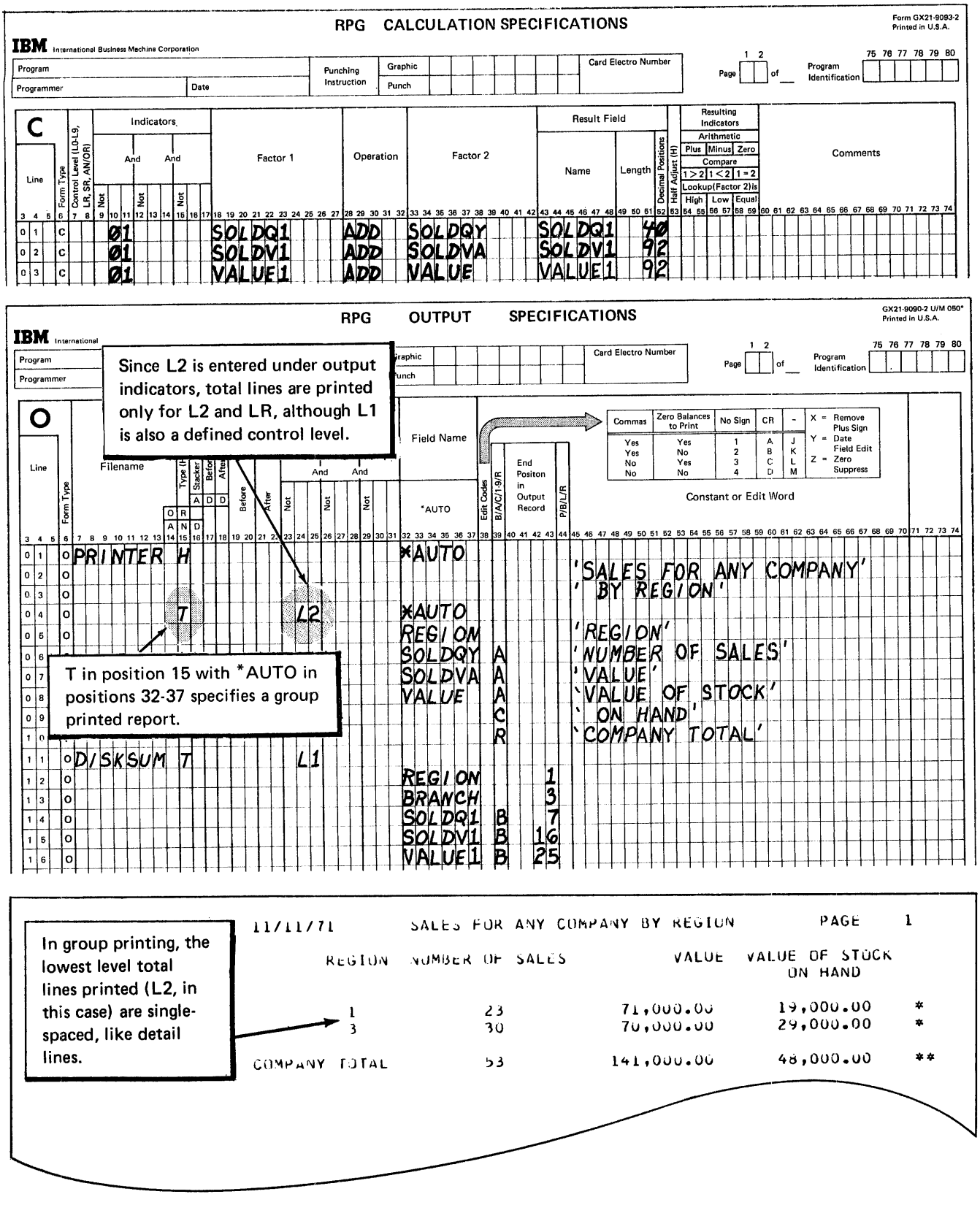


Figure 206. Using *AUTO to Produce a Group Printed Report Showing Region and Final Totals

Example 2

In this example, a group printed report similar to *Example 1* is printed using the same file description and input specifications (Figure 205). Figure 207 shows that by not conditioning the T-*AUTO specification, totals are printed for all defined control levels and LR.

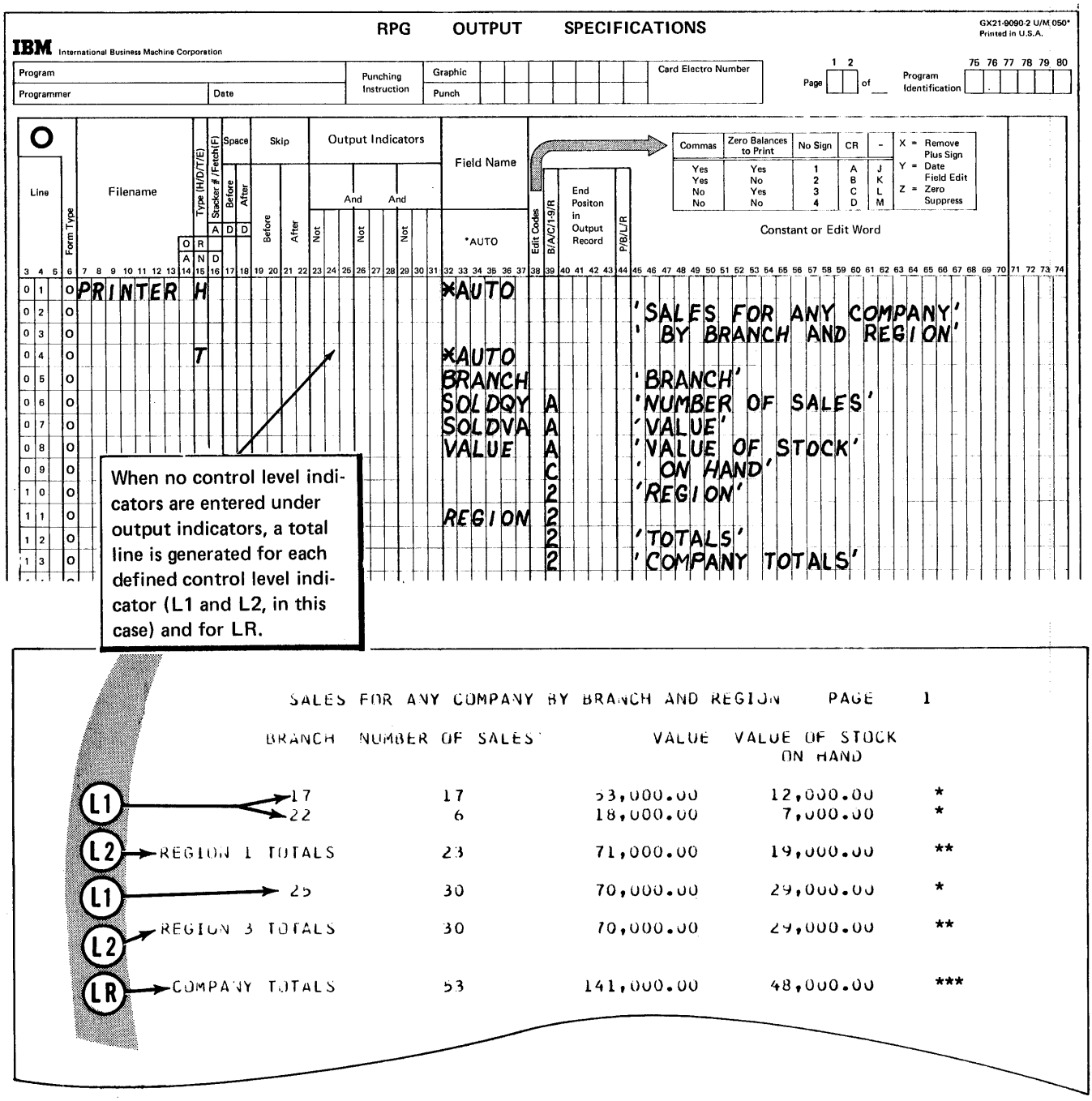


Figure 207. Using *AUTO to Produce a Group Printed Report Showing Branch, Region, and Final Totals

Chapter 6. Auto Report Copy Specifications

The auto report copy function provides a way to include cataloged RPG II source specifications into an RPG II program. The source specifications that are included must reside as a library member on disk. The library member is created using the library maintenance disk utility program. Using the copy function, you can include source specifications that are identical or nearly identical in several different programs and thus reduce the need to repeatedly code specifications that are used in several programs. For example, if file description and input specifications for a particular file are very similar in different programs, these specifications can be placed in the library using the library maintenance program and included in any program in which the file is used by means of the copy function.

Auto report specifications and any valid RPG II specifications, including tables and arrays, can be copied in this manner. The auto report option specifications and other copy statements cannot be copied. See Part 4, Chapter 2, *How To Use RPG II Auto Report* for an example of using the copy function.

The specifications included in an auto report program by means of the copy function are initially placed in the program immediately following the /COPY statement. When all specifications are copied from the library, the entire auto report program is sorted into the order required by the RPG II Compiler (see *Order of Generated Specifications*).

/COPY STATEMENT SPECIFICATIONS

You request the copy function by means of a special statement, the /COPY statement, that is included in the auto report program. This statement identifies the library entry that contains the RPG II specifications to be included in the RPG II source program generated by auto report. /COPY statements must follow the auto report option specifications and they must precede source tables and arrays (file translation tables, alternate collating sequence tables, and compile-time tables and arrays).

The format of the /COPY statement is:

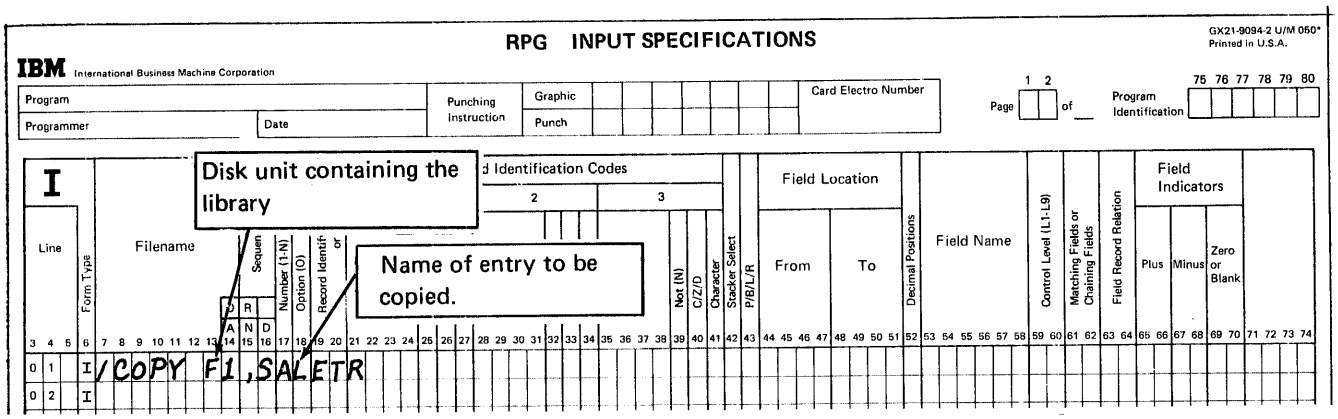
Position	Entry
1-5	Page and line number indicating the placement of the statement in the sequence of auto report source specifications.
6	This position can contain any entry except H or U, or can be blank.
7-11	Enter the characters /COPY.
12	Blank.
13-23	Identifies the library entry to be included. Positions 13-14 contain F1. Position 15 contains a comma. Positions 16-23 contain the name, up to eight characters long.
24-49	Blank.
50-80	Enter any information or comments. The contents of these positions are not read by auto report.

Figure 208 shows an example of the /COPY statement.

MODIFYING COPIED SPECIFICATIONS

You can include statements among your auto report specifications to modify file description and input field specifications as they are copied from the library. No other types of specifications can be modified.

/COPY modifier statements from the source program which add, change, or delete entries on cataloged input field specifications are identified by an X in print position 6 of the auto report listing.



Note: It is convenient to code the /COPY statement on the input sheet if input specifications are to be modified as they are copied. (See index entry: *modifying input specifications*.)

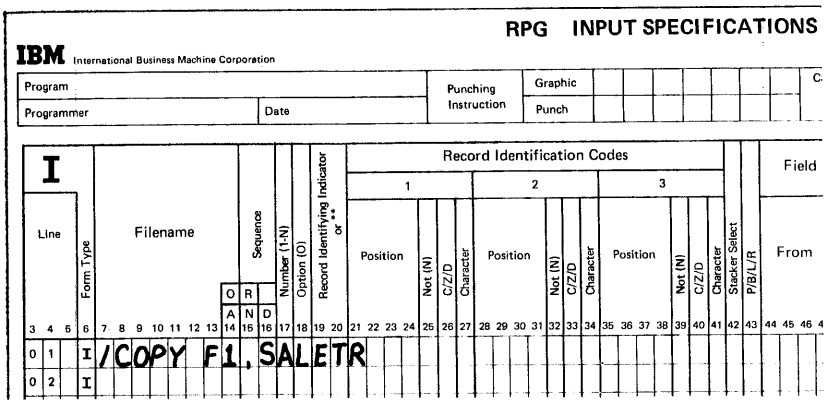
Figure 208. Example of the /COPY Auto Report Statement

Modifying File Description Specifications

To modify a file description specification that is copied from the library, enter the filename in positions 7-14 of a file description specification (F in position 6). Then make only those entries on the line which are to replace existing entries in the copied specification or which are to be included as new entries. Blank entries in the modifier statement do not affect the copied statement.

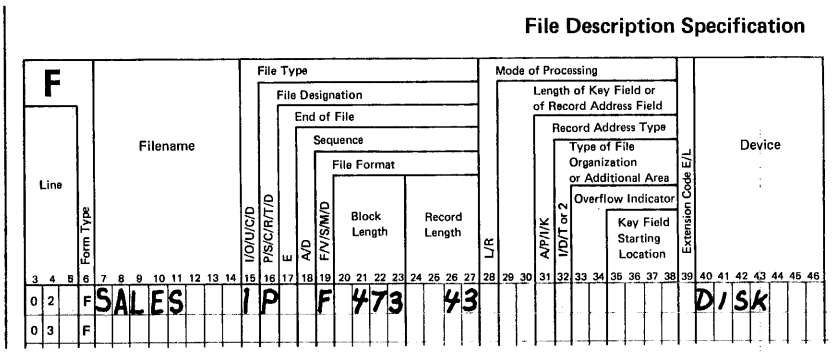
For example, suppose the file description specification for a frequently used file named SALES is to be copied from the library. The original specification contains an I in file type (position 15), defining SALES as an input file (Figure 209, insert A). In a particular job, you want to update the sales file, so you must change position 15 to a U. Therefore, you must include a modifier file description specification (Figure 209, insert B) in the auto report source program. The modifier statement must contain the filename, SALES, and the new file type entry, U. As a result of the modifier statement, the file type on the copied file description specification is changed from I to U (Figure 209, insert C).

/COPY statement to copy specifications for SALES file from the library entry named SALETR.



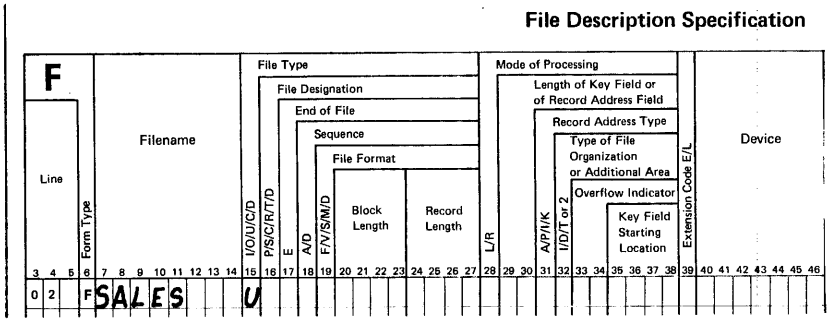
A

File description specification as it is cataloged in the library.



B

Copy function modifier statement.



C

Resulting file description specification that is included in the RPG II source program.

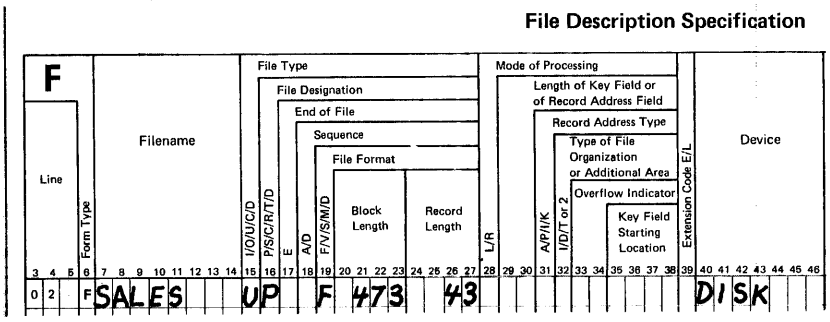


Figure 209. Modifying a Copied File Description Specification

To set an entry to blanks, enter an ampersand (&) in the first position of that entry on the modifier statement, and leave the remaining positions blank. For example, to remove the block length entry (positions 20-23) from the cataloged specification shown in Figure 209, insert A, add an ampersand to the modifier statement in position 20, as shown in Figure 210, and leave positions 21-23 blank.

Modifier statements for file description specifications do not have to be in any particular order in the auto report source program, except that they cannot immediately follow the /COPY statement if input field specifications are also being modified.

Only one file description specification with a particular filename is allowed to come from the library entries and a particular filename can be used only once on a modifier statement.

No modifications are allowed to the file description continuation specifications that accompany a copied file description. New continuation specifications can be added by placing them after a file description modifier statement for the file. A maximum of five continuation specifications are allowed to follow a file description specification (combined total of original and added continuation specifications).

Modifying Input Field Specifications

Only input field specifications (specifications describing individual fields on the input record) can be modified. To modify an input field specification copied from the library, enter the field name in positions 53-58 of an input field modifier statement (I in position 6). Modifier statements for input field specifications must immediately follow the /COPY statement in the auto report program that copies those specifications. The first specification following the /COPY statement which is not an input field specification is considered the end of the input field modifier statements for that /COPY statement. (A comment statement with an I in position 6 is not considered the end of the input field modifier statements.)

The method of replacing, adding, or blanking entries is similar to the method used to modify file description specifications. To replace or add entries, code the new entry in the proper location in the modifier statement; to set an entry to blank, place an ampersand (&) in the first position of that entry in the modifier statement. Figure 211 shows examples of modifying input specifications.

The modifier statement modifies all copied input field specifications which have the same field name. If there is no input field by the same name, the modifier statement is added to the program as a new input field specification. Modifier statements with duplicate field names are allowed (length and number of decimal positions must also be the same), but only the first is used to modify a copied specification. Other field names are added as new input field specifications. You can include at least 20 input field modifier statements per /COPY statement.

Note: For best results, place those statements first which modify existing input field specifications; then place those which are to be added as new input field specifications. This procedure is suggested because input field modifier statements which do not fit into the special main storage table for modifier statements are added to the RPG II source program as new input field specifications. This order of specifying modifier statements increases the likelihood that excess statements, if any, will be valid field descriptions.

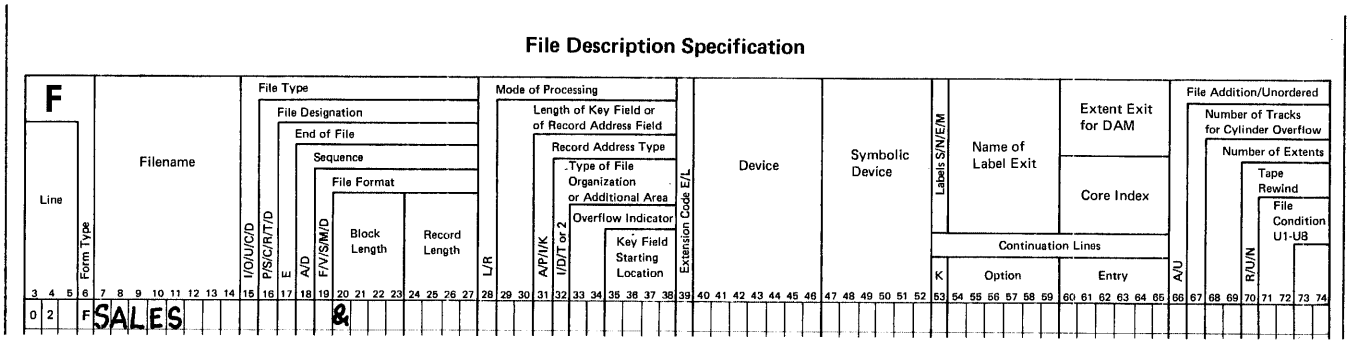
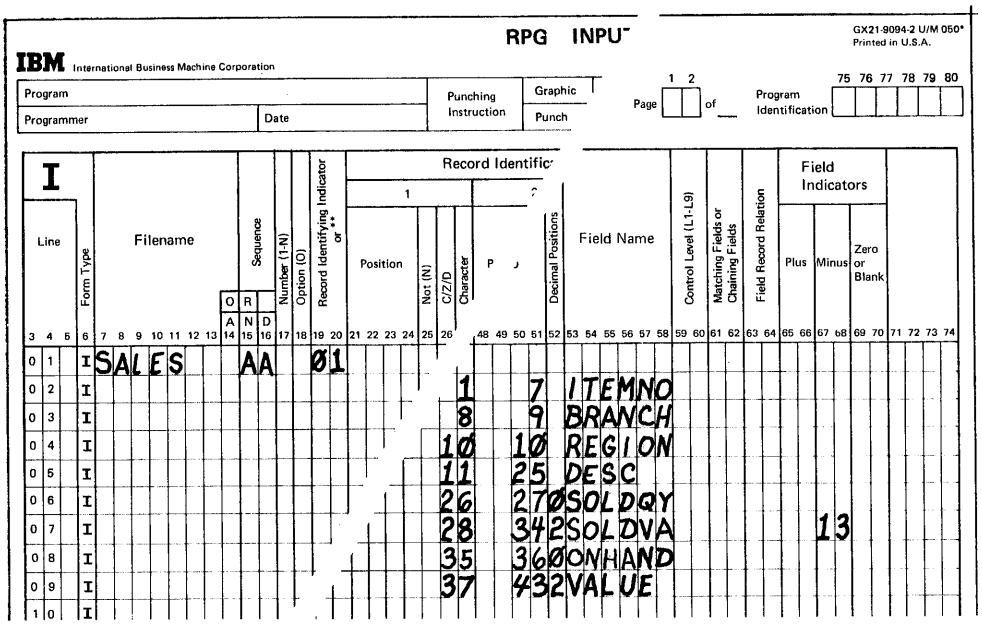


Figure 210. Setting a Copied File Description Entry to Blank

Input specifications as cataloged in the library.

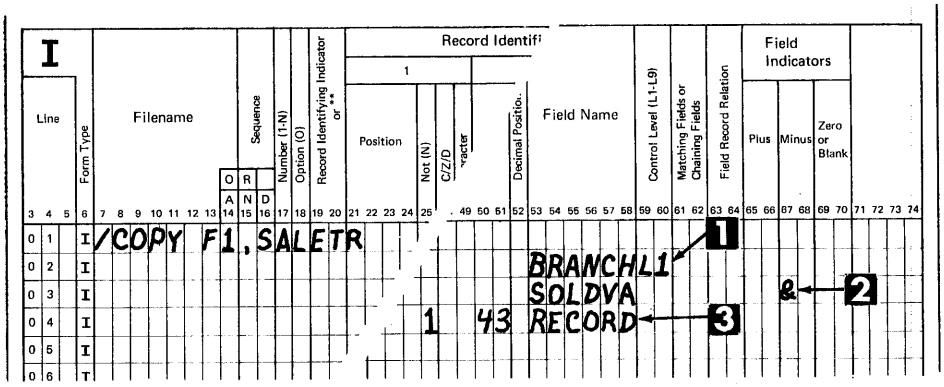


/COPY statement and modifier statements:

1 Add an entry to BRANCH field description.

2 Blank out minus field indicator on SOLDVA description.

3 Add a new field description.



Resulting input specifications for SALES file showing:

1 Added L1 indicator.

2 Blanks in place of minus field indicator.

3 Added field description.

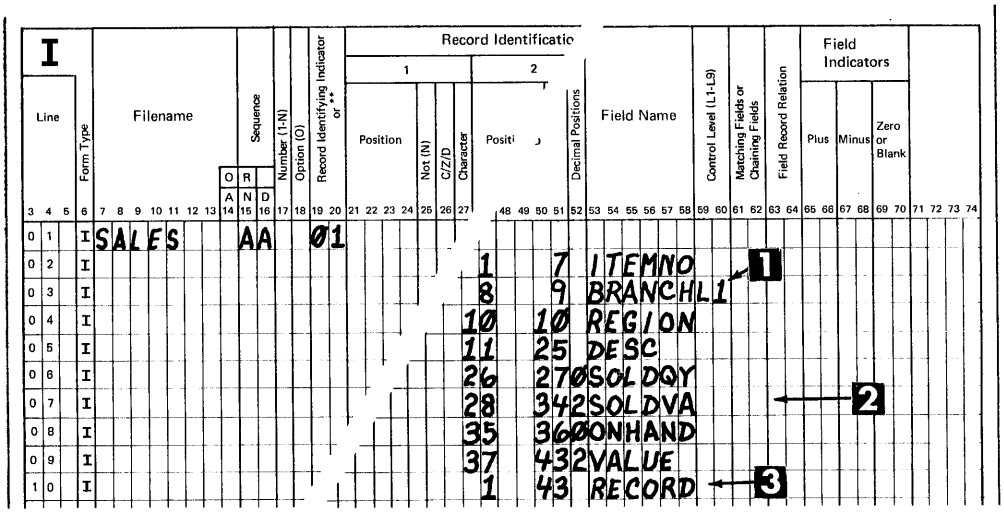


Figure 211. Modifying Copied Input Field Specifications

The RPG II program produced by auto report includes RPG II specifications from the following sources:

- Those included by the programmer in the auto report program (any valid RPG II specifications are allowed).
- Those copied from the library using the /COPY statement.
- Those generated by auto report.

The specifications are in the order required by the RPG II Compiler. Among the generated specifications are calculations to accumulate totals for the specified fields on the D/T-*AUTO specification. The output specifications generated by auto report are also included. These specifications contain the necessary spacing, skipping, and end position entries to produce a formatted report.

FORMAT OF THE GENERATED SPECIFICATIONS

The generated RPG II specifications are in the following format:

Position	Contents
1-4	Sequence number of the specification. This number starts as 0010 on the RPG II control statement and is incremented by 0010 on each specification that follows. If more than 999 specifications are present in the program, the sequence is restarted at 0000.
5	Code that identifies the specification as follows:
	Blank — A standard RPG II specification present in the auto report program.
	C — Specification copied from the library.
	M — Specification copied from the library and modified.
	E — Specification generated by auto report.
6-74	Standard RPG II specification.
75-80	The same characters are present in positions 75-80 of the RPG control statement. (If these positions are blank on the RPG control statement, they are also blank on all specifications in the generated RPG II program.)

Compile-time tables and arrays are not changed by auto report; they remain in standard table/array record format.

GENERATED SPECIFICATIONS

Standard RPG II specifications are generated by auto report and are combined with RPG II specifications included in the input to auto report and specifications copied from the library to produce the final RPG II source program. This section describes the generated RPG II specifications and the order of those specifications in the RPG II source program.

Figures 212 and 213 show auto report specifications for a sales report and the resulting RPG II source specifications that are generated for the report. Numbers are inserted in the figures to identify the auto report functions and to show the specifications that are generated by each function. The auto report specifications in Figure 212 are similar to those under *How to Use RPG II Auto Report, Example 6.* The file description and input specifications for the SALES file are cataloged in the library, as in that example.

The copy function includes the specifications for the SALES file (Figure 212, insert 2). Since BRANCH and REGION are to be control fields for the sales report, modifier statements follow the /COPY statement to add control level indicators to the input specifications.

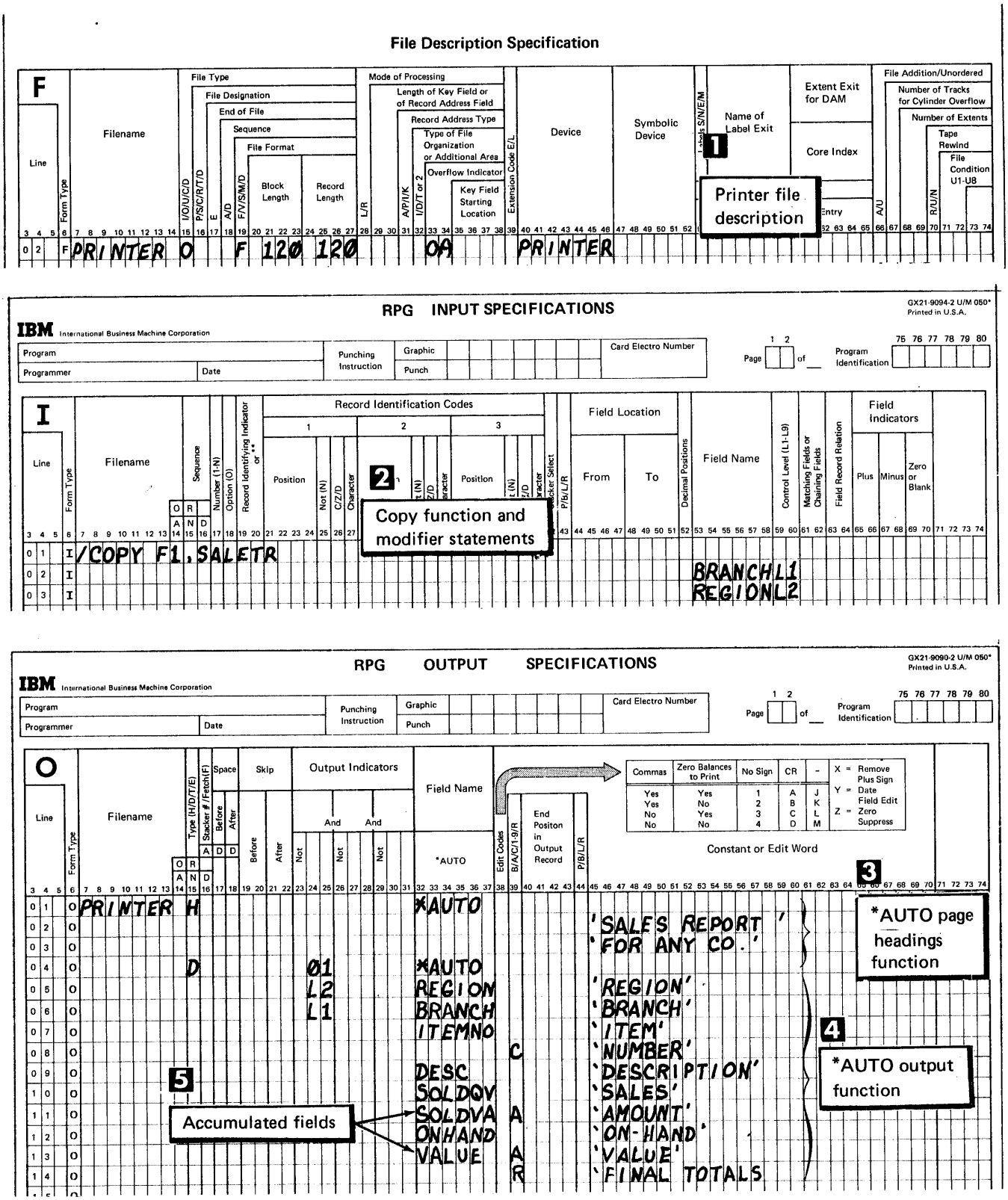
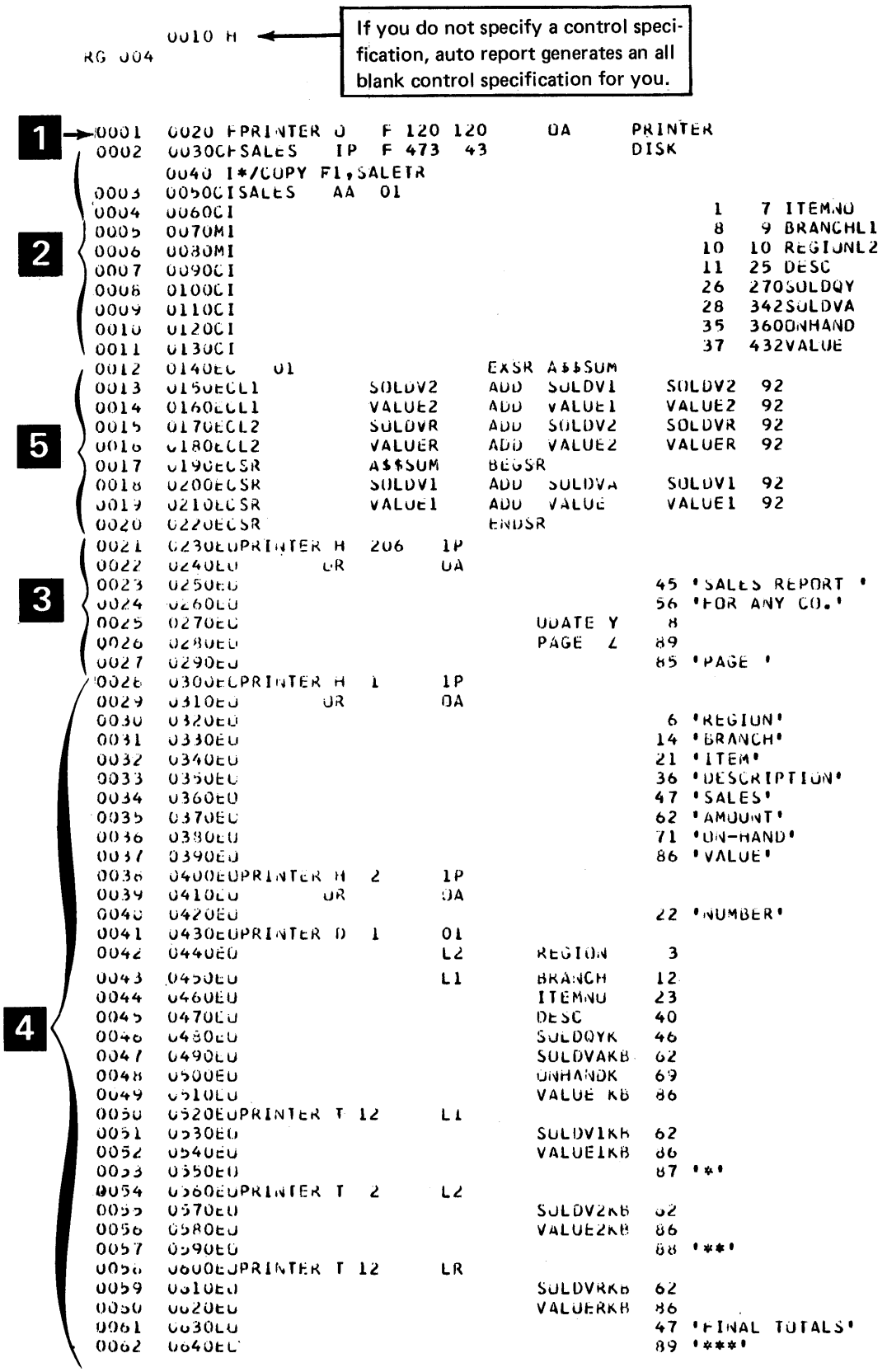


Figure 212. Auto Report Specifications for a Sales Transaction Report




```
RG 004          0010 H
0001  0020 FPRINTER O   F 120 120      OA      PRINTER
0002  0030 FSALES   IP  F 473  43              DISK
      0040 I*/COPY FL,SALETR
0003  0050 ISALES   AA  01
0004  0060 I                                                1   7 ITEMNO
0005  0070 I                                                8   9 BRANCHL1
0006  0080 I                                               10  10 REGIONL2
0007  0090 I                                               11  25 DESC
0008  0100 I                                               26  27 0SOLDQY
0009  0110 I                                               28  34 2SOLDVA
0010  0120 I                                               35  36 0ONHAND
0011  0130 I                                               37  43 2VALUE
0012  0140 C   01                    EXSR A$$SUM
0013  0150 C  L1          SOLDV2     ADD  SOLDV1    SOLDV2   92
0014  0160 C  L1          VALUE2     ADD  VALUE1    VALUE2   92
0015  0170 C  L2          SOLDVR     ADD  SOLDV2    SOLDVR   92
0016  0180 C  L2          VALUER     ADD  VALUE2    VALUER   92
0017  0190 C  SR          A$$SUM     BEGSR
0018  0200 C  SR          SOLDV1     ADD  SOLDVA    SOLDV1   92
0019  0210 C  SR          VALUE1     ADD  VALUE     VALUE1   92
0020  0220 C  SR                     ENDSR
0021  0230 OPRINTER H  206   1P
0022  0240 O        OR        OA
0023  0250 O                                          45 'SALES REPORT '
0024  0260 O                                          56 'FOR ANY CO.'
0025  0270 O                          UDATE Y          8
0026  0280 O                          PAGE  Z         89
0027  0290 O                                          85 'PAGE '
0028  0300 OPRINTER H  1     1P
0029  0310 O        OR        OA
0030  0320 O                                           6 'REGION'
0031  0330 O                                          14 'BRANCH'
0032  0340 O                                          21 'ITEM'
0033  0350 O                                          36 'DESCRIPTION'
0034  0360 O                                          47 'SALES'
0035  0370 O                                          62 'AMOUNT'
0036  0380 O                                          71 'ON-HAND'
0037  0390 O                                          86 'VALUE'
0038  0400 OPRINTER H  2     1P
0039  0410 O        OR        OA
0040  0420 O                                          22 'NUMBER'
0041  0430 OPRINTER D  1     01
0042  0440 O                 L2      REGION    3
0043  0450 O                 L1      BRANCH   12
0044  0460 O                         ITEMNO   23
0045  0470 O                         DESC     40
0046  0480 O                         SOLDQYK  46
0047  0490 O                         SOLDVAKB 62
0048  0500 O                         ONHANDK  69
0049  0510 O                         VALUE KB 86
0050  0520 OPRINTER T 12     L1
0051  0530 O                         SOLDV1KB 62
0052  0540 O                         VALUE1KB 86
0053  0550 O                                          87 '*'
0054  0560 OPRINTER T  2     L2
0055  0570 O                         SOLDV2KB 62
0056  0580 O                         VALUE2KB 86
0057  0590 O                                          88 '**'
0058  0600 OPRINTER T 12     LR
0059  0610 O                         SOLDVRKB 62
0060  0620 O                         VALUERKB 86
0061  0630 O                                          47 'FINAL TOTALS'
0062  0640 O                                          89 '***'
```

Figure 213. RPG II Source Program Generated from Auto Report Specifications

Generated Calculations

Calculations are generated to accumulate totals for fields named on *AUTO field description specifications which have an A in position 39 (Figure 214).

An RPG II subroutine is generated to accumulate the values from these fields into the lowest level generated total fields. The name of the subroutine is always A$#SUM. The subroutine specifications are conditioned differently, depending on whether detail or group printing is specified:

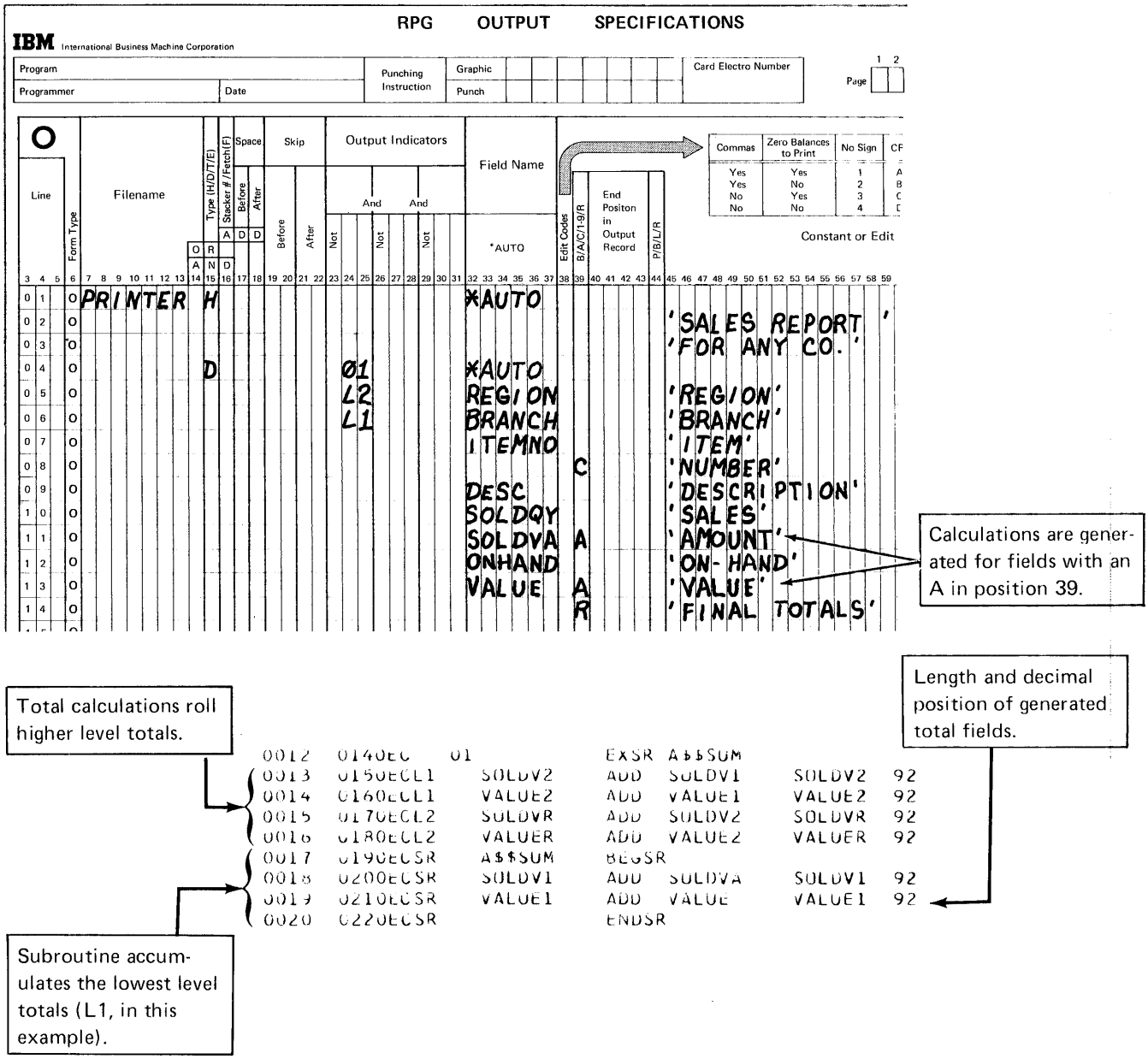
- If detail printing is specified, as in Figure 214, the EXSR statement is conditioned by the same indicator(s) that condition the D-*AUTO specification (01 in this example). Each ADD statement in the subroutine is conditioned by the field indicator(s) specified with the field in its field description specification (none in this example).

- If group printing is specified, the EXSR statement and all ADD statements in the subroutine are unconditioned.

Total calculations are generated to roll the total from the lowest level defined total field through the higher level defined total fields and the final total. The total calculation to add the total from one level to that of the next higher level is conditioned by the control level indicator corresponding to the field name of the lower level. As shown in Figure 214, total calculations to accumulate L2 and LR totals are followed by the subroutine to accumulate the lowest level total, L1.

Generated total fields are defined (given length and number of decimal positions) when the total field is the result field in a generated calculation. In the input specifications, SOLDVA and VALUE are numeric fields defined with a length of seven and two decimal positions. Figure 214 shows that the total fields generated from SOLDVA and VALUE are defined as two positions longer than the original fields, with the same number of decimal positions.

When group printing is specified (T-*AUTO specification), auto report generates total calculations to reset each of the accumulated fields (A in position 39) on the lowest level total line to zero on each cycle. A (Z-ADD) calculation, conditioned by L0, is generated for each accumulated field. These calculations are the first total calculations in the generated RPG II source program.



Note: Placement of the generated calculations in the RPG II source program is shown in Figure 213.

Figure 214. Calculations Generated from Auto Report Coding for Sales Transaction Report

Generated Output Specifications

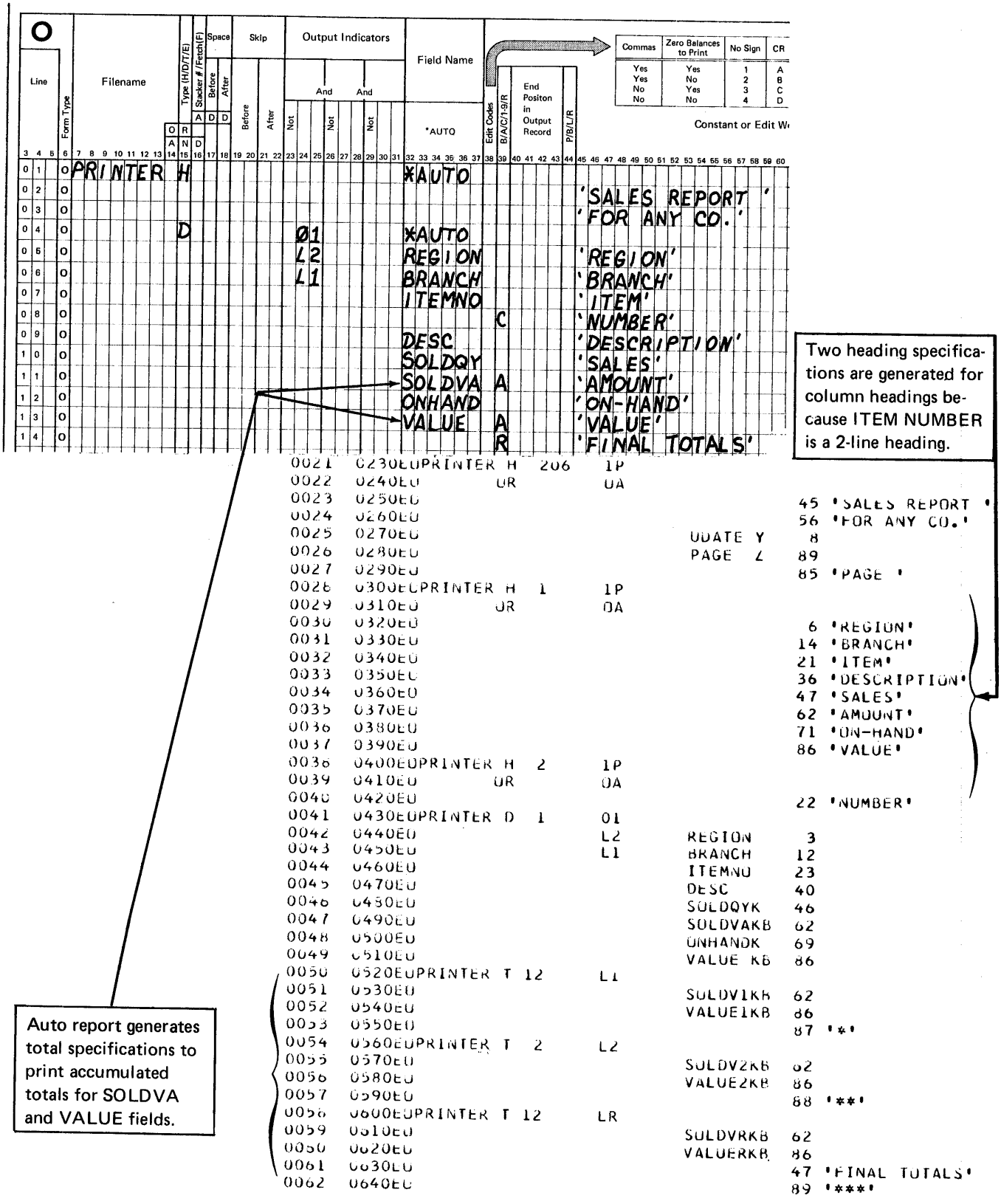
Figure 215 shows the output specifications generated by auto report. Specifications supplied by auto report (column heading specifications, total specifications, conditioning indicators, spacing and skipping values, end position values, blank after) can be identified by comparing the listing with the auto report specifications.

Auto report generates specifications to reset accumulated fields to zero after they are printed. See *Field Description (A in Position 39)* for a discussion of resetting fields to zero. In this example, blank after is generated for accumulated fields.

Order of Generated Specifications

The specifications in the RPG II source program generated by auto report are in the order required by the RPG II Compiler. When specifications are included by means of a /COPY statement, those specifications are initially placed immediately after the /COPY statement. After all specifications are copied and before auto report generates RPG II specifications from the H-*AUTO and D/T-*AUTO specifications, the entire auto report source program is sorted into the following order:

1. Control specifications
2. File description specifications
3. Extension specifications
4. Line counter specifications
5. Telecommunications specifications
6. Input specifications
7. Calculation specifications (in the order: detail, L0, L1-L9, LR, and subroutines)
8. Output specifications
9. Tables and arrays loaded at compilation time (must be placed last among the input statements to auto report)




```
0021  0230EOPRINTER H  206   1P
0022  0240EO         OR      OA
0023  0250EO                                    45 'SALES REPORT '
0024  0260EO                                    56 'FOR ANY CO.'
0025  0270EO                 UDATE Y     8
0026  0280EO                 PAGE  Z     89
0027  0290EO                             85 'PAGE '
0028  0300EOPRINTER H  1     1P
0029  0310EO         OR      OA
0030  0320EO                              6 'REGION'
0031  0330EO                             14 'BRANCH'
0032  0340EO                             21 'ITEM'
0033  0350EO                             36 'DESCRIPTION'
0034  0360EO                             47 'SALES'
0035  0370EO                             62 'AMOUNT'
0036  0380EO                             71 'ON-HAND'
0037  0390EO                             86 'VALUE'
0038  0400EOPRINTER H  2     1P
0039  0410EO         OR      OA
0040  0420EO                             22 'NUMBER'
0041  0430EOPRINTER D  1     01
0042  0440EO                 L2  REGION      3
0043  0450EO                 L1  BRANCH     12
0044  0460EO                     ITEMNO     23
0045  0470EO                     DESC       40
0046  0480EO                     SOLDQYK    46
0047  0490EO                     SOLDVAKB   62
0048  0500EO                     ONHANDK    69
0049  0510EO                     VALUE KB   86
0050  0520EOPRINTER T  12    L1
0051  0530EO                     SOLDV1KB   62
0052  0540EO                     VALUE1KB   86
0053  0550EO                                87 '*'
0054  0560EOPRINTER T  2     L2
0055  0570EO                     SOLDV2KB   62
0056  0580EO                     VALUE2KB   86
0057  0590EO                                88 '**'
0058  0600EOPRINTER T  12    LR
0059  0610EO                     SOLDVRKB   62
0060  0620EO                     VALUERKB   86
0061  0630EO                                47 'FINAL TOTALS'
0062  0640EO                                89 '***'
```

Figure 215. Output Specifications Generated from Auto Report Coding for Sales Transaction Report

Output Specifications

Output heading specifications generated for H-*AUTO specifications appear in the same order they are coded on the output sheet in relation to other RPG II and *AUTO output specifications for the file.

Normally, RPG II output specifications generated from a D/T-*AUTO specifications are in the following order:

1. Heading specifications generated for column headings
2. Detail specifications
3. Total specifications, with the lowest level first and LR last

This group of specifications is placed in the same relative position in the program as the original D/T-*AUTO specification. All other RPG II output specifications remain in their original order.

If, however, the programmer specifies a normal RPG II total output specification conditioned by a positive (no N in position 23) control level indicator in positions 24-25 for the file which has a D/T-*AUTO specification, all output specifications in the program are sorted into the following format:

1. All heading, detail, and exception output specifications. They remain in the same order as they are in the generated RPG II source program. Total specifications which are not conditioned by a positive control level indicator in positions 24-25 remain as they were in the program.
2. Total specifications which are conditioned by a positive control level indicator in positions 24-25. These specifications are sorted into ascending order by the control level indicator in positions 24-25, with LR last.

See Figure 213 and Part 4, Chapter 3, *Sample Program* for examples of the ordering of generated specifications.

Comment Statements

Comment statements (identified by an asterisk in position 7) are allowed among the statements read by auto report. However, since the sort of RPG II specifications is based on the contents of position 6, comments may not occur in the expected order. To ensure that comments remain with the correct specification, place them after that specification and put the same entry in position 6.

Restriction

The order of tables and arrays is not altered when the source specifications are sorted. Therefore, when you include tables and arrays from the library, they may not occur in the correct order after the sort. For example, if a file translation or alternate collating sequence table is present in the auto report source specifications, then any compilation-time tables or arrays included from a library member are out of order. That is, the included tables or arrays are placed ahead of the file translation table. Recall that compile-time tables and arrays must be loaded in the following order:

1. File translation specifications
2. Alternate collating sequence specifications
3. Compilation-time tables and arrays in the order described on the extension specifications sheet

A solution to this restriction is to place your file translation and alternate collating sequence tables in the library and copy them from the library before any other compilation-time tables and arrays are copied. This procedure ensures that your file translation and alternate collating sequence tables are the first compilation-time tables in the generated RPG II source program.

One of the advantages of auto report is that it frees the programmer from the task of specifying the format of his report on the output sheet. Auto report can completely format the report by spacing, skipping, centering lines, and calculating end positions for fields and constants.

SPACING AND SKIPPING

Spacing and skipping can be either left to auto report or specified by the programmer. Figure 216 shows spacing and skipping generated by auto report. For the specifications used to produce the report, see *The Generated RPG II Program*. If positions 17-22 are left blank on an H-*AUTO specification, a skip to line 06 is done before the first heading line is printed and space-two-after is done for the last heading line. If more than one heading line is specified, space-one-after is done for the first and all succeeding lines except the last. If the programmer specifies spacing and skipping entries, he must follow normal RPG II rules for spacing and skipping.

Column heading lines are spaced like page headings. Space-one-after is done for all except the last. Space-two-after is done for a single heading line, or for the last heading line if more than one is specified. The programmer cannot specify his own spacing and skipping entries for column headings. If spacing and skipping entries are made on a D-*AUTO record description specification, the entries apply to the detail line generated. The entries do not apply to column headings or total lines generated by auto report from the D-*AUTO specification. Normal RPG II rules for spacing and skipping must be followed. Space-one-after is assumed for the generated detail line if spacing and skipping entries are not made.

Space-two-after is generated for all total lines produced by auto report from a D-*AUTO specification. In addition, the lowest level total line and the final total line are also generated with a space-one-before.

If spacing and skipping entries are made on a T-*AUTO specification, the entries apply to the lowest level total line generated, but not to column headings or higher level total lines. If spacing and skipping are not made, the lowest level total lines are generated with space-one-after; all higher levels are generated with space-two-after. Space-one-before is always generated for the second-to-the-lowest level total and the final total. (See Figure 207 for an example.)

PLACEMENT OF HEADINGS AND FIELDS

Auto report generates end positions for fields and constants and centers column headings, columns, and report lines (see Figure 216 for an example). However, if the programmer specifies an end position for a field or constant on a D/T-*AUTO field description line, that end position is used on all column heading, detail, and total specifications generated from the field description. (The specified end position may be altered slightly by auto report when the line is centered or when the column heading and field are positioned in relation to each other.) If the specified end position causes an overlay with a previous field or constant, auto report generates a new end position.

Specify end positions only when you want to eliminate the automatic spacing between fields or when you want to spread out or expand a report on the page.

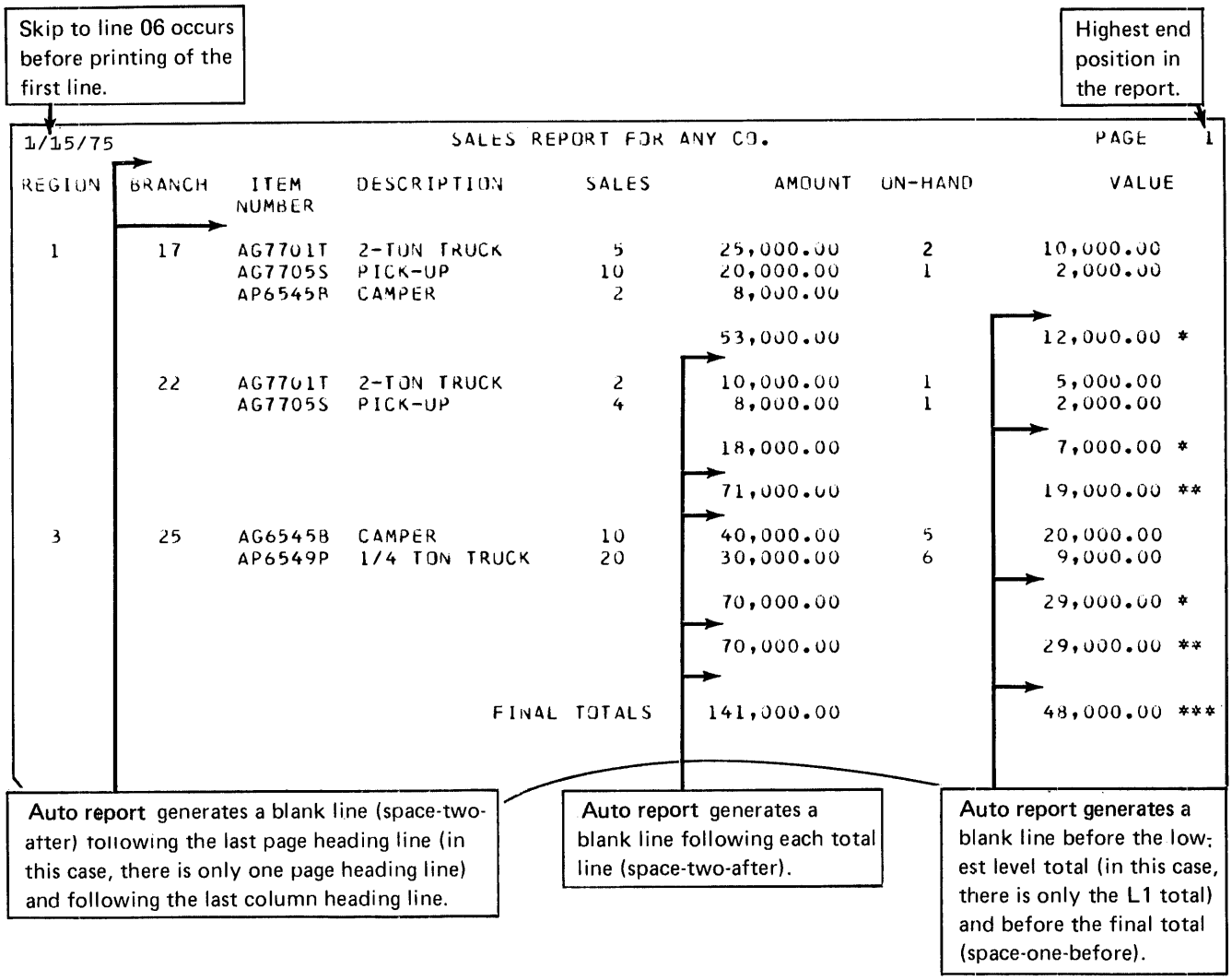


Figure 216. Report Illustrating Format Generated by the Auto Report

Page Headings

If the date and page number are printed on the first *AUTO page heading line (that is, if they are not suppressed by an N in position 27 of the option specifications sheet), the date is always printed in positions 1-8. The page number is printed with an end position equal to the highest end position of the longest line in the report. When the first *AUTO page heading (including date, title, and page number) is the longest line in the report, one blank space separates the title from the date and the word PAGE from the title. If the resulting line exceeds the record length of the printer file, the excess information on the right of the line is not printed.

If a line generated from a D/T-*AUTO specification is the longest report line, that line is printed starting in print position 1 and the title portion of the first page heading line is centered in relation to that line. Additional *AUTO page headings are then centered on the first *AUTO page heading line.

If an *AUTO page heading is the longest line in the report and a D/T-*AUTO specification is present, any other *AUTO page heading lines and the line generated from the D/T-*AUTO specification are centered on the longest page heading.

Fields and constants appear in the order specified in the *AUTO output specifications from left to right. Auto report provides one blank space before and after fields on the heading line. No spacing is provided between constants.

*Reformatting *AUTO Page Headings*

You can reformat an *AUTO page heading line if you do not want to use the end positions for fields and constants that are generated by auto report. If you want to find what end positions are generated for page, date, and title information, see the listing of the generated source program that is produced by the RPG II Compiler (see *Generated Specifications*).

Catalog the generated RPG II source program in the library by specifying the C option in position 7 of the auto report option specifications (see *Source, Position 7*) and change the end positions on the generated source statements.

Body of the Report

Placement of column headings above columns depends on which is longer, the heading or the associated field (including edit characters). If any of the column headings is longer than the associated field, the field is centered under the longest column heading constant. If, however, the field is longer than the longest column heading constant, the column heading is left-justified over an alphameric field and right-justified over a numeric field. When more than one columm heading line is specified, shorter column headings are always centered on the longest column heading.

Fields and constants appear from left to right on a line in the order they are specified by the output specifications. At least two blank spaces appear before each field on the line. However, no spaces are provided before a constant; the programmer must incorporate blanks within constants to provide for additional spacing.

Total indication information (fields and constants specified with 1-9 or R in position 39) is placed to the left of the first total field (A in position 39) on the corresponding total line, followed by two spaces. If two or more such fields or constants are specified for a total line, they appear from left to right in the order specified on the left of the first total on the line. Each field is preceded and followed by one space. No spacing is provided for constants.

*Overflow of the D/T-*AUTO Print Lines*

If the lines generated from a D/T-*AUTO specification are longer than the record length specified for the printer file, a second print line (overflow line) is generated for each column heading line, detail (or group print) line, and total line. (Remember, a second print line is not generated for *AUTO page heading lines.) The excess information is placed on the overflow line in the order specified, right-justified.

Figure 217 shows the result of an overflow condition. The specifications and data for the report are the same as in Part 4, Chapter 3, *Sample Program*, except that the printer record length is reduced from 132 to 96. In the output specifications for Figure 217, no spacing or skipping is specified. If you specify spacing and skipping, however, auto report spaces the report as follows:

- Column heading lines and total lines are spaced as shown in Figure 217.
- The space-before and skip-before entries you specify are for the original detail (or group print) line. Auto report generates space-one-after for this line.
- The space-after and skip-after entries you specify are for the overflow line. Auto report generates blanks for space-before and skip-before for the overflow line.

Auto report prints those columns that cannot be completely contained on the original line on overflow lines.

1/15/75 CASH RECEIPTS REGISTER PAGE 1

REGION	ACCOUNT NUMBER	ACCOUNT NAME	INVOICE NUMBER	INVOICE DATE	DATE PAID / AMOUNT PAID	AMOUNT OWED / BALANCE DUE	DISCOUNT TAKEN / EXCESS DISCOUNT	
1	11243	JONES HARDWARE	27541	7/11/71	7/21/1 23.28	23.75	.47	
1	11352	NU-STYLE CLOTHIERS	27987	7/14/71	7/26/1 40.00	87.07 47.07		
1	11886	MIDI FASHIONS INC	15771	7/04/71	7/14/1 105.08	107.22	2.14	
1	12874	ULOOK INTERIORS	25622	7/09/71	7/23/1 67.95	67.95		
1	18274	STREAMLINE PAPER INC	29703	7/21/71	7/30/1 170.55	274.03 101.10	2.38	
					REGION TOTALS 406.86	560.02 148.17	4.99	*
2	23347	RITE-BEST PENS CO	20842	7/18/71	7/20/1 10.00	15.80 5.80		
2	25521	IMPORTS OF NM	29273	7/20/71	7/27/1 585.47	797.40 200.00	11.93	
2	26723	ALRIGHT CLEANERS	19473	7/07/71	7/23/1 462.00	462.00		
2	28622	NORTH CENTRAL SUPPLY	17816	7/05/71	7/22/1 75.97	75.97		
2	29871	FERGUSON DEALERS	27229	7/10/71	7/22/1 61.91	61.91		
					REGION TOTALS 1,195.35	1,413.08 205.80	11.93	*
3	30755	FASTWAY AIRLINES	26158	7/06/71	7/19/1 725.87	742.72	16.85 1.90	
3	31275	ENVIRONMENT CONCERNS	20451	7/06/71	7/30/1 15.00	29.43 14.43		
3	32457	B SOLE SILOS	27425	7/10/71	7/20/1 110.05	110.05		
3	37945	HOFFTA BREAKS INC	18276	7/06/71	7/23/1 47.23	47.23		
					REGION TOTALS 898.15	929.43 14.43	16.85 1.90	*
4	42622	EASTLAKE GRAVEL CO	16429	7/05/71	7/23/1 29.37	29.37		
					REGION TOTALS 29.37	29.37		*
					COMPANY TOTALS 2,529.73	2,931.90 368.40	33.77 1.90	**

Figure 217. Report Illustrating Overflow of D-*AUTO Print Lines

This section includes information about installation, operating instructions, and operation control language for auto report.

INSTALLATION AND MAINTENANCE

To install auto report, follow the procedures for system installation described in *IBM System/32 Operator's Guide*, GC21-7591.

The RPG II auto report function distributed with the system release includes the following:

- An IBM-supplied library procedure (named AUTO) for loading the auto report program (see *Operation Control Language Considerations*).
- The auto report program, consisting of 14 object modules (O modules), requiring approximately 190 sectors of library space on disk.
- A sample program and the library procedures to compile and execute it (see Chapter 3, *Sample Program* for a description of the program and the procedure to run it).

OPERATING CONSIDERATIONS

To compile an RPG II program that includes auto report specifications, follow the instructions for compiling an RPG II program given in *IBM System/32 Operator's Guide*, GC21-7591. Remember the following differences when compiling an auto report source program:

- The name used in the procedure statement is AUTO; the name on the LOAD statement is #AUTO.
- The auto report option specifications (U in position 6) must be the first specifications in the auto report source program.

OPERATION CONTROL LANGUAGE CONSIDERATIONS

To compile an RPG II program that includes auto report specifications, the auto report program must be loaded into main storage. After the auto report program has generated an RPG II source program, it calls the RPG II Compiler to compile the source program if no terminal auto report diagnostics have been issued. An IBM-supplied procedure can be called to load the auto report program. The OCL statements to include the procedure from the library are:

```
// LOAD      #AUTO
// FILE      NAME-$WORK
             RETAIN-S
             BLOCKS-20
// FILE      NAME-$SOURCE
             RETAIN-S
             BLOCKS-20
// MEMBER    USER1-RPGCMPL1
// MEMBER    PROGRAM1-RPGCMPL1
// MEMBER    PROGRAM2-RPGCMPL2
// COMPILE   SOURCE ?1R'1016'?
// RUN
```

Note: The 1R prompts for the name of user source program.

Library procedures can be modified. OCL statements necessary to modify a library procedure are described in *IBM System/32 System Control Programming Reference Manual*, GC21-7593.

If you want to change the number of tracks in $SOURCE and $WORK, modify the library procedure, calculating tracks as follows:

$$\text{Blocks} = \frac{\text{Number of specifications}}{20}$$

For *number of specifications*, use the greater of the number of specifications read by auto report or the estimated number of specifications in the generated source program. The calculated number of blocks should be used for both $SOURCE and $WORK.

HALTS

Auto report does not diagnose all error conditions in the source program. Diagnostics that are performed by the RPG II Compiler are not duplicated by auto report. If a program cannot be successfully generated because of errors in the auto report specifications, auto report halts. Only recovery option 3 (immediate cancel) is available following this halt.

If an RPG II source program is successfully generated, auto report calls the RPG II Compiler without halting. Normal RPG II compilation halts can occur after compilation has begun. Compilation halts and object program execution halts are explained in the *IBM System/32 Messages Guide—RPG II*, SC21-7617.

COMPILE Statement

The COMPILE statement is designed so that you can enter the procedure name AUTO either with or without the source program name. If the source program name is not entered with the procedure name, a prompt requests the name of the source program.

LOG Statement

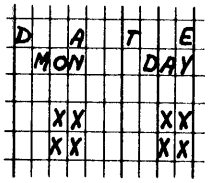
Output of the auto report program listing is governed by the LOG statement. See the *IBM System/32 Control Programming Reference Manual*, GC21-7593, for a description of the LOG statement.

The following chart should be helpful in determing valid *AUTO output entries depending on the contents of position 39:

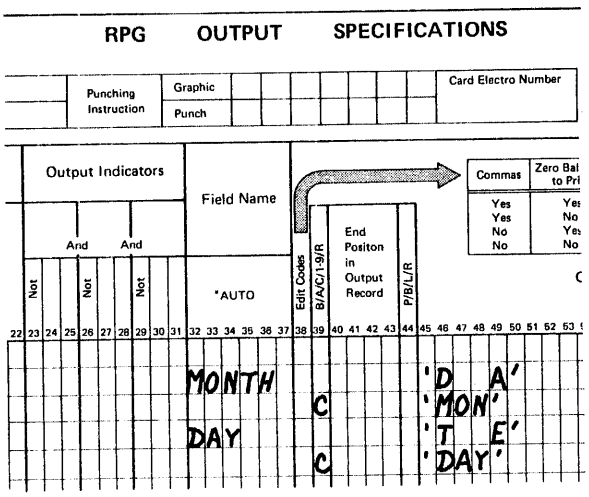
39	7-22	23-31	32-37	38	40-43	44	45-70
Blank	Blank	Blank or Indicators	Field Name	Blank or Edit Code	Blank or End Position	Blank	Blank or Column Heading
	Blank	Blank or Indicators	Blank	Blank	Blank or End Position	Blank	Literal
B	Blank	Blank or Indicators	Field Name	Blank or Edit Code	Blank or End Position	Blank	Blank or Column Heading
A	Blank	Blank or Indicators	Field Name	Blank or Edit Code	Blank or End Position	Blank	Blank or Column Heading
C	Blank	Blank	Blank	Blank	Blank	Blank	Column Heading
1-9, R	Blank	Blank	Field Name	Blank or Edit Code	Blank	Blank	Blank or Edit Word
	Blank	Blank	Blank	Blank	Blank	Blank	Literal

The following miscellaneous programming suggestions may be helpful in specific programming situations:

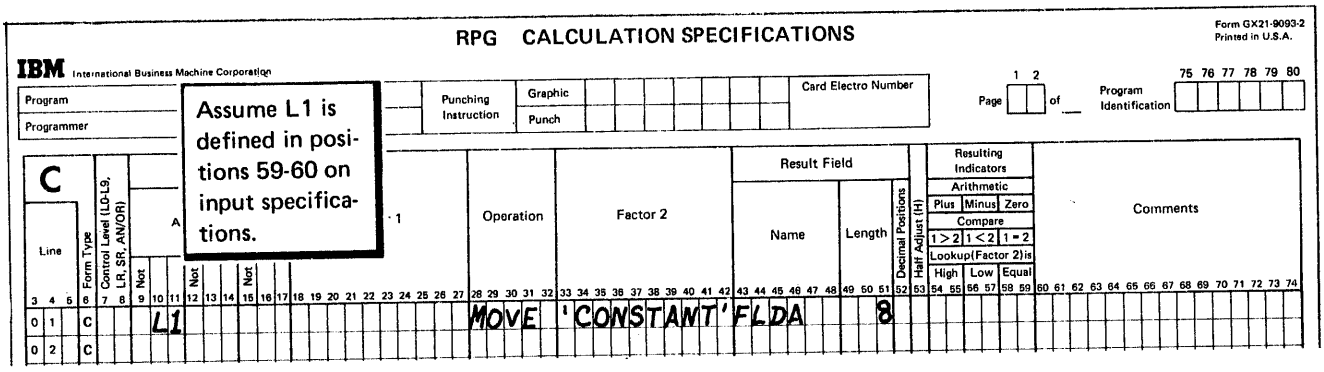
1. One column heading can be printed over two or more fields if automatic column spacing is taken into consideration. For example, if the heading DATE is to print over a month field and a day field as follows:

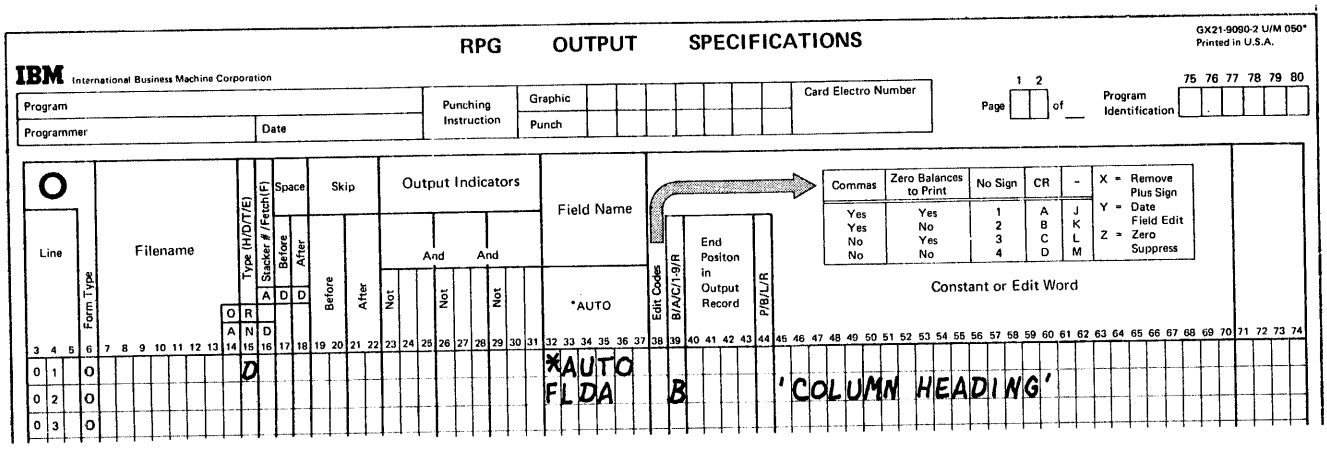


Code the output specifications as follows:



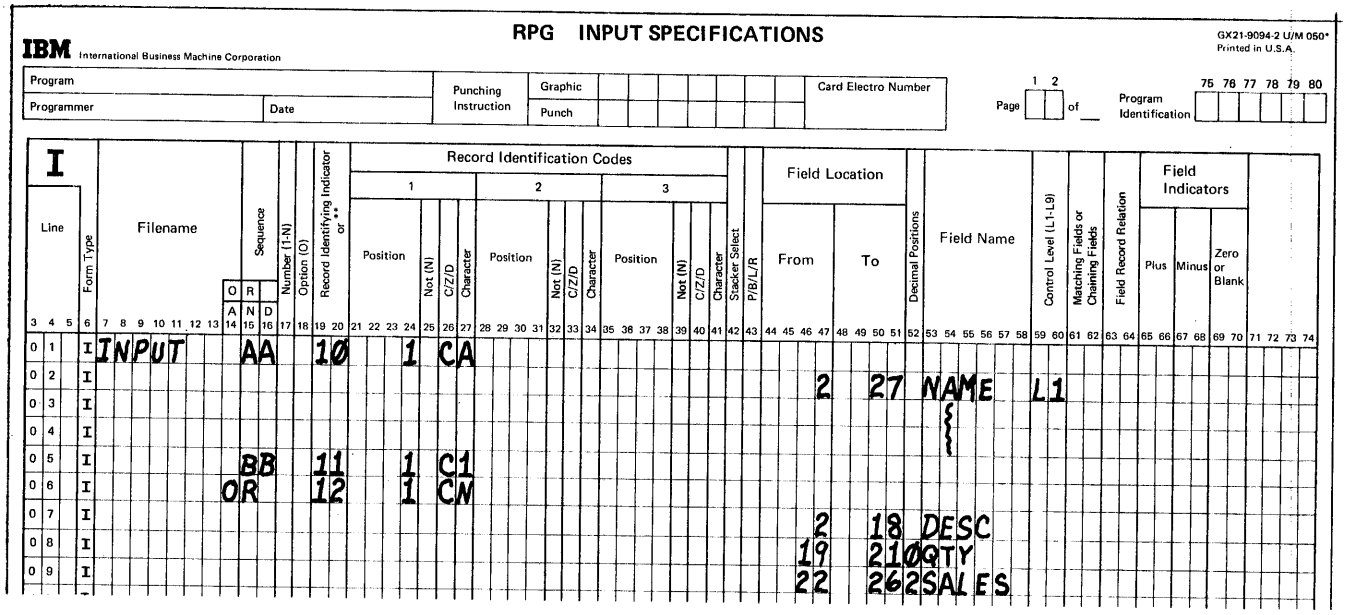
2. To print a constant on only the first detail line under a column heading, move the constant to a field in calculations specifications and print that field.



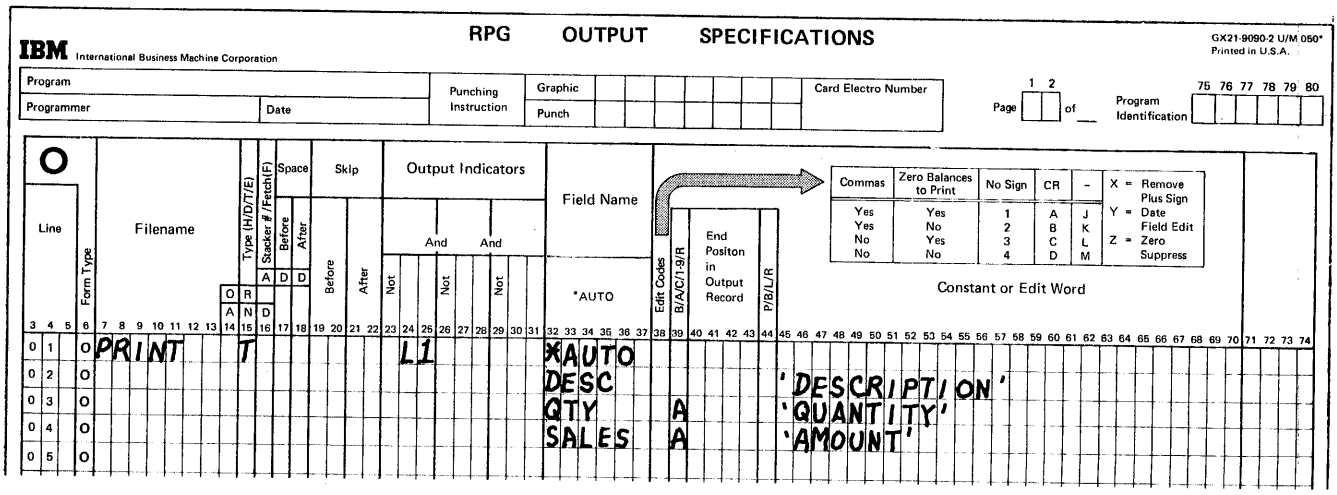


3. To print a column heading using the *AUTO function, a field name must be coded with that column heading. If a need arises to print a column heading for a field that is to be printed only at total time in the RPG II program cycle, create a field of blanks by means of calculations specifications and print that blank field with the desired column heading. Then use a normal RPG II total specification and code the desired field to print using end positions. See programming tip 5 for an example.

4. If group printing is being done and more than one record type is present in the input file, certain precautions must be taken by the programmer. If a field to be accumulated is present in all record types, but only one record type is to be processed, proper total is not generated unless additional coding is used. Given the following input specification:



The following coding gives incorrect results:



The results are incorrect because the T-*AUTO specification causes an unconditioned ADD subrountine to be generated if a field is to be added. In the auto report coding above, QTY is added when the 10 indicator is on and when indicators 11 or 12 are on.

The following coding is a method of obtaining the correct results:

RPG CALCULATION SPECIFICATIONS

Form GX21-9093-2 Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic	Card Electro Number
Programmer	Date		Punch	

Page 1 2 of __ Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / And	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus Minus Zero; Compare 1>2 1<2 1=2; Lookup(Factor 2)is High Low Equal	Comments
0 1	C		Not 11 Not Not		Z-ADD	QTY	QTYA	3	0			
0 2	C		11		Z-ADD	SALES	SALESA	5	2			

RPG OUTPUT SPECIFICATIONS

GX21-9090-2 U/M 050* Printed in U.S.A.

IBM International Business Machine Corporation

Program		Punching Instruction	Graphic	Card Electro Number
Programmer	Date		Punch	

Page 1 2 of __ Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # / Fetch(F)	Space: Before	Space: After	Skip: Before	Skip: After	Output Indicators: And / And	Field Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R	Constant or Edit Word
0 1	O	PRINT	T						L1	*AUTO					
0 2	O									DESC		A			'DESCRIPTION'
0 3	O									QTYA		A			'QUANTITY'
0 4	O									SALESA		A			'AMOUNT'

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

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

5. When printing an invoice register, the total number of invoices is often printed, as shown on the following print chart:

1 2 3 4 5 6 7 8 9 0 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 6 6 6 6
INVOICE NUMBER GROSS AMOUNT DISCOUNT NET AMOUNT NUMBER OF INVOICES
XXXXX XX,XXX.ØØ- XXX.ØØ- XX,XXX.ØØ-
X,XXX,XXX.ØØ- XX,XXX.ØØ- X,XXX,XXX.ØØ- XXX

A method of accomplishing this is shown by the following calculation and auto report output specifications:

RPG CALCULATION SPECIFICATIONS

IBM International Business Machine Corporation

Form GX21-9093-2 Printed in U.S.A.

Program		Punching Instruction	Graphic	Card Electro Number
Programmer	Date		Punch	

Page 1 2 of __ Program Identification 75 76 77 78 79 80

C

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And	And	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus	Minus	Zero	Compare: 1>2	1<2	1=2	Lookup(Factor 2)is: High	Low	Equal	Comments
01	C		01		TOTORD	ADD	1	TOTORD	30												
02	C					MOVE		BLANKS	1												
03	C																				

RPG OUTPUT SPECIFICATIONS

IBM International Business Machine Corporation

GX21-9090-2 U/M 050* Printed in U.S.A.

Program		Punching Instruction	Graphic	Card Electro Number
Programmer	Date		Punch	

Page 1 2 of __ Program Identification 75 76 77 78 79 80

O

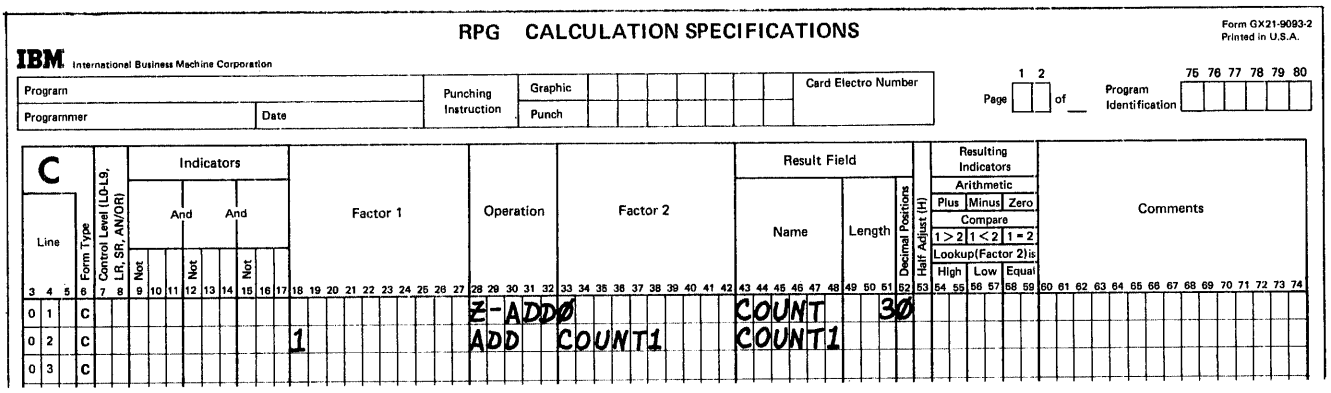
Line	Form Type	Filename	Type (H/D/T/E): O A / R N / D	Stacker # /Fetch(F)	Space: Before	Space: After	Skip: Before	Skip: After	Output Indicators: And	And	Field Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O		D						01		*AUTO					'INVOICE'
02	O										INVNO		C			'NUMBER'
03	O										GROAMT		A			'GROSS AMOUNT'
04	O										DISCNT		A			'DISCOUNT'
05	O										NETAMT		A			'NET AMOUNT'
06	O										BLANKS		A			'NUMBER OF'
07	O												C			'INVOICES'
08	O															
09	O		T						LR							
10	O										TOTORDZ			58		

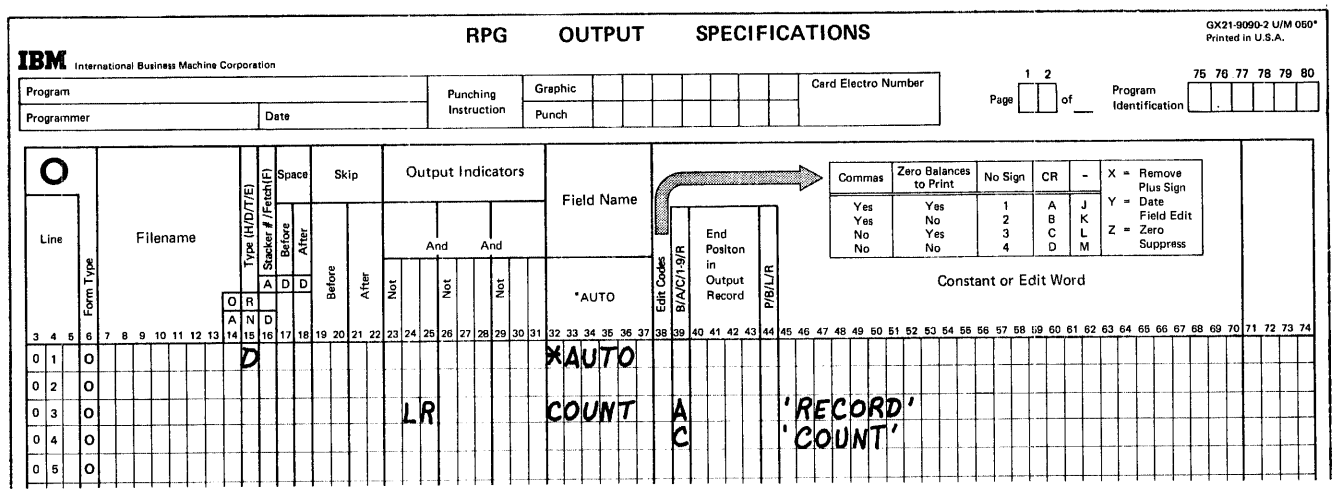
Commas	Zero Balances to Print	No Sign	CR	-
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

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

6. Another method of counting records is shown below. This method is especially applicable when:

   a. You are doing a detail list.
   b. You do not want 1's listed down the page.
   c. You want to take totals by control level.





Calculation Specifications

Line 01 This instruction is needed only to define the field COUNT for accumulation.

Line 02 This instruction accumulates the total for the first control level.

Output Specifications

Line 03 This instruction causes the generation of calculation and output specifications for the detail and total lines. The LR conditioning indicator prevents the generated detail calculation from occurring. It also prevents printing at detail time.

Note: If no control levels are specified in the program, a 1 would be added to COUNTR rather than COUNT1 on the calculation specifications.

Appendix A. Glossary

This glossary contains some terms that are used in this manual. Data processing terms are defined in *IBM Data Processing Glossary*, GC20-1699.

IBM is grateful to the American National Standards Institute (ANSI) for permission to reprint its definitions from the *American National Standard Vocabulary for Information Processing* (Copyright © 1970 by American National Standards Institute, Incorporated), which was prepared by Subcommittee X3K5 on Terminology and Glossary of the American National Standards Committee X3.

ANSI definitions are identified by an asterisk. An asterisk to the right of the term indicates that the entire entry is reprinted from the *American National Standard Vocabulary for Information Processing*; where definitions from other sources are included in the entry, ANSI definitions are identified by an asterisk to the right of the item number.

$SOURCE file: The file into which the RPG II program reads the RPG II source programs.

access method: A technique for moving data between main storage and a disk or diskette. See also file.

accumulating: The process of totaling a particular field's values as records are being processed.

add file: An indexed or sequential disk file defined as an input, output, or update file to which records will be added if the appropriate entries are made on the file description and output specifications sheets.

address: A name, label, or number that identifies a register, location in storage, or any other data source.

ADDROUT file: A record address disk file produced by the sort program. It contains addresses of records in a disk file and can be used to process input or update files that are designated as primary or secondary files.

alphabetic characters: Letters A through Z and the special symbols #, $, and @.

alphameric characters: Composed of alphabetic and numeric characters.

application program: (1) A program written for or by a customer that applies to the customers job. (2) A program that specifies distinct operations to be performed on specific input to obtain unique output. Application programs operate directly on data to meet specific data processing requirements.

array: A systematic arrangement of elements in a table format.

array file: An input file containing array entries.

auto lines: A line which is a part of the auto report specification lines.

auto report function: A function of the RPG II program that accepts simplified specifications and standard RPG II specifications to generate a complete RPG II source program.

auto report program: A set of instructions (program) that use the RPG II auto report function. See *auto report function.*

backup diskette: A backup diskette contains information that has been copied from another disk or diskette. It is used in case the original is damaged.

block: A unit of space assignment for files on the disk. For instance, 1 block = 10 sectors.

byte: The representation of a character with eight binary bits. The amount of storage required for one EBCDIC character.

calculation specifications sheet: An RPG II coding sheet used to describe processing to be done by the program.

chained files: An input, output, or update disk file that uses the chain operation code to read records randomly or to load a direct file.

character: * A letter, digit, or other symbol that is used as part of the organization, control, or representation of data.

CMD: Command key.

collating sequence: The order each character holds in relation to other characters according to the bit structure.

combined file: Used as both an input and output file. Can be assigned only to a special device.

command key: By using upper case shift key, 16 different key functions are available. In RPG II the command keys correspond to the indicators KA-KN, KP, and KQ.

command statement: A command statement is used to request the performance of a particular function. It always contains the command name and may include parameters. Specifically, a command statement is a special form of the / / INCLUDE statement. A command statement evokes a procedure and can pass information to the procedure via parameters included in the statement. The procedure named by the command name is evoked by the command statement.

comments: Words or statements in a program that serve as documentation rather than as instructions to the compiler.

compile: To translate a source program (such as RPG II specs) into an object program (machine language program) using the computer.

compile-time table or array: A table or array compiled with the source program that becomes a permanent part of the object program. See also *execution-time array* and *preexecution-time table or array*.

compiler: A program that translates a source program into an object program.

computer: A programmable device or group of devices capable of accepting, processing, and reporting information.

conditioning: Using indicators to control when calculations or output operations are done.

consecutive processing: A mode of file processing that reads records in the order they appear in the file.

console files: Files used with the interactive data entry function.

constant: A data item that does not change during execution of a program. This item represents itself and is actually used in processing rather than being a field name representing the data. For example, cost is a name representing a field containing data which changes. The constant 100 is actual data used which does not change.

control break: A change in the contents of a control field.

control card and file description specifications sheet: An RPG II coding sheet on which the programmer gives information for computer control and descriptions of files used.

control field: A field within a record that identifies the record's relationship to other records (such as a part number in an inventory record). Control fields are compared from record to record to determine when certain operations are to be performed.

control group: A set of records all having the same control field information.

control level indicator: An indicator used to specify certain fields as control fields and to tell which operations to perform at total time.

control statement: A specification that provides special information about the program and describes the system to the RPG II Compiler.

control unit: An area inside the processing unit that determines from instructions what has to be done. It directs other units or devices to perform the required functions.

CRT: Cathode ray tube. See *display screen.*

data: A collection of facts, numbers, letters, and symbols that can be processed by a computer.

data file: A collection of related data records organized in a specific manner. For example, a payroll data file may consist of one record for each employee showing his name, social security number, pay rate, etc.

data processing: Performing a series of planned instructions on information to achieve a desired result.

demand file: Can be used as input, updated, or combined and is used with the read operation code or key operation code.

descending order: The arrangement of data in a specified field from high to low.

detail record: An output record produced during the detail output operation of the RPG II program cycle.

detail time: An operation in the RPG II program cycle in which calculation and output operations are performed for each record read.

diagnostic message: An output message that identifies RPG II specification errors and their severity.

digit: One of the characters 0 through 9.

direct file: A disk file in which records are assigned specific record positions. Regardless of the order in which records are put in a direct file, they always occupy the assigned position (a specific disk address). Direct files can be processed using the consecutive, random by relative record number, and ADDROUT file processing methods.

disk: (1) The permanently attached magnetic disk storage. (2) A thin circular base coated with magnetic material on which data is recorded as magnetic spots. The System/32 disk is an integral part of the system used primarily for storage on frequently run programs and large volumes of frequently used data.

disk file: A group of related records stored on disk.

diskette: A small flexible magnetic disk permanently enclosed in a protective jacket. Diskettes are a removable storage medium used to store information until it is required for processing. See *diskette interchangeability.*

diskette interchangeability: The ability to read a diskette on a system other than the system used to record the information on the diskette.

display screen: The cathode ray tube screen on which the system displays data, messages, and other information for the operator.

documentation: A written explanation of a program, its use, function and operations.

EBCDIC: Extended binary coded decimal interchange code.

edit: To punctuate a field by suppressing zeros and inserting commas, decimal points, dollar signs, or other constant information.

edit code: A number or letter indicating that editing should be done according to a predefined pattern. This includes zero suppression and punctuation.

element: The smallest addressable unit of a table or array.

end of file: The end of records in a file.

EOF: End of file. The last record in a file.

EOJ: End of job.

erase: Removal of a unit of data.

error message: See *diagnostic message.*

execute: To process input data files according to machine language instructions to produce the desired output.

execution-time array: An array that is loaded by input specifications after actual execution begins. See *compile-time array* and *preexecution-time table or array.*

extension and line counter specifications sheet: An RPG II coding sheet used to provide information about record address, table, and array files used by the program and the number of lines to be printed on the forms that are used.

external indicators: (1) Used to select which files are to be used in multifile processing. (2) Eight indicators (U1-U8) that are normally set by the / / SWITCH statement prior to processing. The indicators can be altered by the job during execution.

factor: In RPG programming, a field name or constant used in a calculation operation.

field: One or more adjacent record positions which contain related information.

field indicator: An indicator used to determine if a given field on an input record is plus, minus, zero, or blank.

field length: The number of columns allowed for a given field, determined by the maximum length of information that will be entered in the field.

field name: In RPG programming, a combination of six or less alphabetic or numeric characters which identifies a field.

file: An organized collection of related records. See *disk file, input file, output file, primary file, secondary file.* When discussing types of file organization see *sequential file, indexed file, direct file.* For types of file processing see *consecutive, sequential by key, random by key, random by relative record number, ADDROUT file.*

filename: The name associated with a file.

first page indicator: An indicator used to specify which lines (such as headings) should be printed on the first page only.

generated program: A program that has been compiled.

half adjust: A method of rounding off a number by adjusting the last digit.

heading: A constant, usually printed at the top of the page, identifying the information or report on that page.

hex: Hexadecimal.

hexadecimal: Pertaining to a number system with a base of 16; valid digits range from 0 (zero) through F, where F represents the highest units position (15).

IDE: Interactive data entry.

IDE files: See *console files.*

indexed file: A file in which the position of the records is recorded in a separate portion of the file called an index. The index contains an index key and disk address of each record in the file. Indexed files can be processed using the consecutive, sequential by keys, sequential within limits, random by key, and ADDROUT file processing methods.

indicator: A 2-digit or 2-character entry on the specification sheet used to tell when certain operations are to be performed. An internal switch used by the object program to remember when a certain event occurs and what to do when the event occurs. See *control level indicator, field indicator, first page indicator, last record indicator, overflow indicator, record identifying indicator, resulting indicator.*

initial program load (IPL): A process that causes the system control program to be loaded into main storage. This prepares the system for execution of jobs.

input: (1) Information to be transferred from disk or keyboard to storage. (2) Data that is to be operated on (processed) by the computer.

input file: A set of records a program uses as source information.

input specifications sheet: An RPG II coding sheet used to identify the different types of records in each input file and to describe the fields of each record.

inquiry mode: The system has received an inquiry request and is performing the inquiry function.

inquiry program: A program that is executed while the system is in inquiry mode.

inquiry request: A request by the operator that stops the job that is running so that another program or function can be done.

instruction: A statement that specifies an operation to be performed by the computer and the locations in storage of all data involved in that operation.

interactive data entry (IDE) mode: Entering data by interaction between the operator and the program. For example, a prompt appears on the display asking the operator to key in data.

IPL: Initial program load.

keyboard: The device with which the operator enters (keys) data into the system and issues requests for system functions. The keyboard has the standard typewriter keys for entering alphameric information, special function keys for requesting system functions and editing data already entered, and a cluster of 10 numeric only keys for keying numeric data rapidly.

keyboard files: A file to be used with either the KEY or SET operation code.

last record indicator: An indicator that signifies when the last data record is processed.

library: An area on the disk that contains procedure members, source members, load members, and subroutine members as well as areas required by the system control program.

library member: A named collection of statements or records in the disk library that can contain source statements, format descriptions, OCL statements, utility control statements, or executable instructions.

limits file: A record address file containing limits records when using sequential within limits processing.

limits record: Consists of the lowest record key and the highest record key of the records in the indexed disk file which are to be read.

literal: A symbol or a quantity in a source program that is itself data, rather than a reference to data.

load member: A collection of instructions, stored in the library, that the system can execute to perform a particular function, whether the function is requested by the operator or specified in an OCL statement.

look ahead field: Allows you to look at information in a field on the next record that is available for processing in any input or update file.

LR indicator: Used to condition all operations that are to be done at the end of job (EOJ).

machine language: A language that can be interpreted and used by a computer.

main storage: The general purpose storage of a computer. Main storage stores program instructions for a job and data to be processed during that job.

master record: A record whose information rarely changes such as a name and address record.

match field: In multifile processing, the field that can specify the order in which records are processed.

match level: The value assigned to the match field (M1-M9). It is used to identify fields by which records are matched during multifile processing.

message identification code: A 4-character identifier associated with a specific error or informational message.

MIC: Message identification code.

null response: Enter key was pressed by the operator without any previous information being keyed.

numeric characters: The characters 0 through 9.

object program: A set of instructions in machine language. The object program is produced by the compiler from the source program.

OCL: Operation control language.

OCL statement: A statement that is used in identifying the job or its requirements to the operating system.

operation: A defined action performed on one or more data items, that is, adding, multiplying, comparing, or moving information.

operation code: A word or abbreviation specified on the calculation specification sheet to identify an operation, that is, SUB for subtract, ADD for addition.

operation control language: The control language used to communicate with the systems control program. OCL is composed of statements with which specific system functions are requested.

OR relationship: The specifying of conditioning indicators such that the operation conditioned is performed when either one or both of the conditions are met.

output: Data transferred from storage to printed form, disk, or to the display screen.

output file: A set of records that is written or printed by the computer.

output specifications sheet: An RPG II coding sheet used to specify the records to be written in each output file and the format of the records.

overflow: The condition that occurs when the last line to be printed on the page has been passed.

overflow indicator: An indicator that signifies when the last line on a page has been printed or passed. It can be used to specify which lines are to be printed on the next page.

overflow line: The line specified to be the last line printed on a page.

overflow page: The new page after an overflow has occurred.

overlay: A program segment or phase that is loaded into main storage. It replaces all or part of a previous loaded segment.

packed data field: One byte is used to store two numeric digits. Bits 0-3 for one digit and 4-7 for the other.

packed decimal format: A data format that allows a byte of disk storage (except for the low order byte) to contain two decimal numbers.

packed field: A field in packed decimal format.

packed keys: An index key in packed decimal format.

preexecution-time table or array: A table or array that is loaded with the object program before actual execution of the program begins. See *compile-time table* and *execution-time table.*

primary file: The main file from which a program first reads records. In multifile processing, it is used to determine the order in which records are selected for processing.

printer: The output media that records information on paper in the form of printed characters.

printer spacing chart: A form used to plan the location of data in the output file.

procedure levels: The level of the procedure when procedures are nested.

procedure member: A named collection of related OCL statements, and possibly, utility control statements stored in the library.

processing: The handling of input according to specific instructions or rules; performing a series of planned actions upon information (data) to achieve a desired result.

processing unit: (1) The part of a computer that provides storage area for the programs and data and performs the operations specified in the program. (2) The control center of System/32. It monitors the flow of information into the system, performs calculations and other operations on data, and regulates the flow of output.

program: (1) A sequence of precise instructions to a system written in a special form the system can interpret. A program tells the system where to get input, how to process it, and where to put the results. (2) A set of instructions that tells the computer which operations are to be done and how to do them. See *object program* and *source program*.

program cycle: A series of operations performed by the computer for each record read.

program listing: A computer printout which gives information about the source program, such as source statements, diagnostic messages, indicators used, storage addresses of fields and constants used.

program name: Column 75 through 80 on the RPG II specifications sheet. Provides a means of identifying a program. You may name the program according to its function or use any letters or numbers to identify the program.

random by key: A mode of processing chained files using the CHAIN operation code. Records to be processed are identified by record keys.

random by relative record number: A mode of processing chained files using the CHAIN operation code. Relative record numbers are used to identify records to be processed.

record: A collection of related data, treated as a unit. For example, one line of an invoice may form a record. A complete set of records may form a file.

record address file: An input file that indicates to your program which records are to be read from a disk file, and the order in which the records are to be read from the disk file.

record identification code: A code placed in a record to identify that record type.

record identifying indicator: An indicator that signifies the type of record to be processed next.

record key: One or more characters within an item of data that are used to identify the data.

record length: The number of characters needed to include all the data for one record.

record types: Records from one file which have different fields and/or format.

relative record number: In a direct file a record is written and retrieved directly by specifying the location of the record in relation to the beginning of the file. This relative position is known as the relative record number.

result field: The name of a field where the outcome of arithmetic calculations is kept.

resulting indicator: An indicator that signifies if (1) The result of a calculation is plus, minus or zero. (2) The field is greater than, less than, or equal to another field.

right justify: The placement of data in a field with least significant digit in the rightmost position.

rollin: The program that has been interrupted by an inquiry is rolled back into main storage when the inquiry is completed.

rollout: When the system receives an inquiry request, the program that is operating at the time is rolled out to a disk area reserved for this purpose, and the inquiry program is executed.

RPG II: A commercially oriented programming language specifically designed for writing application programs that meet common business data processing requirements.

RPG II cycle: Logic steps in the RPG II process.

RPG II source program: The program used as input to the RPG II Compiler. The program is translated into machine language and stored in the library as a load member.

search word: Data for which you want to find a match in a table or array. The search word is specified in the LOKUP statement.

secondary file: Any file other than the primary file used in multifile processing.

sequential by key: A mode of file processing that reads records in the order in which the record keys are arranged in the index portion of the file.

sequential file: A file in which the order of records is determined by the order that they are put in the file. For example, the tenth record entered occupies the tenth record position. Sequential files can be processed using the consecutive, random by relative record number, and ADDROUT file processing methods.

sequence checking: An RPG II function that checks the sequence of records in input, update, or combined files used as primary and secondary files.

source member: A collection of records used as input for a program product such as RPG II specifications. Source members are stored in the library. A source member contains information needed for processing data.

source program: A set of instructions that represents a particular job as defined by the programmer. These instructions are written in a programming language such as RPG II.

special character: A character other than a digit or letter. For example, *, +, %. In RPG II programming, @, #, and $ are considered alphabetic characters.

specification sheets: Forms on which an RPG II program is coded and described. See *control card and file description specifications sheet, extension and line counter specifications sheet, input specifications sheet, calculation specifications sheet, output specifications sheet.*

storage unit: An area inside the processing unit where instructions and data are stored.

subroutine: A group of instructions in a main routine (program) that are executed several times in one program run.

subroutine member: Subroutines that need to be link edited before being loaded for execution. Subroutine members are stored in the library.

system library: An area on the disk used for storing load members, procedure members, subroutine members, and source members.

table: A systematic arrangement of data items.

table file: An input file containing table entries.

total operations: Operations performed only after a group of records has been processed.

total rolling: The transfer of accumulated totals from one field to another during a control break.

total time: That part of the RPG II program cycle in which operations specified for a group of records are done.

unpacked data field: One byte contains one character. Bits 0-3 are the zone portion and 4-7 are the numeric portion.

update file: Disk files from which a program reads a record, updates fields in the record, and writes the record back into the location it came from.

valid RPG II names: The following rules apply to names used in the RPG II programs. (1) RPG II filenames can be from one to eight characters long. (2) RPG II field names can be from one to six characters long. (3) The first character of either a filename or a field name must be an alphabetic character. The remaining characters can be any combination of alphabetic and numeric characters. (4) Blanks cannot appear between characters in a name.

zero suppression: The elimination of preceding zeros in a number. For example, 00057 when zero suppressed becomes 57.

IBM Technical Newsletter

This Newsletter No. SN21-7792
Date 15 May 1975

Base Publication No. SC21-7595-0
File No. S32-28

Previous Newsletters SN21-7809
SN21-5313

IBM System/32
RPG II Reference Manual

This Technical Newsletter, a part of version 02, modification 00 of IBM System/32 RPG II (Program Product 5725-RG1), provides replacement pages for the subject publication. These replacement pages remain in effect for subsequent (versions and modifications) unless specifically altered. Pages to be inserted and/or removed are:

iii, iv	1-39.1, 1-39.2 (added)	1-155 to 1-158	X-9, X-10
ix, x	1-45 to 1-48	2-77, 2-78	X-15, X-16
1-3 to 1-12	1-59, 1-60	2-81, 2-82	
1-15, 1-16	1-59.1, 1-59.2 (added)	2-93 to 2-96	
1-15.1 to 1-15.4 (added)	1-61 to 1-68	3-7, 3-8	
1-19, 1-20	1-67.1, 1-67.2 (added)	3-29, 3-30	
1-23, 1-24	1-99, 1-100	3-37, 3-38	
1-29, 1-30	1-121, 1-122	4-69, 4-70	
1-39 to 1-42	1-149, 1-150	X-1 to X-4	

Changes to text and illustrations are indicated by a vertical line at the left of the change; new or extensively revised illustrations are denoted by the symbol • at the left of the caption.

Summary of Amendments

- BSCA telecommunications support information has been added.
- Shared input/output access method (SIAM) support information has been added.
- Miscellaneous corrections have been made.

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

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

Printed in U.S.A.

IBM **Technical Newsletter**

This Newsletter No.	SN21-5313
Date	2 January 1975
Base Publication No.	SC21-7595-0
File No.	S32-28
Previous Newsletters	None

IBM System/32
RPG II
Reference Manual

This Technical Newsletter, a part of version 01, modification level 00, provides replacement pages for the subject publication. These replacement pages remain in effect for subsequent (versions and modifications) unless specifically altered. Pages to be inserted and/or removed are:

1-35, 1-36

Changes to text and illustrations are indicated by a vertical line at the left of the change; new or extensively revised illustrations are denoted by the symbol • at the left of the caption.

Summary of Amendments

Contains a caution regarding the alteration of files while in inquiry mode.

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

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

Printed in U.S.A.

READER'S COMMENT FORM

IBM System/32
RPG II
Reference Manual

SC21-7595-0

YOUR COMMENTS, PLEASE . . .

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

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

Corrections or clarifications needed:

Page	*Comment*

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

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

Cut Along Line

Fold

Fold

FIRST CLASS
PERMIT NO. 387
ROCHESTER, MINN.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

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

Fold

Fold

IBM S/32 RPG II Reference Man File No. S32-28) Printed in U.S.A. SC21-7595-0

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

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

IBM S/32 RPG II Reference Manual (File No. S32-28) Printed in U.S.A. SC21-7595-0

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

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