MRX/OS RPG II

Reference Guide

2202.003

MEMOREX

Computer System Products

Preliminary Information

This manual precedes initial release publications and therefore may undergo substantial revision.

PRELIMINARY EDITION (JULY 1972)

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

A readers' comments form is provided at the back of this publication. If the form has been removed, comments may be addressed to the Memorex Corporation, Publications Dept., Santa Clara, California 95052.

MRX/OS RPG II
REFERENCE GUIDE

MEMOREX RPG II

REFERENCE GUIDE

This manual serves as an introduction to the MEMOREX RPG II System. It presents a general description of the report program generator and the various specifications that compose its unique programming language.

Section 1 provides a system overview, reviews the types of specification sheets, and discusses the overall performance of the system.

Sections 2 through 8 deal with the use of RPG II's programming language. Each section is devoted to one type of specifications sheet and the coded information that is to be specified on it.

Appendices A through C provide a detailed flowchart of the Memorex RPG II system, supplementary reference tables, and definitions of RPG II terminology.

Appendix D contains a formula for computing block lengths, while Appendix E describes RPG II subroutine linkage and the parameters for special files.

Control Language Statement considerations and examples appear in Appendix F. Further information on this subject appears in Memorex publication 2200.004: Control Language Services—— Extended.

Appendix G contains an explanation of the error message format and lists the error messages by the overlay in which they appear.

Appendix H describes the operational environment for the RPG II compiler and gives the operator options in response to error conditions.

Future releases of the RPG II Reference Guide will include these additions:

- A sample program and programming tips.
- Coding examples for tables and arrays, indicator assignment, and output features.

Memorex brochure 2282.003, RPG II Reference Summary, provides a condensed version of the information found in this manual.

Preliminary Information; supersedes all previous Preliminary Copy and Review Copy releases.

TABLE OF CONTENTS

TOTAL PAGES: 340

TABLE OF FIGURES

SECTION 1. INTRODUCTION

RPG II OVERVIEW

MEMOREX RPG II is a report program generator which consists of a unique symbolic programming language and a compiler which translates that language into machine instructions. The System executes as an application under control of the Memorex Operating System (MRX/OS).

MEMOREX RPG II is a straightforward system that is readily adaptable to most programming situations and capable of managing complex problem situations efficiently. The System is designed to assist the programmer in organizing job requirements to handle a number of important functions with minimum effort. Through simple language entries on specially designed coding sheets, the programmer can use RPG II to perform extensive calculations; to design routines which handle tables and arrays simultaneously; to create tape, disc, punched-card, or console files; to update existing disc files; and ultimately to generate reports that are tailored to a wide variety of user applications. Because of its complete flexibility, RPG II can thus be used for almost any business data processing application.

An RPG II application is accomplished in three job stages:

Analysis
Compilation
Execution

During the Analysis Stage, the programmer is involved in organizing his information and determining the information that is to be generated. He decides what input data, output format, and calculations are necessary, and codes the information he has gathered in symbolic programming language. In this manner, he creates the source program, which is then punched into cards to form the source program deck.

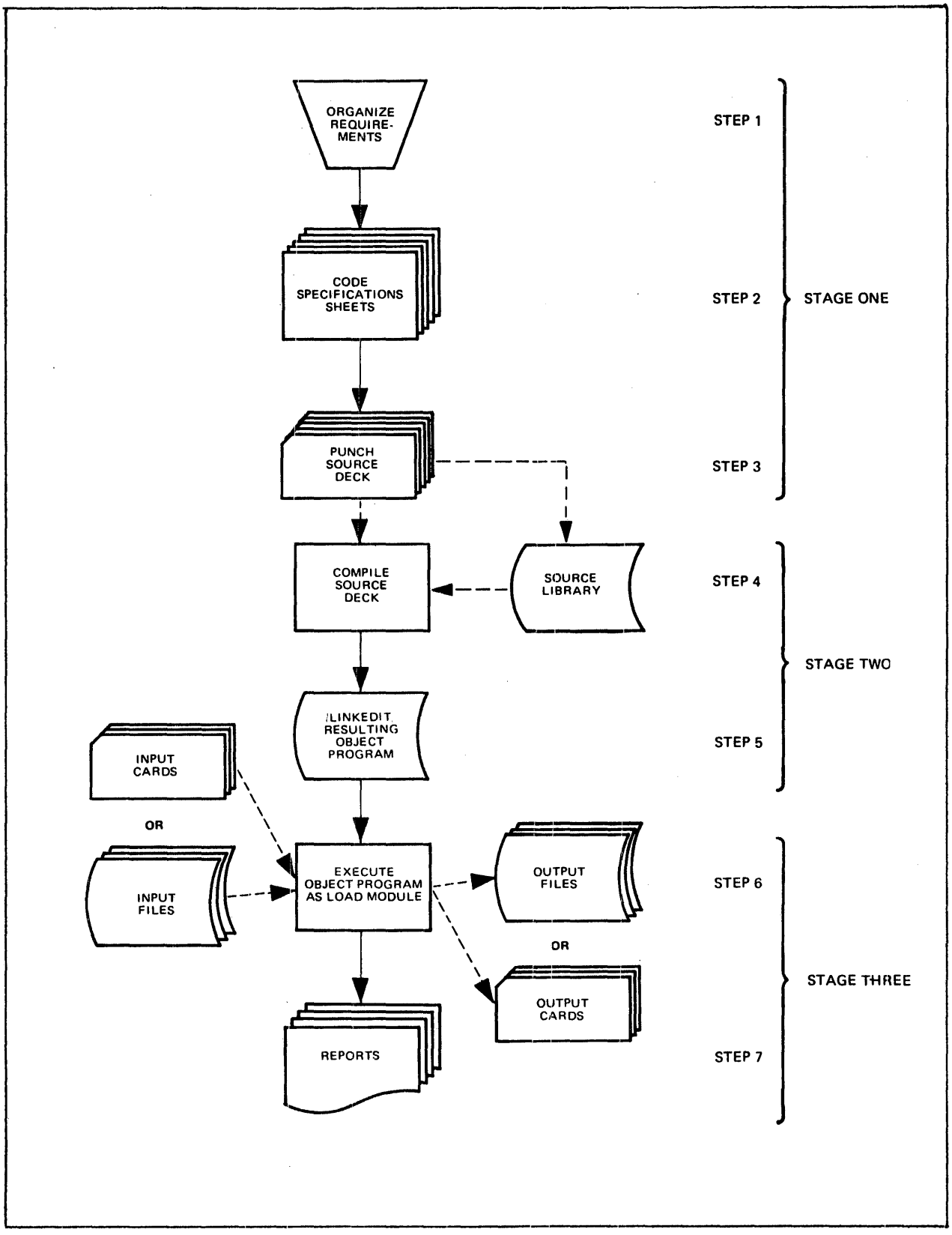


Figure 1-1. RPG II System Processing Overview.

In the Compilation Stage, the compiler translates the source program deck into machine language instructions. Much of the program detail implied by the programmer is actually added by the compiler; the compiler, for instance, assigns the proper disc storage areas and creates routines for such functions as input/output control. At the end of the compilation process the object program that has been generated is ready for the Linkage Editor to prepare it for load module execution of the programmer's instructions.

It is during the third stage, the execution of the object program, that the newly generated machine language instructions manipulate information from the data input files to produce output files and reports. In this stage calculations are performed; new files are created on disc, tape, punched cards, or the console; existing disc files are updated; information is merged for various purposes; and single or multiple reports are generated. (See Appendix A for a detailed flowchart of the object program cycle.)

This three-stage approach to report generation assists the programmer in concentrating on each individual area of activity. The special programming language increases source program code efficiency and the automatic compiler interpretation frees the programmer from delving into detailed machine language logic. MEMOREX RPG II is a versatile, easy-to-use method of report program generation.

Analysis of the three stages shows that each can be further subdivided into chronological steps. Figure 1-1 diagrams the seven operating steps in the three stages and shows the easy, natural flow of the entire process from the program design (Step 1) to the generation of multiple reports and the output files (Step 7). The steps are described in more detail below.

STAGE ONE: ANALYSIS

Step 1 - The programmer analyzes the job requirements and organizes all available information.

Step 2 - The programmer codes the job requirements on the proper RPG II specifications sheets:

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

Step 3 - The data is transferred from the specifications sheets to punched cards. Each line of a specification sheet is punched onto one card, and the completed cards form the source program deck.

STAGE TWO: COMPILATION

Step 4 - Either the source program deck is placed in a card-reading device and read into computer storage, where the compiler program translates the source program into the object program under control of the MEMOREX System; or the deck is put in a source library and read from the library by the compiler. The resulting object program contains all the machine instructions required to produce the output files and reports, but must be linkedited before its actual execution.

Step 5 - The programmer corrects errors and the program is recompiled until all serious errors are eliminated. The object program is then stored on the disc as a relocatable object module, ready to be linkedited. The linkage editor then ties together all subroutines required by the program. When this has been accomplished, the object program is ready for execution.

STAGE THREE: EXECUTION

Step 6 - The load module is read into computer storage and executed. Under control of the load module, the System reads and processes data from available input files specified by the programmer.

Step 7 - The processed data is moved into the specified format by the load module and punched into cards, written on disc, printed in the form of one or more reports, or written to the console.

STAGE ONE: ANALYSIS

As Figure 1-1 shows, the Analysis stage of the System comprises three steps: organizing the requirements, coding the specifications, and punching the source cards.

STEP 1 - ORGANIZING THE REQUIREMENTS

The first step in the process of generating output files or reports is to organize the job requirements. The programmer identifies the information which is to be output and the format it is to take, selects the input fields he can use, and determines the calculations which must be performed to generate the remainder of the output information. He also decides whether the information is to be generated on punched cards, disc, printed pages, or the console, and identifies the necessary system devices to be used. A chart showing the flow of information through the various calculations and decision functions may be helpful in analyzing output requirements.

After the programmer has determined his requirements and the sequence in which they are to be performed, he is ready to transfer the information to the RPG II Specifications Sheets. The specifications sheets deal with seven distinct areas of information, some of which is mandatory and some of which is optional. The sheets are designed to guide the programmer in gathering information and coding the language with a minimum of work.

STEP 2 - CODING THE SPECIFICATIONS

MEMOREX RPG II programming language is coded on five specifications sheets. Collectively the specifications sheets provide the programmer with forms for coding his source information. Individually each form deals with a distinct area of concern such as the input to be used, the format of the output, or calculations that are to be performed. Each specifications sheet listed below is discussed in detail in its own section of this manual.

The Control Card and File Description Specifications Sheet serves a dual function. The control card section provides the compiler with pertinent information about the source program and the system being employed. The file description portion describes the organization and characteristics of the input and output files required by the object program.

The Extension and Line Counter Specifications Sheet also contains two types of specifications. The extension section describes the record address files, tables, and arrays required in the job. The line counter lines associate channel numbers of the printer carriage control with report lines of printer reports so that they can then be output to the disc, instead of the printer.

The Input Specifications Sheet identifies the input records and defines the fields contained in each record type.

The Calculation Specifications Sheet details any arithmetic processing or data manipulation that is to be performed on the input data.

The Output-Format Specifications Sheet describes the type of data that is to be written or punched and specifies its arrangement.

STEP 3 - PUNCHING THE SOURCE PROGRAM DECK

The final step in the Analysis Stage occurs after the specifications sheets coding has been completed. The sheets are arranged in a specified order and submitted for keypunching. Each line of data is transferred to one punched card. When all the coded specifications sheets have been converted to punched cards, the cards are put in order to form the source program deck (Figure 1-2). Since a source program does not necessarily require that all the specifications sheets be coded, those corresponding parts of the source deck that may optionally be included are indicated.

The creation of the source program deck concludes Stage One of RPG II System generation.

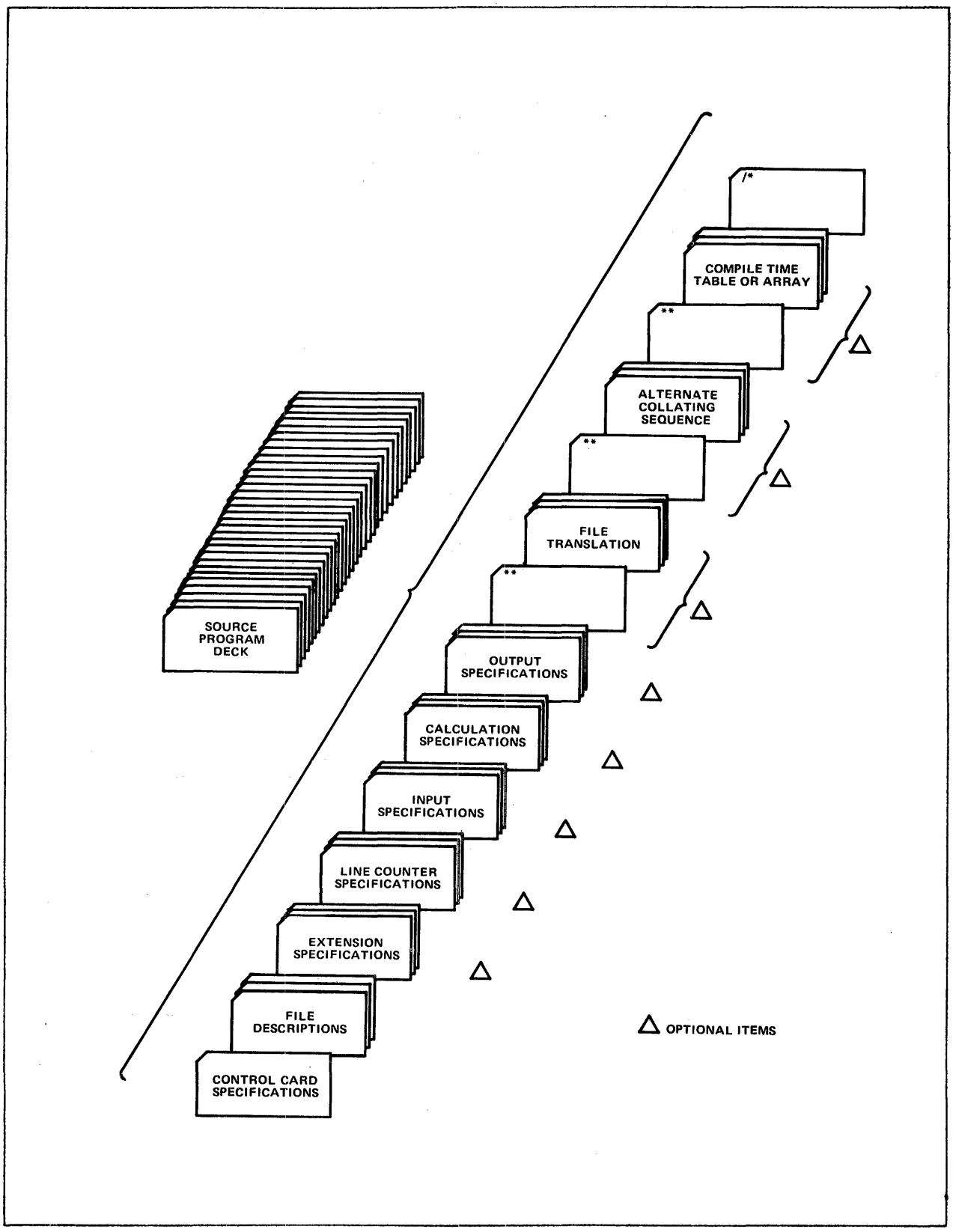


Figure 1-2. Source Program Deck.

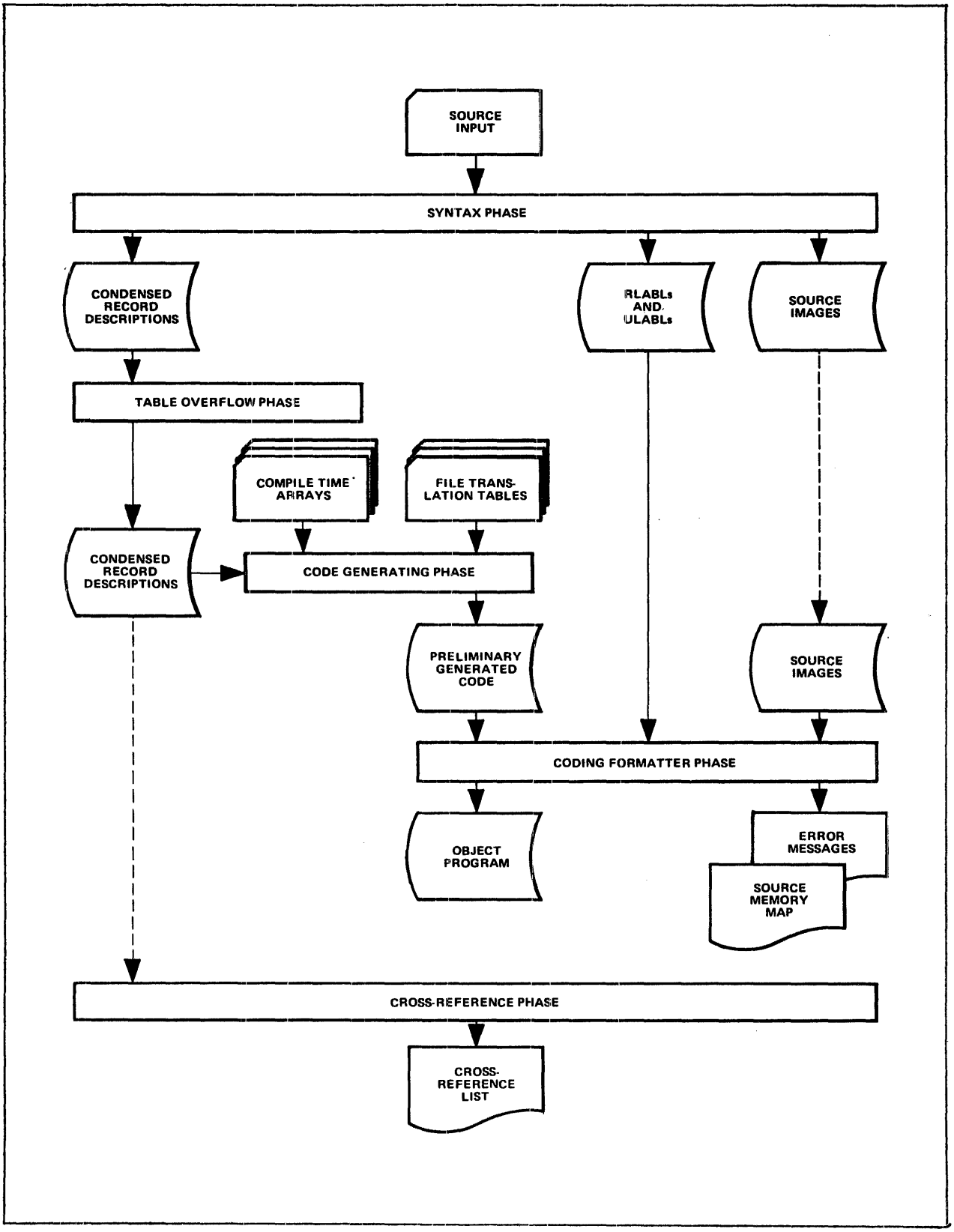


Figure 1-3. RPG II Compiler Structure.

STAGE TWO: COMPILATION

In Stage Two, the emphasis shifts from the language to the compiler that converts the source program deck into machine language instructions (Figure 1-3).

The Compilation Stage of the report generator is divided into two steps: compiling the source program, and linkediting the translated program for execution as a load module.

STEP 4 - COMPILING THE SOURCE PROGRAM

The RPG II compiler contains approximately 25,000 instructions plus object time subroutines, data management routines, tables, and buffers. The compiler is designed for batch processing, and uses a two-pass mode of operation: scan, and code generation. It consists of one resident module, which is the executive routine; and four main phases with fifteen overlays.

To understand the way the compiler works, the six divisions within the compiler must be examined. The executive routine contains the input/output interfaces, table lookup routines, and other common subroutines. It is the primary section of the compiler and is always present in computer storage during compilation. The other five parts are called phases. They are the Syntax Phase, the Table Overflow Phase, the Code Generation Phase, the Code Formatter Phase, and the Cross-Reference Phase.

The Syntax Phase scans the control card, the file descriptions, and all the other input cards, checking the accuracy of the source program deck and locating errors in format and content.

The Table Overflow Phase monitors resident field tables and initiates special passes, when necessary, to correct table overflows.

The Code Generation Phase does the actual translation, converting the source program into machine language instructions, and processing any compile time tables and arrays.

The Code Formatter Phase lists error messages, the source program, a memory map, external references, and entry points; then completes the compilation activity by writing out the generated object program in relocatable format.

The Cross-Reference Phase is optional; it produces a cross-reference list of file names, field names, and indicators, along with their storage locations. The data length and type (numeric or alphanumeric) are included for field names.

The minimum machine configuration required for compiling and executing an RPG II program consists of the following:

16K bytes of computer storage (K = 1,024). 8K bytes of computer storage are required for the compiler and its tables, and 8K bytes are necessary for the Operating System, including its tables and buffers.

One disc storage unit.

One central processing unit.

One operator's console (printer keyboard).

Any devices necessary to the object program, such as a card reader, a printer, or tape units.

Compilation time is a product of machine size and the number of field names and statements used in the source program.

The size of the source program is not limited in using the MEMOREX RPG II System, but the object program generated during compilation is limited to 64K bytes of computer storage (K = 1,024).

To initiate the process of compilation, the source program deck is placed in a card-reading device and read into main storage. Compilation then takes place without further programmer intervention. The source program instructions are processed through the four phases of the compiler and an object program is generated.

STEP 5 - PREPARING THE OBJECT PROGRAM FOR EXECUTION

The object program is automatically stored on disc by the System. Once compilation is completed the System also provides a listing of the object program and a list of any errors it has detected, so that the programmer can undertake corrective measures. After corrections are made to the source program, compilation is repeated until all errors are rectified.

The resulting error-free object program is a relocatable object module which is suitable for input to the linkage editor. This module contains external references to every subroutine it requires, so that it can be linked automatically with those

subroutines via the linkage editor. Supplied by the compiler automatically, the external references consist of fields, entry points, or program names and include all RPG II subroutines, all logical input/output subroutines, and any EXIT or ULABL names in the RPG II program (see Section 7, "Operation Codes").

Stage Two is completed when an error-free object program has been successfully generated and linkedited. The program is then ready to be executed as a load module to perform calculations, create or update files, generate reports, or fulfill other purposes.

STAGE THREE: EXECUTION

The Execution Stage takes care of thm last two steps of the RPG II System: executing the object program, and generating the desired output. Before the programmer can execute the object program, it must be read from the disc on which it is stored into the System, where it can control the System in processing input data as the data is read in from input files.

STEP 6 - EXECUTING THE OBJECT PROGRAM

RPG II program logic consists of six primary segments: housekeeping, detail operations, input operations, total calculations, total output operations, and detail calculations. The detail operations through detail calculations segments constitute the basic RPG II operating cycle.

Housekeeping tasks are performed prior to System entrance to the operating cycle. Housekeeping tasks include opening all files to be used in the program, pre-execution tables and array loading (after which the files are closed), and first page output.

Detail operations are performed at the beginning of the operating cycle. Heading and detail records are written and various program indicators are set off before the System proceeds to the next segment.

In the input operations segment, one input record is read and identified per cycle. The System selects the appropriate record (multifile input) for processing.

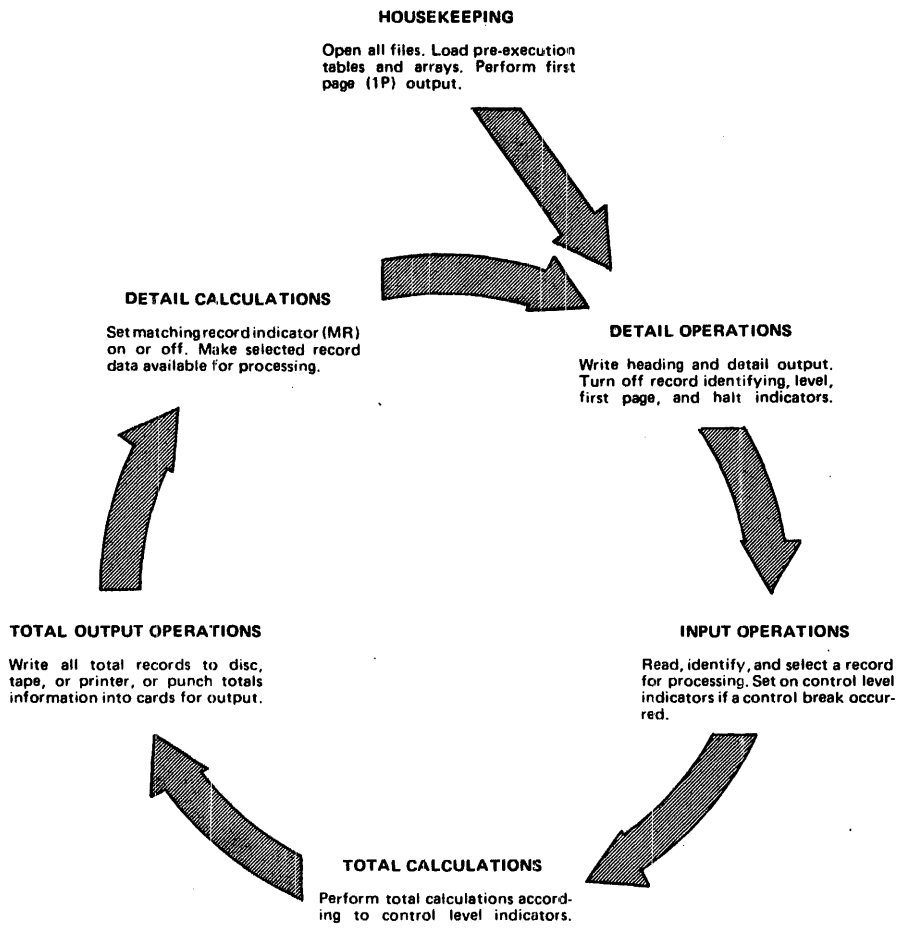


Figure 1-4. Object Program Operating Cycle.

In the next segment, total calculations are performed on the selected record for any control level indicators (L0-L9, LR) which are on. Although indicator L0 is always on, the others may be set on either by the programmer, or by a control break caused by a change in a selected control field. Total calculations are always done after each control break, and again after the last record has been processed. The operations to be performed in this segment are dictated by the status of specified control level indicators.

All records conditioned as totals by output indicators are written in the total output operations segment. End-of-job processing is initiated during this segment if the last record (LR) indicator is on.

The detail calculations segment includes setting the matching record (MR) indicator on or off. Data from the record selected for processing in the input operations segment is made available as the subject of detail calculations.

STEP 7 - GENERATING THE REPORT

The output of Stage Three is the report designed by the programmer in Stage One. The MEMOREX RPG II System helps create the report by providing, via language specifications sheets, a means of interpreting and coding the report requirements first devised by the programmer. During compilation, these requirements are translated into machine language instructions. The final step in the report generation process is the production of the report resulting from the programmer's specifications and the input data.

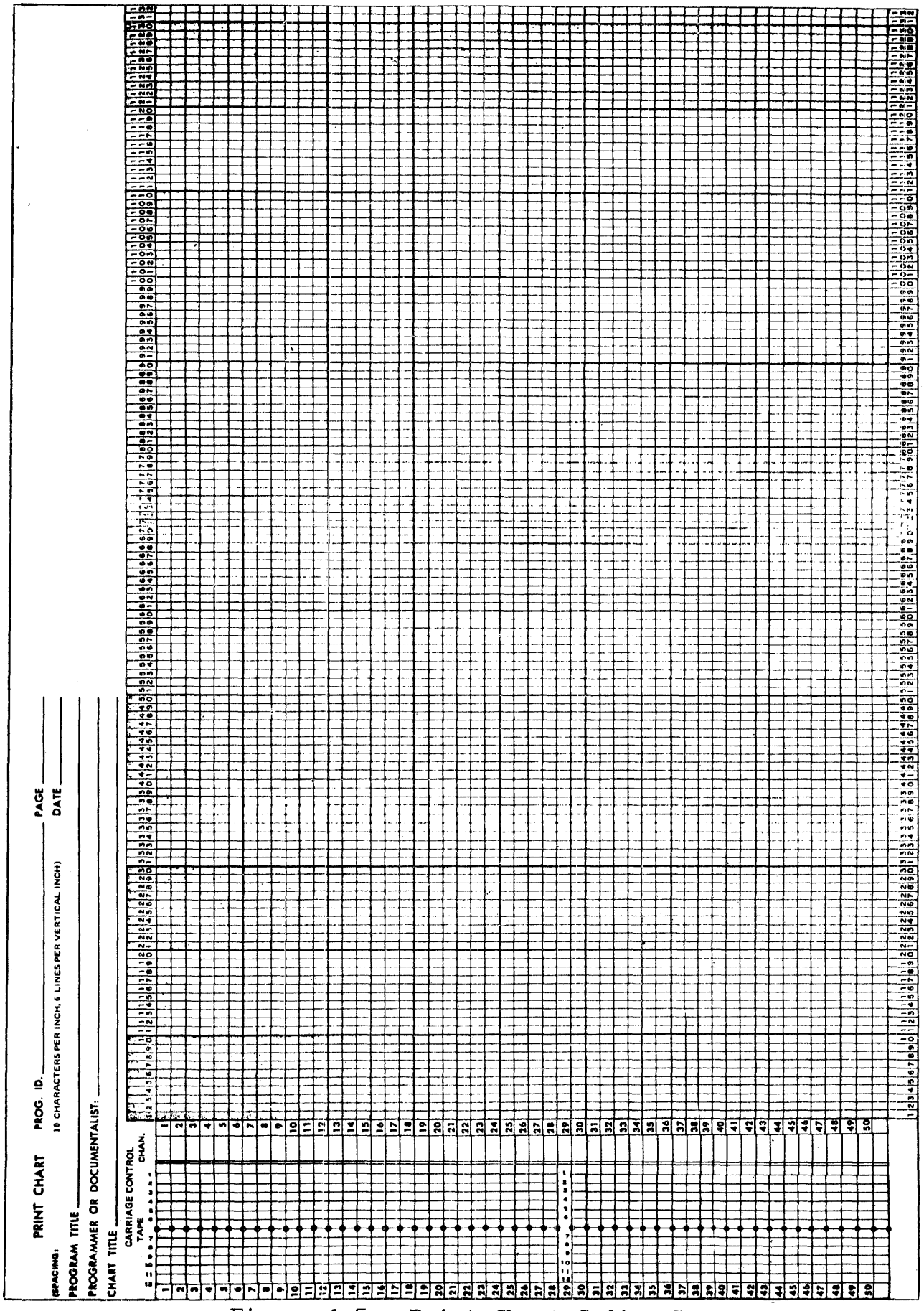


Figure 1-5. Print Chart Coding Form.

CODING INSTRUCTIONS FOR COMMON ENTRIES

Although each RPG II specifications sheet requires coding instructions peculiar to the type of specification, certain of the required entries are common to all the specifications sheets. These are:

PAGE NUMBER	Columns 1-2
LINE NUMBER	Columns 3-5
TYPE OF FORM	Column 6
COMMENTS	Column 7
PROGRAM IDENTIFICATION	Columns 75-80

General rules can be applied in coding these entries for all the specifications sheets.

PAGE NUMBER (Columns 1-2)

Entry	Description
01-99	Specifications sheets are numbered sequentially.

More than one of each type of specifications sheet, except the control card specifications sheet, may be used in the course of coding the source program. To keep the sheets in the proper order, the coded forms must be grouped together sequentially and arranged by type of specifications sheet in the following order:

Control Card and File Description

Extension and Line Counter

Input

Calculation

Output-Format

The specifications sheets should then be numbered in ascending sequence.

LINE NUMBER (Columns 3-5)

Entry	Description
Any number	Line numbers are assigned in ascending sequence.

Line numbers are preprinted in columns 3-4 of the specifications sheet for the programmer's convenience (the control card specification line is always 01). The unnumbered lines below the preprinted numbers can be used for additional lines or to insert a line between completed lines. Column 5 can also be used for this purpose. The line numbers assigned need not be consecutive, but they must be in ascending order.

TYPE OF FORM (Column 6)

Entry	Description
H	Control Card Specifications Sheet (header card).
F	File Description Specifications Sheet.
E	Extension Specifications Sheet.
L	Line Counter Specifications Sheet.
I	Input Specifications Sheet.
C	Calculation Specifications Sheet.
O	Output-Format Specifications Sheet.

Column 6 contains a preprinted letter on all sheets which identifies the type of specifications for each line.

COMMENTS (Columns 7-80)

Entry	Description
* (followed by any characters)	Annotation to assist the programmer in clarifying the program listing.

Comments may be inserted in any of the specifications cards by entering an asterisk in column 7 and any characters in the remaining columns for programmer notations. The compiler then processes only the first seven characters on the card, and ignores columns 8 through 80.

PROGRAM IDENTIFICATION (Columns 75-80)

Entry	Description
Blank	A program identification of RPGOBJ is assumed.
Valid RPG II name	The program identification should be a unique name for the object program.

The program identification can be one to six characters long. The compiler processes the name coded on the control card only; columns 75-80 are ignored on all other card types, but appear on the source listing. The name must begin with a character from A to Z. Other characters in the name may be A - Z, 0 - 9, and the pound sign (#), dollar sign ($), or at sign (@); but no imbedded blanks may be used. No special characters are allowed in the name.

SECTION 2. CONTROL CARD SPECIFICATIONS

Each RPG II source program deck must begin with a control card. The specifications on this card provide the compiler with information necessary to translate the source program into an object program. This information is derived from entries on the control card specifications portion of the RPG II Control Card and File Description Specifications Sheet (Figure 2-1). Since each source program requires only one control card, a single coding line is provided on the specifications sheet.

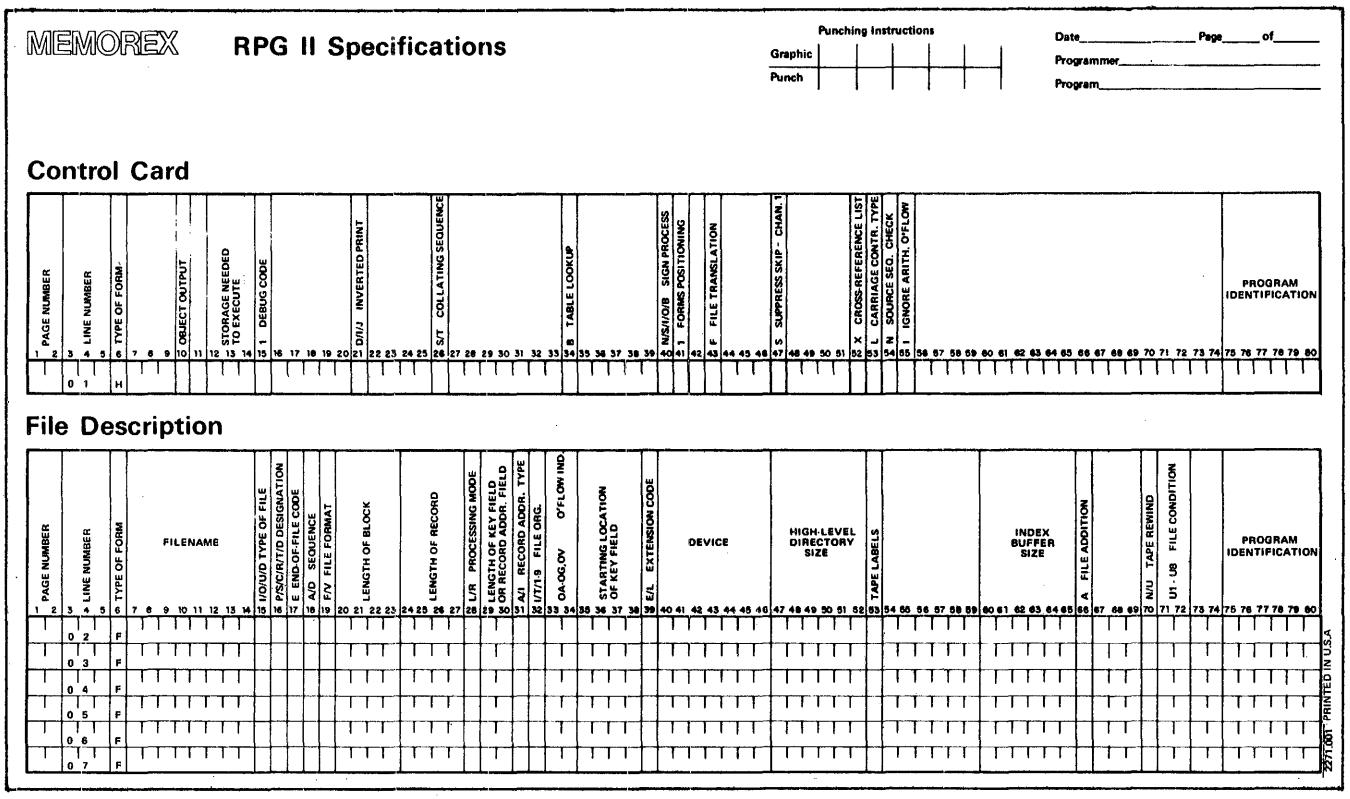


Figure 2-1. RPG II Control Card and File Description Specifications Sheet.

CODING INSTRUCTIONS

Detailed descriptions of the control card entries to be coded on the specifications sheet are defined below.

PAGE NUMBER (Columns 1-2)

Entry	Description
01	The control card page number is always 01.

More than one of each type of specifications sheet, other than the control card, may be used in the course of coding the source program. To keep the sheets in the proper order, the coded sheets must be grouped according to their type and arranged in the following order:

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

The specifications sheets should then be numbered in ascending sequence.

LINE NUMBER (Columns 3-5)

Entry	Description
01	The control card specification line is always 01 and is preprinted in the specifications sheet.

TYPE OF FORM (Column 6)

Entry	Description
H	H identifies the control card specifications to the compiler.

OBJECT OUTPUT (Column 10)

Entry	Description
Blank	The program is saved in the object library.
Any EBCDIC character	The object program is not to be written into the object library.

An EBCDIC character entered in column 10 prevents the object program from being written into the object library for execution. The disposition of the object program is then determined by MRX/OS Control Language Statement cards. If the column is left blank, the object program will be written to the object library.

STORAGE NEEDED TO EXECUTE (Columns 12-14)

Entry	Description
Blank	The amount of computer storage needed for execution is the same as that required for compilation.
004-064	The amount of computer storage needed for execution is entered when it differs from the amount of available storage.

The computer storage allotment is made in multiples of 1K bytes of storage, with K equal to 1,024. The entry must be at least 004. Even when an entry is thought to be unnecessary, careful consideration must be given to the amount of storage the object program will require.

DEBUG CODE (Column 15)

Entry	Description
Blank	If the DEBUG command is specified, it is to be ignored; no DEBUG operation is to be performed.
1	A DEBUG operation is to be performed.

Two entries are required to perform a DEBUG operation. In addition to the 1 entered in column 15 of the control card specifications, the DEBUG operation code should be entered on the Calculation Specifications Sheet (see Section 7, "Debug Operation").

INVERTED PRINT (Column 21)

Entry	Description
Blank	Domestic format is to be used.
D	United Kingdom format is to be used.
I	European Convention format is to be used.
J	European Convention format is to be used with leading zeros remaining for zero balances.

The inverted print entry determines the format and punctuation used for numeric literals, the edit codes used on output, and the order of the UDATE field (see Section 8, columns 32-37). The inverted print resulting from the format chosen is shown in Figure 2-2.

INVERTED PRINT OPTION	NUMERIC LITERAL, PERIOD OR COMMA AS DECIMAL POINT	EDIT CODES WITH PERIOD OR COMMA AS DECIMAL POINT	ZERO SUPPRESSION TO LEFT OR RIGHT OF DECIMAL POINT	UDATE SHOWING SLASH OR COMMA
Blank	1234.56	9,876.54	.12	MM/DD/YY
D	1234.56	9,876.54	.12	DD/MM/YY
I	1234,56	9.876,54	,12	DD.MM.YY
J	1234,56	9.876,54	0,12	DD.MM.YY

Figure 2-2. Inverted Print.

COLLATING SEQUENCE (Column 26)

Entry	Description
Blank	Normal collating sequence is to be used.
S or T	An alternate collating sequence is to be used.

The relative position each alphabetic, numeric, and special character holds in relation to the other characters is called the collating sequence. When the normal RPG II collating sequence (Figure 2-3) is to be used, column 26 of the control card specification is left blank.

MEMOREX Alternate Collating Sequence and Translation Coding Sheet

CODE	MRX GRAPHIC	ENTRY	REPLACES/ REPLACED BY
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	
00110011		33	

CODE	MRX GRAPHIC	ENTRY	REPLACES/ REPLACED BY
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	
01100110		66	
01100111		67	

CODE	MRX GRAPHIC	ENTRY	REPLACES/ REPLACED BY
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	
10011001		99	
10011010		9A	
10011011		9B	

CODE	MRX GRAPHIC	ENTRY	REPLACES/ REPLACED BY
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	
11001100		CC	
11001101		CD	
11001110		CE	
11001111		CF	

CODE	MRX GRAPHIC	ENTRY	REPLACES/ REPLACED BY
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	

227,006 PRINTED IN U.S.A.

Figure 2-3. Alternate Collating Sequence and Translation Coding Sheet

To change the collating sequence, two types of entries must be made. First, an S or T must be placed in column 26 of the Control Card and File Description Specifications Sheet. These two letters both cause the collating sequence to be changed, and may be used interchangeably. Next, the alternate collating sequence must be coded on the Alternate Collating Sequence and Translation Coding

Sheet for punching into change record cards. The alternate collating sequence delineates the changes to be made in the normal collating sequence; it applies only to matching alphanumeric fields, alphanumeric compare operations (see Section 7, "Compare and Testing Operations") and sequence checking. An alternate collating sequence does not affect control levels, numeric comparisons, or lookup (see Section 7, "Lookup Operations").

To cause characters to appear in a sequence different from that used by the computer, or to cause two or more characters to be considered equal (have the same position in the sequence), the alternate collating sequence must be punched into change record cards in the following format:

Positions	Description
1-6	ALTSEQ is entered to identify the entry as one which alters the normal collating sequence.
7-8	These positions are left blank.
9-10	The hexadecimal number of the character whose normal collating sequence is being replaced is entered (see Figure 2-3).
11-12	The hexadecimal number of the character which is to replace the character entered in columns 9-10 is written.
13-16, 17-20, 21-24, etc.	These positions are used in the same manner as positions 9-12. The first two positions contain the character to be replaced. The second two positions contain the character doing the replacing. The first blank position determines the end of the record. As many four-position entries may be made as are necessary to complete the sequence changes. Additional records of the same format may also be used.

The change record cards then form the alternate collating sequence table. The table must be preceded by a beginning record and followed by an ending record. The preceding record is coded with **b in positions 1-3, where b indicates a blank position. The remaining positions of the preceding record may be used for comments or left blank.

The ending record is coded in one of two ways. If the ending record is the last card in the source deck, positions 1-2 are coded with /*. If additional tables and arrays other than the alternate collating sequence table are to be added (see Section 4, "Tables and Arrays"), positions 1-2 are coded with **.

Figure 2-4 shows an example of coding for a change in the collating sequence. All records begin in column 1.

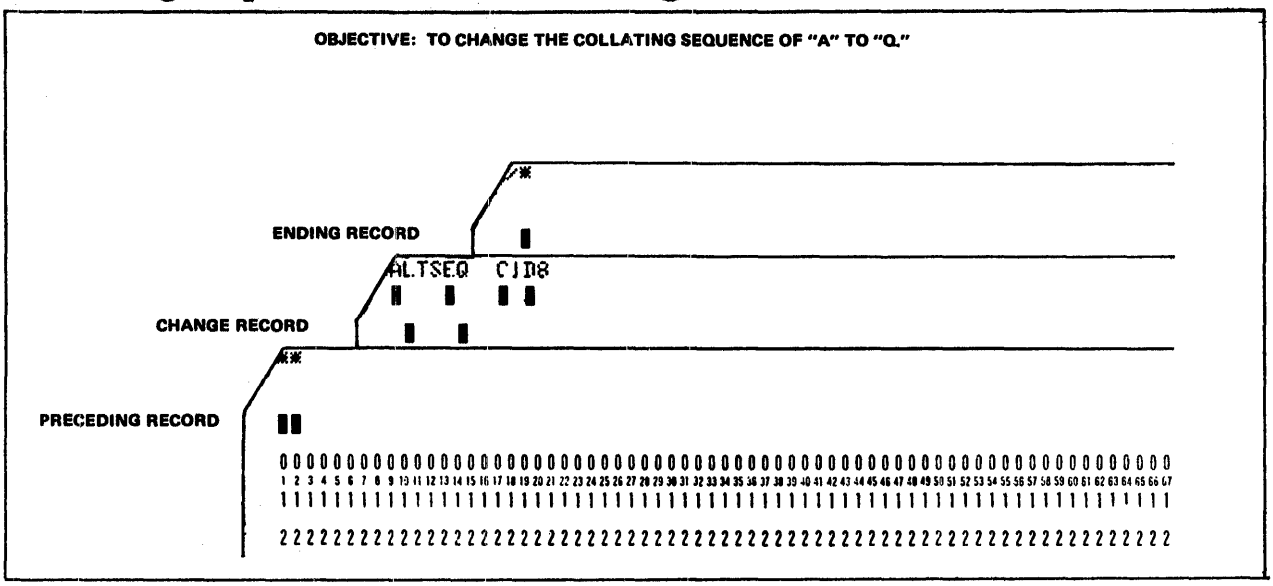


Figure 2-4. Example of a Change in Collating Sequence.

TABLE LOOKUP (Column 34)

Entry	Description
Blank	Either no table lookup is used, or table lookup with sequential search is to be used.
B	Binary search is to be used for table lookup.

If large tables in ascending or descending sequence are used, the binary search specification (B) saves time; however, space can be saved by not using the binary search if the tables are small, if they are rarely used, or if a few entries which will be referenced often are put at the beginning of the table.

SIGN PROCESS (Column 40)

Entry	Description
Blank	Numeric input fields and tables are to be checked for a valid sign.
N	Except for input fields from tape and disc, numeric input fields and tables are to be checked for a valid sign.
S	Standard signs are to be provided for all numeric fields, tables, and literals, and for packed numeric output fields.
I	Input fields are not to be checked for valid signs.
O	No sign is to be forced on output fields and output tables.
B	Sign forcing is to occur on literals (see Section 7, "Literals"), input from tables, and the MOVE, MOVEL, and move zone operations (see Section 7, "Operation Codes").

Several significant details must be kept in mind when the sign process specification is used:

On the MRX machines all bit combinations carry valid signs in arithmetic operations, so that these options are availale for compatibility with other types of RPG II.

In the blank and N sign process specifications, any invalid signs are changed to the positive C sign when they are encountered in numeric input fields.

In the S specification, the standard hexadecimal C is provided for positive contents, and the hexadecimal D for negative contents. If a MOVE, MOVEL or move zone specification contains an alphanumeric field in the factor 2 entry and a numeric field in the result field entry, the sign of the result field will be either a hexadecimal C or a hexadecimal D. The factor 2 and result field entries are thoroughly discussed under the MOVE, MOVEL, and move zone descriptions in Section 7, "Move Operations."

If the factor 2 entry of a MOVE, MOVEL or move zone specification contains a numeric field and the result field is alphanumeric and positive, the result field's sign position is forced to an F-zone. This also occurs with numeric fields which are output in unpacked decimal format.

The I sign process specification should be used only by programmers who are completely familiar with their input data, since input fields are not checked for a valid sign. The I specification causes sign forcing on literals, input tables, output fields, and the MOVE, MOVEL, move zone, and CHAIN operations (see Section 7, "Input and Output Programmed Control Operations") in the same way as an S entry does.

In the O sign process specification, sign forcing occurs on input fields, input tables, literals, and the MOVE, MOVEL, and move zone operations in the same way as it would if an S were entered.

The B sign process specification is merely a combination of the I and O entries.

FORMS POSITIONING (Column 41)

Entry	Description
Blank	The line conditioned by the 1P indicator is to be printed only once.
1	The line conditioned by the 1P indicator can be printed repeatedly.

A 1 entered in column 41 ensures that the printer forms are positioned properly. Forms positioning applies to the first output line conditioned by the 1P (first page) indicator for printer files. Each time the line conditioned by the 1P indicator is printed the program halts for forms repositioning. The page count is not incremented until the forms are positioned properly.

FILE TRANSLATION (Column 43)

Entry	Description
Blank	No files are to be translated.
F	Input, output, or update files are to be translated.

File translation allows the programmer to translate any character code into another character code. A different character code used as input can be translated into the code used internally, or the code used internally by the MEMOREX system can be translated into a different code for output.

Column 43 is used only when information contained in an input, output, or update file is in a form which requires translation. When file translation is specified for an update file, both the input and the output portions of the file are translated.

The input and output characters described here are called external characters, while the characters used for processing within the computer are referred to as internal characters.

In the Control Card and File Description Specifications Sheet, an F is entered in column 43 to specify that files are to be translated. The F indicates one or both of the following:

The external character code used in the input data must be translated into an internal character form that can be used by the program.

The output data must be produced in a character code different from the code used by the program.

File translation records must also be punched into cards to specify the method of translation. All of the individual records together then form a file translation table, which immediately precedes the alternate collating sequence table's beginning card in the source deck sequence.

The entries described below must be punched into a card for each file translation record used in the table. If only certain files are to be translated, each file must be individually named; however, a single entry will serve for the translation of all files.

Positions	Description
1-6	If all input files, all output files, and both the input and output portions of all update files are to undergo translation, *FILES must be entered. The specifications beginning with position 9-10 below are then entered. All files will be translated according to the translation specifications beginning in position 9.
1-8	If only certain files are to be translated, the filename of the specific individual input, output, or update file to be translated must be entered. Both the input and the output portions of the update file will be translated. Specifications are then continued for each file, beginning with positions 9-10 below.
9-10	The hexadecimal equivalent of the external character is the character that is to be translated from input data or for output data.
11-12	The hexadecimal equivalent of the internal character is the character which internally represents the external input or output character.
13-16, 17-20, 21-24, etc.	These groups of positions are used in the same way as positions 9-12. The first two positions of each group indicate the character to be translated into the character named in the last two positions of the group.

All tables for a single file must be kept together. The file translation input records must be preceded by a record card with **Ø punched into positions 1-3. The remaining positions of this record may be used for comments. The file translation records must directly follow the RPG II specifications in the source program sequence.

SUPPRESS SKIP TO CHANNEL 1 (Column 47)

Entry	Description
Blank	A normal skip to channel 1 on the printer file is to be made.
S	A skip to channel 1 at the beginning of RPG II object program execution is suppressed.

An entry in column 47 prevents printing from starting on a new page so that forms can be printed in specific positions.

CROSS-REFERENCE LIST (Column 52)

Entry	Description
Blank	No cross-reference listing is to be printed.
X	A cross-reference listing is to be printed.

An X entered in column 52 causes cross-reference listings of filenames, field names, and indicators to be printed. The filename and indicator listings include the line on which each filename or indicator is defined and the statement that references each of them. The field name cross-reference listing includes the storage location and length of each field name, the kind of data it contains (numeric or alphanumeric), the line on which it is defined, and the statement that references it.

CARRIAGE CONTROL TYPE (Column 53)

Entry	Description
Blank	Carriage control channel numbers are used for spacing.
L	Line numbers on a page are used instead of carriage control channel numbers.

If an L is specified in column 53 the line counter form should specify only the first line and length of the form to be used. A SKIP specification in the Output-Format Specifications Sheet (Section 8) then refers to the line numbers.

When an L is specified in column 53, the generated RPG program simulates skips to a line. It gets its starting point by an initial skip to channel 01, and assumes that channel 01 is on line 06. All subsequent spacing is then relative to that starting point.

Leaving column 53 blank speeds up printing and causes spacing to be governed by carriage control tapes. SKIP specifications in the output-format sheet refer to carriage control channels when the column is left blank.

SOURCE SEQUENCE CHECK (Column 54)

Entry	Description
Blank	A full sequence check is to be performed.
N	No sequence check is to be performed.

If no entry is made in column 54, columns 1-5 are to be checked for sequence. An N entered in column 54 indicates that columns 1-5 are not to be checked for sequence.

IGNORE ARITHMETIC OVERFLOW (Column 55)

Entry	Description
Blank	Arithmetic overflow is to be indicated.
I	Arithmetic overflow is to be ignored by the compiler.

If an I is entered in column 55 the compiler ignores any arithmetic overflow which may result from calculations and processing continues. If column 55 is left blank the compiler issues a message to the console indicating on which line the overflow occurred before processing continues.

PROGRAM IDENTIFICATION (Columns 75-80)

Entry	Description
Blank	A program identification of RPGOBJ is assumed.
Valid RPG II name	The program identific tion should be a unique name for the object program.

No special characters may be used in the program identification. The first character of the name cannot be one of the following characters: #, $ or @; it must be a character from A to Z. The remaining characters can be any combination of alphabetic and numeric characters. The program identification can be one to six characters long. Blanks cannot appear between the characters.

SECTION 3. FILE DESCRIPTION SPECIFICATIONS

A file is a collection of related records treated as a single unit. The records within a file are similar in purpose, form and content.

File description information is required for every file used by a program. This information is entered on the RPG II Control Card and File Description Specifications Sheet (Figure 3-1). A single line is sufficient for describing each file, and a maximum of 20 file description record lines is permitted for each program in a partition whose size is 10K or larger. In a partition smaller than 10K, a maximum of 10 files is allowed.

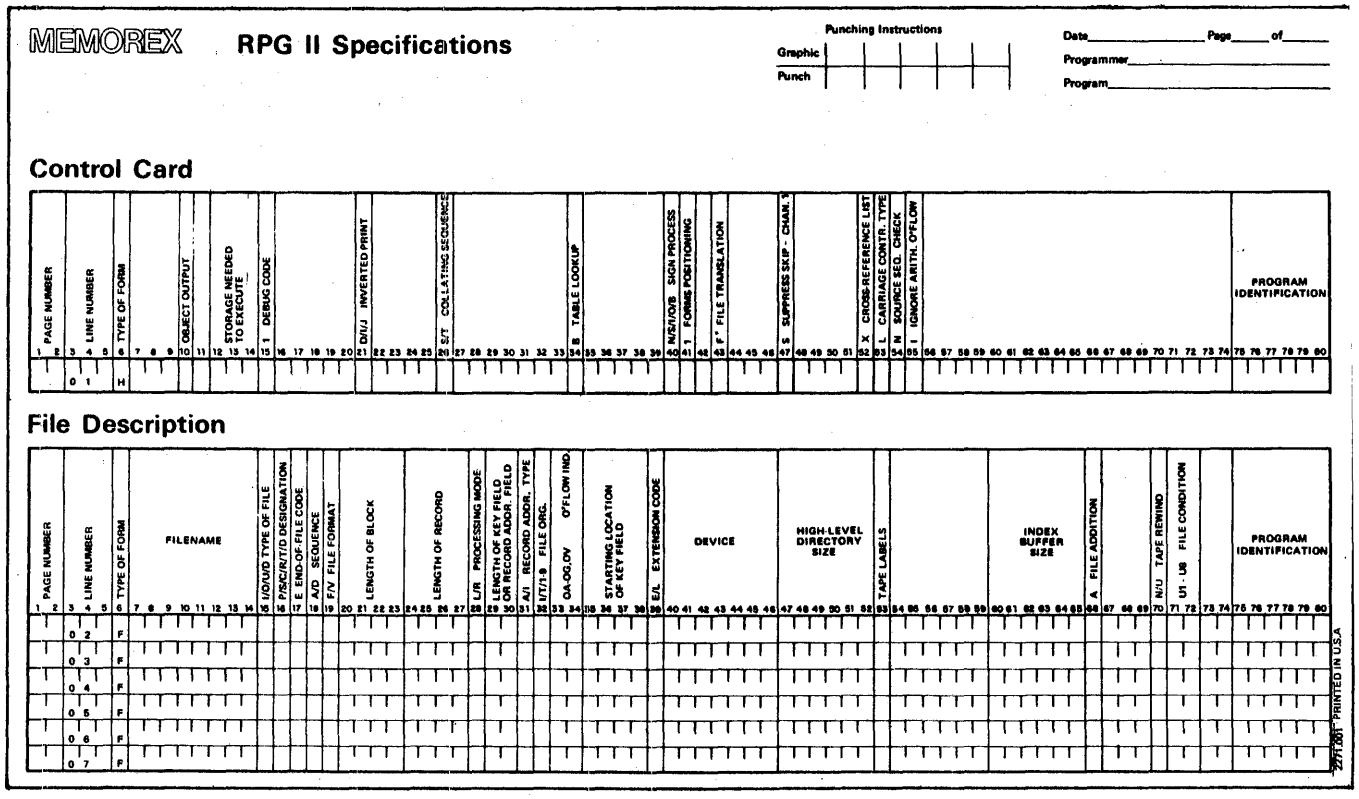


Figure 3-1. RPG II Control Card and File Description Specifications Sheet.

TYPES OF FILES

RPG II uses five types of files: input, output, update, display, and combined.

INPUT FILES

Input files contain records which provide a program's source of data. An input file description in a program indicates that records are to be read from that file. All input files except table files and record address files (see "File Designations" in this section) require additional description on the Input Specifications Sheet (Section 6). Table files and record address files require further description on the Extension Specifications Sheet (Section 4).

OUTPUT FILES

Output files comprise the records that a program writes, punches or prints. All output files except table and array output files require further description on the Output-Format Specifications Sheet (Section 8). See "File Designations" in this section for additional discussion of table and array files.

UPDATE FILES

Update files are disc files from which a program reads a record, updates its fields, and rewrites the record in the location from which it was read. Update files require additional description on both the Input Specifications Sheet (Section 6) and the Output-Format Specifications Sheet (Section 8).

DISPLAY FILES

Display files are composed of information gathered from various fields used by a program. To print a field or record directly from storage, and to key data into a field or record in storage, the DSPLY operation code must be entered on the Calculation Specifications Sheet (Section 7). The only device that can be associated with a display file is a console (printer keyboard).

COMBINED FILES

A combined file comprises an input and an output file. A program reads records from a combined file and simultaneously punches output data into the records via the 8025 Card Reader/Punch. The resulting single file contains both input and output data. Combined files require additional description on both the Input and the Output-Format Specifications sheets (Sections 6 and 8, respectively).

FILE DESIGNATIONS

Six designations are used in RPG II to describe input, output, and update files still further: primary, secondary, chained, demand, record address, and table and array.

PRIMARY FILES

The primary file is the main file from which a program reads records. The program uses the primary file during multifile processing to control the order in which records are selected. A primary file can be an input or update file, but one and only one primary file is required for every program.

SECONDARY FILES

All files involved in multifile processing except the primary file are secondary files. Secondary files are processed in the order in which they are entered on the File Description Specifications Sheet, and can be input or update files.

CHAINED FILES

A chained file is a disc file that is read randomly or loaded directly via the CHAIN operation code (Section 7). A chained file can be read or written during detail calculations time or total calculations time (see Figure 1-4). A maximum of 15 chained and/or demand files is permitted in each program.

DEMAND FILES

Demand files can be input or update files, and can only be processed sequentially. The READ operation code must be used on the Calculation Specifications Sheet (Section 7) in order to read from a demand file. A maximum of 15 demand and/or chained files per program is allowed.

RECORD ADDRESS FILES

A record address file is an input file that indicates which records are to be read from a disc file, and the order in which those records are to be read. Only one record address file can be used in a program.

Record address files contain either relative record numbers in binary format, or record key limits. Those files with record key limits can be disc or card files, or can be entered via the console. They are used with indexed files only (see "File Organization" in this section).

Record address files that contain binary relative record numbers can only be disc files and can be used only with relative files (see "File Organization" in this section).

ADDROUT files are a special type of record address file, since they are produced by the disc sort utility program, rather than by the RPG II source or object program. Execution of the disc sort utility program is controlled by the operating system.

TABLE OR ARRAY FILES

A table or array file is a sequential input file containing table or array entries. Only pre-execution time tables or arrays need be described on the File Description Specifications Sheet.

Pre-execution time table or array entries are read into the program immediately before program execution and are described on both the File Description Specifications Sheet and the Extension Specifications Sheet (Section 4). Table and array files are not involved in record selection and processing, since they are used only as a means of supplying entries for tables or arrays used by the program. When table or array files are read at pre-execution

time, the program reads all the entries from the table or array files before it begins record processing.

Compile time table or array entries are read into the program during compilation, and are described only on the Extension Specifications Sheet (Section 4).

A table or array output file is defined as a normal output file and requires no special entry in the File Description Specifications Sheet.

FILE ORGANIZATION

File organization is determined by the arrangement of the records within the file. The three types of file organization are indexed files, relative files, and sequential files.

INDEXED FILES

Indexed files are disc files whose record locations are kept in a separate portion of the file called an index. The index contains a record key for each record, as well as the record's location. The record key contains the information found in the key field of a record and is used to identify the individual record in the indexed file. Indexed files must contain fixed-length records, and must be loaded in ascending record key sequence.

RELATIVE FILES

Relative files are disc files whose records are directly called by number, rather than by the sequence in which they were loaded. Each record is assigned a specific place within the file and a relative record number which identifies its position in the file in relation to the other records. Before a relative file is loaded, the disc is automatically cleared to blanks.

SEQUENTIAL FILES

A sequential file may be assigned to any device. All files that are not assigned to disc must be sequential files. In a sequential file, records are ordered in the same sequence in which they are put into the file. The first record read always occupies the first record position, the second one read occupies the second record position, and so on.

CODING INSTRUCTIONS

Detailed descriptions of the file description entries to be coded on the specifications sheet are defined below.

PAGE NUMBER (Columns 1-2)

Entry	Description
01-99	Specifications sheets are numbered sequentially.

More than one of each type of specifications sheet may be used in the course of coding the source program. To keep the sheets in the proper order, the coded File Description Specifications Sheets must be grouped together sequentially and arranged with the other specifications sheets in the following order:

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

The specifications sheets should then be numbered in ascending sequence.

LINE NUMBER (Columns 3-5)

Entry	Description
Any number	Line numbers are assigned in ascending sequence.

Line numbers are preprinted in columns 3-4 of the specifications sheet for the programmer's convenience. The unnumbered lines below the preprinted numbers can be used for additional lines or to insert a line between completed lines. Column 5 can also be used to this end. The line numbers assigned need not be consecutive, but they must be in ascending order.

TYPE OF FORM (Column 6)

Entry	Description
F	F identifies the file description specifications to the compiler.

FILENAME (Columns 7-14)

Entry	Description
1-8 characters	The name of the file being described is entered.

A unique name must be assigned to every file the program uses except compile time table and array files, which are described in the Extension Specifications Sheet (Section 4). If the filename is left blank, an error occurs.

The filename must be a valid RPG II name; it may be the same as a field name. A valid filename consists of one to eight characters. The first character must be alphabetic; the remaining characters can be any combination of alphabetic and numeric characters. Special characters are not permitted. Blanks cannot appear between characters in the name.

Pre-execution time table and array filenames are entered on the File Description Specifications Sheet. More than one pre-execution time table or array filename can be entered for the same device (see columns 40-46 in this section).

TYPE OF FILE (Column 15)

Entry	Description
I	An input file is to be used.
O	An output file is to be used.
U	An update file is to be used.
C	A combined file is to be used.
D	A display file is to be used.

The entry in column 15 identifies the way in which the file is to be used by the program. If the type of file entry is left blank, it is considered an error.

Of the four types of files that can be identified, all but the display file require additional entries on other types of specifications sheets. Input files (except record address and table files) require other entries on the Input Specifications Sheet (Section 6). Record address and tables files used as input files must be identified on the Extension Specifications Sheet (Section 4) as well as the File Description Specifications Sheet.

Output files, except table and array files used as output files, require additional description on the Output-Format Specifications Sheet (Section 8). Table and array files used as output files are described on the File Description Specifications Sheet and the Extension Specifications Sheet.

Update and combined files require additional entries on both the Input and the Output-Format Specifications Sheets.

DESIGNATION (Column 16)

Entry	Description
Blank	A display file or output file (except a chained output file) is to be used.
P	A primary file is to be used.
S	A secondary file is to be used.
C	A chained file is to be used.
R	A record address file is to be used.
T	A pre-execution time table or array file is to be used.
D	A demand file is to be used.

The file designation entry in column 16 is a one-character code which further identifies a file by indicating its specific use by the program.

Primary and secondary files involve record selection in multifile processing. For detailed information on primary and secondary file processing, see "Multifile Processing" in Section 6.

END-OF-FILE CODE (Column 17)

Entry	Description
Blank	The program can end before all the records from the file have been processed.
E	The program cannot end until all records from the file have been processed.

The end-of-file code applies only to input and update files designated as primary, secondary, or record address files for multiprocessing. When column 17 is left blank for all of the files described, every record from every file must be processed before the program terminates.

When matching records are being processed with an E specified for the primary file, the job ends only after the System has processed all secondary records that match the last primary record, or after it has encountered the first secondary record without a match field.

SEQUENCE (Column 18)

Entry	Description
Blank	Sequence checking is not to be performed.
A	Sequence checking is to be performed; the file records are in ascending order.
D	Sequence checking is to be performed; the file records are in descending order.

The sequence entry is used to indicate whether or not the program is to check the sequence of the records. The entry applies to all update files, and to all input files except chained, demand, record address, table, and array files, for which it is left blank. Column 18 is left blank for output and display files.

Sequence checking is required when match fields are used in the file records (see Section 6, "Match Fields"). An entry in the sequence field must be further described in columns 61-62 of the Input Specifications Sheet (Section 6) via an entry identifying the match fields which contain the sequence information.

FILE FORMAT (Column 19)

Entry	Description
F	The records in the file are fixed-length.
V	The records in the file are variable-length.

Records in an indexed file must be fixed-length; i.e., all the records in an indexed file must be the same length. Tape and non-indexed disc files may contain records of variable lengths. Either F or V must be specified in the file format entry; if it is left blank, a warning message is issued and F is assumed.

LENGTH OF BLOCK (Columns 20-23)

Entry	Description
Blank	The compiler calculates the block length from the record length and the file device. If the device is disc or tape and the record length is 256 or less, the compiler groups as many records as possible into a block whose size does not exceed 256 (see Appendix D).
1-4096	The length of a block of records is entered. For all devices except disc or tape, the length entered must be the same as the length of a record (columns 24-27). For disc or tape, the length entered may be the record length or a multiple of the record length + 4. If the block length entered is the same as the record length, an optimum block size is calculated for disc or tape (see above). If records are variable-length, the only restriction is that the LENGTH OF BLOCK entry must be greater than or equal to the LENGTH OF RECORD entry.

RPG II allows block lengths for fixed-length records equal to the length of a record or a multiple of its length plus 4. Legal block lengths are governed by the type of device assigned to the file. When the entry is being specified for variable-length records, the block length need not be a multiple of the record length; the maximum block length is entered instead.

All disc and tape records, whether fixed-length or variable-format, are stored in common stored data format: a four-byte header precedes each logical record. Thus if a record length were 100, unblocked, the block length would be 104; and if a fixed-length record had a length of 50, blocked 4, the block length would be 216.

The block length actually assumed by the compiler may be different from the block length specified because of the extra characters needed for carriage control and spooling control. See Appendix D for more information about assumed block lengths.

entries must be right-justified to end in column 23, and leading zeros are not required. When the block length is the same as the length of one record, the field may be left blank.

LENGTH OF RECORD (Columns 24-27)

Entry	Description
1-4096	Either the number of characters in a single fixed-length record, or the maximum record length for variable-length records is specified.

Columns 24-27 are used to indicate the length of the records in the file. The maximum record length allowed and the size of the input/output area assigned depend on the type of device assigned to the file. If the entry is left blank an error occurs. If variable-length records are specified in update files, they need not be the same length after updating occurs as they were prior to updating. The records cannot exceed the maximum record length specified, however, after they are updated. (See Appendix D for information about assumed record lengths.)

PROCESSING MODE (Column 28)

Entry	Description
Blank	The processing mode is either sequential by key, or consecutive.
L	The processing mode is sequential within limits.
R	The processing mode is one of the following: Random by relative record number Random by key By ADDROUT file Direct file load

The processing mode entry indicates the method by which records are to be read from the file, or that a direct file load (random load) is to take place. For disc files specified as primary, secondary, or chained files, the methods possible depend upon the type of file organization (Figure 3-2). For the other types of files, consecutive processing is the only possible method.

PRIMARY AND SECONDARY FILE PROCESSING	
Organization	Possible Methods
Sequential	Consecutively.
Relative	1. Consecutively. 2. By ADDROUT file.
Indexed	1. Sequentially by key. 2. Sequentially within limits. 3. Randomly by key (via record address file or chaining).

CHAINED FILE PROCESSING	
Organization	Possible Methods
Relative	Randomly by relative record number.
Indexed	Randomly by key.

Figure 3-2. Possible Disc File Record Retrieval Methods.

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

Entry	Description
1-99	Specifies the length of the record key in indexed files or in record address files that contain either random keys or limits.
4	Specifies the length of the ADDROUT file record.

This entry applies only to indexed disc files and record address files. If these files are used and the entry in columns 29-30 is left blank, an error occurs.

The maximum length allowed for key fields in indexed file records is 99 characters. All the key fields in indexed file records must be the same length. A length of four is required for records in an ADDROUT file. The leading zero may be omitted, however the entry must be right-justified to end in column 30.

RECORD ADDRESS TYPE (Column 31)

Entry	Description
Blank	One of the following is to occur: Records are to be read consecutively; or a sequential or relative file is to be loaded; or relative record numbers are to be used in processing relative files.
A	Alphanumeric record keys are to be used in processing indexed files.
I	The file is an ADDROUT file, or an ADDROUT File is to be used to process the file.

An entry is specified in column 31 only for indexed and ADDROUT files. No entry is required for the other file types.

FILE ORGANIZATION (Column 32)

Entry	Description
Blank	A sequential or relative file is to be processed. One input/output area is to be used for the file.
I	An indexed file is to be processed.
T	An ADDROUT file is to be processed.
1-9	A sequential file or relative file is to be processed. Two input/output areas are to be used for the file. (The digit 2 is preferred, since a maximum of two input/output areas is permitted.)

The file organization entry can be used to identify the organization of all files except ADDROUT files; to identify ADDROUT files; or to indicate whether one or two input/output areas are to be used for sequential files or relative files.

OVERFLOW INDICATOR (Columns 33-34)

Entry	Description
Blank	No overflow indicator is to be used.
ØA-ØG, ØV	The specified overflow indicator is to be used to condition records in the file.

Overflow indicators are used for output files assigned to the printer to indicate that records to be printed in the file are being conditioned. Overflow indicators used in a program must be unique for each output file assigned to the printer. Only one overflow indicator can be assigned to a file.

Overflow indicators cannot be assigned to a console file. They may be turned on and off by the operation codes SETON and SETOF (see Section 7, "Operation Codes") when they are used to condition overflow printing. Spacing past the overflow line causes the overflow indicator to turn 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.

Overflow printing with the EXCPT operation code (see Section 7, "Operation Codes") involves certain rules:

Overflow indicators cannot condition an exception line, but they can condition fields within an exception record.

Using the EXCPT operation code with the E in column 15 of the Output-Format Specifications Sheet (Section 8) causes the fields to be printed while the calculations are being performed; normally they are printed afterwards. Only the fields identified by an E in column 15 are printed at that time.

Even though the specified fields are printed during calculations, they still have the same effect on the overflow routines as all other lines.

STARTING LOCATION OF KEY FIELD (Columns 35-38)

Entry	Description
1-4096	Indicates the record position in which the key field begins.

An entry in columns 35-38 is required only for indexed disc files. If an entry is not made for an indexed file, an error occurs. The entry must end in column 38. Leading zeros are not required.

The key field of a record is used in the index portion of the file and contains information that identifies the record. The key field in all of the records in a file must be in the same location.

EXTENSION CODE (Column 39)

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

The extension code in column 39 is used for all record address files, for all table and array files to be read during program execution, and for output files assigned to the printer which have a line counter associated with them. The column is left blank for all other files.

The Extension Specifications Sheet must be used to give additional information about table, array, and record address files. The Line Counter Specifications Sheet should be used to give additional information about print files which may be assigned to disc or tape.

If the column is left blank for table, array, or record address files, a warning message is issued and an error occurs.

DEVICE (Columns 40-46)

Entry	Description
DISC	Specifies the disc unit.
PUNCH	Specifies the card punch.
TAPE	Specifies magnetic tape.
READ	Specifies the card reader.
CONSOLE	Specifies the console keyboard.
PRINTER	Specifies the printer.

The entry in columns 40-46 identifies the input/output device to be used for the file being described. The device used depends upon the form of the records, as illustrated in Figure 3-3. Entries must be left-justified to begin in column 40. If the entry is left blank, an error occurs.

FILE	FORM	DEVICE ENTRY
Primary or Secondary Input Files	Cards Tape Disc Keyed in by Operator	READ TAPE DISC CONSOLE
Record Address Files Containing Record-Key Limits	Cards Tape Disc Keyed in by Operator	READ TAPE DISC CONSOLE
ADDROUT File	Disc	DISC
Table Files	Cards Tape Disc Keyed in by Operator	READ TAPE DISC CONSOLE
Chained Input Files	Disc	DISC
Update Files (Primary, Secondary, or Chained)	Disc	DISC
Output Files	Cards Tape Disc Printed Pages	PUNCH TAPE DISC PRINTER or CONSOLE
Display Files	Printed Pages	CONSOLE

Figure 3-3. Devices Possible for Assignment to Files.

Substitute Device Assignment

In addition to the standard device names shown in Figure 3-3, certain other device names are allowed for compatibility with other systems. Figure 3-4 lists the name of each of these devices and the Memorex device that is used in its place, and indicates whether or not the substitute devices issue warning messages.

DEVICE	MEMOREX SUBSTITUTE DEVICE	WARNING MESSAGE
MFCM1 MFCM2 CRP20 MFCU1 MFCU2	READ, if an I is specified in column 15	X
	PUNCH, if an O is specified in column 15	X
PRINTLF	PRINTER	X
PRINTUF	PRINTER	X
PRINTR2	PRINTER	
READ01	READ	
DISK11F	DISC	
PRINTKB	CONSOLE	
INQIPT	CONSOLE	
PUNCH20	PUNCH	
PUNCH42	PUNCH	
DISK	DISC	

Figure 3-4. Substitute Devices Assigned to Files.

Special Device Support

Special input/output devices may be used with RPG II by providing a link to a user-written routine that enables data to be transferred for the special device.

The files read and written are called special files and are designated as such by entering the word SPECIAL in columns 40-46. Special files can only be processed consecutively.

Special files require specific entries in all the columns of the File Description Specifications Sheet, and special entries for the special device feature. The entries are listed below, with asterisks denoting the special entries.

Columns	Entry
7-14	Valid RPG II filename.
15	I, O, or U.
16	P, S, D, or blank.
17	Blank or E.
18	Blank, A, or D.
19	F.
20-23	Block length.
24-27	Record length.
28-31	Blank (mandatory).
32	Blank or 1-9 (dual I/O areas are allowed).
33-39	Blank (mandatory).
*40-46	The word SPECIAL.
47-53	Blank (mandatory).
*54-59	SUBRxx (x can be any alphabetic character. This is the name of the user-written routine that performs data transfer).
60-70	Blank (mandatory).
71-72	Blank or U1-U8.
73-74	Blank (mandatory).

See Appendix E for a discussion of special file linkage.

HIGH-LEVEL DIRECTORY SIZE (Columns 47-52)

Entry	Description
Blank	No high-level directory is to be used.
1-99999	Specifies the number of bytes reserved for a high-level directory.

An entry in columns 47-52 indicates the number of bytes to be reserved for a high-level directory which is kept in memory throughout the job, thereby increasing disc file access speed. The entry must be right-justified; leading zeros are not required. When no entry is made in columns 47-52, the high-level directory is accessed anew each time it is needed.

The high-level directory contains a key and a block number which corresponds to the index directory and is used with randomly accessed disc files.

TAPE LABELS (Column 53)

Entry	Description
Blank	Either no labels are to be used, or the file is not a tape file.
S	Standard labels are to be used.
N	Non-standard labels are to be used.

Column 53 is used for tape files to indicate the type of label information on the tape. RPG II handles the processing of standard labels.

INDEX BUFFER SIZE (Columns 60-65)

Entry	Description
Blank	No index buffer is to be used.
1-9999	The number of bytes reserved for the index buffer is entered.

The index buffer entry applies only to indexed disc files. Up to 9999 bytes can be specified for the index buffer. Entries must be right-justified to end in column 65, however leading zeros are not required.

An index block, which is read into the index buffer, is a table which contains index keys and pointers to the corresponding records. An index buffer must be used for indexed files processed sequentially, but it is not required for indexed files processed randomly. If the entry is left blank for randomly processed indexed files, the index buffer shares the area occupied by the data buffer; thus the amount of time needed to process the file is reduced.

FILE ADDITION (Column 66)

Entry	Description
A	New records are to be added to the file.

Column 66 applies only to sequential, relative, and indexed disc files. If the column is left blank, it is ignored. Records added to a sequential file are added at the end of the file. Records added to relative files must be within the file limits; the file size cannot be extended. Records added to an indexed file can be added to any part of the file, including the end, as long as a record is not already specified for that key. Records may only be replaced by performing an update. Entries for the new records are made in the index.

File addition cannot be specified for indexed files from which records are read using the sequential-within-limits method. New records may be added to a relative file by specifying the file as an update file processed consecutively, or by the CHAIN operation code (see Section 7, "Operation Codes").

TAPE REWIND (Column 70)

Entry	Description
Blank	Rewind only is to occur.
N	No rewind is to take place.
U	Rewind and unload are to occur.

This entry defines the rewind action to be taken on tape files when the end-of-file condition occurs (see column 17).

FILE CONDITION (Columns 71-72)

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

Columns 71-72 apply to primary and secondary input files (excluding table input files), update files, and output files. A record address file may be conditioned by an external indicator if its associated primary or secondary file is conditioned either by the same indicator or by no indicator.

External indicators are set by using a parameter on the MRX/OS Control Language statement for execution of the object program. 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-file condition had been reached; no records can be read from or written in the file.

In addition to their use as file conditioning indicators, external indicators may be used as field record relation indicators (see Section 6, columns 63-64), and to condition calculation or output operations. See Section 7, "Indicator Setting Operations," and Appendix B, Tables B2 and B3, for indicator summaries.)

PROGRAM IDENTIFICATION (Columns 75-80)

Entry	Description
Any characters	These columns may be used for programmer comments.

The program identification field of all cards following the control card may contain any characters, including blanks. The columns may be used for programmer comments, if desired.

SECTION 4. EXTENSION SPECIFICATIONS

Extension specifications are required to complete the description of record address files (see Section 3, "File Designations"), tables, and arrays. Every table and array used by the object program must be described on the RPG II Extension and Line Counter Specifications Sheet (Figure 4-1). A maximum of 60 table and array descriptions and one record address file description is allowed for each program in a partition whose size is 8K or larger. In an 8K partition 12 tables/arrays, minus the number of files used, may be defined.

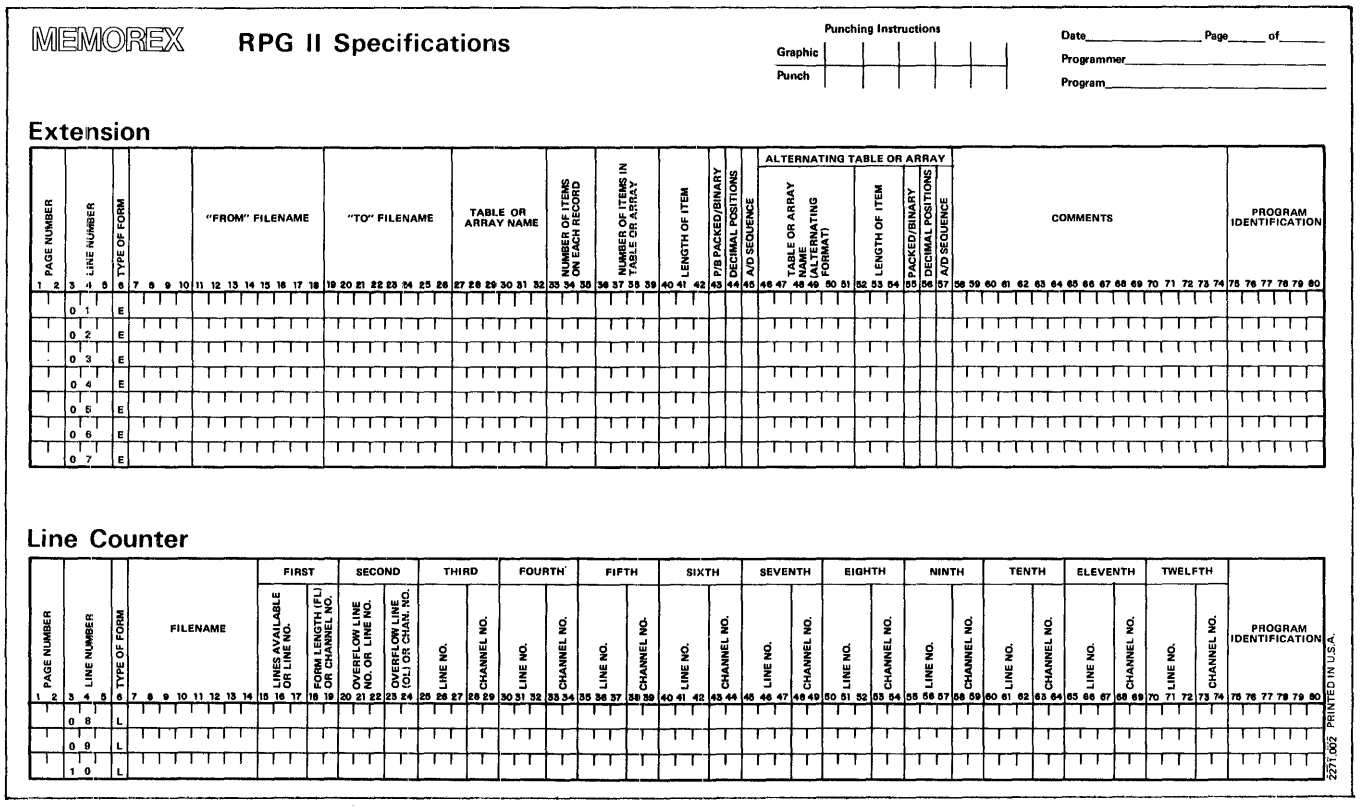


Figure 4-1. RPG II Extension and Line Counter Specifications Sheet.

TABLES AND ARRAYS

Tables and arrays are groups of data items arranged systematically for ready reference. Each item in a table or array must be of the same data type (alphanumeric or numeric) and have

the same field length and number of decimal positions as every other item in the table or array.

Tables and arrays may be searched for the presence of specific items using the LOKUP operation (see "Lookup Operations," Section 7), and they may be used in calculations and output.

Tables are searched one item at a time (unless they are in sequence and the binary search option is used—see Section 2, column 34) until an item is found that matches a certain identifier. That item is then made available for use in calculations. Arrays may also be searched for a given item, but the item is not automatically made available for calculations. A specific array item may be referenced, however, by its location in the array relative to the first item. This kind of relative referencing is called indexing. Indexing may not be used with tables.

When a table name is used as one of the operands in a calculation specifications line (Section 7), the item most recently found in a LOKUP operation on the table is to be used in the calculation. Specifications for which an array name is an operand usually use all of the array items sequentially in the same operation. For instance, a number may be added to every item in a numeric array by using only one specifications line (see "Table and Array Reference" in this section). A single array item must be referenced by the array name followed by a comma and the index of the item. The index may be an actual number or the name of a positive numeric field with zero decimal positions, and it may never have a value less than 1.

TABLE AND ARRAY NAMES

Each table and array used in the program must be given a name from one to six characters long. No two tables or arrays may be given the same name, since the names assigned in the extension specifications are used throughout the program. Each table name must be at least three characters long and begin with the letters TAB. Array names cannot begin with the letters TAB; they must be assigned other identifying characters. Any name assigned in columns 27-32 which does not begin with TAB is considered an array name.

TABLE AND ARRAY TYPES

Tables and arrays described in the extension specifications can be loaded at one of three different times: compilation time, pre-execution time, or execution time. These loading times separate the tables and arrays into three corresponding types.

Compile Time Tables and Arrays

Compile time tables and arrays are loaded at compilation time and compiled along with the source program. They become a permanent part of the resulting object program. To effect a permanent change to a compile time table or array, the source program must be recompiled with the revised table or array.

Pre-Execution Time Tables and Arrays

Pre-execution time tables and arrays are loaded with the object program before any of the non-table or non-array input files are read, calculations are done, or output functions are performed. The load must be completed before actual program execution begins.

Execution Time Arrays

Execution time arrays are loaded after program execution begins. Loading is accomplished when the arrays are read in as input data as described on input specifications forms (Section 6), or when they are created during calculations in the program.

ADDITIONAL TABLE AND ARRAY TYPE DESCRIPTION

Tables and arrays can be further described as short, full, or related.

Short Tables and Arrays

Short tables and arrays are those in which not all of the entries contain data. Each short table or array must contain at least one item. In numeric tables and arrays, the unused items are filled with zeros. In alphanumeric tables and arrays, the unused items are filled with blanks. Items can be added or replaced during execution time by reading them in from input records or creating them through calculation operations.

Full Tables and Arrays

Full tables and arrays are those in which all possible items contain data. The table or array is full, and nothing more can be added to it.

Related Tables or Arrays

Related tables or arrays are tables or arrays that are used together. Each item in the first table or array is associated with a corresponding item in the second table or array which gives additional information. Although all items within one table or array must have the same characteristics, corresponding items in related tables or arrays may have different characteristics from their counterparts. Related tables and arrays should have the same number of entries. Care should be taken if they do not, since a LOKUP search stops at the end of the shorter table (see Section 7, "Operation Codes").

TABLE AND ARRAY INPUT RECORD CREATION

Input records for compile time and pre-execution time tables and arrays must follow certain rules:

The first table or array item entered on each record must begin in the first position of the record.

Two items may not be separated by spaces on a record unless the spaces are part of one of the items.

It is not necessary for the entire record to be filled with items, but an item must be placed in the first position. The remainder of the record may be composed of blanks or comments.

All records for a table or array, except the last record, must contain the same number of items.

An item cannot extend from one record onto the next one. The entire item must be on a single record.

Related tables or arrays can be described either separately or in alternating format, where the corresponding items are described on the input record as a single table or array entry. Each alternating table or array item is entered immediately following the item of the table or array to which it corresponds. The combined item lengths of two tables or arrays described in alternating format must therefore be less than the maximum record length for the device being used. Tables may not alternate with arrays, or vice versa.

TABLE AND ARRAY LOADING

Tables and arrays can be loaded at compilation time, pre-execution time, or execution time. The rules for loading them vary according to the time at which they are loaded.

Compilation Time

Tables and arrays that are loaded at compilation time are compiled along with the RPG II source program, and they become a part of that program. Compile time table or array records are loaded immediately following the source program. Each table or array must be separated by a record with ** in positions 1 and 2. Since these characters are treated as delimiters, they cannot be entered in the first two positions of a data record. The last compile time table or array must be followed by an ending record with /* in positions 1 and 2 (see Figure 1-2 for source deck arrangement and delimiters). Compile time tables and arrays must be loaded in the same order as they are described on the RPG II Extension Specifications Sheet.

A compile time table or array description must have an entry in columns 33-35 (number of items on each record) of the Extension Specifications Sheet. It must not have a "from" filename entry in columns 11-18 or a packed/binary entry in column 43 or 55.

Pre-Execution Time

Pre-execution time tables and arrays do not become part of the source program but are used like any other data file by the object program. Pre-execution time tables or arrays are loaded prior to any other processing. An ending card containing /* in columns 1 and 2 must follow every pre-execution time table or array unless two or more tables or arrays are loaded from the same device. In this case, each table or array except the last is followed by a card with ** in the first two columns. The last table or array is terminated by /*. Tables and arrays loaded from the same device must be loaded in the same order as they are described on the RPG II Extension Specifications Sheet. When errors are encountered during loading, additional information about each error is displayed on the console, provided the console has been defined as the log device.

A pre-execution time table or array description must have a "from" filename entry in columns 11-18 and a number of items on each record entry in columns 33-35. Packed or binary field entries may be made in columns 43 and 55 if they are appropriate for the device.

Execution Time

Execution time arrays may be loaded from input records or by instructions coded on the Calculation Specifications Sheet. Tables may not be defined as execution time tables.

An execution time array description cannot have a "from" filename entry in columns 11-18, a "to" filename entry in columns 19-26, or a number of items for each record entry in columns 33-35 of the Extension Specifications Sheet.

Execution time arrays that are loaded from data in input records must be fully described on the Input Specifications Sheet, as well as on the Extension Specifications Sheet. When execution time array data is to be read in packed or binary format, an entry must be made in column 43 of the Input Specifications Sheet. The field location columns of that sheet (columns 44-51) are then used to describe the record positions the array data occupies in the packed or binary format. Columns 40-42 of the Extension Specifications Sheet are used to define the unpacked decimal length of each array item.

Input specifications may be coded in several different ways. The coding method that is selected depends on the number of input records needed to load the array and the location of the array items on each record. Array information that occupies consecutive positions in a single record can be defined and loaded using a single input specifications line. Array items that are scattered through one or more records must be defined and loaded one or

more at a time. A specifications line may specify several array items with or without an index. If no index is specified, the items are put at the beginning of the array. If an index is specified, the items are put in starting at the place indicated by the index.

When array information is contained in two or more input records, the data in the first record may be defined in the input specifications using either of the two methods stated above. Subsequent array items from the other records should be defined individually. Variable indexes and/or record identifying indicators should be used to avoid overlaying the data from one record with the data from a later record.

Since an RPG II object program processes only one record at a time, the entire array cannot be processed until all records containing array information have been read and all the information has been moved into the array fields. For this reason, it is sometimes necessary to suppress calculation and output operations until the entire array has been read into the system. This can be done using control level indicators (see Section 6).

The input field specifications for describing and loading an array from a single input record are as follows:

I is entered in column 6.

Columns 7 through 42 are left blank.

Column 43 contains a P for packed decimal record format, B for binary format, or blanks for unpacked decimal format.

The field location of either an entire array of consecutive items or an individual item of the array is entered in columns 44-51. The beginning location is entered in columns 44-47 and the ending location is entered in columns 48-51.

Column 52 may be left blank. If a decimal position is entered, it must be identical to the decimal positions entry in the Extension Specifications Sheet.

The name of the array is entered in columns 53-58. The name must agree with the entry made in the Extension Specifications Sheet. The array name and index must be used for each item when individual items are being defined.

Columns 59-62 are left blank.

The field record relation indicator is entered in columns 63-64 in accordance with the rules supplied for field record relation entries in Section 6.

Columns 65-74 are left blank.

TABLE AND ARRAY CHANGES

When a table or array name is used as the result field in an arithmetic or move operation (see Section 7, "Operation Codes"), the table or array items may be changed during the execution of the object program. Additionally, any type of table or array (compile time, pre-execution time, or execution time) can be added to or changed using the Input Specifications Sheet or Calculation Specifications Sheet. Table and array changes are only in effect until execution of the object program is completed. The table or array reverts to its original entries when the program is next executed.

Changes can be made permanent only if the original data is changed. Changes made during program execution can be made permanent by writing or punching them out and using the new records, rather than the original records, in the table or array input file.

TABLE AND ARRAY REFERENCE

Either an entire array or an individual item in an array can be referenced using calculation specifications. The array name may be used without an index with the following operations:

ADD	MHHZO	MOVE	SUB
DEBUG	MHLZO	MOVEL	XFOOT
DIV	MLHZO	MULT	Z-ADD
LOKUP	MLLZO	SQRT	Z-SUB

Except when they are used for the XFOOT and LOKUP operations, factor 1 and factor 2 cannot contain array names unless the result field also contains an array name (see Section 7, columns 18-27, 33-42, and 43-48).

Several operation codes can be used only with an individual item, rather than an array name alone:

BITOF	DSPLY
BITON	TESTB
COMP	TESTZ

The array name, including the comma and index, must fit in the factor or result field in which it is used.

When array names are used without an index in calculations, the following rules apply:

Whenever the factors and the result field are arrays with the same number of items, the operation is performed sequentially. The first item from every array is used first, then the second item, and so forth, until all items in the arrays have been processed. If the arrays do not have the same number of entries, the operation ends when the last item of the array with the fewest items has been processed.

Whenever one of the factors is a field or constant, and the other factor and the result field are both arrays, the operation is performed until every item in the shorter array has been used. The same field or constant is used in all of the operations.

Since multiple operations are being performed, resulting indicators (columns 54-59) cannot be used except for the XFOOT and LOKUP operations.

When a table name is used in calculations, the operation is usually performed using only the table item last found in a LOKUP operation. The item is treated as any other single item would be. Only the LOKUP and XFOOT operation codes can reference all of the items in a table at one time.

TABLE AND ARRAY OUTPUT

When the last record indicator (LR) is on, entire tables and arrays, except for execution time arrays, can be written or punched out at the end of the job under automatic control of RPG II. Execution time arrays cannot be written or punched out at end-of-job under RPG II control. To indicate that a table or array is to be output, the name of the output file to be used is specified in columns 19-26 of the Extension Specifications Sheet. This causes the program to write out the entire table or array with all changes included. The Output-Format Specifications Sheet (Section 8) is used in conjunction with the Extension Specifications Sheet when tables and arrays are to be written out in a format different from the one in which they were read in. If execution time arrays need to be written out, the output-format specifications can be used to do so.

To indicate that an entire table or array is to be written on a single output record, the table or array is described on the Output-Format Specifications Sheet along with any other fields for the record. Columns 32-37 of the output specifications must contain the same table or array name as the one used on the Extension Specifications Sheet, and columns 40-43 must contain the record position where the last field of the array is to end.

If an output record is to contain only certain items from a table or array, the items are described on the output sheet in the same way as normal fields, either by an array name with an index or by a table name used as the field name. Table names on the Output-Format Specifications Sheets cause the last found table entry to be written out instead of the entire table.

When an entire array is to be edited, any editing specified applies equally to all items in the array. If differing editing requirements exist for individual items, these items must be referenced individually. (See Section 8, "Edited Fields," for more information on editing.)

When an edit code or edit word is specified on the output sheet for an entire array, edited array items are separated by two blanks each (see Section 8, "Edited Fields").

TABLE AND ARRAY DESCRIPTION

One extension specifications line is used for each table or array to be defined. The columns in which entries should be made depend upon whether a single table or array or alternating tables or arrays are to be loaded, and the loading time (compilation, pre-execution, or execution). When only one table or array is being described, columns 11-45 are used. When tables or arrays are being described in alternating format, columns 11-45 are used for the first table or array and columns 46-57 of the same line are used for the second.

Compile time tables or arrays are described in columns 27-45. Pre-execution time tables and arrays are described in columns 12-18 and 27-45. Execution time arrays are described only in columns 27-32 and 36-45.

Compile time and pre-execution time arrays can include entries in columns 19-26 ("to" filename) if the table or array described is to be written or punched at the end of the job, and in columns 46-57 to describe an alternating table or array. Execution time arrays cannot have "to" filename and alternating array specifications.

Tables and arrays may be specified on the extension sheet in any sequence; however, the sequence in which they are specified determines the order in which they will be loaded. Compilation time and pre-execution time tables and arrays can be mixed.

CODING INSTRUCTIONS

Detailed descriptions of the extension entries to be coded on the specifications sheet are defined below. Extension specifications are used to complete the description of record address files as well as to describe tables and array. Columns 7-80 may be used for comments (see Section 3, columns 7-14).

PAGE NUMBER (Columns 1-2)

Entry	Description
01-99	Specifications sheets are numbered sequentially.

More than one of each type of specifications sheet may be used in the course of coding the source program. To keep the sheets in the proper order, the coded Extension Specifications Sheets must be grouped together sequentially and arranged with the other specifications sheets in the following order:

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

The specifications sheets should then be numbered in ascending sequence.

LINE NUMBER (Columns 3-5)

Entry	Description
Any number	Line numbers are assigned in ascending sequence.

Line numbers are preprinted in columns 3-4 of the specifications sheet for the programmer's convenience. The unnumbered lines below the preprinted numbers can be used for additional lines or to insert a line between two completed lines. Column 5 can also be used to this end. The line numbers assigned need not be consecutive, but they must be in ascending order.

TYPE OF FORM (Column 6)

Entry	Description
E	E identifies the extension specifications to the compiler.

"FROM" FILENAME (Columns 11-18)

Entry	Description
Blank	Either a compilation time or an execution time table or array is being described.
Record address filename	The name of the record address file used in the file description specifications is entered.
Table or array filename	The name of the pre-execution time table or array file is entered.

The "from" filename entry in columns 11-18 is used to name a record address file or a file used for loading a pre-execution time table or array. Every pre-execution time table or array filename used in the program must be named on an extension specifications line. Each entry must begin in column 11 and can be from one to eight characters long. Since more than one

pre-execution time table or array can be read from the same file, the "from" filename may be the same for more than one table or array. When two or more pre-execution time tables or arrays are read from the same file, the tables or arrays must be in the same order as they are on the file extension cards and must be separated by records with ** in the first two record positions.

No entry should be made in columns 11-18 for compile time or execution time tables or arrays.

"TO" FILENAME (Columns 19-26)

Entry	Description
Blank	Either an execution time array is defined on this line; or the compile time or pre-execution time table or array being defined is not to be written or punched out at the end of the job.
Input or update filename	The name of the file that is processed by the record address file named in columns 11-18 is entered.
Output filename	The name of the output file on which a compile time or pre-execution time table or array is to be written or punched at end-of-job is entered.

If a record address file was described in the "from" filename columns (11-18), the "to" filename must contain the filename of an input or update file. The record address "to" filename is the name of the primary or secondary file to be processed by the record numbers or keys in the record address file.

Columns 19-26 can also be used to specify the filename of the output file to be used to write or punch the tables or arrays named in columns 27-32 and 46-51. Columns 19-26 should be left blank if the tables or arrays are not to be written or punched. Columns 19-26 cannot be used for execution time arrays, since execution time arrays cannot be written at end-of-job.

After all other records have been written or punched, a compile or pre-execution time table or array intended for output will automatically be written or punched in the same format in which it was entered, unless the output table or array is rearranged through output-format specifications (see Section 8). Table or array output can also be formatted by the use of exception lines so that one item is written at a time (see Section 7).

TABLE OR ARRAY NAME (Columns 27-32)

Entry	Description
Blank	A record address file is being described. (No further specifications should be entered on this line.)
Table or array name	The name of the table or array being described is entered.

Columns 27-32 are used to assign a name to the table or array being described. When alternating tables or arrays are being described, columns 27-32 must name the table or array whose item is found first on the input record.

The name assigned to a table or array must begin in column 27 and must follow the rules provided in "Table and Array Names" in the beginning of this section.

NUMBER OF ITEMS ON EACH RECORD (Columns 33-35)

Entry	Description
Blank	Indicates that an execution time array is being described.
1-999	Indicates the number of table or array items on each table or array input record.

Columns 33-35 are used to specify the exact number of table or array items on each table or array input record. The entry must end in column 35. Leading zeros are not required. Every input record for the table or array except the last record must contain the same number of items as the number specified in columns 33-35. The last record may contain fewer entries than that number, but never more.

When two tables or arrays are described in alternating format, each input record must contain the corresponding items from the tables written in alternating form. Corresponding items from alternating tables or arrays are considered one item and must be on the same record. Comments may follow table or array items on the table input records whenever there is room.

Columns 33-35 must be blank for execution time arrays.

NUMBER OF ITEMS IN TABLE OR ARRAY (Columns 36-39)

Entry	Description
1-9999	Indicates the maximum number of table or array items.

Columns 36-39 are used to specify the maximum number of table or array items which can be contained in the table or array named in columns 27-32. The entry must end in column 39. If a table or array is full, the number entered in columns 36-39 specifies the exact number of items in it. If a table or array is not full, the entry specifies the maximum number of items that can be put into the table or array. The entry also specifies the number of items in a table or array in alternating format (columns 46-51), since both must contain the same number.

LENGTH OF ITEM (Columns 40-42)

Entry	Description
1-15	Indicates the number of digits in each item of a numeric table or array.
1-255	Indicates the number of characters in each item of an alphanumeric table or array.

Columns 40-42 are used to specify the length of each item in the table or array named in columns 27-32. The entry must end in column 42. Leading zeros are not required. The length of a numeric item cannot exceed 15 digits, and the length of an alphanumeric item cannot exceed 255 characters.

When a numeric table or array in binary or packed decimal format is described, the unpacked decimal length must be entered (see Section 6, "Numeric Field Format"). 5 is entered for a two-character (two-byte) binary field and 10 is entered for a four-character binary field.

The length specified in columns 40-42 applies to all the items within a table or array. When necessary, leading zeros must be added to numeric items and leading blanks must be added to alphanumeric items to achieve the specified length.

When tables or arrays are described in alternating format, the entry applies to the table or array whose item appears first on an input record.

PACKED OR BINARY FIELD (Column 43)

Entry	Description
Blank	Data is either alphanumeric or in unpacked decimal format.
P	Data for the table or array is in packed decimal format.
B	Data for the table or array is in binary format.

The format of the items in a numeric table or array named in columns 27-32 is specified in column 43 as unpacked decimal, packed decimal, or binary. (See Section 6, "Numeric Field Format," for a discussion of formats.)

If the item is in packed decimal format, the entry must be P. If the item is in binary format, the entry must be B. If the item is in either alphanumeric or unpacked decimal format, column 43 is left blank. Compile time tables and arrays are always in either alphanumeric or unpacked decimal format (column 43 is left blank).

When tables or arrays are described in alternating format, the entry applies to the table or array whose item appears first on an input record.

DECIMAL POSITIONS (Column 44)

Entry	Description
Blank	The table or array items are alphanumeric.
0-9	The number of positions to the right of the decimal in numeric table or array items is specified.

No entry is made in column 44 for alphanumeric tables or arrays, but an entry must be made for every numeric table or array.

As many as nine positions may lie to the right of the decimal in numeric table or array items. If no decimal positions exist, the entry must be zero.

When alternating tables or arrays are described, the entry in column 44 applies to the table or array whose item appears first on an input record.

SEQUENCE (Column 45)

Entry	Description
Blank	The data in a table or array is not in any special order.
A	The data in a table or array is in ascending order.
D	The data in a table or array is in descending order.

The sequence entry is used to specify whether the data in a table or array is in ascending or descending order. The table or array is then checked when it is loaded for the sequence specified (see Section 6, "Sequence"). A severe error occurs if a pre-execution time table or array is out of sequence, and the program halts immediately. Execution time arrays are not checked for sequence, but column 45 must contain an entry if high or low LOKUP is to be used (see Section 7, "Operation Codes").

When alternating tables or arrays are described, the sequence entry applies to the table or array whose item appears first on an input record.

ALTERNATING TABLE OR ARRAY (Columns 46-57)

If alternating tables or arrays are being described, the specifications for the second table or array are entered in columns 46-57. Tables must alternate with tables, and arrays must alternate with arrays. The table or array described in these columns must be loaded in alternating format with the table or array named in columns 27-32 of the same line. An execution time array cannot have an alternating array.

Entries in columns 46-57 must follow the same criteria given for the corresponding columns, 27-32, used for the first table or array described.

COMMENTS (Columns 58-74)

Entry	Description
Any characters	These columns are used for programmer comments.

Since columns 58-74 are disregarded by the compiler, they can be used to make notations. These will appear on the program listing after compilation and may assist the programmer in remembering details about the table or array described on each specification line.

Comments columns serve solely as an aid in documenting the source program, and they may contain any characters in the character set.

PROGRAM IDENTIFICATION (Columns 75-80)

Entry	Description
Any characters	These columns may be left blank or used for program identification or for comments.

The program identification field of all cards following the control card may contain any characters, including blanks. The field may therefore be a continuation of the comments field in columns 58-74.

SECTION 5. LINE COUNTER SPECIFICATIONS

Line counter specifications are used for printer files in a program when external controls, such as sensing channel 12 for overflow, are not applicable or not desirable. The specifications are written on the RPG II Extension and Line Counter Specifications Sheet (Figure 5-1). In addition to their use in controlling spacing by relating line numbers on a form to printer channel numbers, line counter specifications may alternatively be used to indicate the length of the form used in a printer and at what line overflow occurs. If no line counter specifications are made, the form length is assumed to be 66 lines, and line 60 is assumed to be the overflow line.

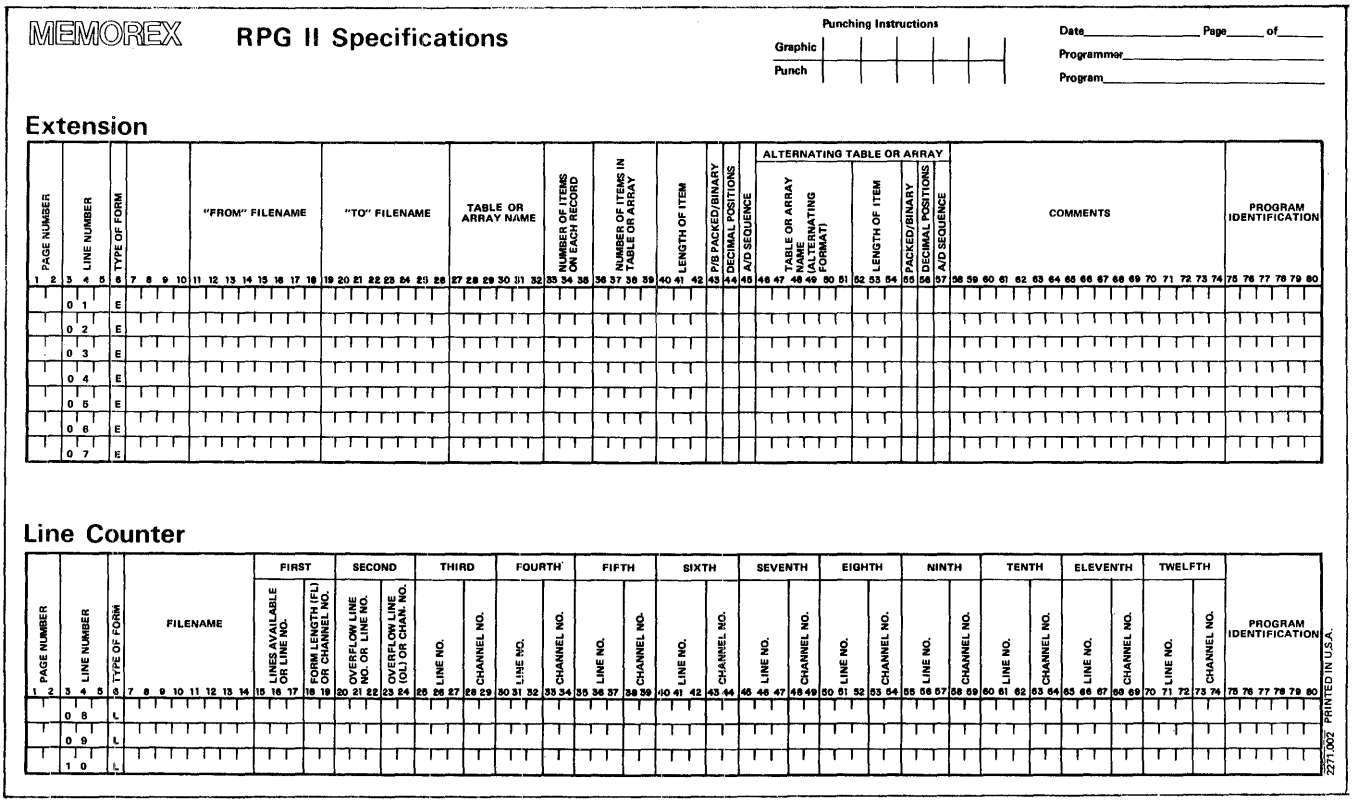


Figure 5-1. RPG II Extension and Line Counter Specifications Sheet.

CODING INSTRUCTIONS

Detailed descriptions of the line counter entries to be coded on the specifications sheet are defined below.

PAGE NUMBER (Columns 1-2)

Entry	Description
01-99	The specifications sheets are numbered sequentially.

More than one of each type of specifications sheet may be used in the course of coding the source program. To keep the sheets in the proper order, the coded Line Counter and Extension Specifications Sheets must be grouped together sequentially and arranged with the other specifications sheets in the following order:

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

The specifications sheets should then be numbered in ascending sequence.

LINE NUMBER (Columns 3-5)

Entry	Description
Any number	Line numbers are assigned in ascending sequence.

Line numbers are preprinted in columns 3-4 of the specifications sheet for the programmer's convenience. The unnumbered lines below the preprinted numbers can be used for additional lines or to insert a line between completed lines. Column 5 can also be used to this end. The line numbers assigned need not be consecutive, but they must be in ascending order.

TYPE OF FORM (Column 6)

Entry	Description
L	L identifies the line counter specifications to the compiler.

FILENAME (Columns 7-14)

Entry	Description
1-8 characters	The output file to be printed is named.

The filename must begin in column 7. Filenames entered on the Line Counter Specifications Sheet must also be defined on the File Description Specifications Sheet. The output device assigned to the file specified on the File Description Specifications Sheet must be a printer.

A valid filename consists of one to eight characters. The first character must be alphabetic; the remaining characters may be any combination of alphabetic and numeric characters. Special characters are not permitted, and the first character cannot be one of the following characters: #, $, or @. Blanks cannot appear between characters in the name. Columns 7-80 may also be used for comments (see Section 3, columns 7-14).

LINES AVAILABLE OR LINE NO. (Columns 15-17)

Entry	Description
1-255	Indicates the number of print lines available on a page or form.

The lines available entry specifies the exact number of lines available on the page or form to be printed. The entry must be right-justified. Leading zeros may be omitted. When no entry is made, the standard value of 66 lines is assumed.

Columns 15-17 may also be used to relate a report line to a channel number. See columns 15-74 for an explanation of this type of specification.

When columns 15-17 are used to specify the number of lines available on the page, an L must be specified in column 53 of the control card.

FORM LENGTH (FL) OR CHANNEL NO. (Columns 18-19)

Entry	Description
FL	Indicates to the program that the form length was specified in columns 15-17.

FL is mandatory in columns 18-19 if the form length has been specified in columns 15-17. When FL is used, an L must be specified in column 53 of the control card.

Columns 18-19 may also be used in conjunction with columns 15-17 to specify a channel number related to a report line. See columns 15-74 for an explanation of this type of specification.

OVERFLOW LINE NO. OR LINE NO. (Columns 20-22)

Entry	Description
1-255	Specifies the line number of the overflow line.

The overflow line number entry specifies the exact point on a page or form at which overflow occurs. The entry must be right-justified. Leading zeros are not required. When the overflow line no. entry is left blank, the standard value of 60 is assumed.

Indicating a line slightly above the bottom of the page as the overflow line enables totals information to be included between the specified overflow line and the bottom of a printed page or form. When a space, skip, or print operation (see Section 8, columns 17-22) carries printing one line below the overflow line, the overflow indicator turns on to signal that the end of the page is near. When the overflow indicator is on, the program

automatically prints the extra information before forms are advanced to the next page. Detail lines are printed, if this part of the program cycle has not already been completed; total lines are printed; and total lines conditioned by the overflow indicator are printed.

The overflow line should be placed high enough on the page to allow for this wrap-up and for any special printing.

Columns 20-22 may also be used to relate a report line to a channel number. See columns 15-74 for an explanation of this type of specification.

OVERFLOW LINE (OL) OR CHANNEL NO. (Columns 23-24)

Entry	Description
OL	Indicates to the program that an overflow line number was specified in columns 20-22.

The OL entry is required in columns 23-24 whenever an overflow line number has been specified in columns 20-22. OL is normally used if an L is specified in column 53 of the control card.

Columns 23-24 may also be used to relate a channel number to a report line. See columns 15-74 for an explanation of this type of specification.

LINE NO. and CHANNEL NO. (Columns 15-74)

Entry	Description
1-255	Specifies the line number.
1-12	Specifies the related channel number.

The fields in columns 15-74 may relate specific line numbers on the report to channel numbers used on the Output-Format Specifications Sheet (Section 8). The line numbers of the form are entered in the line no. columns, and the channel numbers to be related to each line are entered in the associated channel no. columns. The specifications for the channels may be entered in any order. The entries must be right-justified. Leading zeros may

be omitted. The highest permissible line number is 255, and the highest channel number is 12.

A form length should not be specified for a file when channel numbers are being used. Columns 15-19 should be used to specify the first line number-to-channel number relation, or they may be used instead of columns 20-24 to specify the overflow line number.

Channels may be used to control the spacing on a printed page. The printer carriage control tape illustrated on a print chart coding form (see Figure 1-5) has 12 vertical lines on it, one for each channel. Holes may be punched on the tape along channel lines, opposite output lines. If the program encounters a channel skip instruction (see Section 8, columns 19-22), the printed page is automatically advanced to the next output line associated with the given channel. The line-to-channel number relations are specified on the Line Counter Specifications Sheet.

The rules that apply to the printer carriage control tape also apply to line counter channel number assignments. Only the channels used in the output-format specifications should be specified. Several output-format specifications lines may be related to one channel. Only a single channel number may be assigned to a particular line. Only one line of the Line Counter Specifications Sheet may be used for a file.

Channel 1 is pre-assigned to the first print line on the page, and channel 12 is pre-assigned to the overflow line; thus "overflow line"/"OL" and "channel 12" are synonomous terms and should not be used together. If channel 1 is not assigned, line 6 is assumed to be the first print line of the page; and if channel 12 is not assigned, line 60 is assumed to be the overflow line. Both channels 1 and 12 must be assigned if automatic skipping from the end of a form after channel 12 to the beginning of the next form at channel 1 is desired.

Line Counter

PAGE NUMBER	LINE NUMBER	TYPE OF FORM	FILENAME	FIRST: LINES AVAILABLE OR LINE NO.	FIRST: FORM LENGTH (FL) OR CHANNEL NO.	SECOND: OVERFLOW LINE NO. OR LINE NO.	SECOND: OVERFLOW LINE (OL) OR CHAN. NO.	THIRD: LINE NO.	THIRD: CHANNEL NO.	FOURTH: LINE NO.	FOURTH: CHANNEL NO.	FIFTH: LINE NO.	FIFTH: CHANNEL NO.	SIXTH: LINE NO.	SIXTH: CHANNEL NO.	SEVENTH: LINE NO.	SEVENTH: CHANNEL NO.	EIGHTH: LINE NO.	EIGHTH: CHANNEL NO.	NINTH: LINE NO.	NINTH: CHANNEL NO.	TENTH: LINE NO.	TENTH: CHANNEL NO.	ELEVENTH: LINE NO.	ELEVENTH: CHANNEL NO.	TWELFTH: LINE NO.	TWELFTH: CHANNEL NO.	PROGRAM IDENTIFICATION
03	08	L	FLE1A	50	FL	45	OL																					
03	09	L	FLE2B	5	2	10	3	50	12																			
	10	L																										

2271.002 PRINTED IN U.S.A.

Figure 5-2. Report Line-to-Channel Number Relation.

Figure 5-2 shows how entries are made to relate a report line to a channel number. On the first line, entries in columns 15-24 specify a form length of 50 lines (50FL) and assign overflow at

line 45 (450L) for file FLE1A. An L must have been specified in column 53 of the control card (Section 2, column 53).

No form length is specified for file FLE2B, but all the entries in columns 15-29 are used to relate report lines to channel numbers. The fifth line of the report is assigned to channel 2, the tenth line of the report is assigned to channel 3, and the fiftieth line is specified as the overflow line by assigning it to channel 12. The first print line of the report is assumed to be line 6.

Figure 5-2 shows the two ways in which the line counter specifications can be coded. When form length and overflow line entries are used they should be placed in columns 15-24. When channel number assignments are used, specifications must begin in columns 15-19.

PROGRAM IDENTIFICATION (Columns 75-80)

Entry	Description
Any characters	These columns may be used for programmer comments.

The program identification entry of all cards following the control card may contain any characters, including blanks. The columns may be used for programmer comments, if desired.

SECTION 6. INPUT SPECIFICATIONS

Data files, records, and fields of the records to be used as input by an object program are described on the RPG II Input Specifications Sheet (Figure 6-1). Specifications for the descriptions fall into two divisions: file and record type identification entries, specified in columns 7-42; and field description entries, specified in columns 43-74. The field description entries must start at least one line lower than the file and record type identification entries.

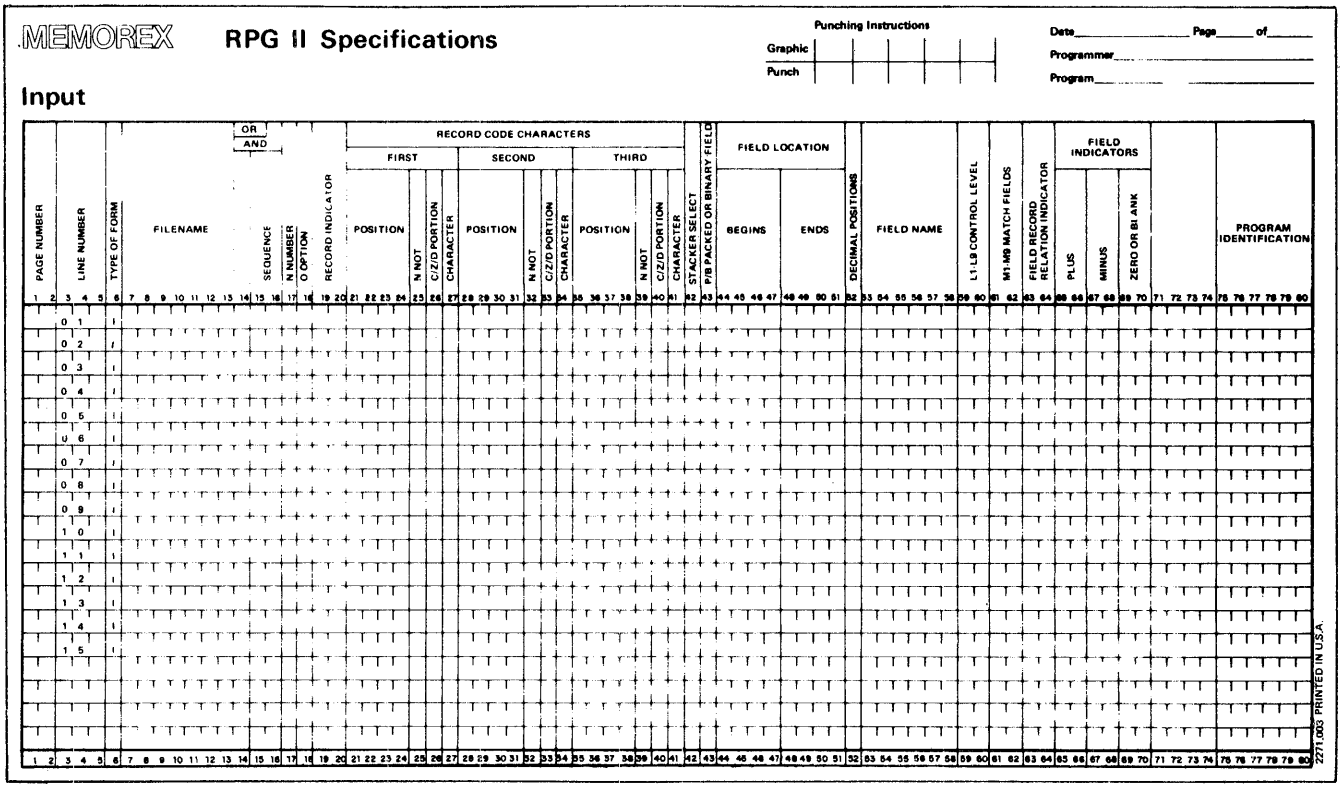


Figure 6-1. RPG II Input Specifications Sheet.

RECORD INDICATORS

A single file can contain more than one type of record, and different operations can be performed for each type. To control the processing of the various record types, a record indicator is assigned to each type in columns 19-20. When a record type is

selected for processing during the execution of the object program, its corresponding record indicator is turned on to identify the record type to the system. (All other record indicators are off unless a chained file is being processed; more than one record in a chain file can be processed at a time.) Once the indicator is turned on, the system knows what record type has been selected for processing and the proper operations can be performed.

Columns 19-20 can be used for three purposes: to specify record identifying indicators, to indicate look-ahead fields, or to designate a spread card trailer. (See Appendix B, Tables B2 and B3, and Section 7, "Indicator Setting Operations," for indicator summaries.)

RECORD IDENTIFYING INDICATORS

Record identifying indicators are used to distinguish between record types or to check for a record type that causes an error condition.

LOOK-AHEAD FIELDS

The use of look-ahead fields allows the System to look at information in fields on the next record available for processing in an input file. In update files, the entry refers to a field in the record currently being processed. Look-ahead fields cannot be specified for chained or demand files, or as array fields.

Once the last record from a file has been processed, every look-ahead field for the file is automatically filled with 9's which remain in the fields until the job is ended.

SPREAD CARDS

Spread cards allow one or more transactions (trailers) or items of information to be linked to a single header. RPG II's spread card capability eliminates the need for a separate header card for each transaction (trailer portion). A single card can contain the header and all of the transactions. The information can be compressed so that a trailer portion may consist of as many fields as are necessary; however the same fields must appear in

each trailer portion following the header. A trailer portion cannot be split between two records.

CHARACTER STRUCTURE

The structure of a character is important in its use as part of the record code (columns 21-41). Each alphabetic, numeric, or special character is represented by a unique combination of punches in a keypunched card. Each punched character consists of two parts: a zone portion, and a digit portion. When the character is read into the computer, it retains this composition even though it has been translated into eight magnetic bits for computer representation in storage. As a result of the translation, the representation of a character in the machine may not be identical to its configuration on a keypunch card; those characters with the same zone punch in the card may have different zone punches in the computer. Characters are stored in the machine in EBCDIC form. It is the EBCDIC machine representation of a character that must be considered in such functions as sequencing, testing, identifying, or comparing records.

All characters can be arranged in a certain order which is governed by the way their zone and digit portions are represented in the machine. Thus each character's machine representation will have a special position in relation to all others in a collating sequence.

NUMERIC FIELD FORMAT

The entry in column 43 describes the format of a numeric field. Three different numeric field formats are used in RPG II: unpacked decimal (Figure 6-2), packed decimal (Figure 6-3), and binary (Figure 6-6).

UNPACKED DECIMAL FORMAT

In unpacked decimal format, each byte of computer or disc storage can hold one alphabetic, numeric, or special character. Each byte

is divided into eight bits: a four-bit zone portion followed by a four-bit digit portion. Each digit in a decimal number contains a zone position which indicates whether the number is positive or negative. In unpacked decimal format, only the zone of the rightmost byte of the rightmost digit serves as the positive or negative sign.

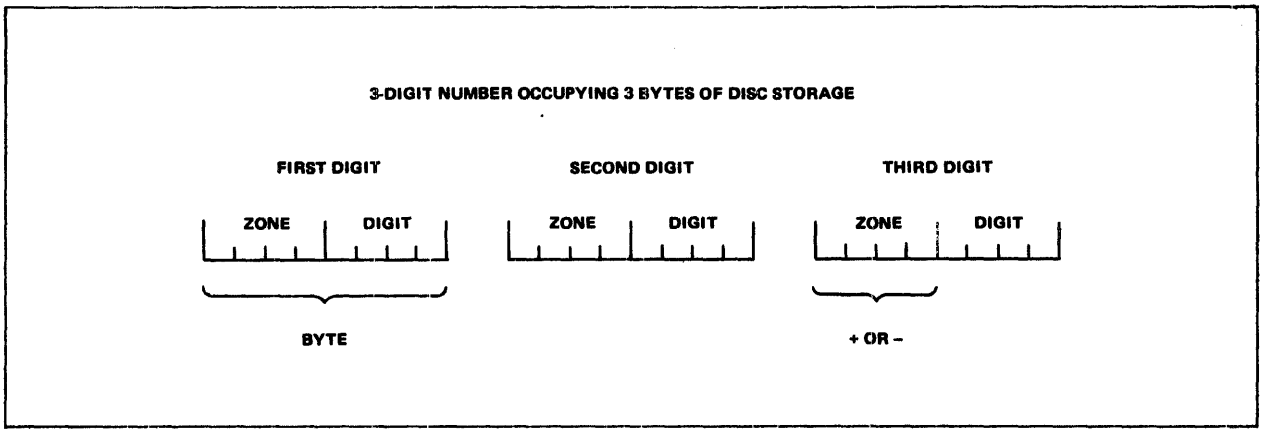


Figure 6-2. Unpacked Decimal Format.

PACKED DECIMAL FORMAT

In packed decimal format, each byte of storage is divided into two four-bit digit portions and no zone is used. The plus or minus sign for a packed decimal field is contained in the rightmost portion of the rightmost byte.

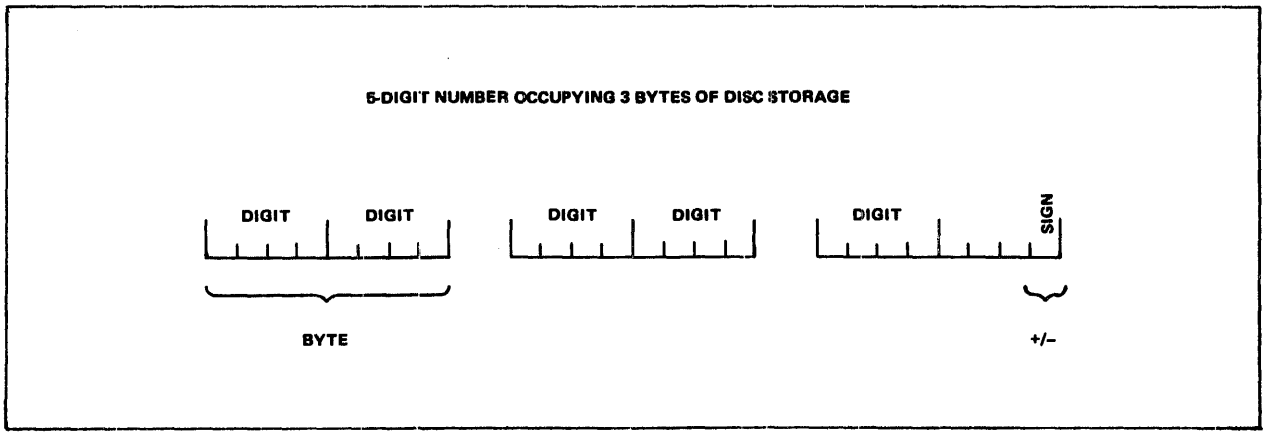


Figure 6-3. Packed Decimal Format.

PACKED DECIMAL REPRESENTATION

The packed decimal coding system uses a four-bit code to represent a numeric character, but to conserve space the zone bits are excluded and a sign bit is added to the least significant byte. If the number is negative, the rightmost four bits of the least significant byte contain 1101. If the number is positive, the rightmost four bits of the least significant byte contain 1100. Figure 6-4 illustrates this convention with the number -4,902.

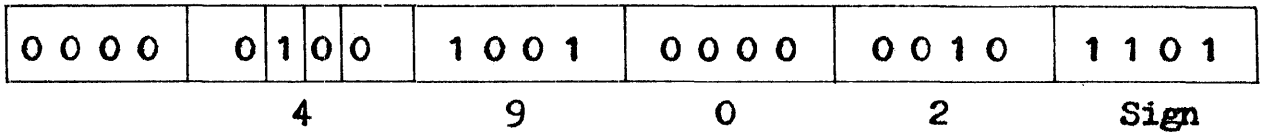


Figure 6-4. Packed Decimal Representation in the Computer.

When an external device enters unpacked numeric data, each digit is preceded by a four-bit zone byte. Figure 6-5A illustrates the greater quantity of memory which would be required if the System could not pack the decimal codes before they were stored, and Figure 6-5B illustrates the number as it would appear in memory if it were packed. Both figures use the number 5,842.

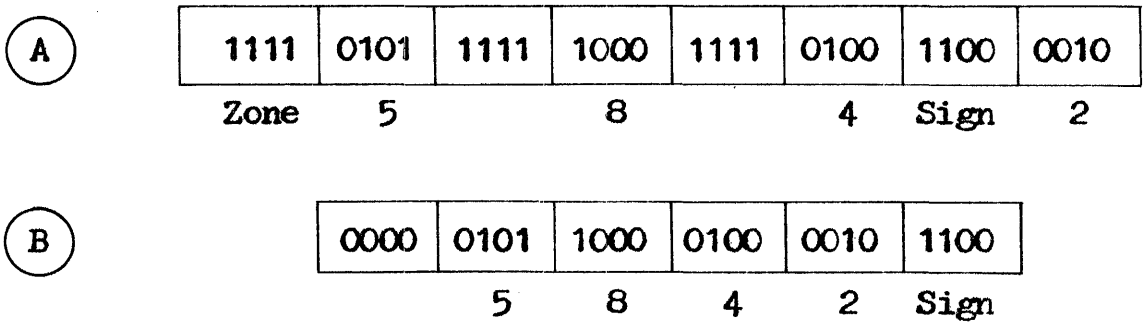


Figure 6-5. Memory Required for Packed vs. Unpacked Numeric Data.

Packed fields are positive if the sign is a hexadecimal F, A, C, E, 0, 2, 4, 6, 7, or 8. Packed fields are negative if the sign is a hexadecimal B, D, 1, 3, 5, or 9.

BINARY FORMAT

Binary format numbers are in twos complement form. In binary format, each field on disc must be either two or four bytes long. Two bytes of disc storage can contain up to five decimal numbers, and four bytes of disc storage can contain up to ten decimal numbers.

Two-byte binary fields contain a one-bit sign followed by a 15-bit numeric value so that in binary format any decimal number up to 32,767 requires only two bytes of disc storage, at the most. For each two-byte binary field stored on disc, the system automatically sets aside five digits of storage to accommodate the field when it is converted to packed format.

Four-byte binary fields contain a one-bit sign followed by a 31-bit numeric value; thus in binary format any decimal number as high as 2,147,483,647 requires a maximum of only four bytes of disc storage. The system automatically sets aside ten digits of memory storage to accommodate each four-byte binary field stored on disc when it is converted to packed format.

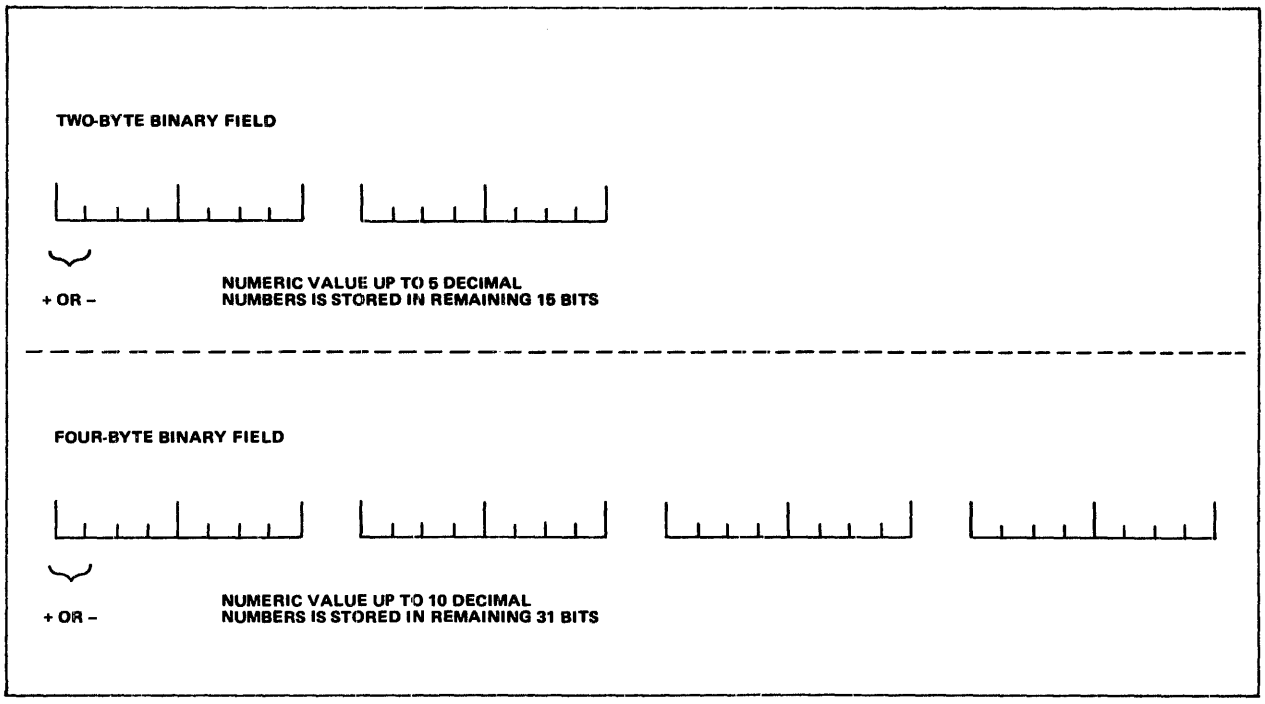


Figure 6-6. Binary Format.

FIELD NAMES

A field name is used in columns 53-58 to identify a particular field, array, or array item in an input record. Each field in each record type must be given a unique name from one to six characters long. The assigned name can then be used throughout the program whenever the field is referenced.

Each field name assigned requires a separate line on the Input Specifications Sheet. Names of the fields must be indicated for all types of records, however only the fields that are used need to be named. Fields which are read in from a card are limited to the length of one punched card.

All fields in a specific record type should be assigned different names, since only the field described last is used if fields are identically named within a record type. If they are the same length and contain the same type of data, however, fields from different record types may have the same name regardless of their location in each record type. Although each field with the same name still requires a separate description, the OR relationship may be used to avoid duplicate coding.

If the form or report to be output consists of several pages, it is usually desirable to number the pages. Page numbering normally begins with page 1 and is handled automatically when the pre-defined RPG II field name PAGE, PAGE1, or PAGE2 is entered in the Output-Format Specifications Sheet (see Section 8, columns 32-37), however the field name entry can be used in the Input Specifications Sheet to indicate that page numbering is to start at a page other than page 1.

CONTROL FIELDS

A control field is a field to which a control level indicator (L1-L9, columns 59-60) has been assigned to indicate the point at which specified operations are to be performed.

Whenever a record which contains a control field is selected, the data in the control field is compared with the data in the control field of the previously selected record. All records with the same information in the control field form a control group. If two control fields are not the same, a control break occurs, and the control level indicator is turned on. A control break allows operations conditioned by the control level indicator to be performed before a new record is processed (see Section 7, columns 9-17 and Section 8, columns 23-31). Undesirable control

breaks may occur if all the record types within a given file do not have the same number of control fields.

A control break most commonly occurs after the first record containing a control field is read, since it is compared to an area in storage which contains no data. Total calculations and total output operations are bypassed for the first record containing a control field because fields from two different records are not being compared.

Within the same record type, record columns in control fields assigned different control level indicators may overlap, but the total number of columns assigned as control fields, when each control level is counted only once, must not exceed 255.

NUMERIC CONTROL FIELDS

If a control field is specified as numeric in column 52, all control fields for which the same control level indicator has been specified are considered numeric.

Control fields are initialized to hexadecimal (logical) zeros, or to the lowest alternate collating sequence value given, before they are processed. When numeric control fields with decimal positions are compared for a control break, the decimal point is ignored; all the digit portions of a specified numeric field are used in the comparison, and all fields are treated as unsigned fields with the number of decimal positions ignored.

SPLIT CONTROL FIELDS

A split control field is a control field that is made up of two or more connected or unconnected fields on a record. The same control level indicator is assigned to all parts of a split control field. The indicator-flagged fields are then combined by the program in the order specified in the input specifications and treated as one control field.

A single control level indicator is assigned to all portions of a split control field. Since field names are ignored in control level operations, fields from different record types which have been assigned the same control level indicator may have the same name.

If the field names are different, the length of the portions of a split control field may vary for different record types, or the field may be split in some record types and not in others. In both cases the total length of the portions of the split field must be the same in all record types.

If the same control level indicator is used in different record types or files, the control fields associated with that control level indicator must agree in length and content, containing either alphabetics or numerics. The format of the portions of a split control field cannot vary; all must be packed decimal, or all must be alphabetic.

A numeric split control field may contain more than 15 characters if any one portion of the split field does not exceed 15 characters and the sum of all the components of the control field does not exceed 255 characters.

CONTROL LEVEL INDICATORS

Control level indicators always turn on after the first record of a control group is read, to indicate that a change in a control field has occurred. Since indicator LO is always on, it can never be used as an assigned indicator in the input specifications.

Control level indicators are used in columns 59-60 to signal that operations are to be performed on only the first record in a control group; that operations are to be performed after all records with the same control field information (all records in the control group) have been read; or to print totals information. (Control level indicators are also used as record indicators in columns 19-20, and as field record relation indicators in columns 63-64.)

Control level indicators are assigned to control fields in the order of the fields' importance, but they need not be assigned in any sequence and gaps can occur in the control levels assigned. Before an indicator is assigned, each field should be assessed for its importance in relation to the other control fields, and the indicator chosen accordingly. Larger numbers are higher in rank than lower numbers. All indicators with a lower rank automatically turn on or off when a higher-ranking level break occurs; for example, when an L3 control break occurs, control level indicators L1, L2, and L3 all turn on.

Control level indicators can be turned on or off by operation codes SETON and SETOF (see Section 7, "Operation Codes"), however control level indicators lower in rank than the one specified do not turn on or off in this case.

MULTIFILE PROCESSING

The selection of records for processing from more than one file is called multifile processing. Although records are always read from a primary file and one or more secondary files in multifile processing, the method used in selecting them is governed by the presence or absence of match fields in the records.

If match fields are not specified, the primary file is processed first and the secondary files are processed in the order in which they are described in the file description specifications. If match fields are specified (columns 61-62), fields from records in the primary file are compared to fields from one or more secondary input or update files for a matching record.

MATCH FIELDS

When the match fields method of record selection is used, records are selected according to the contents of specified fields. One record is read from every file and specified fields in each record are compared to the primary file for a match. Whenever a record from the primary file matches a record from the secondary file, the primary file record is processed first, then the matching secondary file record is processed, unless another file is forced (see Section 7, "Operation Codes - FORCE"). The record indicator (columns 19-20) is on at the time the record is processed to identify the record type just selected and control the type of processing that takes place.

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.

After a record has been selected from a file and processed, the next record from that file is read. It is compared for a match field at the beginning of the next program cycle, using for comparison the records that were not selected during the previous cycle. Match comparing continues in the same manner until all the records in all the files have been processed.

Records can be matched by comparing one field or many fields. When the primary record field matches a field in one or more of the secondary records, the matching record indicator (MR) turns on. The matching record selected for processing is determined by the priority of the files from which the records came.

A record for which no match field has been specified is selected before records with specified match fields, and is processed immediately after the record it follows, with the MR indicator off. If this record type is first in the file, it is processed first even if it is not the primary file. If match fields have not been specified for two or more of the records being compared, the order of their selection is determined by the priority of the files from which the records came.

Match fields are not required for all the files used or for all the record types within one field; however at least one record type from two files must have match fields if files are ever to be matched. Sequence checking, alone, is performed if match fields are specified for only one file.

MATCHING RECORD INDICATOR

The MR indicator is used to condition calculation or output operations for a selected record with a match field. The indicator turns on when a specified field in a primary file record matches a specified field in a secondary file record. The indicator is always set on or off before any calculation operations not conditioned by control level indicators are performed (see Section 7, columns 7-8), and retains the setting for one complete program cycle. If all primary file records match all secondary file records, the MR indicator is always on. If record types are read for which no matching fields have been specified, MR turns off.

CODING INSTRUCTIONS

Detailed descriptions of the input entries to be coded on the Input Specifications Sheet are defined below.

PAGE NUMBER (Columns 1-2)

Entry	Description
01-99	The specifications sheets are numbered sequentially.

More than one of each type of specifications sheet may be used in the course of coding the source program. To keep the sheets in the proper order, the coded Input Specifications Sheets should be grouped together sequentially and arranged in the following order:

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

The specifications sheets should then be numbered in ascending sequence.

LINE NUMBER (Columns 3-5)

Entry	Description
Any number	Line numbers are assigned in ascending sequence.

Line numbers are preprinted in columns 3-4 of the specifications sheet for the programmer's convenience. The unnumbered lines below the preprinted numbers can be used for additional lines or to insert a line between completed lines. Column 5 can also be used to this end. The line numbers assigned need not be consecutive, but they must be in ascending order.

TYPE OF FORM (Column 6)

Entry	Description
I	I is preprinted to identify the input specifications to the compiler.

FILE AND RECORD TYPE IDENTIFICATION ENTRIES

FILENAME (Columns 7-14)

Entry	Description
1-8 characters	The input or update file being described is named.

Every input or update file named on the Input Specification Sheet must previously have been described on the File Specifications Sheet. The exact filename originally chosen must be used and must begin in column 7. If the filename is left blank, the last filename entered is assumed to be the file being described.

The filename is entered on the first line containing information relevant to the records in that file. The information on the following lines is considered part of the description of the file named until a new filename is entered in columns 7-14. Thus it is not necessary to repeat the filename entry when more than one line of description is used for a file. All records and fields for a file must be completely described before a new file is named for description. Columns 7-80 may also be used for comments (see Section 3, columns 7-14).

SEQUENCE (Columns 15-16)

Entry	Description
Any two alphabetic characters	Sequence checking is not to be done.
Any two numeric characters	A special sequence is assigned to different record types in a file.

Sequence checking is not necessary for every file (see Section 8, column 18). If different record types do not need to be in any special order, any two alphabetic characters can be entered in columns 15-16. An alphabetic entry must be made for chained files, however, since sequence checking is not possible for that type of file.

When both alphabetic and numeric sequence entries are specified for record types within a single file, the alphabetic entries must be made before the numeric entries.

A numeric entry in columns 15-16 assigns a sequence number to different types of records within a file so that records with a particular identification code may be checked before records of another type which bear another identification code. The record types are assigned sequence numbers in the order in which they are to appear. The first record type must be given the lowest sequence number, 01, and the remaining record types must be numbered in ascending order. Sequence numbers need not be consecutive as long as they are in ascending order.

Sequence numbers have no connection with control levels. Sequence checking is used solely to ensure that all records of type 01 precede those of type 02, and so forth. Sequence checking cannot be used to designate the order of records within a record type, or to check the data fields within a record (see "Multifile Processing" at the beginning of this section).

Sequence entries are not made for records in an AND or OR line. The sequence entry from the previous line applies to the card in the OR line (see columns 53-58).

NUMBER (Column 17)

Entry	Description
Blank	Sequence checking is not to be performed (the sequence entry in columns 15-16 is an alphabetic entry).
1	Only one record of this type is present in the sequenced group.
N	One or more records of this type may be present in the sequenced group.

Column 17 is used to designate the number of records of one type that are found in the sequence group. When sequence checking is not required for the records being described, column 17 is not used. The 1 entry indicates that only one record of the type specified in columns 15-16 (sequence) can be present in the sequence group. The N entry is used when more than one record of the type specified can be present in the sequence group.

No number entry is required for AND lines and OR lines. The number designated in the preceding line is applied. (See columns

21-41 for a description of AND lines; columns 53-58 for a description of OR lines.)

OPTION (Column 18)

Entry	Description
Blank	The record type must be present in each sequenced group.
O	The presence of the record type is optional.

A blank entry indicates that a record of the specified type must be present in each sequenced group. When sequence checking is to be performed on record types, the O entry indicates that a record of the type specified in columns 15-16 (sequence) may or may not be present. The entry prohibits a sequence error from occurring. An O should be specified only if a numeric entry was made in columns 15-16.

No option entry is required for AND or OR lines (see columns 21-41 and 53-58, respectively). The entry from the preceding line is applied to an AND or OR line.

RECORD INDICATOR (Columns 19-20)

Entry	Description
01-99	The record identifying indicator is used to specify a particular type of record in a file.
L1-L9	The control level indicator is used as a record identifying indicator to indicate that a record type signals the start of a new control group.
LR	The last record indicator is used to indicate that the job is to end.
H0-H9	The halt indicator is used to check for a record type that causes an error condition.

**	Asterisks are used to indicate look-ahead fields in a record.
TR	The trailer specification is used to link trailer portions of spread cards to the header portion.

Record indicator entries are used for a variety of situations. They can be used as record identifying indicators, to indicate look-ahead fields, or to denote spread cards.

Record indicators can be assigned in any order; there is no prerequisite sequence. A single record indicator may be assigned for any number of different record types if the same operations are to be performed on all the types (this is a function of the OR relationship — see columns 21-41). They need not be assigned at all if the difference in record types is not important or if only one type of record is specified with a file.

No record indicator may be assigned for records entered in the AND line of an AND relationship (see columns 21-41). Record indicators may be assigned in the OR line for every record type in the OR relationship that requires special processing (see columns 53-58).

Record identifying indicators comprise four kinds of entries: 01-99, L1-L9, LR, and H0-H9.

01-99

01-99 is the most frequently assigned record identifying indicator. It is used to identify a specific record type in a file that contains two or more types of records.

L1-L9

L1-L9, the control level indicator, is used to identify a record only when a record type, rather than a control field (see "Control Fields" at the beginning of this section), determines the start of a new control group. If a control level indicator is used as a record indicator, only that control level indicator turns on during record selection. All lower level indicators remain off. (See Section 7, "Control Levels," for more information on the control level indicators.)

LR

LR is used to indicate that a job is to terminate. (See Section 7, "Resulting Indicators," for more information on the LR indicator.)

H0-H9

H0-H9, the halt indicator, is used to indicate that checking is to be performed for a record type that causes an error condition. (See Section 7, "Resulting Indicators," for more information on the halt indicator.)

Look-Ahead Fields

Look-ahead fields are indicated by entering asterisks (**) in columns 19-20 (Figure 6-7). Look-ahead fields are specified for a file rather than for a record type within a file. One set of look-ahead fields can be described for each file. The description applies to all records within a file, regardless of their type. The ** entry is made following the normal field description for a file, and the ** line can follow only a file or record type with an alphabetic sequence entry in columns 15-16. Columns 17-18 and 21-74 must be left blank when a look-ahead field indicator is used. Look-ahead field descriptions are made in the field name columns (53-58) on separate lines following the ** line.

Look-ahead fields cannot be altered in the program. They cannot be used as a result field (see Section 7, columns 47-48) or blanked out after they are written (see Section 8, column 39). If the information in a record is to be used both before and after a record is selected for processing, the field must be described once as a look-ahead field, then as a normal field.

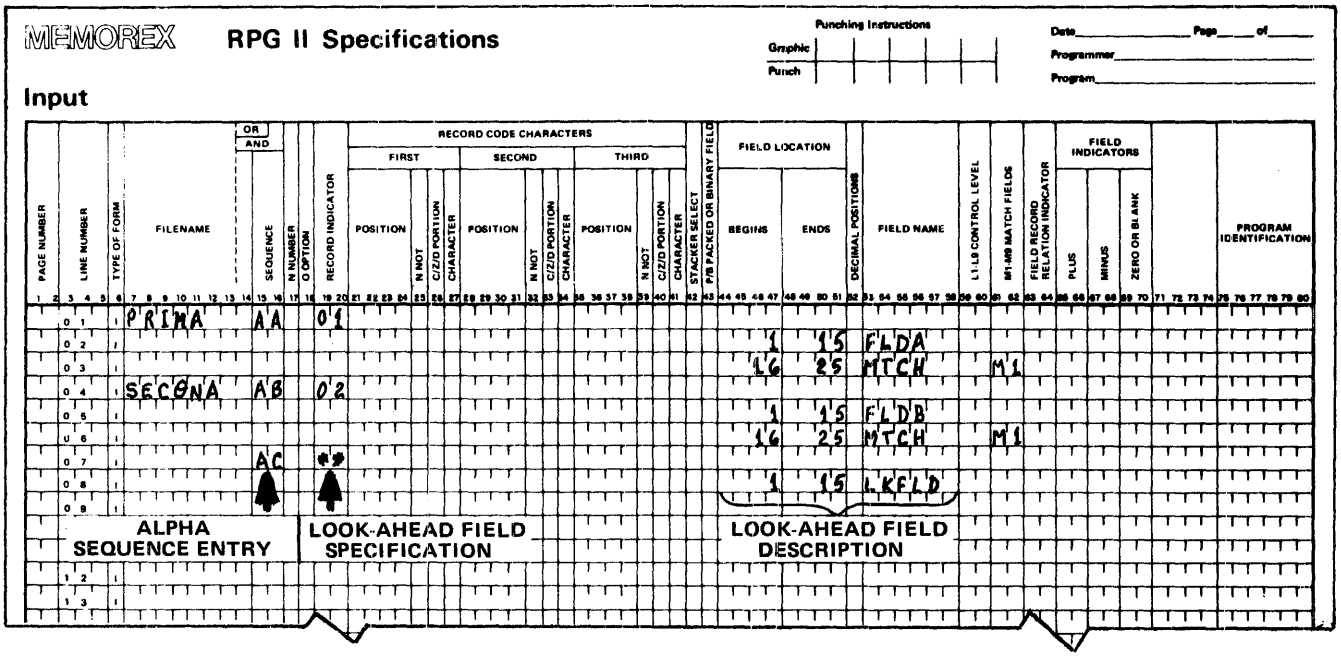


Figure 6-7. Coding for Look-Ahead Fields.

Spread Cards

Spread cards are indicated by entering TR in columns 19-20 (Figure 6-8). Spread cards can be specified only when the input card files are designated as primary or secondary. Look-ahead fields cannot be described for spread cards.

The fields in the header portion of the spread card are described on separate specification lines immediately following the proper file and record type entries. If a numeric entry is made in columns 15-16 of the specifications line containing the file and record type entries, an N must be entered in column 17 of the same line.

Each field in the header portion must be described as would any normal RPG II field. Only those fields in the header portion that are used later in the program need be described. If no field in the header portion is used, the header field specification can be omitted and the TR line specified immediately after the file and record type entries.

TR must be entered in columns 19-20 of a specification line to indicate that the fields in the first trailer portion are described in the specification lines that follow. Columns 7-18 and 21-74 of the TR lines are left blank. Since all trailer portions must be the same length and must include the same fields, only the first one must be described. The compiler uses

this trailer specification to calculate how many trailer portions the record contains and to determine the beginning and ending positions of each.

Trailer field description lines are specified immediately following the TR line. Columns 7-43 and 59-62 must be left blank; otherwise the trailer field descriptions follow the same rules as regular field specifications.

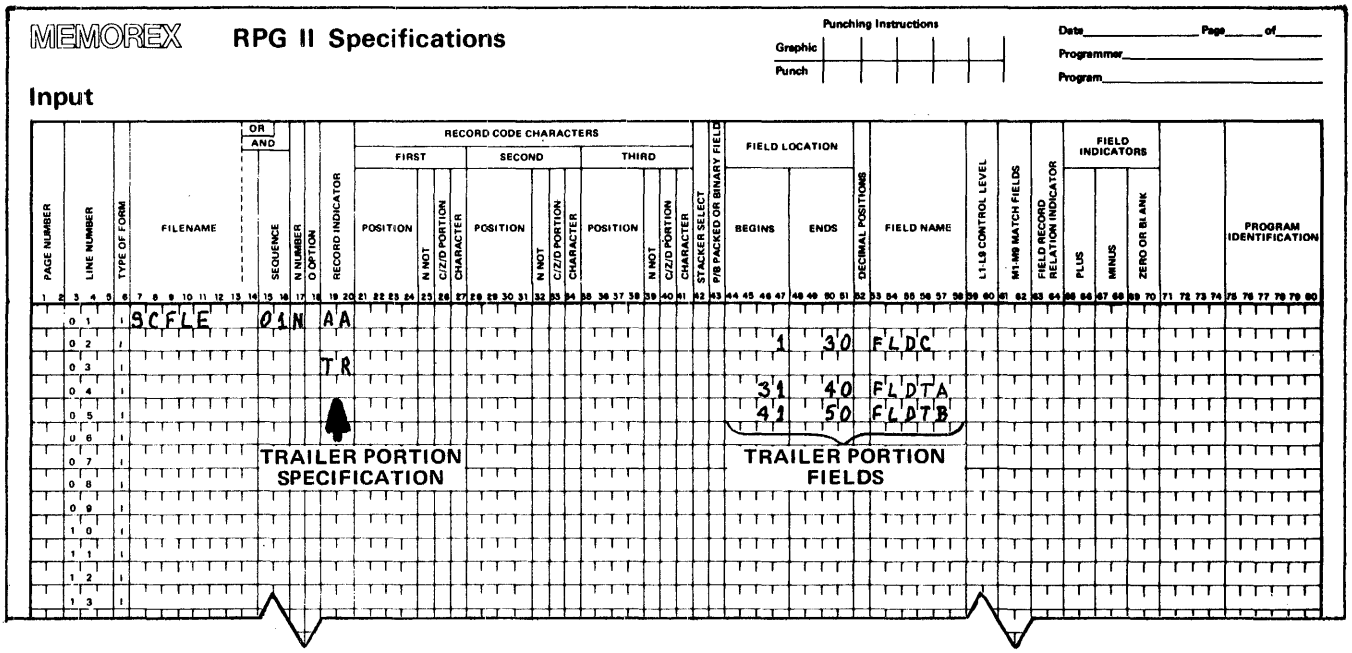


Figure 6-8. Coding for Spread Cards.

RECORD CODE CHARACTERS (Columns 21-41)

Record code characters (columns 21-41) are used to identify each type of record for processing various operations. If all the records are to be processed in the same way, regardless of their type, or if only one record type is to be used, columns 21-41 may be left blank.

When more than one record type is to be processed in a file, each record type is given a special code consisting of a combination of characters in certain positions in the record. Any number of record code characters may be used.

On the Input Specification Sheet, seven columns are required to identify a single character. The first six columns are used for information describing the character, and the seventh column is used for the character itself. Information identifying the first three characters of a record code is specified in columns 21-26,

28-33, and 35-40, respectively. The character code used to identify record types consists of four entries: position, not, portion, and character. The characters themselves are specified in columns 27, 34, and 41. Coding instructions for the three groups of character identification columns immediately follow this discussion.

If more than three record characters are used, the additional characters are described and identified on additional AND lines of the Input Specifications Sheet. AND lines may be used to describe as many characters as are needed after the three entered in the first specifications line. To specify an AND line, the word AND is written in columns 14-16.

When a record type may be indicated by two different codes, OR lines must be used to indicate that either one of the codes may be present to identify the record. To write an OR line, the word OR is placed in columns 14-15.

POSITION (Columns 21-24, 28-31, 35-38)

Entry	Description
Blank	No record identification code is needed.
1-9999	The record position of each character of the record code is indicated.

The position entry gives the record location of each character of the record code. The location of the first character is entered in columns 21-24, the second in columns 28-31, and the third in columns 35-38. Entries must be right-justified, ending in columns 24, 31, and 38, respectively. Leading zeros are not required.

NOT (Columns 25, 32, and 39)

Entry	Description
Blank	A character is to be present in the columns specified by the position entry.
N	A character should not be present in the columns specified by the position entry.

The not entry is used to indicate that a certain character of the record code should not be present in the position specified in the preceding position entry columns.

PORTION (Columns 26, 33, and 40)

Entry	Description
C	The entire character is used.
Z	Only the zone portion of the character is used.
D	Only the digit portion of the character is used.

The portion of a character that is used as part of the record code is specified in columns 26, 33, and 40. The characters are stored in EBCDIC form. Since several characters have either the same zone portion or the same digit portion, they must be chosen carefully when record codes are established, to avoid ambiguity (see "Character Structure" at the beginning of this section.)

CHARACTER (Columns 27, 34, and 41)

Entry	Description
Any character	Indicates a character used to identify the record.

The actual characters used in the record code are specified in columns 27, 34, and 41. Any alphabetic character, special character, or digit may be used.

Characters are stored in the machine in EBCDIC form, where they match characters, zeros, and digits according to the EBCDIC codes, except for the following cases:

The character & (ampersand) has the same zone as the character { (left brace), which is a 12-0 punch; and as A through I.

The character - (minus or hyphen) has the same zone as the character } (right brace), which is an 11-0 punch; and as J through R.

The blank carries no zone punch; neither do 0 through 9.

Although the left and right braces are not printable characters, they are still read and used by the computer.

If the zone portion of the character is being tested, the characters &, -, and blank compare in exceptional ways. If one of these characters or a code with the same zone is punched as the compare character, it will compare as equal. If, however, the compare character is a character with the same EBCDIC zone as the special characters, and a special character is punched in the card, it will also compare as equal.

Since negative numbers are formed by punching a minus sign over a digit, they have a different character structure than positive numbers. When the zone punch (minus sign) is combined with the digit punch (1-9), a different character (a negative number) is formed which is represented in the machine by the characters J-R. If the minus sign is combined with a zero, the right brace character (}) is formed.

Positive numbers, formed by punching an ampersand (&) over a digit, are represented in the computer by the characters left brace ({) and A through I.

STACKER SELECT (Column 42)

Entry	Description
Blank	The input cards will automatically fall into the stacker.

This column is reserved for future implementation of the stacker select feature.

FIELD DESCRIPTION ENTRIES

PACKED OR BINARY FIELD (Column 43)

Entry	Description
Blank	The input field is not numeric, or is in unpacked decimal format.
P	The input field is in packed decimal format.
B	The input field is in binary format.

The entry in column 43 indicates the format of the input field named in columns 53-58. The entry is used only if the field is numeric. If the field is alphabetic, alphanumeric, or in unpacked decimal format, no entry is made. Arrays in packed decimal or binary format require an entry in column 43.

If a numeric field is in binary or unpacked decimal format, it must be converted to packed decimal format before it can be processed. Only the digits and sign are recognized in the process of conversion to packed decimal format.

FIELD LOCATION (Columns 44-51)

The length of the input field named in columns 53-58 must be specified in columns 44-51. The maximum field length for a numeric field is 15 positions (8 if the format is packed decimal and 4 if it is binary). The maximum length for an alphanumeric field is 255 characters. The record position of the beginning of a field is specified in columns 44-47; the record position of the end is specified in columns 48-51. These are described below as separate entries.

BEGINS (Columns 44-47)

Entry	Description
1-4 digit number	Designates the beginning location of the field named in columns 53-58.

The record position of the first character of the input field named in columns 53-58 is specified in the begins entry. The entry must be right-justified to end in column 47. Leading zeros are optional.

ENDS (Columns 48-51)

Entry	Description
1-4 digit number	Designates the ending location of the input field named in columns 53-58.

The record position of the last character of the input field named in columns 53-58 is specified in the ends entry. The entry must be right-justified to end in column 51. Leading zeros may be omitted.

DECIMAL POSITIONS (Column 52)

Entry	Description
Blank	The field is alphanumeric.
0-9	The number of decimal positions in the numeric field is indicated.

If a numeric field is named in columns 53-58, the number of positions to the right of the decimal must be specified in column 52. The number of decimal positions cannot exceed the number of digits in the field. Fields that are used in arithmetic operations and fields that are edited or zero-suppressed (see Section 8, column 38 and columns 45-70) must be defined as numeric.

FIELD NAME (Columns 53-58)

Entry	Description
1-6 alpha-numerics	The field, array, or array item being specified is named.
PAGE PAGE1 PAGE2	Special word entries specify that page numbering is to be done.

Fields in a record must be assigned field names in columns 53-58. A separate line is used for each field description. The field name must be a valid RPG II name. It can be from one to six characters long and must begin in column 53. The first character must be alphabetic. The remaining characters can be any combination of alphabetic and numeric characters. Special characters are not permitted, and blanks cannot appear between characters in the name. The name assigned to each field must be used throughout the program whenever that field is referenced.

To eliminate duplicate coding of identical fields from different record types (Figure 6-9A), the OR relationship may be used to indicate that the fields named may be found in any of the record types (Figure 6-9B).

The OR relationship is used when two or more record types have the same fields in the same positions, or when two or more record types have some fields which are identical and some fields which differ in location, length, or type of data (numeric or alphanumeric).

In entering an OR relationship on the specification sheet, OR is written in columns 14-15 following the record identification line, and the record codes of the alternate record type are specified. The field names are then entered in columns 53-58 of the lines following.

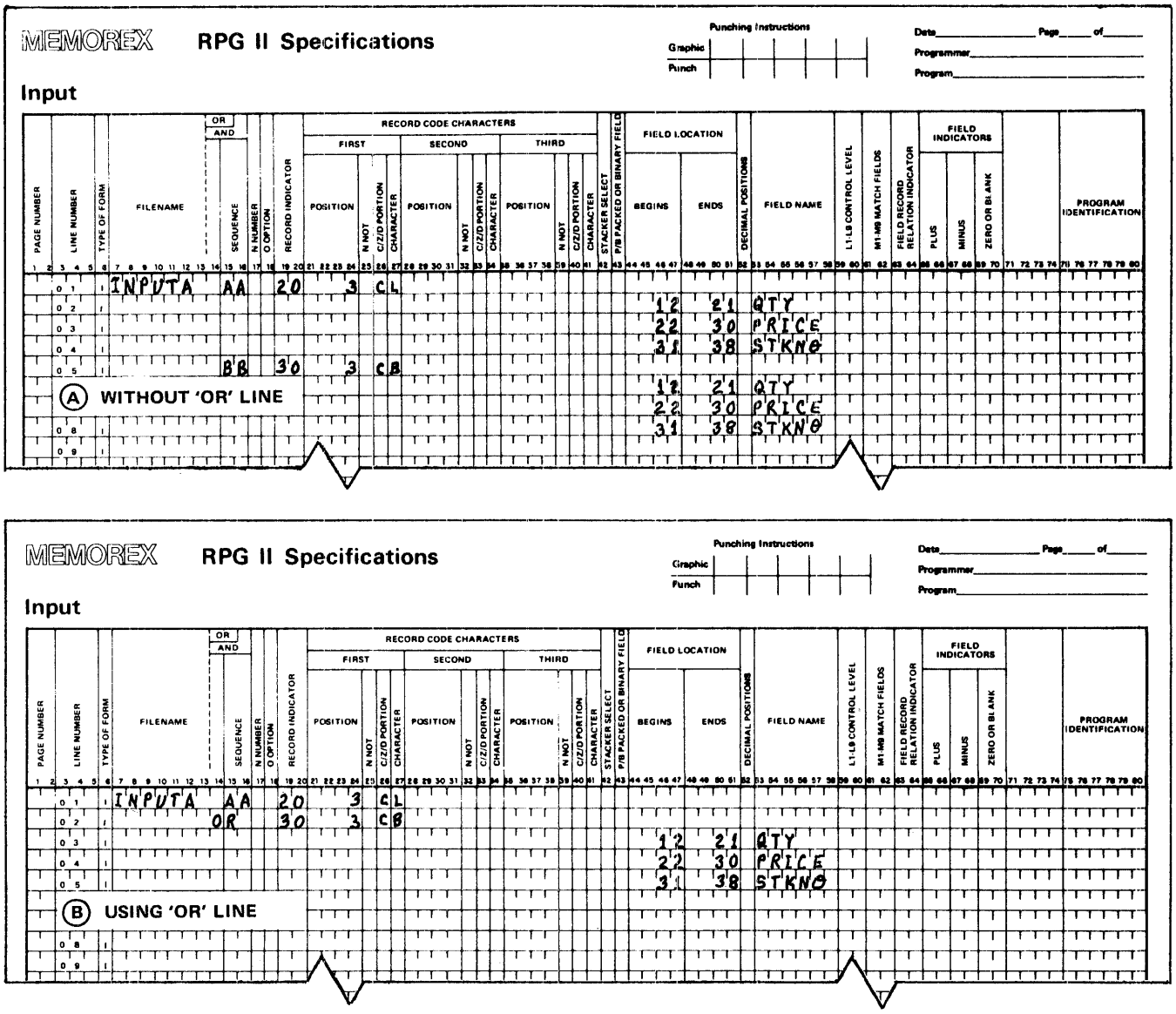


Figure 6-9. Coding for Record Types with Identical Fields.

If array items are referenced, additional entries may be required (see Section 4, "Tables and Arrays"). If an index is used with the array name, it must have been defined previously as a numeric literal. An index must be numeric with zero decimal positions.

The special words PAGE, PAGE1, and PAGE2 are field names used to indicate that page numbering on an output form or report is to start with a number other than page 1. The alternate page number is entered in a field of an input record and the field is named PAGE in columns 53-58. Three possible PAGE entries (PAGE, PAGE1, and PAGE2) are provided for naming the PAGE field of up to three different output files.

Page numbering can be restarted during a program by entering a number in a PAGE field of any input record. The number entered in the PAGE field must be one number less than the actual starting page number, and must be right-justified. For example, if page numbering were to start with 50, the entry in the PAGE field would be 0049. The PAGE field can be defined and used in calculations like any other field (see Section 7, columns 43-48).

CONTROL LEVEL (Columns 59-60)

Entry	Description
Blank	No control level indicator is to be used.
L1-L9	A control level indicator is to be used.

The entry in columns 59-60 assigns a control level indicator to an input field. Control level indicators are used to identify the point at which specified operations are to be performed. They may not be assigned to binary fields, or to input fields in chained or demand files.

Control level indicators with higher numbers should be assigned to the more important fields, so that those fields will be processed first (see "Control Level Indicators" at the beginning of this section).

MATCH FIELDS (Columns 61-62)

Entry	Description
Blank	Sequence checking is not to be performed.
M1-M9	Fields are to be matched and/or sequence checked.

Sequence checking is automatically done for all record types with match field specifications. The contents of the fields to which M1-M9 have been assigned are checked for correct sequence. All match fields must be in the same order, either all ascending or all descending, and all files with match fields must be in the same sequence.

If only one input or update file is used, an entry in columns 61-62 causes sequence checking of the data in the field to which the entry has been assigned (see columns 15-16 for record type sequence checking).

If an alternate collating sequence is defined for a program, alphanumeric fields are matched according to the sequence that was specified. Match fields contain a corresponding initial alternate collating sequence value; they are set to the lowest alternate sequence value if ascending sequence is specified, and to the highest alternate sequence value if descending sequence is specified.

As many as nine fields (M1-M9) may be used to sequence check. The sequence (ascending or descending) of the record file must be specified in the file description specifications (see Section 3, column 18).

If more than one input or update file is used, the M1-M9 entry also causes records from the files to be compared and determines when they match. 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 indicator (MR) turns on. M1-M9 are used only to identify fields by which records are matched, not as indicators; but they do cause MR to turn on when a match occurs. Matching is allowed with primary and secondary files only. It cannot be used with chained or demand files.

The same number of match fields must be specified for all record types used in matching. If more than one match field is specified for a record type, all the fields are combined and treated as one continuous match field. They are combined according to the ascending sequence of matching record values. Match fields may not be split; thus the same match field value cannot be used twice for one type of record.

The same matching record values (M1-M9) must be used for all types. Whenever more than one matching record value is used, all match fields must match before the MR indicator turns on. For example, if match fields M7, M8, and M9 are specified, a match on only the M7 and M8 fields will not turn on the MR indicator.

Since field names are ignored in matching record operations, fields from different record types which have been assigned the same match level may have the same name. Record columns of different match fields may overlap, but the total length of all fields must not exceed 255 characters.

Match fields may be either alphanumeric or numeric, however all match fields given the same matching record value (M1-M9) are considered numeric if any of those matching fields is described as numeric. Numeric match fields contain only the digits 0-9

(matching fields of 068 and ⍽68, where ⍽ denotes blank, would compare equal). When numeric fields having decimal positions are matched, they are treated as if they had no decimal position. Only the digit portions of numeric match fields are compared and signs are ignored. Even though a field is negative it is treated as a positive field, and the sign of the numeric field is disregarded (a -8 will match with a +8).

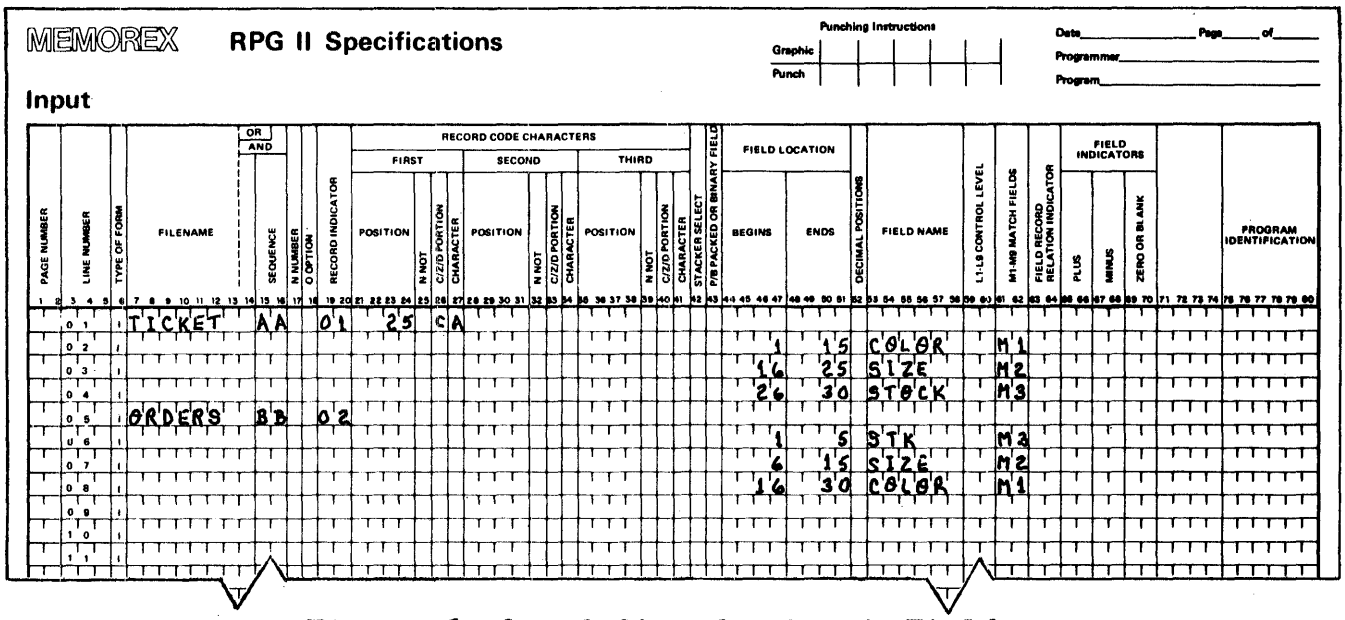


Figure 6-10. Coding for Match Fields.

FIELD RECORD RELATION INDICATOR (Columns 63-64)

Entry	Description
Blank	The field is associated with all the record types in a specified OR relationship.
01-99	The record identifying indicator relates a field to a record of a specific record type.
L1-L9	The control level indicator regulates the acceptance of data from a field if a control break occurs.
MR	The matching record indicator regulates the acceptance of data from match fields.

U1-U8	The external indicator conditions a field or a file.
H0-H9	The halt indicator relates a field to a record in an OR relationship.

The entry in columns 63-64 serves a variety of specific purposes which are governed by the type of indicator specified. Its primary purpose is to relate the field named in columns 53-58 to a particular record type. Although each type of indicator must be handled on an individual basis, some general coding rules apply to all of them.

All fields, including matching and control fields, that require no field record relation indicator entry should be entered before those that do.

All portions of a split control field must be assigned the same field record indicator. All fields with the same field record relation indicator, whether split or not, must be entered as a group in consecutive lines of the specifications sheet; thus memory storage is used to its fullest efficiency. The fields or field portions need not be entered in any particular order within the indicator groupings, however.

Five kinds of indicators may be entered in columns 63-64: record identifying indicators (01-99), control level indicators (L1-L9), the matching record indicator (MR), external indicators (U1-U8), and halt indicators (H0-H9).

Record Identifying Indicators

Record identifying indicators (01-99) are used in columns 63-64 when several record types have been specified in an OR relationship. If all the record types in a file have the same fields, no field record relation indicator is required; but if record types in the OR relationship have a mixture of fields, with some that are the same and some that are not, a record identifying indicator is required to relate a field to a certain record. The indicator entered in columns 63-64 must be the same as the record identifying indicator found in columns 19-20 of the record type on which the field is located.

The 01-99 record identifying indicators can also be used as field record relation indicators to relate control fields and match fields entered in columns 59-60 and 61-62 to a particular record type in an OR relationship. 01-99 may be assigned to just one match field when two match fields have the same matching level entry. In this case only the specification with the field record relation indicator is used when that indicator is on. If none of

the field record relation indicators is on for the control or match field, the specification that has no field record relation indicator is used. Control fields and match fields cannot have an L1-L9 or MR entry in columns 63-64.

Control Level Indicators

Control level indicators (L1-L9) may be used in columns 63-64 to cause the acceptance and use of data from a particular field only when a control break occurs. When the indicator turns on, data from the field named in columns 53-58 is accepted.

Matching Record Indicator

The matching record indicator (MR) may be used in the same way as control level indicators; data from a particular field is accepted and used only when match fields are encountered in columns 53-58.

External Indicators

External indicators (U1-U8) may be used to condition a specification. The external indicator is set prior to processing conditions to indicate whenever a field is to be used in the program. When the indicator is on, the field is read; when the indicator is off, the field is not read. Although they are most commonly used when file conditioning is to be done (see Section 3, columns 71-72), external indicators may be used to condition when a specification should or should not be done, if file conditioning is not specified.

Halt Indicators

Halt indicators (H0-H9) are used to relate a field to a record in relationship when a halt indicator is specified in columns 19-20.

FIELD INDICATORS (Columns 65-70)

Field indicators are assigned to the field named in columns 53-58 of the input record. Any indicator except LO can be used to test that field for a plus, minus, zero, or blank condition. If the condition is true, the indicator turns on; if the condition is not true, the indicator remains off. Halt indicators (H0-H9) may be assigned in these columns to test for an error condition in the data.

The field indicators are actually used in the calculation specifications (Section 7) and output-format specifications (Section 8); field indicators are specified in the input specifications only to condition operations. If indicators are being specified for this purpose, the programmer must know when they will be off and when they will be on.

One input field may be assigned two or three field indicators, but only the one which signals the result of the test turns on; the others are turned off.

The status of a single field indicator assigned to fields in different record types is always based on the last record type selected. When different field indicators are assigned to fields in different record types, however, a field indicator turned on will remain on until another record of that type is read. Similarly, a field indicator assigned to more than one field within a single record type will always reflect the status of the last field defined. Field indicators assigned in these columns may be SETON or SETOF in calculation specifications. The three field indicators are described as separate entries below.

PLUS (Columns 65-66)

Entry	Description
Blank	No indicator is being used for this field.
Any indicator except LO	An indicator is being used.

Indicators are used to test a field for a plus condition. Any valid indicator entered except LO turns on if a field named in columns 53-58 is numeric and greater than zero.

Indicators for the plus condition are off at the beginning of the program. They are not turned on until the plus condition is satisfied by the field being tested on the card just read. Halt

indicators specified in columns 65-66 cause the program to halt after the record which caused the indicator to turn on is completely processed. The data can then be checked for an error condition, such as zeros in the field when none should be there. In this case the zero field is found, the halt indicator turns on, the record with the zero field is processed, and processing stops so that the data can be corrected.

MINUS (Columns 67-68)

Entry	Description
Blank	No indicator is being used for this field.
Any indicator except LO	An indicator is being used.

Indicators are used in columns 67-68 to test for a minus (not true) condition. Any valid indicator entered except LO turns on if a field named in columns 53-58 is numeric and less than zero. Indicators for the minus condition are off at the beginning of the program and are not turned on until the minus condition is satisfied by the test field on the card just read.

Halt indicators (H0-H9) can be specified in columns 67-68 to cause the program to halt so that an error condition in the data can be checked.

ZERO OR BLANK (Columns 69-70)

Entry	Description
Blank	No indicator is being used for this field.
Any indicator except LO	An indicator is being used.

Indicators assigned in columns 69-70 cause testing for a zero or blank condition. Any of the indicators entered except LO turns on if a field named in columns 53-58 is all zeros and numeric, or if the field is alphanumeric and wholly composed of blanks.

An indicator assigned to test for zeros or blanks is off at the beginning of the program and remains off until the field being tested is zeros or blanks. A numeric field which is composed wholly of blanks will cause the zero indicator to turn on; however an alphanumeric field that is all zeros will not cause an indicator specified for all blanks to turn on.

Halt indicators (H0-H9) can be used to test for error conditions in the data.

PROGRAM IDENTIFICATION (Columns 75-80)

Entry	Description
Any characters	These columns may be used for programmer comments.

The program identification entry of all cards following the control card may contain any characters, including blanks. The columns may be used for programmer comments, if desired.

SECTION 7. CALCULATION SPECIFICATIONS

Specifications are written on the Calculation Specification Sheet (Figure 7-1) to describe both the calculations to be performed on the data, and the order in which they are to be performed.

Each calculation specification can usually be divided into three parts. The first part (columns 7-17) tells when the operation is to be performed, and under what conditions the operation specified is to be done; the second part (columns 28-32) indicates the kind of operation to be performed on the data specified in factor 1 (columns 18-27) and/or factor 2 (columns 33-42); and the third part (columns 54-59) indicates the tests to be made on the results of the operation. The indicators entered here signal the result of the operation and may be used to condition other operations.

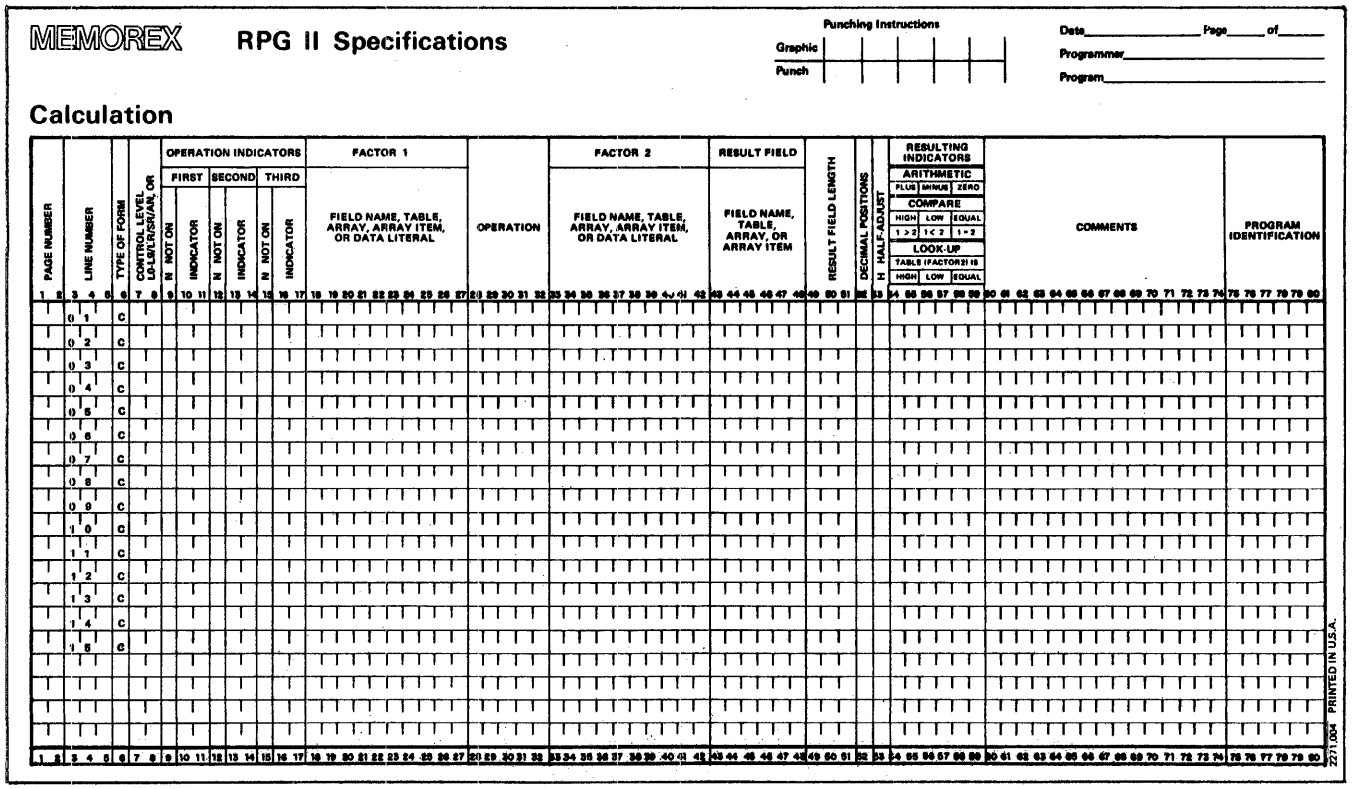


Figure 7-1. RPG II Calculation Specifications Sheet.

LITERALS

A literal is the actual data used in an operation, rather than a field name representing that data. Although a literal may be either alphanumeric or numeric, coding for an alphanumeric is different from the way in which a numeric literal is coded.

Alphanumeric literals may contain any combination of eight or fewer characters, excluding the two enclosing apostrophes. An apostrophe which is included as part of the body of a literal is represented by two apostrophes. The literal 'HARRY'S' would be written as 'HARRY''S', for example. Alphanumeric literals may not be used for arithmetic operations.

Numeric literals (Figure 7-2) are used in the same way as numeric fields. They may contain any combination of the digits 0-9. A decimal point or sign may also be included, but blanks cannot appear in numeric literals, and apostrophes must not be used. The maximum total length of a numeric literal is ten characters, including an optional sign and decimal point. If a sign is present, it must be the leftmost character. Unsigned numeric literals are treated as positive numbers.

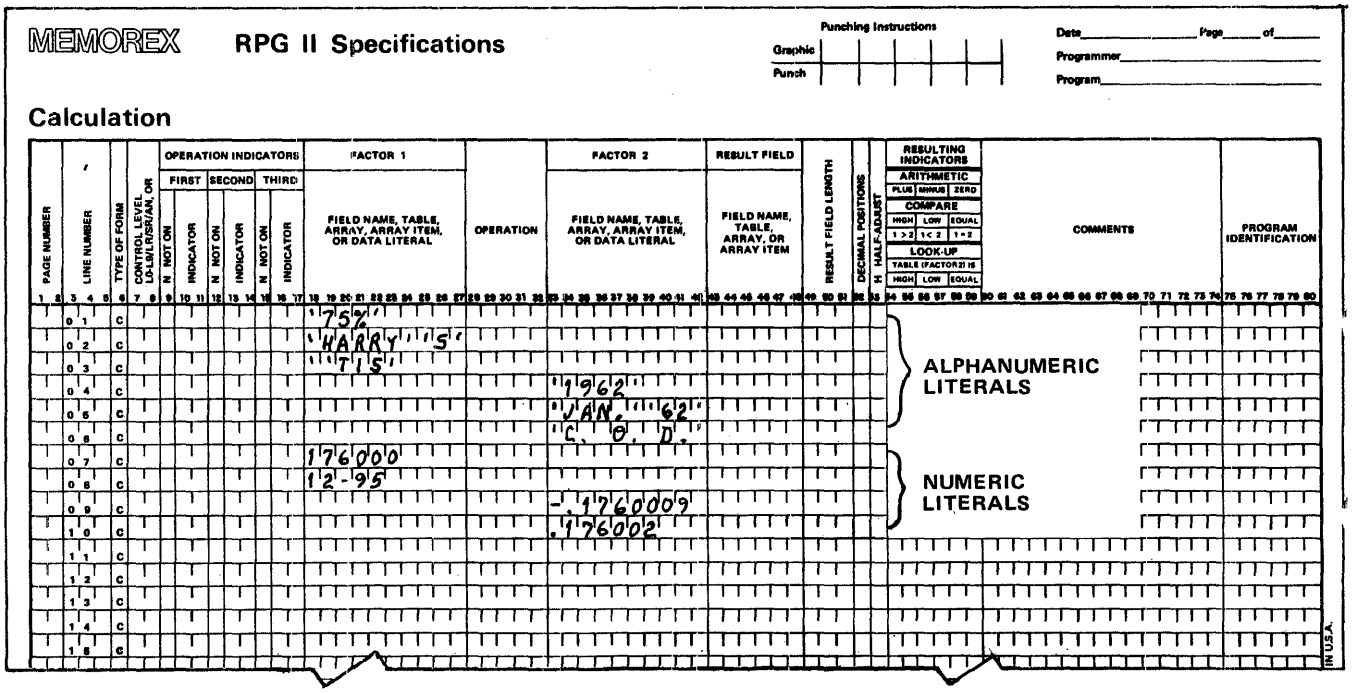


Figure 7-2. Examples of Alphanumeric and Numeric Literals.

OPERATION CODES

Use of RPG II allows many different types of operations to be performed on data. Special codes which indicate the operation to be carried out must be used to specify the required operation. The codes are easy to work with, since they are usually an abbreviated form of the operation to be performed.

Calculation operations are either internal or external. Internal calculation operations are performed in the computer within the processing of a single RPG II program. Internal calculation operations fall into the following categories:

Arithmetic Operations
Move Operations
Move Zone Operations
Compare and Testing Operations
Binary Field Operations
Indicator Setting Operations
Branching Operations
Lookup Operations
Subroutine Operations
Input and Output Programmed Control Operations

External calculation operations comprise those operations which are performed by a subroutine or program that has been compiled separately from the main RPG II program. They comprise the external linkage operations and the debug operation. Valid operation codes are summarized in Appendix B, Table B4.

ARITHMETIC OPERATIONS

Arithmetic operations can be performed only on numeric fields, arrays, array items, or literals. The result field must also be numeric. When arithmetic operations are used in which all three fields occur, the factor 1 and factor 2 fields (columns 18-27 and 33-42) and the result field (columns 43-48) may all be different fields; they may all be the same field; factor 1 and factor 2 may be the same field but different from the result field; or either factor 1 or factor 2 may be the same as the result field.

The length of the result field must be kept in mind when arithmetic operations are performed. When the result is aligned, decimal positions in excess of the number of decimal positions in the result field are dropped from the right side; integer digits in excess of the integer portion of the result field's size are dropped from the left side. The results of all operations are

signed plus or minus (+ or -). Any data placed in the result field completely replaces the previous contents of the field.

Arithmetic operations are performed through the use of codes for nine functions: add, zero and add, subtract, zero and subtract, multiply, divide, move remainder, square root, and crossfoot.

ADD (ADD)

In the add function, factor 2 is added to factor 1 and the sum is placed in the result field. Factor 1 and factor 2 are not changed by the operation.

ZERO AND ADD (Z-ADD)

In the zero and add function, the result field is first set to zeros, then factor 2 is added to the result field. The result field then contains factor 2 in the right positions, and zeros in the left positions not occupied by factor 2. Factor 1 is not used in the Z-ADD operation.

SUBTRACT (SUB)

The subtract function causes factor 2 to be subtracted from factor 1 and the difference to be placed in the result field. Factor 1 and factor 2 are not changed by the operation. Subtracting two fields which are the same is a method of setting the result field to zero.

ZERO AND SUBTRACT (Z-SUB)

In the zero and subtract function, the result field is first set to zeros, then factor 2 is subtracted from the result field and placed in the result field, so that the complement of f ctor 2 ends up in the result field. Factor 1 is not used. This operation can be used to change the sign of a field.

MULTIPLY (MULT)

The multiply function causes factor 1 to be multiplied by factor 2, and the product to be placed in the result field. Factor 1 and factor 2 are not changed. When a field described as the result field is used as a factor, the length of the result field must be checked to be sure it is sufficient to hold the product.

DIVIDE (DIV)

In the divide function, factor 1 (dividend) is divided by factor 2 (divisor) and the result (quotient) is placed in the result field. Factor 1 and factor 2 are not changed. If factor 1 is zero, the result of the divide operation will be zero, however factor 2 cannot be zero. Any remainder resulting from the divide operation is lost unless the move remainder operation (MVR) is specified as the next operation; if it is, the result of the divide operation cannot be half-adjusted (rounded).

MOVE REMAINDER (MVR)

The move remainder function moves the remainder from the previous divide operation to a separate field named under the result field entry. The maximum length of the remainder is 15, including decimal positions. The number of significant decimal positions is the greater of either the number of decimal positions in factor 1 of the previous divide operation, or 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 are equal to the whole-number positions in factor 2 of the previous divide operation. This operation must immediately follow the divide operation and should be conditioned by the same indicators. Factor 1 and factor 2 must not be used with MVR.

SQUARE ROOT (SQRT)

The square root function derives the square root of the field named in factor 2 and places it in the result field. Factor 1 is not used. Factor 2 and the result field are numeric fields from 1 to 15 digits long, including up to nine decimal places. For every digit to the left of the decimal place in the result field there

should be two digits to the left of the decimal place in factor 2. For every digit to the right of the decimal place in the result field there should be two digits to the right of the decimal place in factor 2. A negative number causes a halt, thus factor 2 cannot be a negative number. When the SQRT operation is used, the result field length must be greater than or equal to the decimal positions entry. The result field (root) is automatically half-adjusted. A whole array can be used in a SQRT operation if factor 2 and the result field entry contain array names. In this case, the square root of each item of the array named in factor 2 is placed in the corresponding item of the array named in the result field.

CROSSFOOT (XFOOT)

The crossfoot function is used only on arrays with numeric items. It adds all the items of the array together and places the sum in the result field. The total can be half-adjusted in the result field. Factor 1 is not used; factor 2 contains the name of the array. Resulting indicators can be used. If the result field is an item of the same array used in factor 2, the value of that item before the XFOOT operation is used in arriving at a total.

MOVE OPERATIONS

Move operations move factor 2, or part of it, to the result field, but do not change factor 2 itself. Factor 1 is not used in move operations. Resulting indicators may not be assigned. Move operations can be used to change numeric fields to alphanumeric fields and alphanumeric fields to numeric fields. To change a numeric field to an alphanumeric field, the name of the numeric field is placed in factor 2 and an alphanumeric result field is used. To change an alphanumeric field to a numeric field, the process is reversed: the name of the alphanumeric field is placed in factor 2 and a numeric result field is used.

When data is moved into numeric fields by means of move operations, any decimal positions in the data are ignored. For example, if 1.72 is moved into a numeric field with one decimal position, the result is 17.2.

Move operations can be used to perform two functions: move, which moves and right-adjusts the data; and move left, which moves and left-adjusts the data.

MOVE (MOVE)

The MOVE operation code causes characters from factor 2 to be moved to the rightmost positions in the result field. Moving starts with the rightmost character. 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 characters in the result field that lie to the left of the data just moved in remain unchanged.

Alphanumeric fields or constants may be changed into numeric fields by moving them into numeric fields. When this type of move occurs, 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, and the zone portion of the rightmost alphanumeric character is converted to a corresponding sign. The sign is then moved to the rightmost position of the numeric field, where it becomes the sign of that field. Conversely, numeric fields may be changed into alphanumeric fields by moving them into alphanumeric fields. When this type of move is specified, all digits are transferred, including the digit and zone of the rightmost character.

MOVE LEFT (MOVEL)

When the move left operation is used, the characters from factor 2 are moved to the leftmost positons in the result field, beginning with the leftmost character of factor 2.

When MOVEL is specified to change an alphanumeric field or constant into a numeric field, 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. The zone of the rightmost character of factor 2 is used as the sign of the result field, whether or not the rightmost character is moved.

When a numeric field is moved into an alphanumeric field, it is changed into an alphanumeric field. Digits are transferred left to right. The digit and zone portions of the rightmost character are converted to one byte each, if that character is to be moved.

If factor 2 is longer than the result field, the excess rightmost characters of factor 2 are not moved. Certain other conditions apply if factor 2 is longer than the result field:

If both fields are numeric, the sign from the rightmost position of factor 2 is moved over the rightmost digit of the result field.

If both fields are alphanumeric, only the number of characters needed to fill the result field is moved.

If factor 2 is numeric and the result field is alphanumeric, the result field contains only digits after factor 2 is moved.

If factor 2 is alphanumeric and the result field is numeric, the zone from the rightmost character of factor 2 is moved to the sign position of the result field, and the remaining result field positions contain only digits.

If the result field is longer than factor 2, the characters to the right of the data just moved in remain unchanged, and if a numeric field is used, its sign also stays the same. Two other conditions apply for the MOVEL operation if factor 2 is shorter than the result field:

If the result field is numeric and factor 2 is either numeric or alphanumeric, the digit portion of factor 2 replaces the contents of the leftmost positions in the result field, and the sign in the rightmost position of the result field is not changed.

If the result field is alphanumeric and factor 2 is either numeric or alphanumeric, the characters in factor 2 replace the equivalent number of leftmost positions in the result field, and no change is made in the zone of the rightmost position of the result field.

When factor 2 and the result field are the same length, the following conditions apply with MOVEL:

If factor 2 and the result field are both numeric, the sign is moved with the rightmost digit of factor 2.

If factor 2 and the result field are both alphanumeric, all the factor 2 characters are moved.

If factor 2 is numeric and the result field is alphanumeric, the factor 2 sign is moved with the rightmost digit. Only the digits in the other positions are moved.

If factor 2 is alphanumeric and the result field is numeric, the zone and digit portions of the rightmost digit of factor 2 are moved. Zones in other positions are not moved.

MOVE ZONE OPERATIONS

Move zone operations are used to move only the zone portion of a character. Four varieties of the move zone operation can be used: move high to high zone, move high to low zone, move low to low zone, and move low to high zone.

In the following discussion the word high refers only to an alphanumeric field, and the word low refers to either an alphanumeric or a numeric field.

MOVE HIGH TO HIGH ZONE (MHHZO)

The move high to high zone operation moves the zone from the leftmost position of factor 2 to the leftmost position of the result field. Both factor 2 and the result field must be alphanumeric.

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 alphanumeric, but the result field may be either alphanumeric or numeric.

MOVE LOW TO LOW ZONE (MLLZO)

MLLZO moves the zone from the rightmost position of factor 2 to the rightmost position of the result field. Factor 2 and the result field may both be either alphanumeric 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 either numeric or alphanumeric, but the result field must be alphanumeric.

COMPARE AND TESTING OPERATIONS

Compare and testing operations are used to test fields for certain conditions. No fields are changed by these operations; instead the result of the test on the fields is shown by the resulting indicators assigned in columns 54-59.

COMPARE (COMP)

The compare operation causes factor 1 to be compared with f0 tor 2. The results of the comparison are shown by assigned indicators that are turned on or off. The nigh indicator is turned on if factor 1 is greater than factor 2; the low indicator if factor 1 is less than factor 2; the equal indicator if factor 1 is equal to factor 2.

The compare operation cannot be used with entire arrays, and factors 1 and 2 cannot be compared unless both are either alphanumeric or numeric. If the fields are alphanumeric, they are automatically aligned to their leftmost character before they are compared. If one factor is shorter, its unused positions are automatically filled with blanks.

If the fields to be compared are numeric, they are automatically aligned according to the decimal point, and any missing digits are filled in with zeros. The maximum length for numeric fields to be compared is 15 digits.

If an alternate collating sequence is defined in the specifications, alphanumeric fields are compared according to that sequence.

TEST ZONE (TESTZ)

This operation is used on alphanumeric result fields to test the zone of the leftmost character (see Section 6, "Character Structure"). The test determines whether the zone is positive (+), negative (-), or blank. Resulting indicators are assigned to show the results of the test. The plus indicator is turned on by the zone portion of the characters & and A-I, while the minus indicator is turned on by the zone portion of the characters } (right brace), - (minus), and J-R. All other characters turn on the blank indicator.

The TESTZ operation cannot be used on numeric result fields, nor does the operation use factor 1 and factor 2.

BINARY FIELD OPERATIONS

Three binary field operation codes are provided to set and test individual bits used as switches in a program. The codes are BITON, BITOF, and TESTB; they are entered in columns 28-32. When one of these codes is used, factor 2 can contain either a field name, or one to eight bit specifications.

One or more bits per operation, up to a maximum of eight, may be set on, set off, or tested. Bits may be specified in a literal by numbering them 0-7, from left to right, and enclosing them in apostrophes. They need not be specified in any order. To specify bits 1, 4, 6, and 7, for example, '1467' would be entered in columns 33-38 of factor 2. Bits which are not specified in factor 2 are not changed.

Bits may also be specified using the name of a one-position alphanumeric field name or table or array item. In this case, the bits which are on in the field or array item are turned on, turned off, or tested in the result field, and bits which are not on are not affected.

SET BIT ON (BITON)

The use of the BITON operation code in columns 28-32 causes the bits identified in factor 2 to turn on in the field named in the result field entry. Conditioning indicators can be used in columns 7-17. Any entry under field length must be 1.

The factor 1, decimal positions, half-adjust, and resulting indicators entries are not used with the BITON operation.

SET BIT OFF (BITOF)

The use of the BITOF operation code in columns 28-31 causes bits identified in factor 2 to turn off in a field named as the result field. All other specifications are the same as those for the BITON operation.

TEST BIT (TESTB)

Use of the TESTB operation code causes bits identified in factor 2 to be tested for an on or off condition in the field named as the result field, and the condition of the bits to be shown by resulting indicators in columns 54-59. All other specifications are the same as those for BITON and BITOF.

As many as three resulting indicators can be named for one TESTB operation, but at least one must be used. Although two indicators may be the same for one TESTB operation, three cannot. If factor 2 contains bits which are all off, no resulting indicators are turned on.

Resulting indicators have different meanings for different columns.

An indicator in columns 54-55 is turned on if each bit specified in factor 2 is off in the result field.

An indicator in columns 56-57 is turned on if two or more bits were tested and some bits were found to be on and others off. The field named in factor 2 must contain more than one bit which is on if an indicator appears in columns 56-57.

An indicator in columns 58-59 is turned on if each bit specified in factor 2 is on in the result field.

INDICATOR SETTING OPERATIONS

Indicator setting operation codes are used to turn indicators in columns 54-59 on or off. The headings shown in the resulting indicators field (plus, minus, zero, etc.) do not apply in indicator setting operations. The two indicator setting operations which can be used are SETON and SETOF. 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.

Indicators L1-L9 and the record identifying indicators are always turned off after detail output operations are completed, regardless of the previous SETON or SETOF operation. Setting on or setting off a control level indicator (L1-L9) does not automatically set on the lower control level indicators.

SET ON (SETON)

The SETON operation code causes indicators entered in columns 54-59 to be turned on. Any indicator may be turned on by the SETON operation.

If the LR indicator is turned on by a SETON operation conditioned with a control level indicator (columns 7-8), processing stops after all total output operations are finished. If it is turned on by a SETON operation not so conditioned, processing stops after the next total output operation is completed.

If halt indicators (H0-H9) are set on and not turned off before the detail output operations are complete, the system types a message on the console indicating which halt indicator is on. The operator may then either continue the job or terminate it.

SET OFF (SETOF)

This operation causes indicators in columns 54-59 to be turned off. Any indicator may be turned off by the SETOF operation.

NUMERIC INDICATORS

Numeric indicators (01-99) may be used in a variety of ways to fulfill a number of purposes. During the input cycle they may be used as field record relation indicators, as record identifying indicators, or as field indicators. Specifying them as field record relation indicators causes them to be tested; specifying them as record identifying indicators causes them to be turned on or off. It is important to remember that if an input field is specified again as an output field with a blank after entry (column 39 of the output sheet), a zero or blank field indicator specified on the input field will turn on as soon as that field is blanked, and will affect calculation and output specifications with which it is used.

During the calculations cycle, numeric indicators 01-99 may be used either as operation indicators or as resulting indicators. Specified as operation indicators, they determine whether or not an operation is to be performed; as resulting indicators they are turned on or off.

During the output cycle, 01-99 may be specified as output indicators to condition output fields.

HALT INDICATORS

Halt indicators (H0-H9) are turned on and off in the same manner as numeric indicators, with two exceptions:

Halt indicators are tested at the end of each detail cycle, and if any are on a message is typed to the operator (see Appendix G, "Error Messages"). If the operator indicates that the job is to be continued, the indicator is turned off.

If halt indicators are on at the end of the detail cycle, the job is halted and processing stops.

OVERFLOW INDICATORS

One overflow indicator (OA-OG, OV) is assigned to each printed output file in the file description specifications. The overflow indicator may be turned on or off when it is specified as a field indicator on input or as a result indicator for calculations; or it may be tested when it is specified as an operation indicator

for calculations or as a conditioning indicator for output. Although overflow indicators cannot condition exception lines, they may condition fields within the exception record.

An overflow indicator is turned on when spacing goes past an overflow line, but not when the overflow line is skipped. A skip to a new page from a line not conditioned by an overflow indicator causes the indicator to be turned off.

CONTROL LEVEL INDICATORS

Control level indicators (L1-L9) are used to indicate a change in a control field. RPG turns them on when a change is encountered in the value of the control field of a subsequent record. They are turned off at the end of the detail cycle following the control field change. Control level indicators may also be turned on or off by calculations.

When RPG turns on a control level indicator, all lower level control indicators are turned on as well; however a control level indicator turned on or off as a record, field, or resulting indicator does not turn the lower level control indicators on or off.

EXTERNAL INDICATORS

External indicators (U1-U8) are used on file description specifications to condition files. They may also be used as field record relation indicators (input specifications), operation indicators (calculation specifications), or conditioning indicators (output specifications).

External indicators provide a means of passing yes or no information from Control Language Statements (Appendix F) to a program and from one program to the next. They may be turned on or off with the CLS parameter card (//PAR, Appendix F), or by being specified as field indicators. The status of external indicators remains unchanged from the end of one job step to the beginning of the next job step.

FIRST PAGE INDICATOR

The first page indicator (1P) is used to produce heading and detail lines on the first page of output. It is turned on by RPG at the beginning of the first program cycle, before any input records are read, and turned off by RPG just before the first detail record is read. It is also turned on when specified as a field indicator in the input specifications.

The 1P indicator may be used as an operation indicator, but will remain off the first time through the program cycle unless it is intentionally set on.

BRANCHING OPERATIONS

Branching operations allow operations to be performed in a different order than is specified on the Calculation Specifications Sheet. They can be used to cause several operations to be performed over and over again; to cause certain operations to be performed for certain record types and not for others; or to cause several operations to be skipped when certain conditions occur.

Two operation codes can be used: GOTO, and TAG.

GO TO (GOTO)

The GOTO operation code allows instructions to be skipped if some other instruction is specified to which the program is to go (see TAG). The skip may be to an earlier line or to a later specification line, but it cannot be from a calculation within a subroutine to a calculation outside of that subroutine, or vice versa.

A skip can be made from a calculation that is conditioned by a control level indicator (columns 7-8) to one that is not, however the programmer should be familiar with the RPG program logic before he uses this feature.

If there is no entry in columns 7-8 of a GOTO line, but the associated TAG line contains a control level indicator (L0-L9), the program skips from the point of the GOTO line in detail calculations to the point following the TAG line in total calculations, and no detail output occurs. Data from the record

being processed remains available; data from the next record is not transferred to the input work area. The same input fields are repeatedly transferred to the RPG-assigned locations before detail calculations are performed.

Total calculations following the TAG line are once again sequentially executed, subject to the status of conditioning indicators in columns 7-17; total output again occurs, unless another GOTO instruction causes it to be bypassed; overflow output is processed if OA, OF, or OV is on; and detail calculations are again performed.

Several events take place during detail calculations preceding the GOTO operation. If field indicators or the MR indicator were turned on or off, they are reset after each total output to the setting they had immediately before original detail calculations for the current record. If OA-OF, or OV was turned on or off, it is reset before overflow output to its status at the end of the preceding total output. It is turned on, however, if it was off at the end of the preceding output and a carriage-tape channel 12 punch is encountered during one of the repeated total output times. Once it has been turned on by channel 12, it is not turned off again until the next detail output has been completed. L0-L9 or record type indicators turned on or off by the SETON or SETOF operation or as resulting indicators retain their status.

Factor 2 must contain the name of the point to which the operation is to go. Factor 1 and the result field are not used in the GOTO operation. GOTO may be conditioned by any indicators. If it is not conditioned, the operation is always executed.

TAG (TAG)

The TAG operation code defines the point to which a skip is specified in the GOTO operation. The name must begin in column 18 of factor 1. The same name may not be used for more than one TAG instruction, and it cannot be the name of a field used in the program.

Factor 2 and the result field are not used in the TAG operation, and no indicators may be entered in columns 9-17. Control level indicators may be used, however, if a skip is to be made at total time.

LOOKUP OPERATIONS

Lookup operations are used when a table or array is searched for a special item. The lookup operation can be used to search a single table, to search an array, or to search using two related tables. Only one code is required for lookup operations: LOKUP.

LOOKUP (LOKUP)

The LOKUP operation code causes a search to be made for a particular item in a table or array named in factor 2. Factor 1 contains the search word for which a match is to be found in the table or array named. The search word may be an alphanumeric or numeric constant, an array item, a field name, or a table name. The search word must be the same length and contain the same format (alphanumeric or numeric) as the table or array item, but need not have the same alignment.

When a table is named in factor 1 for a LOKUP operation, the specification is not for the whole table, but rather for the item last selected from the table.

ASSIGNING INDICATORS

Resulting indicators are always used in connection with LOKUP, first to indicate the type of search desired, and then to reflect the result of the search. In all cases the resulting indicator is turned on only if the search is successful.

An indicator assigned to LOW (columns 56-57) instructs the program to locate an entry in the table that is nearest to the search word, yet lower in sequence than the word. The first higher entry found causes the indicator assigned to HIGH to turn on.

A resulting indicator assigned to EQUAL (columns 58-59) instructs the program to search for a table or array item equal to the search word. The indicator turns on only if such an item is found. If several items are identical to the search word, the program selects the first one it encounters.

At least one resulting indicator must be assigned for a LOKUP operation, but no more than two can be used. Resulting indicators should not be assigned to both HIGH and LOW, since the indicator

assigned to LOW will be ignored, but indicators can be assigned to EQUAL and HIGH, or to EQUAL and LOW. The program then searches for an entry that satisfies either condition, but checks for an EQUAL item first. If no EQUAL entry can be found, the nearest item lower or higher in sequence is selected.

A search may be made on HIGH, on LOW, on HIGH and EQUAL, or on LOW and EQUAL only if the table or array is in sequence. No resulting indicator is turned on if the entry searched for is not found. (See Appendix B, Tables B2 and B3, for indicator summaries.)

SEARCHING A SINGLE TABLE

When a single table is searched, factor 1, factor 2, and at least one resulting indicator must be specified. Conditioning indicators may also be specified in columns 7-17. Resulting indicators are always set on to reflect a successful result of the search, and off if the search was unsuccessful.

Every time 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 and the previous contents of the storage area are destroyed. If the search is not successful, the storage area remains untouched.

SEARCHING WITH TWO TABLES INVOLVED

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 made available for use. The two tables involved should be the same length. If the table that is searched is longer than its related table, the search stops at the end of the shorter table.

Factor 1 must be named as 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 to be made available for use. Resulting indicators must also be used. Conditioning indicators may be specified in columns 7-17 if they are needed.

REFERENCING FOUND TABLE ITEMS

Whenever a table name is used in an operation other than LOKUP, the table name does not refer to the table itself, but to the data item placed in the special storage area as a result of the last successful search. This allows data items from a table in calculation operations to be used.

The table may also be used as the result field in operations other than the LOKUP operation, so that the contents of the table can be modified by calculation operations. In this case the contents of the special storage area are changed by the calculation operation, and the corresponding table item in the table itself is also changed.

If a table is named in factor 1 in a LOKUP operation, the contents of the special storage area are used as the search word. In this way a data item from a table can itself become a search word.

SEARCHING AN ARRAY

LOKUP operations are specified for arrays in the same way as for tables except that the result field cannot be used if factor 2 names an array. If just the array name is used in referencing the array, the search begins at the first item in the array. Indicators must be used to determine if a match was found. The indicators reflect only that the desired item is in the array; the programmer does not have ready access to a found item.

The array name may be used with an index consisting of a field name or a literal. If an index is used, the search begins at the item identified by the index. If a match is found, the number of the array item containing the match is placed in the field used as the index. If no match is found, the index field is set to 1.

If a literal is used as an index, indicators must be designated to determine if a match was found. The content of the item referenced by the literal is not changed.

SEARCHING AT A SPECIFIED ARRAY ITEM LOCATION

To save processing time it is possible to start the LOKUP search at a particular item in an array. The search starts at the specified item and continues until the desired item is found or

until the end of the array is reached. This type of search is indicated by additional entries in columns 33-42. The name of the array to be searched is entered in these columns, followed by a comma and either a numeric literal or the name of a numeric field with no decimal positions. The numeric literal or field shows the number of the item at which the search is to start; it is termed an index because it points to a certain item in the array. All other columns are used in the same way as they are for the normal lookup operation.

When an index field is used, an unsuccessful search causes that field to contain the value of one. If an array item is found which satisfies the conditions of the LOKUP operation, however, the number of the item, counting from the first item in the array, is placed in the index field. A numeric literal used as an index is not changed to reflect the result of the search.

SUBROUTINE OPERATIONS

The subroutine operation codes are used only for subroutines that are parts of other main routines. The subroutines are coded in specification lines on the calculation sheet following all detail and total calculation operations. Even though they are specified last on the calculation sheet, they may be performed at any point in the calculation operations. Each subroutine must have a name entered in factor 1, and the operation code BEGSR entered on the same line. The subroutine name can be 1-6 characters long and must begin with an alphabetic character in column 18. The remaining characters can be any combination of alphabetic or numeric characters, but may not contain special characters. Blanks may not appear between characters in the name, and no two subroutines used in the same program may have the same name.

Except for AN and OR lines, each specification line within the subroutine must have SR in columns 7-8 to identify it as a subroutine line.

All possible RPG II operations may be performed within a subroutine. The operations may be conditioned by any valid indicator in columns 9-17. Individual operations within a subroutine cannot be conditioned by a control level indicator in columns 7-8, however, since SR must appear in those columns.

Fields used in the subroutine may be defined either inside or outside the subroutine. In either case the fields can be used by both the main routine and the subroutine.

A subroutine cannot be written within a subroutine. Although it may call another subroutine, a subroutine cannot call itself or

the subroutine which called it. When GOTO is used in a subroutine, branching outside to a statement in another subroutine or branching from outside the subroutine to a statement within the subroutine causes an error condition. A skip can be made only from one statement within a subroutine to another statement within the same subroutine.

Three operation codes are used in defining a subroutine: BEGSR, ENDSR, and EXSR.

BEGIN SUBROUTINE (BEGSR)

This operation code is coded as the first statement of a subroutine and serves as its beginning point. Factor 1 must contain the name of the subroutine.

END SUBROUTINE (ENDSR)

ENDSR serves as the last statement of the subroutine to define the end of the subroutine. Factor 1 may contain a name which then indicates a point to which a skip can be made by a GOTO statement within the subroutine. The ENDSR operation ending the subroutine automatically causes a branch back to the next statement after the EXSR operation.

EXECUTE SUBROUTINE (EXSR)

This operation causes a skip to be made to the beginning of a subroutine. Factor 2 must contain the name of the subroutine that is to be executed, and the same name must appear on a BEGSR instruction. EXSR may appear anywhere in the program to indicate that the subroutine is being called. Whenever EXSR appears, the subroutine is automatically executed. EXSR may be conditioned by indicators to cause the subroutine to be executed only at the time when all specified conditions are satisfied. Any valid indicators may be used in columns 7-17. If no indicators are used, the subroutine is always executed when EXSR is encountered. Entire subroutines can be conditioned by control level indicators when the control level indicator is used with the EXSR operation. After all operations in the subroutine are done, the operation in the line following the EXSR specification is performed.

INPUT AND OUTPUT PROGRAMMED CONTROL OPERATIONS

In the normal RPG II processing cycle, a record is read, calculations are performed, and records are written. This cycle can be altered, however, to allow input and output operations to be executed during calculations. Five operations can be used to alter the cycle: exception, force, display, read, and chain.

EXCEPTION (EXCPT)

This operation allows records to be written at the time calculations are being done. It is used primarily when the number of similar or identical records (either detail or total) to be written in one program cycle is variable rather than exact.

When the exception operation is used, EXCPT is entered in columns 28-32. Columns 7-17 may have entries, but all other columns must be blank. The line or lines to be written out during calculation time are indicated by an E in column 15 of the Output-Format Specifications Sheet.

FORCE (FORCE)

FORCE statements are used with primary or secondary input or update files to enable selection of the file from which the next record is to be taken for processing. 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 selected at the start of the next program cycle. If two or more FORCE statements are executed during the same program cycle, all but the last one are ignored.

Although the first record to be processed is always selected by the normal method, FORCE statements override multifile processing selection for the remaining records. When end-of-file is encountered on a forced file, a record is not retrieved from the file; instead the method of selection reverts to multiprocessing and the record to be processed is selected by normal methods.

DISPLAY (DSPLY)

The display operation allows either or both of two events to occur:

A field, table element, array element, or literal up to 120 characters long can be printed on the console without a program halt during program execution.

A field, table element, or array element up to 120 characters long can be printed on the console with a program halt to allow that field to be changed. Literals may not be changed with the DSPLY operation code.

When the DISPLAY operation is used, factor 1 holds the field to be displayed; factor 2 gives the filename of the console; and the result field contains the field to be changed. Both factor 1 and the result field are optional; however, either one or the other must be used. If desired, both may be used.

When data is entered during program execution, alphanumeric fields are left-justified after all characters are keyed, and numeric fields are right-justified after all characters are keyed. Leading zeros are not required for numeric data. After the data is entered the END key must be depressed if the data is correct, or the CANCEL key if data is to be re-entered. If no characters are entered or the space bar is not depressed, the result field is not changed. Alphanumeric fields entered during program execution are blanked out; numeric fields are zeroed out before the actual input field is moved in.

READ (READ)

The READ operation is used to call for demand file information to be input during the calculations in the program cycle. This operation differs from the FORCE operation in that FORCE specifies input on the next program cycle rather than the present one. When the program is reading from several demand files during the same RPG II cycle, record identifying indicators assigned to the demand files remain on throughout the cycle if the previous READ operations were executed successfully.

The READ operation code must appear in columns 28-32. Factor 2 must contain the name of the file from which a record will immediately be read. An indicator should be specified in columns 58-59 to turn on after each READ operation if an end-of-file condition is reached. When columns 53-59 (resulting indicators) are blank, a halt condition occurs on an end-of-file condition and on subsequent READ operations after the end-of-file condition is reached. Indicators may be specified in columns 7-17.

The READ operation is similar to the CHAIN operation, except that the READ file is processed sequentially and the CHAIN file is processed randomly. Demand files can be processed only by the READ operation; CHAIN cannot be used. When the READ operation is used for demand files, the MR indicator cannot be entered in columns 63-64 on the input sheet. Numeric sequence testing on the input sheet is not allowed for demand files, nor are control levels, match fields, and look-ahead fields. When a demand file is conditioned by a U1-U8 indicator which is not on, no records are read from that file and the end-of-file indicator in columns 58-59 does not turn on.

The following files can appear as factor 2 in a READ operation, provided they are designated as demand files by a D in column 16 of the file description sheet:

Sequential files processed consecutively and specified as input or update files in column 15 of the file description sheet.

Indexed disc files processed sequentially by key and specified as input or update files.

Indexed disc files processed sequentially within limits and specified as input or update files.

CHAIN (CHAIN)

The CHAIN operation causes a record to be read from a disc file during calculations. One record is allowed to be read in when the operation code CHAIN appears in columns 28-32 of the calculation sheet. Indicators may be used in columns 7-17, but the result field, field length, decimal position, and half-adjust (columns 43-53) entries must be blank. File conditioning indicators (U1-U8) can be used to condition a chained file, and columns 54-55 should contain a resulting indicator entry. If the record is not found, the indicator specified in these columns will turn on. If an indicator is not specified in columns 54-55 and the record is not found, the program will have a halt condition. No output is permitted to a chained update file when the specified record is not found. Columns 56-59 must always be blank for chain operations.

The chain operation is used for two purposes: random processing of an indexed or direct file; and direct file loading.

In random processing, a record must be identified by a relative record number before it can be read from a direct file, or by a record key before it can be read from an indexed file. The

relative record number or record key can be contained in a field specified for that purpose. In the chain operation, the operation code CHAIN is required in columns 28-32 of the calculation sheet. Factor 1 entries must be a relative record number or key. Relative record numbers must be numeric. Factor 2 must contain the name of the file from which the record will be read. This file is then the file that is chained to, or the chained file.

A direct file is created for direct file loading when it is defined as a chained output file on the file description sheet. Factor 1 in the calculation specifications must then contain a relative record number, columns 28-32 must contain the operation code CHAIN, and factor 2 must contain the name of the direct disc file to be loaded. Relative record numbers define the record position for each record in the direct disc file. The relative number can be all or part of a field in input records, or can be generated by the RPG II program. After the disc file is loaded, the relative record numbers are used for record identification of the disc records.

When a direct file is loaded as a chained output file, the system blanks out the disc space required for the direct file before it is loaded. The relative record number is used to chain to the corresponding relative record position in the disc file. The information in the file then replaces the blanks when it is written on disc. If a record is not loaded, the space reserved for that record in the disc file remains blank until the proper record is loaded at a later time.

Once the direct file is loaded, records can be inserted or changed in the file by defining the direct file as an update file processed either consecutively or by the chain operation.

Inserting records in direct disc files is a very different process from record addition to sequential or indexed files. A new record is added to a sequential disc file in the first position available at the end of the file. The same process occurs for an indexed file, except that the record key and disc address are added to the file index. Space is already reserved for any new records inserted in a direct disc file, hence the record is inserted in its proper place rather than added to the physical end of the file.

EXTERNAL LINKAGE OPERATIONS

External linkage operations permit interface between the main RPG II program, which performs internal calculations, and external subroutines and user programs. (See Appendix E for a detailed explanation of subroutine linkage.) Codes are provided for three external linkage operations: exit to a subroutine, RPG II label, and user's label.

EXIT TO A SUBROUTINE (EXIT)

The EXIT operation code allows the programmer to transfer control from the RPG II program to a user subroutine whose name is contained in factor 2. The name of the subroutine cannot be greater than six alphanumeric characters; the first character must be alphabetic. Factor 1 and the result field are not used with the EXIT operation.

Normal linkage uses register 6 for the parameter list pointer, but no parameter list is associated with the EXIT operation. When the EXIT operation is used in calculations, it links to a user subroutine with a register usage conversion which sets the save area address in register 7. The user must save any registers he uses in the save area pointed to by that register.

The save area format is as follows:

Words	Contents
1	Return address.
2	Previous save area address (always zero).
3	Status word (zero).
4-11	Register save area.

RPG II LABEL (RLABL)

RLABL allows a subroutine external to the RPG II program to reference a field in the RPG II program. The name of the field to be referenced is entered in the result field. It must be a valid numeric or alphanumeric field, an indicator, or a table, and may be from one to six alphanumeric characters long. The first character must be alphabetic.

The field length and decimal positions must be defined in the result field of an RLABL entry unless they are defined by a preceding entry in either the input or the calculation specifications. If UDATE, UMONTH, UDAY, or UYEAR is used in the result field of an RLABL operation, the field length or number of decimal positions need not be specified; however if they are specified, UDATE must have a field length of six with zero decimal positions, and the others must have a field length of two with zero decimal positions. Indicators, factor 1, and factor 2 are not used. To specify an indicator, the first three characters must be IND, followed by the two-character indicator; e.g., IND02, INDL1.

USER'S LABIL (ULABL)

The user's label operation enables the RPG II program to reference a field contained in a user subroutine whose name is entered in the result field. The name may be from one to six alphanumeric characters long; the first character must be alphabetic. The field length and decimal positions must be defined. Indicators, factor 1, and factor 2 are not used. A ULABL field can be specified on the calculation or output-format sheet, but cannot be specified on input sheets.

DEBUG OPERATIONS

The debug operation is an RPG II function used to locate errors in a program which is not working properly. The operation causes one or more records to be written which contain information helpful in finding programming errors, and is coded with the word DEBUG. The operation code produces results only if a 1 has been entered in column 15 of the control card specifications. If the control card entry has not been made, the operation code DEBUG is treated as a comment. Because additional processing considerations are involved, care must be exercised in writing debug records to a direct or indexed file.

In specifying the DEBUG code, use of factor 1 is optional. If it is used, it may contain a literal or field name which identifies the particular debug operation; this literal or the value of the field named is then written on record 1. Factor 2 must contain the name of the output file on which the records are written. The same output file name must appear in factor 2 for all DEBUG statements in a program. The result field may be a field, a table item, an array item, or a whole array whose contents are to be written on record 2. Any valid indicators may be used in columns 7-17. Columns 49-59 (length of result field, decimal positions, half-adjust, and resulting indicators) must be left blank.

The DEBUG operation code may be placed at any point or at several points in the calculation operations. Record 1 is required, and is written whenever it is encountered, but record 2 is written only if a specification has been entered for it. Record 1 contains a list of all indicators which are on at the time the DEBUG code was encountered. Record 2 shows the contents of any one field.

SPECIFYING RECORD 1

Record 1, which is always written, is specified in the following format:

Positions	Entry
2-7	DEBUG.
8	Blank.
9-16	The constant entered in factor 1 or the statement number of the DEBUG operation code in the program.
17	Blank.
18-31	The words INDICATORS ON.
32-any position, depending on the number of indicators on	The names of all indicators which are on, each separated by a blank. The word NONE is entered if no indicators are on. More than one record may be needed.

SPECIFYING RECORD 2

Record 2 is optional and need be specified only when there is a result field. The record is written in the following format:

Positions	Entry
2-12	The words FIELD VALUE or TABLE VALUE or ARRAY VALUE.
13-14	Blank.
15-any position, depending on field length	The contents of the result field or array (up to 255 characters per item). More than one record may be needed.

When the result field is written in record 2, a blank is used to separate each array element. When applicable, a negative sign is written following an array item, table item, or field. When the result field cannot be contained in a record, a continuation begins in position two of the following record. When one or more

array items can be written on a single record, but the next item cannot be entirely contained on the record, the next item is written in position two of the next record.

CODING INSTRUCTIONS

Detailed instructions for coding the calculation entries on the specifications sheets are defined below.

PAGE NUMBER (Columns 1-2)

Entry	Description
01-99	Specifications sheets are numbered sequentially.

More than one of each type of specifications sheet may be used in the course of coding the source program. To keep the sheets in the proper sequence, the coded Calculation Specifications Sheets must be grouped according to their type and arranged with the other specifications sheets in the following order:

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

The specifications sheets should then be numbered in ascending sequence.

LINE NUMBER (Columns 3-5)

Entry	Description
Any number	Line numbers are assigned in ascending sequence.

Line numbers are preprinted in columns 3-4 of the specifications sheet for the programmer's convenience. The unnumbered lines below the preprinted numbers can be used for additional lines or to insert a line between completed lines. Column 5 can also be used to this end. The line numbers assigned need not be consecutive, but they must be in ascending order.

TYPE OF FORM (Column 6)

Entry	Description
C	C identifies the calculation specifications to the compiler.

CONTROL LEVEL (Columns 7-8)

Entry	Description
Blank	The calculation operation is not part of a subroutine and may be performed only for detail calculations.
L0, L1-L9	The calculation operation is done when the appropriate control break occurs or an indicator is set on (L0 is always on).
LR	The calculation operation is done after the last record has been processed or after the LR indicator has been turned on by a SETON operation.
SR	The calculation operation is part of a subroutine.
AN,OR	An AN or OR relationship is being established between lines of indicators.

The type of calculations to be performed must be specified in the following order: detail (blank), total (L0, L1-L9, LR), subroutine (SR). (AN/OR lines can appear within any of the calculations specifications.)

If columns 7-8 are left blank, detail calculations are performed: the operation specified on the same line is performed each time a record is read, provided the operation is allowed by indicators in columns 9-17 of that line or by AN/OR lines associated with that line. Operations conditioned by control level indicators in columns 9-17 are done at detail calculation time immediately following the control break.

Total calculations are specified next, by LO, L1-L9, or LR in columns 7-8. In one program cycle, all operations conditioned by control level indicators in columns 7-8 are done at total calculation time. All LO calculations must appear before L1-L9, and LR calculations must appear after L1-L9 calculations. The LR indicator is used to condition all operations that are to be done only at the end of the job. When it automatically turns on after the last input record has been processed, all other control level indicators also turn on automatically, and the job ends after all total operations have been performed. The LR indicator can also be turned on by a SETON operation. This does not, however, cause all the other control level indicators used to turn on.

Subroutine calculations, specified by SR in columns 7-8, are used to indicate that a line is part of a subroutine (see "Subroutine Operations"). Subroutine lines must be specified last.

AN or OR specified in columns 7-8 shows that lines of indicators are in an AND or OR relationship. By using the AND/OR relationship, many lines of indicators may be grouped together to condition an operation. A maximum of seven AN, OR, or AN/OR lines may be used to condition an operation. The first line of a group linked by AN/OR lines must contain blanks in columns 7-8, or LO-L9 or LR if the group of lines is conditioned by a control level indicator, or SR if the group 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 indicators on each line are in an AND relationship. Although each AN/OR group requires at least one indicator, it is not necessary to have three indicators on each AN and OR line. All lines in the group prior to the last line must contain blanks in the columns for the factor 1, factor 2, operation, and resulting indicator entries. The last line of the group must contain the operation and the necessary operands.

Columns 7-80 may also be used for comments (see Section 3, columns 7-14).

OPERATION INDICATORS (Columns 9-17)

One to three indicators per line may be used in columns 9-17 to control whether or not an operation is to be done. Three separate fields appear on each line (columns 9-11, 12-14, and 15-17), one

for each indicator. The three indicators are in an AND relationship with each other. If AN or OR entries are used in columns 7-8, many indicators can be used to condition one operation. Up to seven AN or OR lines are permitted in any combination. Each operation indicator is specified by two entries: N, and the indicator itself.

NOT ON (Columns 9, 12, 15)

Entry	Description
Blank	The indicator specified in the next two columns is on.
N	The indicator specified in the next two columns is not on.

If an indicator must not be on to condition an operation, an N is placed in the column immediately preceding the indicator specification.

INDICATOR (Columns 10-11, 13-14, 16-17)

Entry	Description
Blank	The operation is performed every time it is encountered, if columns 7-8 are not IO, L1-L9, or SR.
01-99	The operation is performed if a resulting indicator used elsewhere in the program is on/off.
LO-L9	A control level indicator previously assigned is used.
LR	The last record indicator is used.
MR	The matching record indicator is used.
HO-H9	A halt indicator assigned elsewhere is used.
U1-U8	An external indicator previously set is used.

0A-0G 0V	An overflow indicator previously assigned is used.
1P	The first page indicator is used. (This indicator is normally off, even the first time through the object program cycle, unless it has been purposely set on.)

Up to three indicators may be entered on a line. Additional indicators may be entered on following AN/OR lines. The indicators on one line, or indicators in grouped lines, plus the control level indicator (if used in columns 7-8) must all be exactly as specified before the operation can be performed.

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

Entry	Description
1-10 character literal	Any alphanumeric or numeric literal may be used in either or both factors.
1-6 character name	Any subroutine, table, array name, or array item may be used in either or both factors. NOTE: An array item specification may exceed 6 characters because of the addition of a comma and an index.
UDATE UMONTH UDAY UYEAR	Special word date field names may be used in either or both factors.
PAGE PAGE1 PAGE2	Special word page fields may be used in either or both factors.
1-6 character label	A label for a TAG, BEGSR, or ENDSR operation may be used in factor 1 only.
1-6 character label	A label for a GOTO or EXSR operation may be used in factor 2 only.

Columns 18-27 and 33-42 are used to name the fields or to give the actual data (see "Literals" at the beginning of this section) on which an operation is to be performed.

An entry in factor 1 must begin in column 18; an entry in factor 2 must begin in column 33. The entries to be used depend upon the operation being described (see columns 28-32). Some operations

need entries in both factor 1 and factor 2, some operations need entries in only one of the factors, and some operations need no entries at all (see columns 28-32).

OPERATION (Columns 28-32)

Entry	Description
ADD	Adds factors 1 and 2 and places the sum in the result field.
Z-ADD	Replaces the result field with factor 2.
SUB	Subtracts factor 2 from factor 1 and places the difference in the result field.
Z-SUB	Changes the sign of the field by subtracting factor 2 from the result field and placing the negative of factor 2 in the field.
MULT	Multiplies factor 1 by factor 2 and places the product in the result field.
DIV	Divides factor 1 by factor 2 and places the quotient in the result field.
MVR	Moves the remainder from the previous divide operation to the result field.
SQRT	Derives the square root of factor 2 and places it in the result field.
XFOOT	Adds array items and puts the sum in the the result field.
MOVE	Moves characters from factor 2 to the rightmost positions of the result field.
MOVEL	Moves characters from factor 2 to the leftmost positions of the result field.
MHHZO	Moves the zone from the leftmost factor 2 position to the leftmost result field position.
MHLZO	Moves the zone from the leftmost factor 2 position to the rightmost result field position.

MLLZO	Moves zone from the rightmost factor 2 position to the rightmost result field position.
MLHZO	Moves zone from the rightmost factor 2 position to the leftmost result field position.
COMP	Compares factor 1 with factor 2 and turns on/off resulting indicators.
TESTZ	Tests the zone of the leftmost result field character and turns on/off a resulting indicator.
BITON	Turns on the result field bits identified in factor 2.
BITOF	Turns off the result field bits identified in factor 2.
TESTB	Tests the result field bits identified in factor 2 for an on/off condition and turns on/off resulting indicators.
SETON	Turns resulting indicators on.
SETOF	Turns resulting indicators off.
GOTO	Skips instructions backward or forward to continue processing at the named instruction.
TAG	Names the instruction to which the GOTO skip is to be made.
LOKUP	Searches for specific items in tables or arrays.
BEGSR	Defines the beginning point of a subroutine.
ENDSR	Defines the end of a subroutine.
EXSR	Causes a subroutine to be executed.
EXCPT	Writes records while calculations are being performed.
FORCE	Identifies the file from which the next record is to be selected.

DSPLY	Prints one or two fields, table items, literals, or array items up to 125 characters. A special case of the DSPLY operation causes a program halt and allows the programmer to blank out or change a field or item where contents have been displayed.
READ	Reads input from a demand file during the current program cycle.
CHAIN	Reads a record from a chained disc file during calculations.
EXIT	Transfers control from the RPG II program to a user subroutine.
RLABL	Allows an external subroutine to reference a field internal to the RPG II program.
ULABL	Allows the RPG II internal program to reference a field in an external subroutine.
DEBUG	Locates programming errors in a malfunctioning program.

Columns 28-32 are used to specify the kind of operation to be performed using resulting indicators, factor 1, factor 2, and/or the result field. The operation code must begin in column 28. The operations are performed in the order specified on the calculation sheet. All operations conditioned by control level indicators in columns 7-8 must follow those which are not conditioned by control level indicators. All operations that are part of a subroutine (SR in column 7-8) must follow all other calculations in a program. Every operation code used requires certain entries on the same specification line. See "Operation Codes" at the beginning of this section for more information.

RESULT FIELD (Columns 43-48)

Entry	Description
1-6 valid characters	Contains the name of a field, table, array, or array item.

Columns 43-48 are used to name the field, table, array, or array item that is to hold the result of the operation specified in

columns 28-32. The result field name must begin with an alphabetic character in column 43 and contain no blanks or special characters. A new field may be defined, or the name of a field, table, array, or array item that has already been defined in extension specifications, input specifications, or elsewhere in the calculation specifications may be used. If the name of a field that has already been defined is being entered, entries in columns 49-52 are not necessary; but if they are specified they must agree with the previous definition of that field.

If the name of a field that has not been defined elsewhere is being entered, columns 49-52 should also contain entries. A new field may be defined by entering a field name that has not already been used; the field will then be created when the program is compiled. The named field may be either numeric or alphanumeric, but it must be numeric if it is used in arithmetic operations or if it is edited or zero-suppressed in output-format specifications.

RESULT FIELD LENGTH (Columns 49-51)

Entry	Description
Blank	The alphanumeric or numeric field to be used is described elsewhere.
1-255	Defines the number of digits in a numeric result field (1-15), or the number of characters in an alphanumeric field (1-255).

Columns 49-51 are used to specify the length for any result field. If a result field that has not been used before is being named, the new form of the data and its new length after performing the operation must be considered.

DECIMAL POSITIONS (Column 52)

Entry	Description
Blank	Either the field is described elsewhere, or the field is alphanumeric.
0-9	The number of decimal places in a numeric result field is entered.

Column 52 is used to indicate the number of positions to the right of the decimal point in a numeric result field. If the numeric result field contains no decimal positions, a zero is entered.

The number of decimal positions must never be greater than the length of the field. The number may, however, be larger or smaller than the number of decimal positions that actually result from an operation. If the number is greater, zeros are filled in to the right. If it is smaller, the rightmost digits are dropped.

The decimal positions entry must be left blank if the result field is alphanumeric. It may also be left blank if the result field is numeric but has been previously described in the extension, input, or calculation specifications, in which case the field length columns (49-51) must also be blank.

HALF-ADJUST (Column 53)

Entry	Description
Blank	Half-adjusting does not take place.
H	The contents of the result field are to be half-adjusted.

Column 53 can be used only with arithmetic operations to indicate that the contents of the result field are to be half-adjusted (rounded). Half-adjusting is done by adding a 5 to the number at the right of the last decimal position specified for a positive field, or -5 if the field is negative. All decimal positions to the right of the position specified for that field are then dropped.

RESULTING INDICATORS (Columns 54-59)

Entry	Description
01-99	A numeric indicator is used.
H0-H9	A halt indicator is used.
L0-L9	A control level indicator is used.

LR	A last record indicator is used.
MR	The matching record indicator is used.
OA-OG, OV	An overflow indicator specified on the file description sheet is used.
U1-U8	A user-defined internal indicator is used.

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 change by a SETON or SETOF operation; or to indicate end-of-file for the READ operation code.

Entering an indicator in columns 54-59 specifies that the result field is to be tested after the operation specified in columns 28-32 has been performed. Normally, only indicators 01-99 and H0-H9 are used for testing. Column headings in columns 54-59 have no meaning for the SETON and SETOF operations, however the indicators to be turned on or off by the operations SETON or SETOF may be entered. Any indicators to be turned on or off by the SETON or SETOF operations are specified from left to right in the three resulting indicator fields.

If an 01-99 or H0-H9 indicator is specified, it is turned on only if the result field satisfies the condition for which it is being tested, otherwise it is turned off. Three fields can be used for this purpose, each of which tests for different conditions. Columns 54-55 test for plus or high, columns 56-57 test for minus or low, and columns 58-59 test for zero or equal. An indicator can be tested for more than one of the conditions.

An indicator is placed in columns 54-55 (PLUS or HIGH) to test whether the result field in an arithmetic operation is positive; whether factor 1 is higher than factor 2 in a COMP or table or array LOKUP operation; or to test the results of one of the following operations: CHAIN (not found), TESTB (all zeros), or TESTZ (C zone).

An indicator is placed in columns 56-57 (MINUS or LOW) to test whether the result field in an arithmetic operation is negative; whether factor 1 is lower than factor 2 in a COMP operation or table or array LOKUP operation; or to test the results of a TESTB (mixed) or TESTZ (D zone) operation.

An indicator is placed in columns 58-59 (ZERO or EQUAL) to test whether the result field in an arithmetic operation is zero; whether factor 1 is equal to factor 2 in a COMP or table or array LOKUP operation; or to test the results of one of the following operations: READ (end-of-file), TESTB (all ones), or TESTZ (not C or D zone).

COMMENTS (Columns 60-74)

Entry	Description
1-15 characters	Alphabetic, numeric, and special characters may be used in any combination.

Any meaningful information the programmer desires can be entered in columns 60-74 to help in understanding or remembering what is being done on each specification line. Comments are not instructions to the RPG II program; they serve only as a means of documenting a program.

PROGRAM IDENTIFICATION (Columns 75-80)

Entry	Description
Any characters	These columns may be used for programmer comments.

The program identification entry of all cards following the control card may contain any characters, including blanks. The columns may be used for programmer comments, if desired.

SECTION 8. OUTPUT-FORMAT SPECIFICATIONS

Output is the result or product of processing. Output records consist of data that is gathered, calculated, or generated during object program operation. The output records are stored in files originally designated as output or update files on the file specifications sheet.

The Output-Format Specifications Sheet (Figure 8-1) is used to describe these output records. Two kinds of entries can be made on this sheet: record description entries that describe which output records are to be written; and field description entries, which indicate the position and format of the information contained in the output records. Field description entries must always be entered one line lower than record description entries.

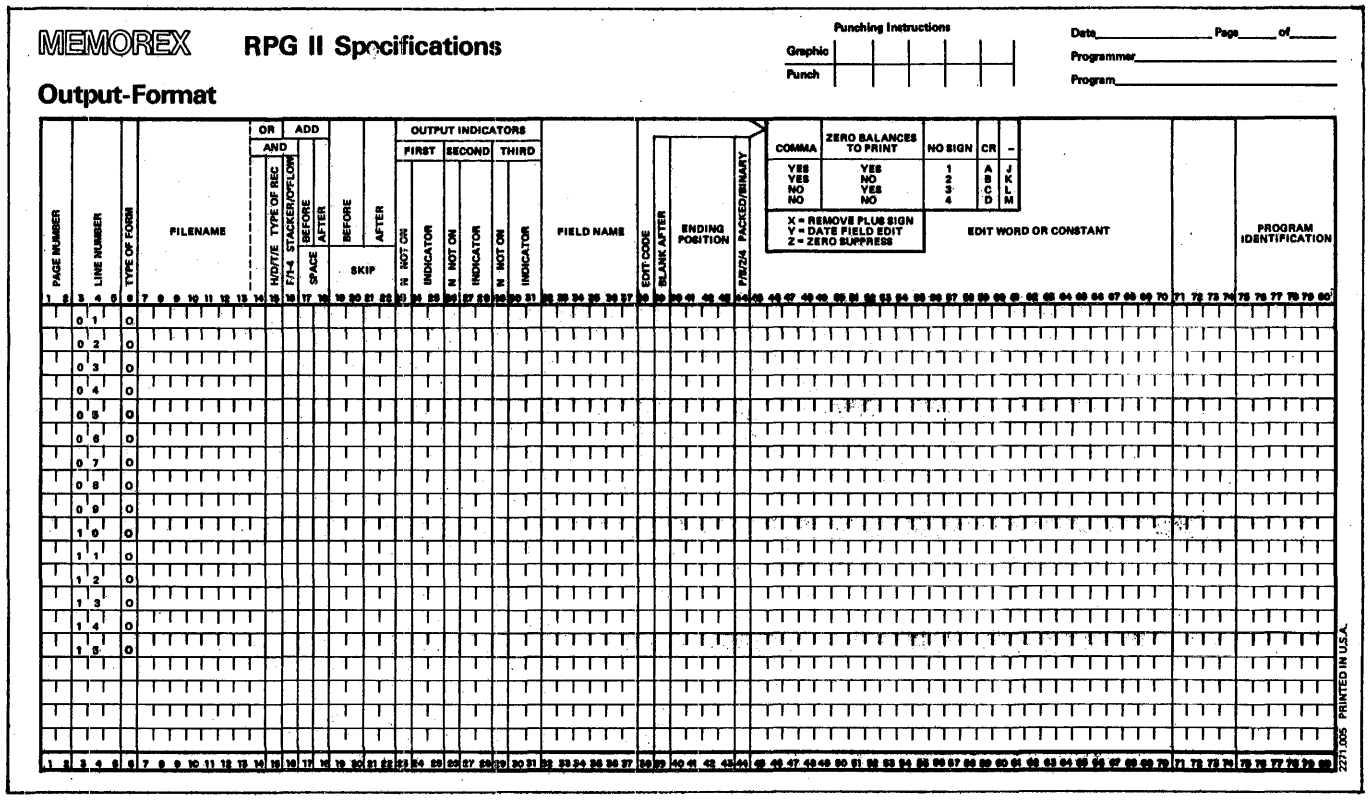


Figure 8-1. RPG II Output-Format Specifications Sheet.

TYPES OF OUTPUT RECORDS

The four types of output records are heading records, detail records, total records, and exception records.

Heading records contain identifying information which does not usually change from page to page, such as the report name, the date, and the column titles.

Detail records contain information that comprises the subject matter of the report. Most of the information in detail records is gathered directly from input records or calculated during program operation.

Total records are generally cumulative sums of the calculations in detail records, and are usually written last. They may not be specified for primary or secondary update files.

Exception records are always written during calculation time, and can only be used when the EXCPT operation code is entered on the Calculation Specifications Sheet (Section 7).

Within each type, records are processed in the order in which they appear in the output specifications.

OVERFLOW INDICATOR

An overflow indicator may be assigned for a print file on the File Description Specifications Sheet (Section 3). The indicator is turned on when a line is printed on or below the overflow line. If overflow occurs during header and detail output, the indicator is turned off after the header and detail records of the next cycle are written to output. If overflow occurs during total output or during EXCPT output (see Section 7, "Operation Codes"), the indicator is turned off after the header and detail records of the following cycle are written to output.

NO OVERFLOW PROCESSING

If an overflow indicator is assigned in the file description specifications but is never used to condition an output record, the RPG II System does no special processing for the overflow condition.

AUTOMATIC SKIP TO TOP OF FORMS

If no overflow indicator is assigned in the file description specifications for a print file, the RPG II System generates a skip to channel 1 in place of normal overflow processing. The timing of this skip cannot be altered by anything in the program.

OVERFLOW ROUTINE

All records conditioned by the overflow indicator are output by the overflow routine. Under normal conditions the overflow routine is called automatically, however in certain cases the overflow routine may be called or "fetched" ahead of the normal time (see column 16, this section).

Normal Processing

If the assigned overflow indicator is turned on during normal output, processing continues in the usual manner until all total, header, and detail records not conditioned by the indicator have been written to output. At this point in the program cycle the overflow routine is called.

The overflow routine first prints all total records that have been conditioned by the indicator, then all header and detail records conditioned by the indicator. A skip to channel 1 should be entered in the first header record, since skips to a new page are not automatically generated by the RPG II System in normal overflow processing.

Fetching the Overflow Routine

Problems can occur under normal overflow processing when several detail and total lines must be printed during each regular program cycle, causing printing to go past the end of the page. Although the programmer could avoid this situation by setting the overflow line high enough on the page to accommodate all the possible lines after overflow, some empty pages would result.

To circumvent the problem entirely, the overflow line should be assigned just high enough on the page to include only the number of lines needed to print all total records conditioned by the overflow indicator, then the overflow routine should be fetched

for the given printer line. If this procedure is followed, the overflow indicator is first checked, and if it is on, the overflow routine is executed before that line is printed.

PAGE NUMBERING

Automatic page numbering takes place when the special word PAGE is entered in the field name columns (32-37). When PAGE is entered in the Output-Format Specifications Sheet without being defined elsewhere, the program treats it as a four-position numeric field with no decimal positions, and automatically increments it by 1 each time a new page is encountered. Page numbering may start at a number other than 1 only through a specification on the input sheet (see Section 6, columns 53-58) or the calculation sheet (see Section 7, columns 43-48).

A PAGE field defined in the input or calculation specifications sheets may contain as many as 15 positions, but must be defined with zero decimal positions. Leading zeros are suppressed, and the sign must be specified via an edit word or code before it can be printed in the rightmost position. Page numbering then begins with 1, unless otherwise specified, and is automatically incremented by 1 each time the PAGE field is written.

Up to three print files can be individually numbered concurrently by entering special words PAGE, PAGE1, and PAGE2 in the field name columns of the output specifications, however each special word can be assigned to only one file.

ITERATED FIELDS

Entering *PLACE in columns 32-37 causes all fields named before it on the specifications sheet for each to be repeated in the same relative locations ending at the position defined in the *PLACE statements. An end position must be specified for every *PLACE line; the leftmost position of the fields to be moved by the *PLACE specification is always assumed to be position 1. The end position should therefore be twice the number of the highest end position used in the previously named fields, to avoid overlaying previous data.

REPORT DATES

Special words may be used in columns 32-37 to cause the date to appear in a specified format on printed reports or output records. Four such words allow the programmer to vary his format as desired: UDATE, UDAY, UMONTH, and UYEAR. UDATE produces a six-character numeric date field in either Domestic format (mmddyy) or United Kingdom/European Convention format (ddmmyy). UDAY may be used to obtain dates consisting of days only; UMONTH to obtain months only; and UYEAR to obtain years only. Slashes and periods may be inserted via editing to separate from each other the day, month, and year portions of a date.

EDITED FIELDS

Editing can be specified for the information in numeric output fields to enhance the legibility of a field in several ways:

By inserting special characters or delimiters, such as commas, periods, spaces, etc., between characters of the field.

By indicating a negative field with a minus sign (-) or credit sign (CR).

By suppressing leading zeros and replacing them with blanks.

By suppressing leading zeros and printing asterisks instead (asterisk fill).

By inserting a dollar sign immediately before the most significant non-zero digit in the field (floating dollar sign).

EDIT CODES

One-character edit codes provide the above editing features for frequently used formats and take into consideration the inverted print option of the control card specification (Section 2, column 21).

EDIT WORDS

Edit words give the programmer more flexibility by allowing him to lay out explicitly the entire resulting output field, character by character. This is done using replaceable and non-replaceable characters. Replaceable characters are those positions which are to be occupied by digits of the edited field.

CONSTANTS

Constants may be used in columns 45-70 instead of edit words; or they may be used within edit words. Constants comprise any unchanging information, such as a column heading or title, that is entered by a specification. They must be enclosed in apostrophes when they are entered on the specifications sheet. Up to 24 characters of constant information may be entered on a specifications line.

CODING INSTRUCTIONS

Detailed instructions for coding the output-format entries on the specifications sheets are defined below.

PAGE NUMBER (Columns 1-2)

Entry	Description
01-99	Specifications sheets are numbered sequentially.

More than one of each type of specifications sheet may be used in the course of coding the source program. To keep the sheets in the proper order, the coded Output-Format Specifications Sheets must be grouped together, numbered sequentially, and placed immediately after the Calculation Specifications Sheets.

LINE NUMBER (Columns 3-5)

Entry	Description
Any number	Line numbers are assigned in ascending sequence.

Line numbers are preprinted in columns 3-4 of the specifications sheet for the programmer's convenience. The unnumbered lines below the preprinted numbers can be used for additional lines or to insert a line between completed lines. Column 5 can also be used to this end. The line numbers assigned need not be consecutive, but they must be in ascending order.

TYPE OF FORM (Column 6)

Entry	Description
O	O identifies the output-format specifications to the compiler.

FILENAME (Columns 7-14)

Entry	Description
1-8 characters	The output file being described is named.

Every output file to which records are to be written must be named previously in the file description specifications. The exact filename originally assigned must be entered, beginning in column 7.

The output filename must appear on the first line containing information relevant to a particular file, but need be specified only once. The information on the following lines refers to the previous filename until a new filename is entered in columns 7-14.

Columns 7-80 may also be used for comments (see Section 3, columns 7-14).

AND/OR (Columns 14-16)

Entry	Description
AND	More than three indicators are used to condition the output operation.
OR	Output indicators are used to condition an OR relationship.

The AND/OR entry is used in conjunction with the output indicators entry (columns 23-31). AND lines can be used when more than three output indicators are used to condition an output operation. The word AND is entered in columns 14-16 of the next lines following the first line on which the first three indicators are specified. The condition of all indicators assigned must then be satisfied (on or off) before the output operation can be performed.

Output indicators may also be assigned in an OR relationship. If either or both of the OR conditions are met, the output operation is performed. OR lines are indicated by the word OR in columns 14-15.

Both AND and OR lines are allowed in an output operation. A maximum of 20 such lines may be used for each record description. AND and OR lines cannot be used to condition a field.

TYPE OF RECORD (Column 15)

Entry	Description
H	The record is a heading record.
D	The record is a detail record.
T	The record is a total record.
E	The record is an exception record.

The entry in column 15 is used to specify the type of output record to be printed, punched, or written on disc. If column 15 is left blank, an error condition occurs. See columns 23-31 for information concerning the heading and detail specifications used with control level and overflow indicators that specify when output records are to be written.

If it is necessary to continue the description of indicator conditions under which output should occur, the specifications should be made in AND or OR specifications lines following the type of record entry.

ADD (Columns 16-18)

Entry	Description
ADD	A record is to be added to the file.

When a record is to be added to an input, output, or update file, the word ADD is entered in columns 16-18 of the first line for each record to be added. Disc must be specified as the output device for the file to which the record is to be added, and an A must be coded in column 66 of the File Description Specifications Sheet for the file. Columns 16-18 may also be used for the stacker/overflow and space specifications for printer or card files.

STACKER/OVERFLOW (Column 16)

Entry	Description
Blank	The overflow routine is not to be fetched.
1-4	(For card files this specification is reserved for future implementation.)
F	The overflow routine is to be fetched for a printer file.

Column 16 may be used to specify that the overflow routine is to be used for a printer file.

An F entered in column 16 is used to specify the fetch overflow routine and causes overflow lines to be printed ahead of the usual time. The fetch overflow routine does not automatically cause forms to advance to a new page, but merely prohibits the printing of remaining detail and total lines until overflow lines are printed. To achieve page advancing, a skip to channel 1 must be specified on a line conditioned by the overflow indicator.

The overflow routine is fetched only if all conditions specified by the indicators in columns 23-31 are met, and an overflow has occurred. The 1P indicator may not be used with fetched overflow lines.

SPACE (Columns 17-18)

Columns 17-18 are used to specify line spacing for one page of an output file to be printed. Line spacing may be required both before and after a line, thus the entry is divided into before and after columns. Up to three spaces may be specified both before and after a line of print, so that as many as six spaces can be left between two printed lines.

If columns 17-18 are left blank, single spacing occurs automatically after each line is printed.

When the destination of a spacing operation is beyond the overflow line but not on a new page, the overflow indicator is turned on until all overflow lines are printed.

When space and skip are specified on the same line, they are done in the following order:

Skip before the line of print.
Space before the line of print.
Skip after the line of print.
Space after the line of print.

BEFORE (Column 17)

Entry	Description
Blank	No spacing is required before a line is printed.
0	No spacing is required.
1	One space is to be left before each line is printed.
2	Two spaces are to be left before each line is printed.
3	Three spaces are to be left before each line is printed.

If column 17 is left blank, no spacing occurs before print; otherwise one, two, three, or no lines are left blank before the next line is printed, according to the number entered.

AFTER (Column 18)

Entry	Description
Blank	No spacing is required after a line is printed.
0	No spacing is required.
1	One space is to be left after each line is printed.
2	Two spaces are to be left after each line is printed.
3	Three spaces are to be left after each line is printed.

If columns 17 and 18 are left blank, single spacing occurs automatically; otherwise one, two, three, or no lines are left blank after each line is printed, according to the number entered.

SKIP (Columns 19-22)

Skipping is controlled by the printer carriage control tape. A skip can be made to channel 01 for a new page, or to any of the other 11 channels assigned in the line counter specifications for succeeding pages. Skips can be specified to occur both prior to a line of print (columns 19-20) and after the line is printed (columns 21-22). If the columns are left blank, no skipping occurs.

If an L is specified in column 53 of the control card, the generated RPG program only simulates skips to a line. It gets its starting point by an initial skip to channel 01, and assumes that channel 01 is on line 06. All subsequent spacing is then relative to that starting point.

BEFORE (Columns 19-20)

Entry	Description
Blank	No skipping is to be done before a line is printed.
01	A skip is to be made to a new page before a line is printed.
02-12	A skip is to be made to the indicated channel number before a line is printed.
01-99	An L has been entered in column 53 of the control card specifications sheet; a skip is to be made to the line indicated.
A0-A9	An L has been entered in column 53 of the control card specifications sheet; a skip is to be made to the specified line in the range 100 (A0) through 109 (A9).
B0-B2	An L has been entered in column 53 of the control card specifications sheet; a skip is to be made to the specified line in the range 110 (B0) through 112 (B2).

If printing is to be continous, columns 19-20 are left blank; otherwise skips can be made to other print channels before printing starts to begin a new page or format the printing on the current page.

AFTER (Columns 21-22)

Entry	Description
Blank	No skipping is to be done after a line is printed.
01	A skip is to be made to a new page after a line is printed.
02-12	A skip is to be made to the indicated channel number after a line is printed.
01-99	An L has been entered in column 53 of the control card specifications sheet; a skip is to be made to the line indicated.
A0-A9	An L has been entered in column 53 of the control card specifications sheet; a skip is to be made to the specified line in the range 100 (A0) through 109 (A9).
B0-B2	An L has been entered in column 53 of the control card specifications sheet; a skip is to be made to the specified line in the range 110 (B0) through 112 (B2).

An entry in columns 21-22 causes skipping to the specified channel to be performed after a line is printed. An 01 specification, for example, would cause the next line to start at the top of a new page.

OUTPUT INDICATORS (Columns 23-31)

The output indicator columns are divided into three entries. Up to three indicators can be entered per line. One indicator can be entered in columns 23-25, another in columns 26-28, and a third in columns 29-31. If the actual state of the indicators coincides exactly with that described in columns 28-31 (and all AND/OR lines), and the line is an output field line, the field is put into its output record. If the state of the indicators coincides and the line is a record description line, the record is output. (See Appendix B, Tables B2 and B3, for indicator summaries.)

NOT ON (Columns 23, 26, 29)

Entry	Description
Blank	The indicator specified in the next two columns is on.
N	The indicator specified in the next two columns is not on.

If an indicator is not required to be on to condition an output operation, an N must be placed in the column immediately preceding the indicator specification. The indicator must then be exactly as specified (not on) before the output operation can be performed.

INDICATOR (Columns 24-25, 27-28, 30-31)

Entry	Description
01-99	A resulting indicator, field indicator, or record identifying indicator previously assigned to the record or field is to be used.
L1-L9	A control level indicator previously assigned to the record or field is to be used.
H0-H9	A halt indicator previously assigned to the record or field is to be used.
ØA-ØG,ØV	An overflow indicator previously assigned to the file containing this record or field is to be used.
MR	The matching record indicator is to be used.
LR	The end-of-job indicator is to be used.
1P	The first page indicator is to be used. This indicator cannot be used with record types T and E.
LO	The level zero indicator is to be used.
U1-U8	An external indicator is to be used.

Up to three indicators may be entered on a line. When a field is to be printed, the output indicators in columns 23-31 must be entered on the same line being used to specify the field name (columns 32-37).

A file named in the output-format specifications may be conditioned by an external indicator (U1-U8) in the file description specifications. External indicators can also be used to condition a record or field. No output can occur to a file if an external indicator conditions the file and that indicator is off.

Fields can be prevented from printing through the use of indicators.

FIELD NAME (Columns 32-37)

Entry	Description
Blank	A constant is to be entered in columns 45-70 (edit word or constant entry).
1-6 characters	A field name previously assigned in the program or the name of a table, array, or array item is to be used.
*PLACE	This special word is used to cause a repeat of a field or a set of fields within a record or report.
PAGE PAGE1 PAGE2	These special words are used in numbering the pages of the report.
UDATE UDAY UMONTH UYEAR	These special words are used in dating the report.

If a field name is used in columns 32-37, it must be the same as the field name assigned on the input sheet or the calculations sheet. No entry can be made in columns 7-22 of the same line and although an edit word may be entered in columns 45-70, a constant cannot.

Fields can be entered in any order. The sequence in which they will be moved to the output record buffer is determined by the order in which they appear on the specifications sheets. When a later output field overlaps the first field specified, the overlayed original data is lost. Detail printing of a selected field occurs when entries are specified for the field name, columns 32-37, and ending position, columns 40-43, in the Output-Format Specifications Sheet. Output indicators may also be specified. When a field is to be printed, the associated output indicators (columns 23-31) must be entered on the same line being used to specify the field name (columns 32-37).

Numeric field signs (+ or -) lie in the units position containing the rightmost digit. A minus sign in the rightmost digit of a numeric field prints as a letter unless the field is edited (see column 38).

Special Word *PLACE

The special word *PLACE is used to cause specified fields to be repeated ending in a specified position of an output record.

*PLACE must be specified after the field names which are to be placed in different positions in one line. The entry then causes all fields in a record type which are listed above it to be written. *PLACE must appear on a separate specification line for every additional time the field or group of fields is to be written or punched.

An end position must be specified for every *PLACE line. The end position specified for *PLACE must be at least twice the highest previously specified field end position. When *PLACE is specified for card output, the fields and constants named above *PLACE will be repunched. A *PLACE specification must not be conditioned by indicators in columns 23-31. *PLACE is automatically conditioned by the same indicators which condition the field or fields to be repeated. Columns 38, 39, and 44-70 should be blank for *PLACE.

Special Words PAGE, PAGE1, PAGE2

Special word PAGE, PAGE1, or PAGE2 is entered in columns 32-37 when the report pages are to be numbered. The same entry cannot be used for two different output files. The page numbering sequence can be restarted at any time by setting the PAGE field to zero before it is printed. This can be done via the blank after entry (column 39) or by using an output indicator. A PAGE field is always printed, regardless of whether or not it is conditioned by an indicator. If the indicator is as specified in columns 27-31, the PAGE field is set to zero, 1 is added, and the page number is written as 1.

A page field is assumed to be a four-position numeric field with no decimal positions. Leading zeros are suppressed and the sign is not printed in the rightmost position unless the field is so edited.

Special Words UDATE, UMONTH, UDAY, UYEAR

Special words UDATE, UMONTH, UDAY, and UYEAR are entered in columns 32-37 to record the system date in the report. UDATE produces a six-character numeric date field in mmddyy or ddyymm format (specified in column 21 of the control card). Using edit code Y, the edited date field is eight characters long, in one of three formats:

Domestic (MM/DD/YY).
United Kingdom (DD/MM/YY).
European Convention (DD.MM.YY).

UDAY, UMONTH, and UYEAR may be used for days only, months only, and years only, respectively. The special word date fields may not be changed by any operations specified in the calculations or output sheet.

EDIT CODE (Column 38)

Entry	Description
Any valid edit code	The edit code specifies the type of editing to be done on the specified numeric field.

Edit codes can be used only for numeric fields. The table shown in Figure 8-2 summarizes the edit codes provided on the Output-Format Specifications Sheet.

COMMA	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 8-2. Edit Code Table from Output-Format Specifications Sheet.

Each edit code punctuates a numeric field differently, as Figure 8-3 shows. All codes suppress leading zeros except in the European Convention output format, specified by a J entered in column 21 of the control card. In this case all zero balances, as well as all balances with zeros to the left of the decimal comma, are written with one leading zero: .00 is written 0,00 and .07 is written 0,07. An edit code specified to print zero balances cannot suppress the zero to the left of the decimal comma; that zero will always print.

If an edit code is used to punctuate an array, two spaces are left between array items. An edit word cannot be used in columns 45-70 if an edit code is used in column 38, except in the following cases:

When leading zeros are to be replaced by asterisks, '*' must be entered in columns 45-47 of the line containing the edit code.

When a dollar sign (floating) is to appear before the first digit in the field, '$' should be entered in columns 45-47 of the line containing the edit code.

It is also possible to cause a dollar sign to appear before the asterisk fill by placing a dollar-sign constant one space before the beginning of the edited field, then placing '*' in columns 45-47 of the line containing the edit code.

Asterisk fill and the floating dollar sign may not be used with the X, Y, and Z edit codes.

(Edit codes are summarized in detail in Appendix B, Table B5.)

EDIT CODE	CHARACTER PRINTED			
	COMMA	ZERO BALANCE	CR	MINUS SIGN
1	Yes	Yes	No	No
2	Yes	No	No	No
3	No	Yes	No	No
4	No	No	No	No
A	Yes	Yes	Yes	No
B	Yes	No	Yes	No
C	No	Yes	Yes	No
D	No	No	Yes	No
J	Yes	Yes	No	Yes
K	Yes	No	No	Yes
L	No	Yes	No	Yes
M	No	No	No	Yes
X	Removes the plus sign.			
Y	Edits the date field.			
Z	Suppresses zeros.			

Figure 8-3. Edit Code Functions.

Effects of using the inverted print option (control card specifications, column 21) in punctuating the date to obtain domestic, United Kingdom, and European Convention formats are illustrated in Section 2, Figure 2-2.

BLANK AFTER (Column 39)

Entry	Description
Blank	The field being described is not to be reset to zeros or blanks after writing.
B	The field being described is to be reset to zeros or blanks after writing.

An entry is used in column 39 to reset a field to zeros or blanks. When the field is numeric, it is reset to zeros; when the field is alphanumeric, it is reset to blanks. No entry can be made for look-ahead fields or date fields assigned the names UDATE, UMONTH, UDAY, or UYEAR.

Resetting fields to zeros is usually necessary when totals are being accumulated and printed for several control groups. Once the total for one group is found and printed, totals start accumulating for the next group. To prevent their being added in the totals for the previous group, the fields should be reset to zeros so that accurate totals will result. Entering a B in column 39 causes the total field to be reset to zero automatically after it is printed. The B must be entered on the last output line for that field; otherwise the field will be reset before all required output is printed.

When a table name is specified in the field name together with the blank after entry, the table item that was looked up last will be reset to blanks or zeros.

ENDING POSITION (Column 40-43)

Entry	Description
1-4 characters	The number of the print or punch position of the rightmost character in the field being described is entered.

The number of the printing position of the rightmost character in a field or constant is entered in columns 40-43 to indicate its location on the output record to be written.

Enough space must be allowed in the output record to hold edited fields. The largest number that can be used to indicate the end position for disc output is the record size. The largest number that can be used for printer output is governed by the number of print positions on the printer. When *PLACE is specified in columns 32-37 for the printer, the ending position entry indicates the end position of the last field of the group to be printed.

PACKED OR BINARY FIELD (Column 44)

Entry	Description
Blank	The field is in unpacked numeric format or contains alphanumeric data.
P	The field is to be written on disc or punched into a card in packed decimal format.
B	The field is to be written on disc or punched into a card in binary format.
2	The field is to be written on disc as a two-byte binary field.
4	The field is to be written on disc as a four-byte binary field.

An entry must appear in column 44 when a decimal number is to be written on disc or punched into a card in packed decimal or binary format. Packed decimal and binary fields should not be printed; they can only be written on disc or punched into cards.

After decimal fields are processed they can be left in unpacked numeric format, but disc space is more efficiently used if they are converted into packed decimal or binary format. Fields containing five or less digits are converted to two bytes of binary data for output; fields containing six to ten digits are converted to four bytes. To control the binary size directly, a 2 must be entered for two-byte binary output, or a 4 for four-byte binary output.

EDIT WORD OR CONSTANT (Columns 45-70)

Entry	Description
1-24 valid RPG II characters	Either an edit word is used to punctuate a numeric field and columns 32-37 contain a field name; or constant information is to be entered in the record and columns 32-37 are blank.

Edit words and constants are specified in columns 45-70. Edit words are direct specifications which indicate whether or not specific punctuation characters are needed. Because they are directly specified, edit words give more flexibility than edit

codes. When an edit word is specified, an edit code cannot be used in column 38. The name of a numeric field must be entered in the field name (columns 32-37) and the ending position must be specified in columns 40-43. An edit word cannot be longer than 24 characters, and must be enclosed in apostrophes, with the leading apostrophe entered in column 45. The edit word itself must begin in column 46. The number of replaceable characters in an edit word must be equal to the length of the field to be edited.

The replaceable characters are listed below.

Character	Position	Use
∅	Any	Indicates that a digit from the edited field is to be placed in this position.
0	Where zero suppression is to end	Indicates that leading zeros are to be replaced by blanks up to and including this position.
*	Where asterisk fill is to end	Indicates that leading zeros are to be replaced by asterisks up to and including this position.
$	Left of the leftmost zero	Indicates that a floating dollar sign is to be placed before the most significant digit.

All leading zeros are suppressed up to and including the first zero or asterisk.

The number of replaceable characters in the edit word must be equal to the length of the field to be edited, except in two cases:

An extra space must be left in the edit word for a floating dollar sign to ensure a print position for the dollar sign if the output field is full.

An extra space can be left in the edit word when the first character in the edit word is a zero. As a result, the field to be edited is not zero-suppressed, but all other specified editing is done.

All characters except ⌀, 0, *, and $, listed above, are non-replaceable characters. Non-replaceable characters always print in the positions indicated in the edit word, except in two cases:

When CR and the minus sign (-) are used to the right of the rightmost replaceable character, they are printed only when the field is negative. Blanks are printed otherwise. All characters between the CR or minus sign, plus the rightmost replaceable character, are printed only when the field is negative.

All characters inserted between digits of the field are zero-suppressed along with the digits of the field.

An ampersand (&) is used in an edit word to represent a blank. A double apostrophe should be used in either an edit word or a constant to represent an apostrophe.

Constants (Figure 8-4) comprise any unchanging information that is entered by a specification. The data is entered exactly as it is to appear in the output record. Columns 32-39 and 44 should be blank. The constant should be enclosed in apostrophes with the leading apostrophe in column 45.

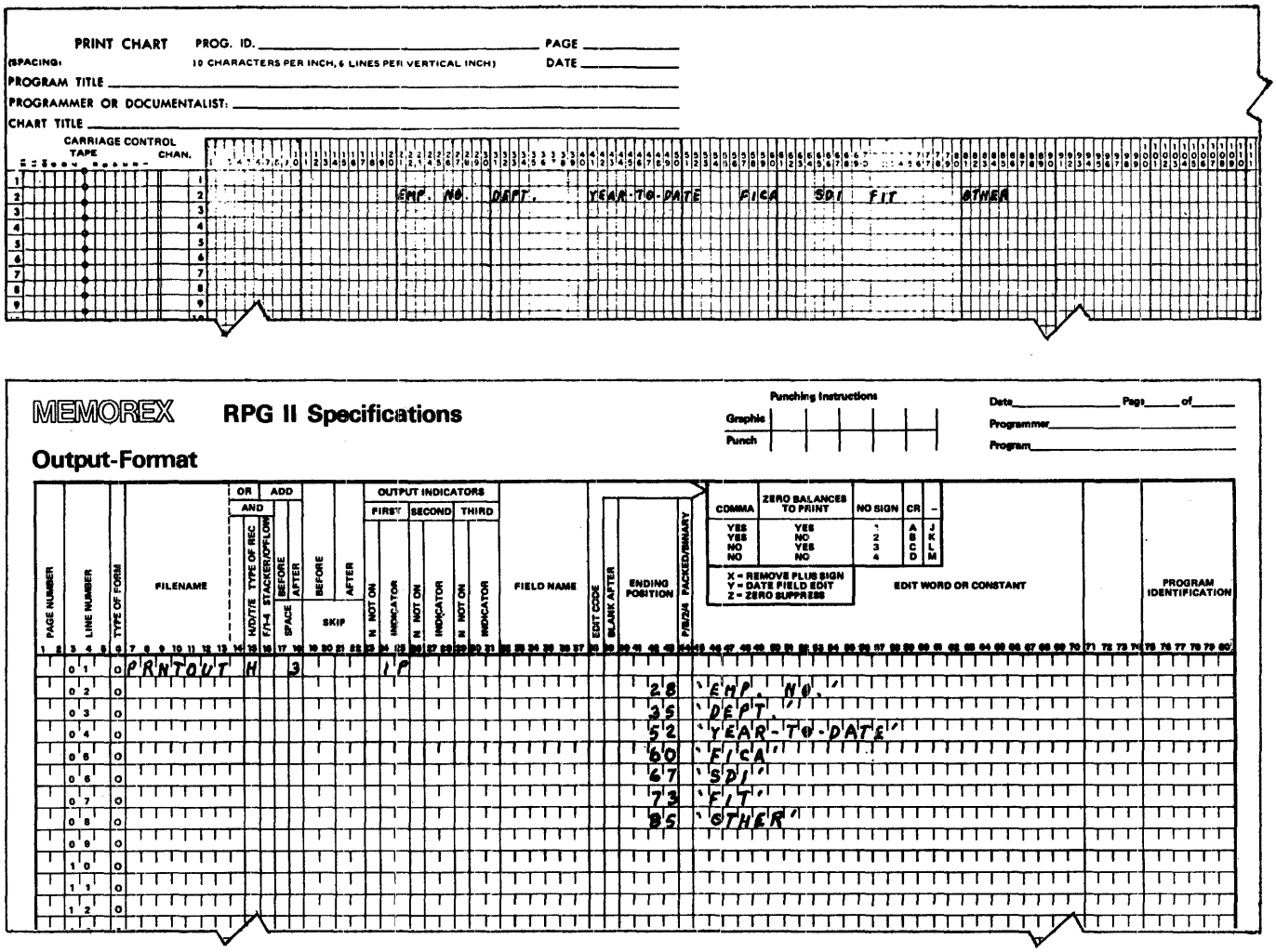


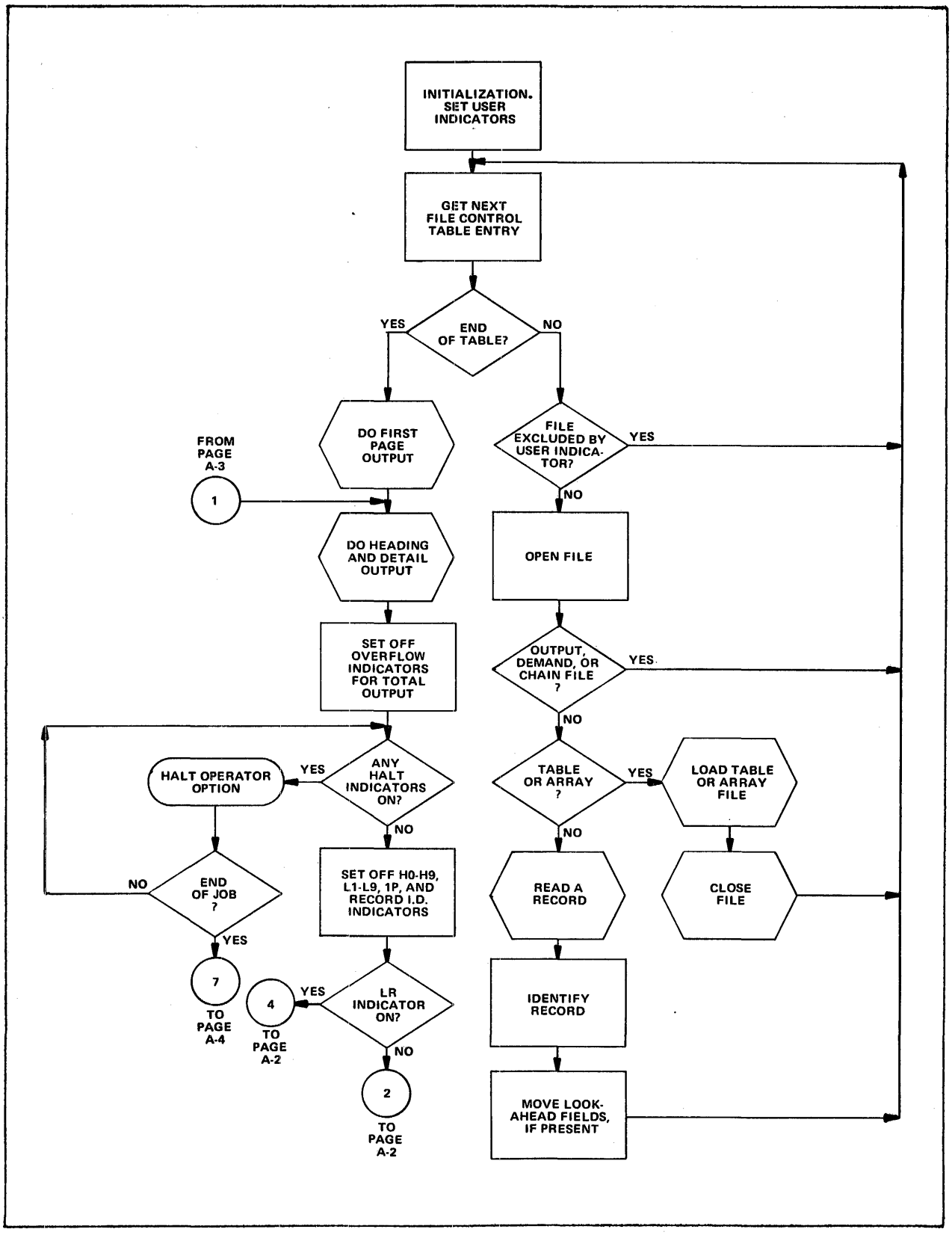
Figure 8-4. Examples of Constant Information.

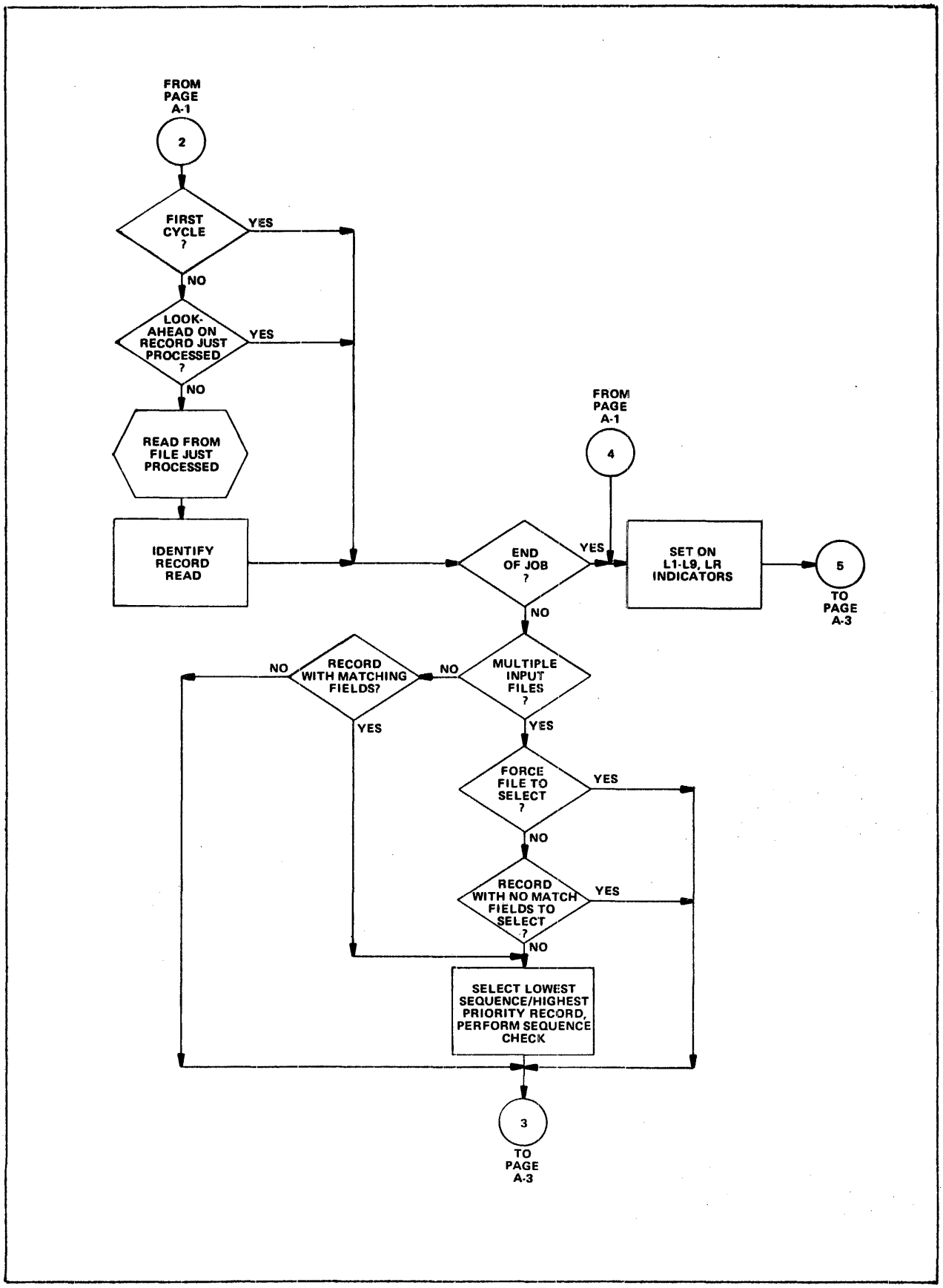
PROGRAM IDENTIFICATION (Columns 75-80)

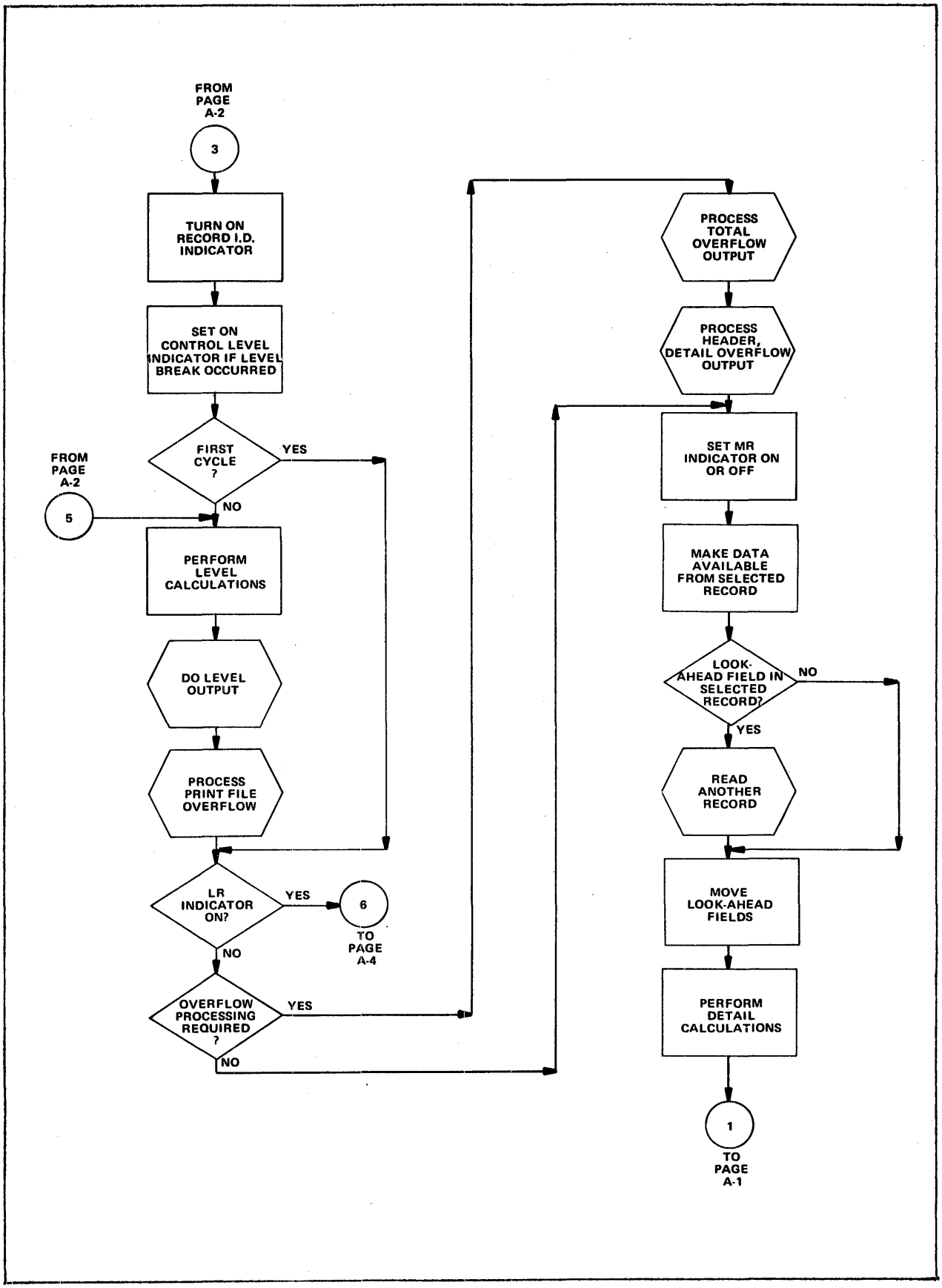
Entry	Description
Any characters	These columns may be used for programmer comments.

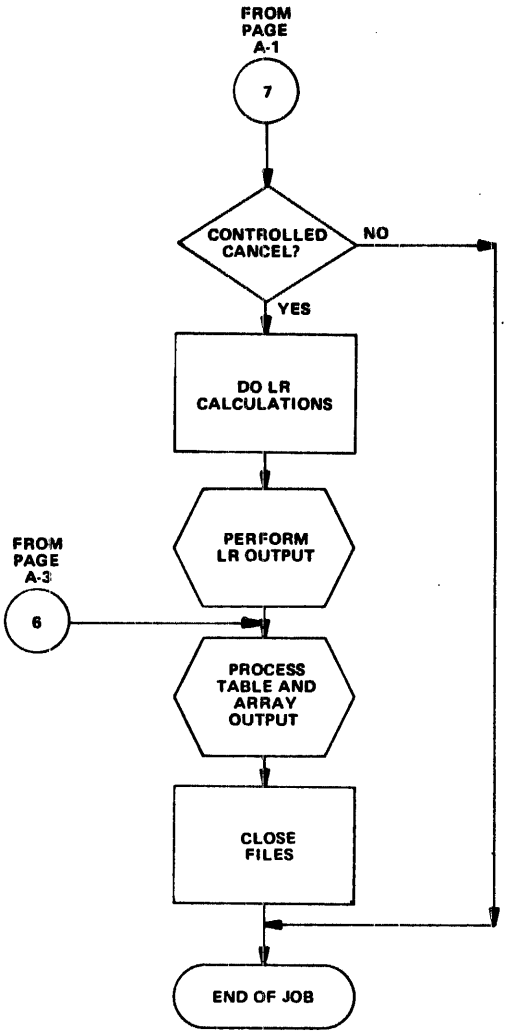
The program identification entry of all cards following the control card may contain any characters, including blanks. The columns may be used for programmer comments, if desired.

APPENDIX A. DETAILED RPG II OBJECT PROGRAM CYCLE




FROM PAGE A-1
2
FIRST CYCLE ?
YES
NO
LOOK-AHEAD ON RECORD JUST PROCESSED ?
YES
NO
READ FROM FILE JUST PROCESSED
IDENTIFY RECORD READ
FROM PAGE A-1
4
END OF JOB ?
YES
NO
SET ON L1-L9, LR INDICATORS
5
TO PAGE A-3
MULTIPLE INPUT FILES ?
NO
YES
RECORD WITH MATCHING FIELDS?
NO
YES
FORCE FILE TO SELECT ?
YES
NO
RECORD WITH NO MATCH FIELDS TO SELECT ?
YES
NO
SELECT LOWEST SEQUENCE/HIGHEST PRIORITY RECORD, PERFORM SEQUENCE CHECK
3
TO PAGE A-3


FROM PAGE A-2
3
TURN ON RECORD I.D. INDICATOR
SET ON CONTROL LEVEL INDICATOR IF LEVEL BREAK OCCURRED
FIRST CYCLE ?
YES
NO
FROM PAGE A-2
5
PERFORM LEVEL CALCULATIONS
DO LEVEL OUTPUT
PROCESS PRINT FILE OVERFLOW
LR INDICATOR ON?
YES
6
TO PAGE A-4
NO
OVERFLOW PROCESSING REQUIRED ?
YES
NO
PROCESS TOTAL OVERFLOW OUTPUT
PROCESS HEADER, DETAIL OVERFLOW OUTPUT
SET MR INDICATOR ON OR OFF
MAKE DATA AVAILABLE FROM SELECTED RECORD
LOOK-AHEAD FIELD IN SELECTED RECORD?
NO
YES
READ ANOTHER RECORD
MOVE LOOK-AHEAD FIELDS
PERFORM DETAIL CALCULATIONS
1
TO PAGE A-1


FROM PAGE A-1
7
CONTROLLED CANCEL?
NO
YES
DO LR CALCULATIONS
PERFORM LR OUTPUT
FROM PAGE A-3
6
PROCESS TABLE AND ARRAY OUTPUT
CLOSE FILES
END OF JOB

APPENDIX B. RPG II REFERENCE TABLES

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

Table B1. RPG II Collating Sequence.

	File Description Specifications		Input Specifications				Calculation Specifications			Output-Format Specifications
Indicators	Overflow Indicators (33-34)	File Conditioning (71-72)	[1]Record Identifying Indicator (19-20)	Control Level (59-60)	[1]Field Record Relation (63-64)	Field Indicator (65-70)	Control Level Indicator (7-8)	Operation Indicator (9-17)	Resulting Indicator (54-59)	Conditioning Indicator (23-31)
01-99			X		X	X		X	X	X
H0			X			X		X	X	X
H1-H9			X		X	X		X	X	X
1P						X		X[6]		X[3]
MR					X[2]	X		X	X	X
OA-OG,OV	X					X		X	X	X[4]
L0							X	X	X	X
L1-L9			X	X	X[2]	X	X	X	X	X
LR			X			X	X	X	X	X
U1-U8		X[5]			X	X		X	X	X

X	Denotes the indicator that may be used.
1	Not valid on look-ahead fields.
2	When field named is not a match or control field.
3	Only for detail or heading lines.
4	Cannot condition an exception line, but may condition fields within the exception record.
5	Not valid for table input files.
6	Normally off, even the first time through the object program cycle, unless it is purposely set on.

Table B2. Valid Indicators.

Table B3. Summary of Indicator Specifications (Part 1 of 3).

TYPE OF INDICATOR	ENTRY	WHERE SPECIFIED	SPECIFICATION	COLS.	WHERE USED	SPECIFICATION	COLS.	TURNED ON	TURNED OFF	NOTES
Level Zero Internal	LO	Calculation Sheet	RESULTING INDICATORS	54-59	Calculations Output	CONTROL LEVEL OPERATION INDICATORS OUTPUT INDICATORS	7-8 9-17 23-31	At the beginning of the program cycle.	By calculations specifications.	
Control Level	L1-L9	Input Sheet Calculation Sheet	RECORD INDICATOR CONTROL LEVEL FIELD INDICATORS RESULTING INDICATORS	19-20 59-60 65-70 54-59	Input Calculations Output	FIELD RECORD RELATION INDICATORS CONTROL LEVEL OPERATION INDICATORS OUTPUT INDICATORS	63-64 7-8 9-17 23-31	When the value in a control field changes (all indicators of the lower levels are also turned on) or individually by input or calculations specifications.	At the end of the following detail cycle, or by calculation specifications.	1
End-of-Job Internal	LR	Input Sheet Calculation Sheet	FIELD INDICATORS RECORD INDICATOR RESULTING INDICATORS	65-70 71-72 54-59	Calculations Output	CONTROL LEVEL OPERATION INDICATORS OUTPUT INDICATORS	7-8 9-17 23-31	After processing the last record of the last form (see column 17 of the file description specifications).	At the beginning of processing.	1 (can not be SETOF) 2
Matching Record Internal	MR	Input Sheet Calculation Sheet	FIELD RECORD RELATION INDICATORS FIELD INDICATORS RESULTING INDICATORS	63-64 65-70 54-59	Calculations Output	OPERATION INDICATORS OUTPUT INDICATORS	9-17 23-31	If the matching field contents of the record of a secondary file match the match-field contents of a record in the primary file; or by calculations specifications.	When all total calculations and output are completed for the last record of the matching group; or by calculations specifications.	
Overflow Internal	OA,OB OC,OD OE,OF OG,OV	File Description Sheet Input Sheet Calculation Sheet	OVERFLOW INDICATOR FIELD INDICATORS RESULTING INDICATORS	33-34 65-70 54-59	Calculations Output	OPERATION INDICATORS OUTPUT INDICATORS	9-17 23-31	If the destination of a skip, space, or print operation falls within the forms overflow area.	At the end of the detail cycle.	3

Table B3. Summary of Indicator Specifications (Part 2 of 3).

TYPE OF INDICATOR	ENTRY	WHERE SPECIFIED	SPECIFICATION	COLS.	WHERE USED	SPECIFICATION	COLS.	TURNED ON	TURNED OFF	NOTES
Field Zero & Blank; Plus; Minus	01-99	Input Sheet	FIELD INDICATORS	65-70	Calculations Output	OPERATION INDICATORS OUTPUT INDICATORS	9-17 23-31	By blank or zero in specified field; by plus in specified field; by minus in specified field.	Before this field status is to be tested the next time.	1
Record Identifying	01-99	Input Sheet	RECORD INDICATOR	19-20	Input Calculations Output	FIELD RECORD RELATION INDICATORS OPERATION INDICATORS OUTPUT INDICATORS	63-64 9-17 23-31	When specified record has been read and before total calculations are executed.	Before the next record is read during the next processing cycle.	1
Resulting Plus Minus Zero Compare Operation High Low Equal Look-up Operation High Low Equal Testz Operation High Low Equal	01-99	Calculation Sheet	RESULTING INDICATORS	54-59	Calculations Output	OPERATION INDICATORS OUTPUT INDICATORS	9-17 23-31	By a positive balance in the field, by a negative balance in the field, by a zero balance in the field. If factor 1 > fac.2 If factor 1 < fac.2 If factor 1 = fac.2 If table > factor 1 If table < factor 1 If table = factor 1 If C zone present. If D zone present. If no C or D zone; by a no-record-found condition.	The next time a calculation is performed for which the program specifies the indicator as a resulting indicator and the condition is not satisfied.	1
First Page Internal	1P	Calculation Sheet	OPERATION INDICATORS	9-17	Output	OUTPUT INDICATORS	23-31	At the beginning of processing, before any input records are read.	Before the first detail record is read.	4
Halt	H0-H9	Input Sheet Calculation Sheet	RECORD INDICATOR FIELD INDICATORS RESULTING INDICATORS	19-20 65-70 54-59	Input Calculations Output	FIELD RECORD RELATION INDICATORS OPERATION INDICATORS OUTPUT INDICATORS	63-64 9-17 23-31	Whenever the specified field status or record identification condition is satisfied.	Internally, at the end of the detail cycle.	1

TYPE OF INDICATOR	ENTRY	WHERE SPECIFIED	SPECIFICATION	COLS.	WHERE USED	SPECIFICATION	COLS.	TURNED ON	TURNED OFF	NOTES
External	U1-U8	File Description Sheet Input Sheet Calculation Sheet	FILE CONDITION FIELD INDICATORS RESULTING INDICATORS	71-72 65-70 54-59	Input Calculations Output	FIELD RECORD RELATION INDICATORS RESULTING INDICATORS OUTPUT INDICATORS	63-64 54-59 23-31	Prior to object program execution by a MRX/OS control language parameter.	By input or calculations specifications.	

Note 1. Turning indicators on or off can also be accomplished by using SETON and SETOF operation codes.
Note 2. All control level indicators (L1-L9) are also turned on when LR is turned on.
Note 3. The overflow indicator remains on during the following detail calculations and output cycles.
Note 4. This indicator is used to condition printing of the first page of the report.

Table B3. Summary of Indicator Specifications (Part 3 of 3).

COMMAND	FACTOR 1								FACTOR 2								RESULT							
	LIT	FLD	ARR	TAG	SUB	FLN	ALB	NUM	LIT	FLD	ARR	TAG	SUB	FLN	ALE	NUM	LIT	FLD	ARR	TAG	SUB	FLN	ALB	NUM
ADD	X	X	X	-	-	-	-	X	X	X	X	-	-	-	-	X	-	X	X	-	-	-	-	X
Z-ADD	-	-	-	-	-	-	-	-	X	X	X	-	-	-	-	X	-	X	X	-	-	-	-	X
SUB	X	X	X	-	-	-	-	X	X	X	X	-	-	-	-	X	-	X	X	-	-	-	-	X
Z-SUB	-	-	-	-	-	-	-	-	X	X	-	-	-	-	-	X	-	X	X	-	-	-	-	X
MULT	X	X	X	-	-	-	-	X	X	X	X	-	-	-	-	X	-	X	X	-	-	-	-	X
DIV	X	X	X	-	-	-	-	X	X	X	X	-	-	-	-	X	-	X	X	-	-	-	-	X
MVR	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	-	-	-	-	-	X
SQRT	-	-	-	-	-	-	-	-	X	X	X	-	-	-	-	X	-	X	X	-	-	-	-	X
XFOOT	-	-	-	-	-	-	-	-	-	-	X	-	-	-	-	X	-	X	-	-	-	-	-	X
MOVE	-	-	-	-	-	-	-	-	X	X	X	-	-	-	X	X	-	X	X	-	-	-	X	X
MOVEL	-	-	-	-	-	-	-	-	X	X	X	-	-	-	X	X	-	X	X	-	-	-	X	X
MHHZO	-	-	-	-	-	-	-	-	X	X	X	-	-	-	X	-	-	X	X	-	-	-	X	-
MHLZO	-	-	-	-	-	-	-	-	X	X	X	-	-	-	X	-	-	X	X	-	-	-	X	X
MLLZO	-	-	-	-	-	-	-	-	X	X	X	-	-	-	X	X	-	X	X	-	-	-	X	X
MLHZO	-	-	-	-	-	-	-	-	X	X	X	-	-	-	X	X	-	X	X	-	-	-	X	-
COMP	X	X	-	-	-	-	X	X	X	X	-	-	-	-	X	X	-	-	-	-	-	-	-	-
TESTZ	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	-	-	-	-	X	-
BITON	-	-	-	-	-	-	-	-	X	X	-	-	-	-	X	-	-	X	-	-	-	-	X	-
BITOF	-	-	-	-	-	-	-	-	X	X	-	-	-	-	X	-	-	X	-	-	-	-	X	-
TESTB	-	-	-	-	-	-	-	-	X	X	-	-	-	-	X	-	-	X	-	-	-	-	X	-
SETON	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X
SETOF	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
GOTO	-	-	-	-	-	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-	-	-	-	-

Table B4. Valid Operation Codes (Part 1 of 2).

COMMAND	FACTOR 1								FACTOR 2								RESULT							
	LIT	FLD	ARR	TAG	SUB	FLN	ALB	NUM	LIT	FLD	ARR	TAG	SUB	FLN	ALB	NUM	LIT	FLD	ARR	TAG	SUB	FLN	ALB	NUM
TAG	-	-	-	X	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
LOKUP	X	X	-	-	-	-	X	X	-	-	X	-	-	-	X	X	-	-	X	-	-	-	X	X
BEGSR	-	-	-	-	X	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ENDSR	-	-	-	X	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
EXSR	-	-	-	-	-	-	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-	-	-	-
EXCPT	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FORCE	-	-	-	-	-	-	-	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-	-	-
DSPLY	X	X	-	-	-	-	X	X	-	-	-	-	-	X	-	-	X	X	-	-	-	-	X	X
READ	-	-	-	-	-	-	-	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-	-	-
CHAIN	X	X	-	-	-	-	X	X	-	-	-	-	-	X	-	-	-	-	-	-	-	-	-	-
DEBUG	X	X	-	-	-	-	X	X	-	-	-	-	-	X	-	-	-	X	X	-	-	-	X	X
EXIT	-	-	-	-	-	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-	-	-	-	-
RLABL	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	-	-	-	-	-
ULABL	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	-	-	-	-	-

LIT = Literal.
FLD = Field (or array item, or table item).
ARR = Array.
TAG = Tag.
SUB = Subroutine name.
FLN = Filename.
ALB = Alphabetic.
NUM = Numeric.

Table B4. Valid Operation Codes (Parts 2 of 2).

Table B5. Summary of Edit Codes.

Edit Code	Commas	Decimal Point	Sign for Negative Balance			Print-Out On Zero Balance*			Zero Suppress
			No Sign	CR	- (Minus)	Domestic And United Kingdom	European Convention - I	European Convention - 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**									
Y***									Yes
Z									Yes

* Zero balances for the European Convention format are written in two ways, depending on the entry made in column 21 of the control card specifications.

** The X code performs no editing.

*** 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

If a data field of six digits is packed on disc and the Y edit code is used with the data field, an error will occur. The problem is solved by moving the data field to another field.

APPENDIX C. GLOSSARY OF TERMS

ADDROUT file
A record address file which is written out by the utility sort program.

alphabetic characters
The 26 alphabetic EBCDIC characters and three other EBCDIC characters: #, $, and @.

alphanumeric characters
Any of the 256 EBCDIC characters.

alphanumeric fields
All fields for which a decimal-positions specification has not been made in the appropriate column of the specifications form. Alphanumeric fields can contain any of the 256 EBCDIC characters.

alternate collating sequence
A character sequence in which the relative position of some or all alphanumeric characters in relation to the other characters is changed from the normal sequence used by the computer.

alternating format
A method of describing two tables or two arrays which are used together, so that each item in the first table or array is associated with a corresponding item in the second table or array which gives additional information.

alternating table or array
The second of two related tables or arrays.

array
A group of data items arranged systematically for ready reference.

array file
A sequential input file which contains array entries specified by the programmer and is processed before program execution or during program compilation.

asterisk fill
Asterisks printed in place of leading zeros.

binary format
A format in twos complement form wherein each field on disc must be either two or four bytes long. Two-byte fields contain a one-bit sign followed by a 15-bit numeric value and can contain up to five decimal numbers. Four-byte fields contain a one-bit sign followed by a 31-bit numeric value and can contain up to ten decimal numbers.

binary search
A time-saving method of searching through a large table in ascending or descending sequence.

bit
A single binary digit in a binary number, represented by 0 if it is negative, or 1 if it is positive.

branching
Skipping backward or forward to an instruction other than the next one in the sequence.

byte
A sequence of eight adjacent binary digits (bits) operated upon as a unit.

chained file
A disc file that is read randomly or loaded directly via the CHAIN operation code.

character structure
The unique combination of zone and digit punches in a card that represent a specific alphanumeric character.

Code Formatter Phase
The phase of the compiler in which error messages, the source program, a memory map, external references, and entry points are listed and the object program is written out in relocatable format.

Code Generation Phase
The phase of the compiler in which the source program is translated into machine language instructions and compile time tables and arrays are processed.

collating sequence
The order in the computer in which each alphabetic, numeric, or special character is related to all others (see Appendix B, Table B1).

compile time table or array
A table or array loaded at compilation time and compiled along with the source program.

constant
Any unchanging information, such as a column header or title, that is entered by a specification. Constants are enclosed in apostrophes.

control break
A change in the control field information of two records which causes the control level indicator(s) to turn on and the operations conditioned by the indicator(s) to be performed before a new record is processed.

control card
The initial card in the RPG II source program deck. The card provides the compiler with the control information necessary to translate the source program into an object program.

control field
A field to which a control level indicator (L1-L9) has been assigned to indicate the point at which specified operations are to be performed.

control group
A group of records which all have the same information in the control field.

control level indicator
An indicator which is turned on to indicate that a change in a control field has occurred and operations are to be performed or information is to be printed.

Cross-Reference Phase
The optional phase of the compiler in which cross-reference lists of file names, field names, and indicators are generated.

demand file
A file from which selected input or update information can be sequentially read for calculations, via the READ operation code, as required.

detail calculations
The segment of the object program operating cycle in which data from a selected record is made available for processing and the matching record (MR) indicator is turned on or off.

detail operations
The first segment of the object program operating cycle, in which heading and detail records are written and selected program indicators are set off.

detail record
An output record containing information that comprises the subject matter of the report.

device
The input or output hardware used for a particular file.

display file
A file composed of information gathered from various fields used by a program. Through use of the DSPLY operation code, fields or records in a display file can be printed directly from storage and data can be keyed into a display field or record in storage. The console is the only device that can be associated with a display file.

domestic date format
MM/DD/YY, where the numerics representing the month (M), the day (D), and the year (Y) are separated by slashes.

EBCDIC
Extended Binary-Coded-Decimal Interchange Code.

EBCDIC notation
The 256-character machine language code used in the MEMOREX System (see Appendix B, Table B1).

edit code
A one-character code which causes a specific type of punctuation editing to be carried out on a numeric field.

editing
Enhancing the legibility of a field by inserting special characters, including spaces, before or between the characters of a field.

edit word
The explicit character-by-character layout of an entire resulting output field, using replaceable and non-replaceable characters. Edit words are direct specifications which indicate whether or not specific punctuation characters are needed.

European Convention date format
DD.MM.YY, where the numerics representing the day (D), month (M), and year (Y) are separated by periods.

exception record
A record which is written during calculation time.

executive routine
The portion of the compiler program which controls execution of the compiler program. The executive routine contains input/output interfaces, the table lookup routines, and variables common to all phases of compiler processing.

external calculation operation
An operation performed by a subroutine or program that has been compiled separately from the main RPG II program.

external characters
The characters used in input and output data.

fetching the overflow routine
Requesting the overflow routine ahead of time to print several detail and total lines on each regular program cycle.

field indicator
An indicator used to test a named input field for a plus, minus, zero, or blank condition, or to halt a program so that an error condition in the data can be checked.

field name
A unique name which identifies a particular field, array, or array item in an input record.

field record relation indicator
An indicator whose primary purpose, among a variety of specific purposes, is to relate a field name to a particular record type.

file
A collection of related records treated as a single unit.

file designation
A method of describing the input, output, and update file types still further. The six designations are primary, secondary, chained, demand, record address, and table and array.

filename
A unique name assigned to every file a program uses except compile time table and array files.

file organization
The arrangement of records within a file. The three types of organization are indexed, relative, and sequential.

file translation
The translation of any character code into another, such as the translation of an input character code to an internal machine code.

file types
The four categories into which all RPG II files fall: input, output, update, and display.

fixed dollar sign
A dollar sign which always has the same position in an edited field.

fixed-length
The same length. All records in an indexed file must be fixed-length.

floating dollar sign
A dollar sign inserted before the most significant non-zero digit in a field.

full table or array
A table or array in which all possible items contain data and nothing more can be added.

half-adjusting
Rounding, by adding a 5 to the number at the right of the last decimal position specified for a positive field, or -5 if the field is negative, then dropping all decimal positions to the right of the position specified for that field.

halt indicator
An indicator which turns on to indicate that the program is to be stopped and an error message written for an error condition.

heading record
An output record containing identifying information which does not usually change from page to page, such as the report name, the date, and the column titles.

hexadecimal notation
A number system written to the base-16 notation (see Appendix B, Table B1).

housekeeping tasks
Tasks performed prior to System entrance to the object program operating cycle, in which all files to be used in the program are opened, pre-execution time tables and arrays are loaded, and the first page is output.

index
The portion of an indexed file which contains a key for each fixed-length record and the record's location within the file.

indexed file
A disc file whose record locations are kept in a separate portion of the file called an index.

indexing
A method of relative referencing by which an array item may be referenced by its location in the array relative to the first item. Indexing may not be used with tables.

input
All data files, records, and record fields used by the object program.

input file
The file from which a program reads its source data.

input operations
The segment of the object program operating cycle in which one input record is selected for processing, read, and identified.

internal calculation operation
An operation performed in the computer within the processing of a single RPG II program.

internal characters
Characters used for processing within the computer.

inverted print option
An option used to determine the punctuation and format used for numeric literals and the date field, and the edit codes used on output.

item
A single element in a table or array which is coded as an entry in the specifications sheets. An item cannot extend to more than one record.

last record indicator
An indicator which is turned on to indicate that a job is to terminate.

literal
The directly specified data used in an operation. It may be either alphanumeric or numeric.

look-ahead field
A feature which enables the System to look at information on the next record available for processing in an input record, or on the record currently being processed in an update record.

machine instruction
A code element, recognizable by a particular machine, that causes a predefined sequence of operations to be executed by the machine.

machine language
The set of characters, symbols, or signs, and the rules for combining them, that directly conveys instructions or information to a computer.

match fields
Selected record fields in the primary file which are compared to fields from one or more secondary input or update files for a matching record. Records can be matched by comparing one to nine fields.

matching record indicator
An indicator which turns on when a selected record field in the primary file matches a field in one or more secondary file records. The indicator is used to condition calculation or output operations for a selected record with a match field.

multifile processing
The selection of records for processing from a primary file and one or more secondary files.

non-replaceable characters
Those characters of an edit word which will remain constant in the edited field or will be replaced by blanks. All characters except ϕ, 0, *, and $ are non-replaceable.

non-resident program
A program or routine which is read in from an external library, rather than stored in memory.

numeric characters
The EBCDIC characters 0-9.

numeric fields
All fields having a decimal-positions specification in the appropriate columns of the specifications forms.

object program
The set of machine language instructions which result from the compiler program's translation of the programmer's RPG specifications. The object program contains all the machine instructions necessary to manipulate information from the data input files and produce output files and reports.

off indicator
An indicator which reflects a negative condition (is set to 00).

on indicator
An indicator which reflects a positive condition (is set to F0).

output
The result or product of processing.

output file
A file containing the records that a program writes, punches, or prints.

output record
A record containing data that has been gathered, sorted, calculated, or generated during object program operation.

overflow indicator
An indicator which turns on to signal that the end of a page is near.

overflow line
The point on a page or form at which overflow occurs and after which detail or total printing starts.

overlaying
Reading input data in on top of data already in storage. The new data replaces only the amount of existing data that is overlayed.

packed decimal format
A format in which each byte of computer or disc storage is divided into two four-bit digit portions and no zone is used.

PAGE, PAGE1, PAGE2
Special words used to cause page numbering on output to start with a number other than 1, or to differentiate between page numbering systems for one to three different output files. PAGE is a four--position numeric field with no decimal positions.

*PLACE
A special word which causes output fields to be repeated without respecifying the field names.

pre-execution time table or array
A table or array loaded with the object program before any input files are read, calculations are done, or output functions are performed.

primary file
The main file from which a program reads records. The program uses the primary file during multifile processing to control the order in which the records are selected.

program cycle
The cycle of operations through which one record must go to be completely processed.

record address file
An input file that indicates which records are to be read from a disc file, and the order in which those records are to be read.

record code characters
Character codes used to identify each type of record for processing various operations.

record identifying indicator
An indicator which is turned on to distinguish between record types or to indicate a record type that causes an error condition.

record key
The key field of each record, which is used in the index portion of an indexed file to identify the record.

related tables or arrays
Two tables or two arrays which are used together, so that each item in the first table or array is associated with a corresponding item in the second table or array which gives additional information.

relative file
A disc file whose records are called by relative record number, rather than by the sequence in which they were loaded.

relative record number
A number assigned to each record in a relative file to indicate the record's position in the file in relation to the other records.

replaceable characters
Those positions of an edit word which will be occupied by digits of the edited field. The replaceable characters are ø, 0, *, and $.

resident program
A program or routine which is stored in computer memory at all times, rather than read in from a library.

resulting indicator
An indicator that is turned on according to the result of a calculation operation.

search word
An alphanumeric or numeric constant, an array item, a field name, or a table name for which a match is to be found in a table or array that is being searched.

secondary file
All files involved in multifile processing except the primary file.

sequential file
A sequential file in which records are ordered in the same sequence in which they are put into the file.

short table or array
A table or array that is not full.

source program
All the symbolic programming language entries coded by the programmer on a specific set of RPG II specifications sheets. The source program is then punched into cards to form the source program deck, which is in turn translated by the compiler into machine instructions for generating output files and reports.

special device
An I/O device, other than the usual one used, which has been selected by the programmer and linked to RPG II by a user-written routine that enables data to be transferred for the device. See Section 3, "Special Device Support," for rules governing specifications sheet entries for the special device feature.

special file
A file on a special I/O device linked to RPG II by a user-written routine that enables data transferred to or from that device. Special files can only be processed consecutively. See Section 3, "Special Device Support," for the specifications rules for special files.

special word
A word used as a field name to cause a particular action to be performed. The special words are PAGE, PAGE1, PAGE2 for page numbering; *PLACE for iterated fields; and UDATE, UDAY, UMONTH, UYEAR for the date format.

split control field
A control field which is composed of two or more connected or unconnected fields on a record.

spread cards
A feature which allows one or more transactions to be linked to a single header on the same card.

standard I/O devices
The disc unit, the card punch, the card reader, the console, the printer, and magnetic tape.

Syntax Phase
The phase of the compiler in which all the input cards are checked for accuracy.

table
A group of data items arranged systematically for ready reference.

table file
A sequential input file, containing table entries specified by the programmer, which is processed before program execution or during program compilation.

Table Overflow Phase
The phase of the compiler in which field names not resolved in the syntax phase are finally resolved.

total calculations
The calculation segment of the object program operating cycle invoked if a level break occurs.

total output operations
The segment of the object program operating cycle in which all total records are written and end-of-job processing is initiated if a level break has occurred.

total record
An output record written out when a level break occurs that contains cumulative sums of the fields in detail records.

UDATE
A special word whose specification produces a six-character numeric date in either domestic (mmddyy) or United Kingdom/European Convention (ddmmyy) format.

UDAY
A special word which may be used to obtain dates consisting of days only.

UMONTH
A special word which may be used to obtain dates consisting of months only.

United Kingdom date format
DD/MM/YY, where the numerics representing the day (D), month (M), and year (Y) are separated by slashes.

unpacked decimal format
A format in which each byte of computer or disc storage holds one eight-bit character comprising a four-bit zone portion followed by a four-bit digit portion.

update file
A disc file from which a program reads a record, updates its fields, and rewrites the record in the location from which it was read.

UYEAR
A special word which may be used to obtain dates consisting of years only.

APPENDIX D. RPG II ASSUMED RECORD AND DATA BLOCK LENGTHS

If no LENGTH OF RECORD entry is made in columns 24-27 of the File Description Specifications Sheet, a warning message is issued and the compiler assumes the length to be the maximum record length allowed for the file device. The maximum record lengths allowed for the various devices are:

DISC	4096
PUNCH	80
TAPE	4096
READ	80
CONSOLE	120
PRINTER	132

If no LENGTH OF BLOCK entry is made in columns 20-23 of the File Description Specifications Sheet and the file device is other than disc or tape, the compiler assumes the length to be equal to that specified in the LENGTH OF RECORD entry, columns 24-27.

If the device is disc or tape, and the block length is blank or equal to the record length, the compiler computes an integer blocking factor according to the following formula:

blocking factor = 256/(record length + 4)

The block length is then computed as follows:

block length = blocking factor X (record length + 4)

The block and record lengths actually used by the compiler may differ from the entered or computed values for files assigned to printer, punch, or card devices. The compiler automatically adds 1 to the block and record lengths of printer and punch files for carriage and stacker control, respectively. If a file is spooled, the compiler also adds 4 to the block length for spooling record headers.

APPENDIX E. RPG II SUBROUTINE LINKAGE

Since standard system linkage is used in RPG II, an RPG II program can call a COBOL, FORTRAN, or assembly language program. Because RPG II numerical data is always in packed decimal format and FORTRAN can only be used with binary or floating-point numerical data, however, some conversion of data would be required by the user.

A COBOL, FORTRAN, or assembly language program cannot call RPG II programs because RPG II can never return to the calling program. For the same reason, an RPG II program may not be linked with another RPG II program--the results would be unpredictable.

Linkage to an external subroutine from RPG II can be accomplished by either of two methods: through the EXIT operation, or by defining a SPECIAL file.

Register six (6) holds the parameter list address used only for reading SPECIAL files. This list is not used by the EXIT operation.

The save area address will be set in register seven (7).

The return address will be in the first word of the save area.

The format of the save area (pointed to by register seven) is as follows:

Words	Contents
1	Return address.
2	Previous save area address (always zero).
3	Status word (zero).
4-11	Register save area.

SPECIAL FILE PARAMETER LIST

A parameter list pointed to by register 6 is for SPECIAL files only. The format of that list is as follows:

Words	Contents
1	Length of list (always 11).
2	Bits 0-7 (function code) = 0 GET = 1 PUT = 4 OPEN = 5 CLOSE Bit 12 = 1 (variable length records are specified, and word 7 must point to the record size). Bit 13 = 1 (end-of-file return address specified in word 9).
3	Error return code -- set by the external routine (if non-zero the job will be stopped and the error return code will be printed out).
4-6	Not used.
7	Record size address - points to a one-word (two-byte) location which will contain the record size (bit 12 of word 2 must be equal to 1). For GET, the record size address and record size are set by the external program. For PUT they are set by RPG II.
8	Not used.
9	End-of-file return address (always specified - bit 13 of word 2 will always be equal to 1).
10-11	Not used.
12	Address of record. For GET this address is supplied by the external routine. For PUT this address is supplied by the RPG II compiler.

APPENDIX F. CONTROL LANGUAGE STATEMENTS

REQUIREMENTS

To perform an RPG II compilation, an //EX (execute) Control Language Statement card with a PGM=RPG parameter is necessary. A minimum of four //DEF cards is then required to define the files, as follows:

Card	File
//DEF ID=$RGERRS,FIL=RPGERRMSG,STA=(P,I)	Error message file
//DEF ID=LIST	File to which source listing is to go
//DEF ID=INPUT	File from which source input program comes
//DEF ID=OUTPUT	Library to which relocatable object library is to be written

In addition to these specifications, any or all of the four work files may be assigned to specific volumes to increase compilation speed. If they are not, they are automatically assigned to the System's volume.

The ID's used for the four work files are:

ID=MRTEXT01
ID=MRTEXT02
ID=MRTEXT03
ID=MRSIFIL

The most efficient compilation is achieved by assigning the following ID's to different volumes:

MRTEXT01, MRSIFIL, and INPUT
MRTEXT02, MRSIFIL, OUTPUT, and LIST
MRTEXT01 and MRTEXT03

OUTPUT and INPUT may be on the same volume.

//PAR CARD PARAMETERS

The //PAR card is an optional CLS card available to the RPG II programmer for naming an input and/or output library member and indicating to the compiler the source input file's approximate size. This determines the size in the first allocation of the RPG II work files.

More than one //PAR card may be used. The parameters to be used on the card are IMEM=name, OMEM=name, and either NUM=nnnnn or NUMBER=nnnnn.

IMEM=name

This parameter specifies the name to be used for the input library member in the file defined on the //DEF ID=INPUT card. The parameter is required if and only if the source input is in a library. The name may be up to six characters long.

OMEM=name

This parameter specifies the name of the library member in which the relocatable object module is to be placed. The name is for the output library member in the file defined on the //DEF ID=OUTPUT card; it may be up to six characters long. Presence of the OMEM=name ensures that an RPG object program will be output, even though column 10 of the control card indicates otherwise. If the OMEM=name parameter is not used, the member name is taken from columns 75-80 of the RPG II control card (see Section 2); otherwise the name specified overrides the name in the control card. If a member name is not specified in either the parameter or the control card, the default name of RPGOBJ is used.

NUM=nnnnn/NUMBER=nnnnn

This parameter specifies the number of source statements to be compiled. The default value is 1000 source statements. If the value is incorrect, the program will still be compiled; however in large programs, the correct value should be used to ensure that the initial work file allocations will not require extension.

LINKAGE EDITOR CLS SPECIFICATIONS

The user must specify a linkage editor //DEF ID=INPUT card with the filename of the object library in which his RPG II program has been placed. He must also give the member name of the relocatable object module in a linkage editor //PAR PAR=name card.

He may either compile, linkedit, and execute his program all in one job, or he may perform each of the three steps separately, whichever is more convenient. If he chooses to do all three in one job step, he can condition his linkedit on successful compilation (no fatal errors) with an //IF Control Language Statement, written as follows:

```
//IF      CODE=F,GO=EOJ
```

An F error code (fatal compilation error) then allows him to bypass the linkedit and execute steps.

CLS REQUIRED FOR COMPILATION

The following examples show the CLS used in compiling an RPG program. Five common compilation situations were chosen for the examples. Whenever possible, default values are used for the RPG compiler parameters. Information peculiar to an individual program being compiled is underlined in the examples.

A complete description of MRX/OS CLS is provided in the Memorex publication Control Language Services—Extended.

EXAMPLE 1. CARD-TO-COMPILER COMPILATION

MEMOREX **Assembler Coding Form**

Punching Instructions — Graphic / Punch — Date / Prog / Prog

NAME	OPERATION	OPERAND
//JOB	NAM=RPGTEST1	
//EX	PGM=RPG	
//DEF	ID=$RGERRS,FIL=RPGERRMSG,STA=(P,I)	
//DEF	ID=LIST,DEV=PRINTER	
//DEF	ID=OUTPUT,FIL=TEMP-OBJ-LIB,SIZ=256,NUM=1500,BLK=1,CSD=NO	
//DEF	ID=INPUT,DEV=READER	
//DATA	FIL=CARD	
	source program	card deck
/*		
//EOJ		

Example 1 deals with compiling an RPG source program deck read directly from the card reader. The RPG compiler parameters (IMEM, OMEM, NUM) are assigned default values by the compiler. When the CLS shown in Example 1 is used, the following occurs:

- The compile listing is printed directly on the printer as specified by the 4th statement.
- The object module is placed in a temporary library (TEMP-OBJ-LIB) named and described in the 5th statement. The name of the library member is taken from the RPG control card in the absence of the OMEM parameter.
- The 6th and 7th statements indicate that the RPG source program is in card format and is to be read directly from the card-reading device.
- The source program is inserted in the CLS deck following the 7th statement.

EXAMPLE 2. CARD-TO-DISC COMPILATION

MEMOREX **Assembler Coding Form**

Punching Instructions — Graphic / Punch — Date / Prog / Prog

NAME	OPERATION	OPERAND
//JOB	NAM=RPGTEST2	
//EX	PGM=RPG	
//DEF	ID=$RGERR,S,FIL=RPGERRMSG,STA=(P,I)	
//DEF	ID=LIST,DEV=PRINTER	
//DEF	ID=OUTPUT,FIL=TEMP-OBJ-LIB,SIZ=256,NUM=1000,BLK=1,CSD=NO	
//DEF	ID=INPUT,FIL=RPG-SOURCE	
//DATA	FIL=RPG-SOURCE	
	source program card deck	
/*		
//EOJ		

Example 2 deals with compiling an RPG source program placed in the jobstream and spooled by the job monitor. The RPG compiler parameters (IMEM, OMEM, NUM) are assigned default values by the compiler. When the CLS shown in Example 2 is used, the following occurs:

- The compile listing is printed directly on the printer as specified in the 4th statement.
- The object module is placed in a temporary library (TEMP-OBJ-LIB) named and described in the 5th statement. The name of the library members is taken from the RPG control card in the absence of the OMEM parameter.
- The 6th and 7th statements specify that the source module is to be spooled from the jobstream into the file called RPG-SOURCE.
- The source program is inserted in the CSL deck following the 7th statement.

EXAMPLE 3. DISC COMPILATION USING DEFAULT VALUES

MEMOREX **Assembler Coding Form**

Punching Instructions — Graphic / Punch — Date / Prog / Prog

NAME	OPERATION	OPERAND
//JOB	NAM=RPGTEST3	
//EX	PGM=RPG	
//DEF	ID=$RGERRS,FIL=RPGERRMSG,STA=(P,I)	
//DEF	ID=LIST,DEV=PRINTER	
//DEF	ID=OUTPUT,FIL=TEMP-OBJ-LIB,SIZ=256,NUM=1000,BLK=1,CSD=NO	
//DEF	ID=INPUT,FIL=$SYSSRCLIB,STA=(P,I)	
//PAR	IMEM=CB231	
//EOJ		

Example 3 deals with compiling an RPG program previously stored on disc. The RPG compiler parameter IMEM is used to identify the source module to be compiled. The parameters OMEM and NUM are assigned default values by the compiler. When the CLS shown in Example 3 is used, the following occurs:

- The compile listing is printed directly on the printer as specified by the 4th statement.
- The object module is placed in a temporary library (TEMP-OBJ-LIB) named and described in the 5th statement.
- The 6th statement identifies the System source library ($SYSSRCLIB) from which the source program is read.
- The compiler parameter IMEM is used in the 7th statement to identify the source module to be compiled.
- The //EOJ card concludes the compilation deck. Since the source program was previously stored on disc, it is not included in the CLS deck.

EXAMPLE 4. DISC COMPILATION USING SPECIFIED PARAMETERS

MEMOREX **Assembler Coding Form**

Punching Instructions — Graphic Punch — Date — Prog — Prog

NAME	OPERATION	OPERAND
//JOB	NAM=RPGTEST4	
//EX	PGM=RPG	
//PAR	NUM=1500	
//DEF	ID=$RCERRS,FIL=RPGERRMSG,STA=(P,I)	
//DEF	ID=MRTEXT01,FIL=DUMMY,VOL=D115	
//DEF	ID=MRTEXT02,FIL=DUMMY,VOL=D037	
//DEF	ID=MRTEXT03,FIL=DUMMY,VOL=D037	
//DEF	ID=MRSIFIL,FIL=DUMMY,VOL=D037	
//DEF	ID=LIST,DEV=PRINTER	
//DEF	ID=OUTPUT,FIL=TEMP-OBJ-LIB,SIZ=256,NUM=2000,BLK=1,CSD=NO	
//PAR	OMEM=TSTMOD1	
//DEF	ID=INPUT,FIL=$SYSSRCLIB,STA=(P,I)	
//PAR	IMEM=CB231	
//EOJ		

Example 4 deals with compiling an RPG program previously stored on disc. All RPG parameters (IMEM, OMFN, NUM) are specified by the programmer rather than assigned default values by the compiler:

- The 3rd statement specifies the number of statements in the source program to be compiled.
- The 5th, 6th, 7th, and 8th statements are optional DFF statements used to place the RPG compiler work files on specific DASD volumes. If these statements are not included, the System assigns the volumes to be used for allocation.
- The compile listing is printed directly on the printer as specified by the 9th statement.
- The object module is placed in a temporary library (TEMP-OBJ-LIB) named and described in the 10th statement.
- The compiler parameter OMEM is used in the 11th statement to name the object module stored in the temporary library.

- The 12th statement identifies the System source library ($SYSSRCLIB) from which the source program is read.
- Compiler parameter IMEM in the 13th statement identifies the member name of the source module to be compiled.
- The //EOJ card concludes the compilation deck. Since the source program has been previously stored on disc, it is not included in the CLS deck.

EXAMPLE 5. COMPILE, LINKEDIT, AND EXECUTE

MEMOREX **Assembler Coding Form**

Punching Instructions — Graphic / Punch — Date / Prog / Prog

NAME	OPERATION	OPERAND
//JOB	NAM=RPGTEST5	
//EX	PGM=RPG	
//DEF	ID=$RGERR	S,FIL=RPGERRMSG,STA=(P,I)
//DEF	ID=LIST,DEV=PRINTER	
//DEF	ID=OUTPUT	,FIL=TEMP-OBJ-LIB,SIZ=256,NUM=1000,BLK=1,CSD=NO
//PAR	OMEM=TSTMOD	
//DEF	ID=INPUT,	FIL=RPG-SOURCE
//IF	CODE=F,GO=EOJ	
//EX	PGM=LNKEDT	
//PAR	LST=XREF	
//DEF	ID=LIST,DEV=PRINTER	
//DEF	ID=OUTPUT	,FIL=TEMP-LOAD-LIB,SIZ=256,NUM=1000,BLK=1,CSD=NO
//DEF	ID=INPUT,	FIL=TEMP-OBJ-LIB
//PAR	PGM=TSTMOD	
//IF	CODE=F,GO=EOJ	
//EX	PGM=TSTMOD	,LIB=TEMP-LOAD-LIB
//PAR	...	
//DEF	...	
//DATA	FIL=RPG-SOURCE	
	source program card deck	
/*		
//DATA	...	
	test data card deck	
/*		

MEMOREX **Assembler Coding Form**

Punching Instructions — Graphic / Punch — Dat / Pro / Pro

NAME	OPERATION	OPERAND
//EOJ		

Example 5 is a three-step job. It deals with compiling, linkediting, and executing an RPG program.

The 1st through the 8th statements make up the compilation step. RPG parameters IMEM and NUM are assigned default values. CLS specifies the following in Example 5:

- The compile listing is printed directly on the printer as specified by the 4th statement.

- The object module is placed in a temporary library (TEMP-OBJ-LIB) named and described in the 5th statement.
- The object module is given the library member name TSTMOD in the 6th statement.
- The 7th statement causes the source program to be spooled from the jobstream into the file called RPG-SOURCE.
- The 8th statement instructs the System to skip the link-edit and test-execute steps if the compile step fails.

The 9th through the 15th statements make up the linkedit step. They specify the following:

- The module output (linkedited program) is to be stored in a temporary library (TEMP LOAD LIB) named and described in the 12th statement.
- The 13th statement specifies that the object module to be linkedited is input from the temporary library allocated in the 5th statement.
- The 14th statement gives the name of the object module to be read as input to the link editor.
- The 15th statement instructs the System to skip the test-execute step if the linkedit step fails.

The 16th through the 25th statements make up the test-execute step and complete the job deck. They specify the following:

- The 16th statement names the compiled and linkedited program to be executed. The name given is the same as that used in the 14th statement.
- The filename specified on file description cards in the RPG source program must be used as the identification (ID=filename) on the //DEF card for object program execution.
- The CLS required for program execution is inserted in the job deck after the 16th statement (lines 17 and 18).
- The source program to be compiled is inserted immediately after the last Control Language Statement required by the test-execute step (lines 19 and 20).
- The data to be processed by the test program is inserted last in the job deck (lines 22 and 23).
- The entire deck is completed by an end-of-job (//FOJ) card.

APPENDIX G. ERROR MESSAGES

FORMAT

All error messages produced on the 132-character printer are preceded by a fourteen-character program-generated code consisting of a standard eight-character identification, a blank, a letter which indicates either a warning (W) or a fatal error (F), and four more blanks. A fatal (F) error indicates that the object code may not be executable; in such cases the line is ignored, and in the majority of cases the source deck should be corrected and recompiled. A warning (W) message indicates that the object code may produce invalid results, even though the code is executable. The compiler attempts to recover from errors producing warning messages and continues processing, whether or not recovery is successful.

Each message is printed immediately following the line on the source listing to which it applies. The format of the printed messages is as follows:

Characters	Contents
1-2	The letters RG.
3-4	The number of the overlay in which the error was encountered.
5-7	The error number within the overlay.
8	Zero (0). (Reserved for future use.)
9	Blank.
10	W (warning) or F (fatal error).
11-14	Blanks.
15-84	Error message text.

LIST OF MESSAGES BY OVERLAY NUMBER

The error messages on succeeding pages are arranged according to the overlay area in which the corresponding error is encountered. Error messages do not appear for some overlay areas because error-producing conditions do not arise. In the following overlay area list, those areas which do not produce error messages are indicated by an asterisk preceding the overlay number.

	Number	Overlay	Error Ranges
*	1	Compile Executive Program	
	2	Header Card Scan (Control Card)	RG020010-0310
	3	File Description Scan	RG030010-0900
	4	File Extension and Line Counter Scan	RG040010-0610
	6	Input Specifications Scan	RG060010-0910
	7	Calculation Specifications Scan	RG070010-0570
	8	Output Specifications Scan	RG080010-0640
*	9	Table Overflow	
*	10	Input/Output Generation	
	11	File Extension/Line Counter Generation	RG110010-0120
*	12	Record Handling Generator	
	13	Calculation Generator	RG130600-1000
	14	Output Generator	RG140010-0150
	15	Code Formatter	RG150010-0040

Error messages are listed sequentially by overlay number, beginning on the next page. Each successive overlay error message list begins a new page. Error code numbers are listed at the left, followed by F or W (fatal error or warning) and the text of the error message.

Variable fields within error messages are represented by the quantity of X's or N's required to satisfy the length of the field. X's represent alphanumeric fields and N's represent numeric fields.

(Execution-time error messages and compile-time diagnostic messages issued to SYSOUT are listed and described in Appendix H, "Operational Environment.")

OVERLAY 2. HEADER CARD SCAN (CONTROL CARD)

RG020010 F No source cards for RPG II program

RG020020 W No header card. Assume default specifications

RG020030 W Columns 7-9 should be blank. Assume blank

RG020040 W Invalid object output entry in column 10. Assume blank

RG020050 W Column 11 should be blank. Assume blank

RG020060 W Columns 16-20 should be blank. Assume blank

RG020070 W Inverted print entry in column 21 is not D,I,J or blank. Assume blank

RG020080 W Columns 22-25 should be blank. Assume blank

RG020090 W Collating seq. entry in column 26 is not S, T or blank. Assume blank

RG020100 W Columns 27-33 should be blank. Assume blank

RG020110 W Table lookup entry in column 34 is not B or blank. Assume blank

RG020120 W Columns 35-39 should be blank. Assume blank

RG020130 W Error deleted

RG020140 W Error deleted

RG020150 W Invalid sign process entry in column 40. Assume blank

RG020160 W Forms positioning entry in column 41 is not 1 or blank. Assume blank

RG020170 W Column 42 should be blank. Assume blank

RG020180 W File translation entry in column 43 is not F or blank. Assume blank

RG020190 W Columns 44-46 should be blank. Assume blank

RG020200 W Suppress skip to channel 1 (column 47) is not S or blank. Assume blank

RG020210 W Columns 48-51 should be blank. Assume blank

RG020220 W Debug entry in column 15 is not 1 or blank. Assume blank

RG020230 W Invalid core size needed to execute entry in cols 12-14. Assume blank

RG020240 W Error deleted

RG020250 W Error deleted

RG020260 W Invalid program identification entry in columns 75-80. Assume RPGOBJ

RG020270 W Cross reference list entry in col 52 is not X or blank. Assume blank

RG020280 W Carriage control type entry in col 53 is not L or blank. Assume blank

RG020290 W Ignore arithmetic overflow (col 55) is not T or blank. Assume blank

RG020300 W Columns 56-73 should be blank. Assume blank

RG020310 W Sequence check entry in column 54 is not N or blank. Assume blank

OVERLAY 3. FILE DESCRIPTION SCAN

RG030010 F Too many file description specs. lines used. Ignore extra specs.

RG030020 F Filename in columns 7-14 is missing or invalid

RG030030 F Filename has already been used. Ignore current definition

RG030040 F Device name in columns 40-46 is missing or invalid. Assume disc

RG030050 W Input card file may not be spooled if block and record lengths not 80

RG030060 W Dual printer carriage is not supported. Assume device is printer

RG030070 F File type in column 15 is not I, O, U, C or D. Assume default for device

RG030080 F File type in col 15 invalid for device. Assume type default for device

RG030090 W Invalid file designation (col 16). Assume desig. default for file type

RG030100 W File desig. invalid for device. Assume desig. default for file type

RG030110 W A primary file has already been designated. Assume secondary file

RG030120 F A record address file has already been defined. Ignore current file

RG030130 F More than 15 demand and/or chained files defined. Ignore those over 15

RG030140 W Processing mode entry in column 28 is not L, R or blank. Assume blank

RG030150 W Processing modes L, R valid for disc device only. Assume col 28 blank

RG030160 W Random processing mode invalid for demand file. Assume col 28 blank

RG030170 W R or L processing mode invalid for file desig. or device. Assume blank

RG030180 W Limits processing mode invalid for chained file. Assume random mode

RG030190 W Blank processing mode invalid for chained file. Assume random mode

RG030200 W End-of-file code in col 17 not E or blank. Assume E if valid for file

RG030210 W End-of-file code in column 17 is not E or blank. Assume blank

RG030220 W E end-of-file code invalid for file type and proc. mode. Assume blank

RG030230 W Sequence entry is not A, D or blank. Assume sequence of previous file

RG030240 W A or D sequence invalid for file type or designation. Assume blank

RG030250 W Length of record missing or invalid. Assume default length for device

RG030260 W Length of record invalid for device. Assume default length for device

RG030270 W Invalid length of block entry. Assume equal to length of record

RG030280 F Length of block should be a multiple of length of record + 4 for device

RG030290 W Block length should equal record length for this device. Assume equal

RG030300 W Invalid record address type in col 31. Assume A if mode=L, else blank

RG030310 W K entry in column 31 for record address type invalid. Assume type=A

RG030320 W A record address type invalid for non-disc device. Assume blank

RG030330 W Record address type invalid for file type and desig. Assume blank

RG030340 W Record address type invalid for limits processing mode. Assume type=A

RG030350 W Invalid file organization entry in column 32. Assume blank

RG030360 W Indexed file org. invalid for non-disc device. Assume column 32 blank

RG030370 W Limits mode invalid for a non-indexed file. Assume indexed (col 32=I)

RG030380 F ADDROUT file must be an input record address file. Assume col 32 blank

RG030390 W File org. entry invalid for type and desig. or device. Assume blank

RG030400 W File org. entry must be blank or 1-9 for non-disc device. Assume blank

RG030410 F File org. entry invalid for proc. mode or RA type file. Assume blank

RG030420 F Starting loc. of key field (cols 35-38) missing or invalid. Assume 1

RG030430 F Starting location of key field is 0 or exceeds record length. Assume 1

RG030440 W Length of RA field (cols 29-30) invalid for an ADDROUT file. Assume 4

RG030450 W Length of record in cols 24-27 must be 4 for an ADDROUT file. Assume 4

RG030460 F Length of key or RA field is missing, invalid or over 99. Assume 4

RG030470 F Length of key or RA field plus start loc. of key exceeds record length

RG030480 W Columns 29-30 should be blank with these specifications. Assume blank

RG030490 W Columns 35-38 should be blank with these specifications. Assume blank

RG030500 W Overflow indicator should be blank for non-printer file. Assume blank

RG030510 W Invalid overflow indicator entry in columns 33-34. Assume blank

RG030520 W Overflow indicator in columns 33-34 already defined. Assume blank

RG030530 W Extension code in column 39 is invalid or E for print file. Assume L

RG030540 W Extension code in column 39 is not E for table or RA file. Assume E

RG030550 W Extension code in column 39 invalid for device or desig. Assume blank

RG030560 W Hi-level directory valid for random indexed file only. Assume blank

RG030570 W Tape labels entry (col 53) invalid or invalid for device. Assume blank

RG030580 W Columns 54-59 should be blank. Assume blank

RG030590 F Error deleted

RG030600 W Tape labels entry should be blank for special device. Assume blank

RG030610 F Error deleted

RG030620 W Invalid index buffer size entry in columns 60-65. Assume blank

RG030630 W Inv. index buf. size value. Assume blank if random, sys default if seq.

RG030640 W File addition entry in column 66 is not A or blank. Assume blank

RG030650 W File addition should be blank for non-disc device. Assume blank

RG030660 W File addition should be blank for relative, RA or T file. Assume blank

RG030670 W Column 67 should be blank. Assume blank

RG030680 W Tape rewind entry (col 70) invalid or invalid for device. Assume blank

RG030690 W Invalid file condition entry in columns 71-72. Assume blank

RG030700 W File condition entry in cols 71-72 invalid for file type. Assume blank

RG030710 W Columns 73-74 should be blank. Assume blank

RG030720 W File format entry in column 19 missing or invalid. Assume F

RG030730 W File format entry in column 19 invalid for device. Assume F

RG030740 W Columns 68-69 should be blank. Assume blank

RG030750 W No primary file specified. Assume first secondary file is primary

RG030760 F No file description specifications

RG030770 W Error deleted

RG030780 F Error deleted

RG030790 F Block length is more than 255 times record length. Assume block=record

RG030800 W Error deleted

RG030810 W High-level directory size entry in cols 47-52 not 1-9999. Assume blank

RG030820 W Index buffer size (cols 60-65) missing. Assume SYSGEN default value

RG030830 F RA-processed file is not compatible with RA file (keys vs. rcd. nos.)

RG030840 F More than one record address-processed file has been defined

RG030850 F RA file is not compatible with RA-processed file (keys vs. rcd. nos.)

RG030860 F Key length of RA file does not equal key length of RA-processed file

RG030870 F Record address-processed file defined but no RA file defined

RG030880 F Record address file defined but no RA-processed file defined

RG030890 W Labels entry in column 53 should be S for disc file. Assume S

RG030900 W I record address type (col 31) invalid for file device. Assume blank

OVFRLAY 4. FILE EXTENSION AND LINE COUNTER SCAN

RGC40010 W Columns 7-10 should be blank. Assume blank

RGC40020 F Invalid FROM filename. Assume T file if cols 27-57 not blank; else RA

RGC40030 F FROM filename undefined. Assume T file if cols 27-57 not blank

RGC40040 F FROM file neither RA nor T file. Assume T if columns 27-57 not blank

RGC40050 W 'E' not specified on file description specifications for FROM file

RGC40060 F An RA file has already been described for this program

RGC40070 F Filename of RA-processed file is missing or invalid

RGC40080 F Filename of RA-processed file has not been defined by file specs.

RGC40090 F RA-processed file is not a primary, secondary or demand file

RGC40100 F RA-processed file is not an input or update file

RGC40110 F RA file is being used to process more than one file

RGC40120 W Columns 27-57 should be blank for an RA file. Assume blank

RGC40130 W TO file record length is not long engough for T/A items. Ignore TO file

RGC40140 W Invalid TO filename for table or array. Assume blank

RGC40150 W TO filename for table or array is undefined. Assume blank

RGC40160 W TO file for table or array is not an output file. Ignore TO filename

RGC40170 F T/A name missing or invalid. Assume table if name or alt. name=TABXXX

RGC40180 F More than 60 tables and arrays defined. Ignore those over 60

RGC40190 F Invalid entry for number of items in each record, cols 33-35. Assume 1

RG040200 W No. of items/rcd entry missing for pre-exec or comp time T/A. Assume 1

RG040210 F Invalid entry for no. of items per T/A. Assume equals no. items/rcd.

RG040220 W No. of items per T/A less than no. of items per record. Assume equal

RG040230 F Invalid entry for length of item in cols 40-42 or 52-54. Assume 1

RG040240 F FROM or compile file record length is not long enough for T/A items

RG040250 W Packed/binary entry is not blank for compile time T/A. Assume blank

RG040260 W Invalid packed or binary field entry in col 43 or 55. Assume blank

RG040270 W Unpacked length of binary item cannot be less than 4. Assume length 5

RG040280 W Unpacked length of binary item is not 4, 5, 9 or 10. Assume length 10

RG040290 W Invalid decimal positions entry. Assume 0 if P, B or length LE 15

RG040300 W Decimal pos. entry not numeric for packed or binary item. Assume 0

RG040310 F Length of item is greater than 255. Assume file record length or 255

RG040320 F Length of numeric item is greater than 15. Assume 15

RG040330 W Invalid sequence entry in column 45 or 47. Assume blank

RG040340 F Duplicate table/array name. No storage assigned

RG040350 F Table or array is too large. Allow 256 bytes for table or array

RG040360 F Data area used up. Table/array storage allocation will be in error

RG040370 W No extension specifications on card

RG040380 F Specifications missing for first table/array. Ignore alt. T/A specs.

RG040390 F Alt. array cannot be specified for exec. time array. Ignore alt. array

RG040400 F Alternate table or array name is missing or invalid.

RG040410 F First table/array and alternate table/array not both tables or arrays

RG040420 F More than 60 tables and arrays defined. Ignore those over 60

RG040430 W No extension specifications found for file XXXXXXXX

RG040440 W No line counter specifications on card

RG040450 F Filename in columns 7-14 is missing or invalid

RG040460 F Filename in columns 7-14 has not been defined by file descr. specs.

RG040470 W 'L' not specified on file description specifications for this file

RG040480 W Line counter specifications have already been processed for this file

RG040490 F File is not a print file

RG040500 W Line number entry is not a valid number or blank

RG040510 W Line number entry is not within the range 1-255. Assume 255

RG040520 W The same line number is assigned to more than one channel

RG040530 W Channel number entry is not 01-12, OL or FL

RG040540 W More than one line number is assigned to channel 01, 12 (=OL) or FL

RG040550 W Overflow line number is greater than form length. Assume equal

RG040560 W Chan 1 line is not less than overflow line. Assume 6 if OL GT 6; else 1

RG040570 W No line counter specifications found for file XXXXXXXX

RG040580 W Decimal positions entry is greater than item length. Assume equal

RG040590 W Non-blank entries follow a blank line number. Such entries ignored. (NOTE: A blank line number will be allowed in columns 15-17 if columns 18-19 are blank or FL and column 53 of the control card is L)

RG040600 F Too many table and array names used. No more storage assigned

RG040610 W Columns 18-19 should be FL when L spec'd in Col. 53 of the control card

RG040620 W Blank line number for FL, OL and/or 01 channel. Assume default values

OVERLAY 6. INPUT SPECIFICATIONS SPAN

RG060010 F Length of control level differs from previous def. Assume prev. def.

RG060020 F Invalid filename entry in columns 7-14. Assume previous filename

RG060030 F Filename has not been defined by file specs. Assume previous filename

RG060040 F File is not an input file. Assume previous filename

RG060050 F Filename has already been used on input specs. Ignore previous use

RG060060 F Filename missing from first input spec. Assume input sequential file

RG060070 F No. of match fields on record less than previous def. Assume 1st def.

RG060080 W Field definitions missing for previous record

RG060090 F Invalid filename entry in columns 7-14. Assume previous filename

RG060100 F AND/OR line illegal following look-ahead record identification

RG060110 F AND/OR line illegal following trailer record identification

RG060120 W AND/OR line does not follow a record identification line

RG060130 W AND line may not have a record indicator in cols 19-20. Ignore ind.

RG060140 W Stacker select in col 42 should be blank for AND line. Assume blank

RG060150 W No. and option (cols 17-18) should be blank for AND line. Assume blank

RG060160 W Numeric sequence no. invalid for chained file. Assume alpha sequence

RG060170 W Sequence numbers (cols. 15-16) not in ascending order. Assume they are

RG060180 W First numeric seq. entry for file is not 01. Accept number specified

RG060190 W Number entry in col 17 should be 1 or N for numeric sequence. Assume N

RG060200 W Option in column 18 is not 0 or blank. Assume 0 for numeric sequence

RG060210 W Invalid sequence entry in columns 15-16. Assume alphabetic

RG060220 F Alpha sequence in cols 15-16 invalid after numeric seq. Assume numeric

RG060230 W Number, option (cols 17-18) not blank for alpha sequence. Assume blank

RG060240 W Indicator in cols 19-20 is not a valid record indicator. Assume valid

RG060250 F Invalid record indicator in columns 19-20. Ignore indicator

RG060260 W Spread card header with numeric sequence must have N in column 17

RG060270 W Look-ahead specified with spread card trailer. Assume valid

RG060280 F More than one trailer specified for spread card. Ignore all but first

RG060290 W Invalid specs. made for TR or look-ahead. Assume invalid cols blank

RG060300 W Cols 43-70 should be blank for record identifying line. Assume blank

RG060310 F More than one set of look-ahead fields defined for file. Use first set

RG060320 F Look-ahead fields invalid for chained or demand file

RG060330 W Look-ahead fields invalid with numeric sequence. Ignore look-ahead

RG060340 F Position entry in columns 21-24, 28-31 or 35-38 missing or invalid

RG060350 F Position entry in cols 21-24, 28-31 or 35-38 is not within record size

RG060360 W 'NOT' entry in column 25, 32 or 39 is not N or blank. Assume N

RG060370 W C/Z/D entry in column 26, 33 or 40 is not C, Z or D. Assume C

RG060380 W AND/OR line follows line without record codes. Assume new record spec.

RG060390 W Stacker select entry in col 42 invalid for file device. Assume blank

RG060400 W Packed or binary field entry in col 43 not P, B or blank. Assume blank

RG060410 F Beginning field loc. entry in cols 44-47 missing or invalid. Assume 1

RG060420 F Ending field location entry in columns 48-51 missing or invalid

RG060430 F Ending field location in columns 48-51 is greater than record length

RG060440 F Beginning field location is greater than ending field location

RG060450 F Invalid decimal positions entry in col 52. Assume zero dec. positions

RG060460 F Dec. pos. entry exceeds field length. Assume field length=dec. pos.

RG060470 F Binary input field length is not 2 or 4. Assume 4

RG060480 F Numeric field length is greater than 15. Assume 15

RG060490 F Alphanumeric field length is greater than 255. Assume 255

RG060500 F Invalid field name in columns 53-58

RG060510 F Field name in columns 53-58 is missing

RG060520 F Table name may not be used as field name in columns 53-58

RG060530 F Date field is being redefined or page is being redefined illegally

RG060540 F Look-ahead field is being redefined

RG060550 W Dec. pos. for array item differs from def. on E spec. Assume E def.

RG060560 F Array field length is not a multiple of item length on E spec.

RG060570 F Binary input specified for array items more than 10 digits long

RG060580 W Number of array items on input record exceeds number of items in array

RG060590 W Alpha field specified as packed or binary. Ignore P or B entry

RG060600 F Invalid array index. Ignore index

RG060610 W Error deleted

RG060620 F Array index is greater than the number of items in the array

RG060630 F Array index name has not been previously defined. Ignore index

RG060640 F Array index is not a numeric field with zero dec. pos. Ignore index

RG060650 W Field name has an index but is not an array name. Ignore index

RG060660 W Invalid field record relation indicator (cols 63-64). Ignore indicator

RG060670 W Invalid control level indicator in columns 59-60. Ignore indicator

RG060680 F Data area used up. Storage allocation will be in error

RG060690 W Control or match field specified both as alpha and num. Assume numeric

RG060700 F Split control fields must be defined on consecutive lines

RG060710 W Fields without record relation indicators should precede those with

RG060720 W Record rel. ind. used with match or control field not prev. defined

RG060730 F Control field length is greater than 255

RG060740 F Match field entry in columns 61-62 is not M1-M9

RG060750 F No. of match fields on record exceeds previous def. Assume first def.

RG060760 F Match field level has already been used for a field in this record

RG060770 F Match field length differs from previous def. Assume previous def.

RG060780 F Total match field length for record is greater than 255

RG060790 F Match or control fields are valid only for primary or secondary files

RG060800 F Match or control level invalid for look-ahead or trailer field

RG060810 F Match or control field cannot be an array item

RG060820 W Invalid field indicator in cols 65-66, 67-68 or 69-70. Ignore indicator

RG060830 F Field size or type differs from previous definition. Assume prev. def.

RG060840 F No input specifications found for file XXXXXXXX

RG060850 F Files with matching records not all either ascending or desc. sequence

RG060860 W Record indicator in columns 19-20 is never referenced

RG060870 W Field record relation indicator in columns 63-64 is never defined

RG060880 W PLUS field indicator in columns 65-66 is never referenced

RG060890 W MINUS field indicator in columns 67-68 is never referenced

RG060900 W Zero or blank field indicator in columns 69-70 is never referenced

RG060910 W PLUS and MINUS field indicators invalid with an alphanumeric field

OVERLAY 7. CALCULATION SPECIFICATIONS SCAN

RG070010 W Invalid control level in columns 7-8. Assume latest control level

RG070020 F ULABL field length and decimal positions missing or incomplete

RG070030 F AN/OR line out of order or invalid. Ignore AN/OR line

RG070040 F More than 7 AN/OR lines used together. Ignore entire indicator group

RG070050 W 'AN' or 'OR' not specified in cols 7-8. Ignore entire indicator group

RG070060 F Operation not recognized

RG070070 W 'NOT' entry in column 9, 12 or 15 is not N or blank. Assume N

RG070080 F Invalid operation indicator. Ignore entire indicator group

RG070090 W Indicator missing for 'NOT' entry in column 9, 12 or 15. Ignore 'NOT'

RG070100 F Invalid resulting indicator. Ignore indicator group

RG070110 F Invalid factor 1 entry in columns 18-27

RG070120 F Invalid factor 2 entry in columns 33-42

RG070130 F Invalid filename entry for factor 2 (columns 33-42)

RG070140 F Filename in factor 2 has not been defined by file description specs.

RG070150 F Invalid index in factor 1 (columns 18-27)

RG070160 F Invalid index in factor 2 (columns 33-42)

RG070170 F Invalid result field entry in columns 43-48

RG070180 F Invalid result field length entry in columns 49-51

RG070190 F Invalid result field length entry in columns 49-51

RG070200 F Invalid decimal positions entry in column 52

RG070210 W Result field length missing but dec. pos. specified. Ignore dec. pos.

RG070220 W Half-adjust entry in column 53 is not H or blank. Assume H

RG070230 W Field attribute not the same as in previous def. Use previous def.

RG070240 W Result field has an index but is not an array name. Ignore index

RG070250 W Invalid index in result field (columns 43-48)

RG070260 F Result field is missing (columns 43-48)

RG070270 W Data area used up. Storage allocation will be in error

RG070280 F Factor 1 is missing (columns 18-27)

RG070290 F Factor 2 is missing (columns 33-42)

RG070300 W A resulting indicator must be specified for this operation

RG070310 W Missing AN/OR line. Ignore entire operation indicator group

RG070320 W Missing ENDSR statement. Assume ENDSR

RG070330 W Operation indicators missing on AN/OR line

RG070340 W Indicators in cols 9-17 not allowed for this operation. Assume blank

RG070350 W Indicators in cols. 54-59 not allowed for this operation. Assume blank

RG070360 W Factor 1 (cols 18-27) should be blank for this operation. Assume blank

RG070370 W Factor 2 (cols 33-42) should be blank for this operation. Assume blank

RG070380 W Result field should be blank for this operation. Assume blank

RG070390 F Field used as result field is a non-alterable field. Ignore spec.

RG070400 W Page field used as result has already been redefined. Use prev. def.

RG070410 F Name already has another use

RG070420 F GOTO operation may not branch into or out of a subroutine

RG070430 F GOTO operation may not branch out of one subroutine into another

RG070440 F Name already has another use

RG070450 F Name referenced by EXSR is not a subroutine name

RG070460 F Name referenced by GOTO is not a TAG name

RG070470 F A subroutine may not call itself

RG070480 W BEGSR statement missing. Assume BEGSR

RG070490 F BEGSR out of sequence. Ignore BEGSR line

RG070500 F Decimal pos. entry is greater than field length. Assume zero dec. pos.

RG070510 W Field length and dec. pos. (cols 49-52) should be blank. Assume blank

RG070520 W Decimal positions entry (column 52) should be blank. Assume blank

RG070530 W Half-adjust entry (column 53) should be blank. Assume blank

RG070540 W BEGSR statement should have SR in columns 7-8. Assume SR

RG070550 F Field length is over 255 for alphanumeric field or over 15 for numeric

RG070560 F TAG or subroutine name XXXXXX is undefined. Referenced NNN time(s)

RG070570 W TAG or subroutine name XXXXXX has not been referenced

OVERLAY 8. OUTPUT SPECIFICATIONS SCAN

RG080000 F AND/OR line out of order. Ignore all statements until next filename

RG080010 F Filename for output not on first output-format specifications line

RG080020 W No fields described for previous record

RG080030 W Output-format specifications missing

RG080040 W 1P indicator should be used with external indicators only

RG080050 W One or more output files have been defined but not referenced

RG080060 W Forms positioning is specified on control card; 1P indicator not used

RG080070 W Columns 16-22 should be blank for a field description. Assume blank

RG080080 F Invalid filename in columns 7-14

RG080090 F Filename has not been defined by file description specifications

RG080100 F File is not an output, update, combined or input 'ADD' file

RG080110 F T or E in column 15 invalid for combined or non-chained update file

RG080120 W Cols 32-70 should be blank for record identifying spec. Assume blank

RG080130 W Type of record in column 15 is not H, D, T, E or blank. Assume H

RG080140 W Output to input 'ADD' file requires 'ADD' in cols. 16-18. Assume 'ADD'

RG080150 W 'ADD' invalid for device or file type. Assume columns 16-18 blank

RG080160 W Error deleted

RG080170 W Stacker/overflow entry (col 16) invalid for file device. Assume blank

RG080180 W Stacker/overflow (col 16) not valid stacker number or F. Assume blank

RG080190 W Space/skip entries in cols 17-22 invalid for file device. Assume blank

RG080200 W Invalid skip entry in columns 19-22. Assume blank

RG080210 F More than 20 AND/OR lines used for this record. Ignore those over 20

RG080220 W Invalid indicator in cols 24-25, 27-28 or 30-31. Assume blank

RG080230 W Overflow indicator not assigned to file in file descr. Assume valid

RG080240 W 'NOT' entry in column 23, 26 or 29 is not N or blank. Assume N

RG080250 F Overflow indicator cannot condition EXCPT record. Ignore spec. line

RG080260 F Fetch overflow invalid with overflow indicator. Ignore fetch overflow

RG080270 F Overflow ind. used is not the same one assigned this file. Ignore spec

RG080280 W Fetch overflow invalid with 1P indicator. Ignore fetch overflow

RG080290 W 1P indicator cannot be used with combined file

RG080300 F 1P indicator cannot condition total record

RG080310 F 1P indicator cannot condition EXCPT record

RG080320 W Indicators missing on AND line

RG080330 W Space/skip entries should be blank on AND line. Assume blank

RG080340 W Indicators missing on OR line

RG080350 W 'ADD' in cols 16-18 illegal on OR line. Assume columns 16-18 blank

RG080360 W Invalid space entry in columns 17-18. Assume blank

RG080370 W Packed/binary field entry in col 44 not P,B,2,4, or blank. Assume blank

RG080380 W Blank after entry in column 39 not B or blank. Assume blank

RG080390 W Cols 45-70 can only be '$', '*' or blank with edit code. Assume blank

RG080400 W Packed/binary entry should be blank for an edited field. Assume blank

RG080410 W Columns 45-70 must be blank for edit code X, Y or Z. Assume blank

RG080420 F Invalid field name in columns 32-37

RG080430 F Field name in columns 32-37 has not been previously defined

RG080440 F Invalid index in field name (columns 32-37)

RG080450 W Page field name has already been used for another file. Ignore spec

RG080460 W Cols after end apostrophe in cols 45-70 should be blank. Assume blank

RG080470 W Punch field follows print only field (* in col 40). Assume print only

RG080480 F Invalid edit word in columns 45-70. Ignore specifications line

RG080490 F End apostrophe missing for constant in columns 45-70. Assume present

RG080500 F Leading apostrophe missing for constant in cols 45-70. Assume present

RG080510 W Invalid edit code in column 38. Assume blank

RG080520 F Invalid entry for ending position in columns 40-43. Assume 1

RG080530 F End position entry exceeds record length. Assume equals record length

RG080540 F Print only field (* in col 40) invalid for non-MFCU file. Ignore spec

RG080550 W Indicators cannot be used to condition *PLACE. Ignore indicator(s)

RG080560 F End position for *PLACE is not at least twice previous high position

RG080570 F *PLACE or *PRINT must be preceded by a field description

RG080580 W Cols 38-39 and 44-74 should be blank for *PLACE, *PRINT. Assume blank

RG080590 W Ending position in cols 40-43 should be blank for *PRINT. Assume blank

RG080600 F *PRINT specification invalid after * has been used in col 40. Ignore

RG080610 F *PRINT specification invalid for non-MFCU file. Ignore specification

RG080620 F Field name (cols 32-37) and constant (cols 45-70) both blank

RG080630 W Packed or binary output invalid for file device. Assume col 44 blank

RG080640 W Packed or binary entry should be blank for constant. Assume blank

OVERLAY 11. FILE EXTENSION/LINE COUNTER GENERATION

Those error messages which are followed by a printed card image and an asterisk (*) showing the error location are indicated by an asterisk, in parentheses, following the message.

RG110010 F Invalid numeric value in compile time table or array. Card number NNN of XXXXXX (*)

RG110020 F Compile time table or array item is out of sequence. Card number NNN of XXXXXX (*)

RG110030 F Too much data given for compile time table or array. Card number NNN of XXXXXX (*)

RG110040 W Compile time table or array data missing. Items set to zeros or blanks in XXXXXX

RG110050 W Compile time table or array short. Empty items set to zeros or blanks in XXXXXX

RG110060 W Invalid char. in file trans. or alt. seq. record. Ignore rest of rcd. (*)

RG110070 W Char. already referenced in trans. or alt. seq. table. Ignore 2nd ref (*)

RG110080 W File translation filename XXXXXXXX invalid. Ignore translation record

RG110090 W File translation records for XXXXXXXX not together. Ignore later rcd.

RG110100 W Cols 1-8 of alt. seq. record are XXXXXXXX, not 'ALTSEQ␣␣' Ignore rcd

RG110110 W File translation table missing

RG110120 W Alternate collating sequence specifications missing

OVERLAY 13. CALCULATION GENERATOR

RG130600 W Operation indicator XX is never defined

RG130610 F Factor 2 is not a table or array for a LOKUP command

RG130620 F Result field length specified in columns 49-51 for blank result field

RG130630 F Factor 1 and/or 2 should be an array if result is a non-indexed array

RG130640 F Factor 1 is not a valid type of field for this operation

RG130650 F Factor 2 is not a valid type of field for this operation

RG130660 F Result field is not a valid type of field for this operation

RG130670 F Array index is not a numeric field or literal with zero dec. positions

RG130680 F Factor 1 and/or factor 1 index has not been defined

RG130690 F Factor 2 and/or factor 2 index has not been defined

RG130700 F Result field and/or result field index has not been defined

RG130710 F Result field must be an array if factor 1 or 2 is a non-indexed array

RG130720 W Resulting indicator XX is never referenced

RG130730 W Result field size may be too small to contain the operation result

RG130740 F Invalid bit specifications for BITON, BITOF or TESTB operation

RG130750 F Literal index is greater than the number of items in the array

RG130760 W Result field size may be too small to contain the operation result

RG130770 F MVR operation does not follow a DIV operation

RG130780 F Factor 2 must be an array name for a XFOOT operation

RG130790 F Factor 1 and factor 2 must be both alphanumeric or numeric for compare

RG130800 F Record length of file used for debug must be 34 or greater

RG130810 W Both high and low indicators may not be used with LOKUP. Ignore low

RG130820 F Unsequenced table or array may not be searched for hi or lo condition

RG130830 F LOKUP may only be used with one array at a time. Ignore LOKUP

RG130840 W Search table longer than related table. Search ends with shorter one

RG130850 F Search word is not the same size as table/array items. Ignore LOKUP

RG130860 W Columns 54-57 should be blank for READ operation. Assume blank

RG130870 W Indicator should be specified in columns 58-59 for end-of-file on read

RG130880 W Columns 56-59 should be blank for CHAIN operation. Assume blank

RG130890 W Indicator should be specified in columns 54-55 for unsuccessful chain

RG130900 F File used with FORCE is not input, update, or combined; primary or sec.

RG130910 F File referenced by READ command is not a demand file

RG130920 F File referenced by CHAIN command is not a chained file

RG130930 F Key in factor 1 is not same length or type as key field of chain file

RG130940 F File used with DEBUG operation is not an output file

RG130950 F File used with DEBUG operation may not be a chained file

RG130960 F More than one output file specified for DEBUG operation

RG130970 F File used with DSPLY command is not a display file

RG130980 F Either factor 1 or the result field must be specified for DSPLY

RG130990 F Factor 2 or result field is invalid for a bit operation

RG131000 W Field or literal to be displayed is longer than 100 chars. Display 100

OVERLAY 14. OUTPUT GENERATOR

RG140010 F Field name has not been defined or is used as another kind of name

RG140020 F Array index is not a numeric field. Ignore specifications line

RG140030 W Packed or binary entry should be blank for alpha field. Assume blank

RG140040 W Edit word in cols 45-70 should be blank for alpha field. Assume blank

RG140050 W Edit code in column 38 should be blank for alpha field. Assume blank

RG140060 F Invalid field length for Y edit code. Ignore specifications line

RG140070 F No. of replaceable characters in edit word incorrect for field length

RG140080 F Field name has an index but is not an array. Ignore spec. line

RG140090 F Index has more than zero decimal positions. Ignore spec. line

RG140100 F Index cannot be an array name. Ignore specifications line

RG140110 F Size of field exceeds ending position or record length. Ignore spec.

RG140120 W Indicator used to condition record has not been previously defined

RG140130 W Blank after entry invalid for non-alterable field. Assume col 39 blank

RG140140 W Indicator used to condition field has not been previously defined

RG140150 F Literal index is too large for the array. Ignore specifications line

OVERLAY 15. CODE FORMATTER

RG150010 W Page no. and/or line no. entry in cols 1-5 invalid or out of sequence

RG150020 F Type of form entry in col 6 is invalid or out of order. Ignore line

RG150030 F Too many external references used in program

RG150040 F Generated object program is too large for core

APPENDIX H. OPERATIONAL ENVIRONMENT

SOFTWARE

The RPG II compiler runs under the Memorex Operating System (MRX/OS) using block input/output, and requires the date and time functions. A relocatable object module is produced, ready for the linkage editor. All subroutines required by the generated program are automatically called in by external references (EXTRNS).

The generated RPG II program runs under MRX/OS using logical input/output. It requires the DATE and user switch functions, and the commercial instructions set. No SYSGEN parameters are required.

RPG II user switches U1-U8 are external switches which are initially set by a //SET SWITCH= Control Language Statement card and which correspond to the eight CLS switches. If U1-U8 are reset in a job step, the action is passed on to the next job step.

To activate the RPG II compiler, all the operator need do is arrange the source card deck in the proper order (see Figure 1-2) and place the cards in the card reader; processing from that point on is automatic.

COMPILE-TIME ERROR MESSAGES WRITTEN TO SYSOUT

Two compile-time messages may be written to SYSOUT. The first of these messages is issued by the compiler when it encounters a syntax error in the //PAR card. Its format is:

**** RPG PARAMETER CARD SYNTAX ERROR **** CARD IS

The message is immediately followed by the card image. When this type of error occurs, the entire card is invalidated, the job is cancelled, and the program automatically halts. Compilation can be resumed only after the //PAR card has been corrected and the job resubmitted.

The other message is written to SYSOUT if the user partition size is insufficient to compile the program. Its format is:

POOLSIZ ALLOCATION LEAVES INSUFFICIENT MINIMUM
BUFFER/TABLE SPACE-TO CORRECT SET POOLSIZ=180 IN
COMPILER LINK-EDIT STEP

This message also halts the program automatically. The programmer must then reduce the poolsize to 180 and resubmit the job. If the compiler continues to issue the same message, the program requires a larger user partition. The minimum System partition size acceptable is 8K.

OPERATOR RESPONSES TO EXECUTION-TIME ERROR CONDITIONS

The RPG II compiler does not communicate directly with the operator; however, at run time it informs the operator of any error conditions via messages on the console. The compiler supplies variable field information when the message is printed so that the operator can isolate the error, and the operator may take his choice of responses. Differing responses are legal for different messages. The user may enter a one-character response code, as follows:

0 To continue processing where he left off.

1 To go to the beginning of the next cycle.

2 To request a controlled cancel, in which tables and arrays are written out and files are closed before the job is ended.

3 To cancel immediately.

The run-time error messages and the legal response options available are listed on the following page.

Error No.	Message	Variable Fields	Permitted Responses			
			0	1	2	3
RGRT0001	IND ON IS Hn	n = indicator 0-9	x		x	x
RGRT0002	NEG. SQ. RT. AT LINE nnnnn	nnnnn = line number	x		x	x
RGRT0003	ARITH OVERFLOW AT LINE nnnnn	nnnnn = line number	x		x	x
RGRT0004	DIVIDE BY 0 AT LINE nnnnn	nnnnn = line number	x		x	x
RGRT0005	VARIABLE ARRAY INDEX IS 0, MINUS, OR TOO LARGE AT LINE nnnnn	nnnnn = line number	x		x	x
RGRT0006	TABLE/ARRAY SEQ ERROR, T/A-RECORD NO. S nnnnn-nnnnn * FILE IS xxxxxxxx	nnnnn-nnnnn = table/array number and record number; xxxxxxxx = file identification	x		x	x
RGRT0007	TABLE/ARRAY MISSING, T/A NO. nnnnn * FILE IS	nnnn = table/array number; xxxxxxxx = file identification	x		x	x
RGRT0008	EXCESS TABLE/ARRAY ENTRIES,T/A NO. nnnnn * FILE IS xxxxxxxx	nnnn = table/array number xxxxxxxx = file identification	x		x	x
RGRT0009	PARTITION SIZE EXCEEDED	None				x
RGRT0010	1P FORMS ALIGNMENT	None	x	x		
RGRT0011	RECORD SEQ ERROR * FILE IS xxxxxxxx	xxxxxxxx = file identification		x	x	x
RGRT0012	MATCHING RECORD SEQ ERROR * FILE IS xxxxxxxx	xxxxxxxx = file identification		x	x	x
RGRT0013	UNIDENTIFIED RECORD * FILE IS xxxxxxxx	xxxxxxxx = file identification		x	x	x
RGRT0014	INVALID NUMBER AT LINE nnnnn	nnnnn = line number	x		x	x
RGRT0015	CHANNEL NOT DEFINED ON LINE COUNTER * FILE IS xxxxxxxx	xxxxxxxx = file identification	x			x
RGRT0016	BINARY CONVERSION OVERFLOW AT LINE nnnnn	nnnnn = line number	x			x
RGRT0017	TABLE/ARRAY BINARY OVERFLOW, T/A NO. nnnnn * FILE IS xxxxxxxx	nnnnn = table/array number; xxxxxxxx = file identification	x			x
RGRT0018	SPECIAL FILE I/O ERROR nnn	nnn = error code			x	x
RGRT0019	EOF INDICATOR MISSING ON READ OR CHAIN OPERATION * FILE IS xxxxxxxx	xxxxxxxx = file identification			x	x

MRX/SYSTEM 3/MODEL 20 COMPARATIVE ANALYSIS

The following table presents an analysis of the RPG features supported by Memorex as compared to IBM's System/3 and Model 20 support.

Feature	MRX		Support System/3		Mod 20	
	Yes	No	Yes	No	Yes	No
AN/OR calculations	x		x			x
Arrays	x		x			x
Binary input and output fields	x		x			x
Binary operations	x		x			x
Binary search	x		x		x	
Combined files		x	x		x	
DEBUG statement	x		x			x
Demand files	x		x			x
Display files	x		x			x
Dual-feed printer		x	x		x	
Fetch overflow	x		x			x
Input chaining		x		x	x	
Lookahead	x		x			x
MFCM		x		x	x	
MFCU		x	x			x
Multiple printing of first line	x		x			x
READ calculations	x		x			x
Sign forcing	x			x	x	
Special field names PAGE1,PAGE2, PAGE3	x		x			x
Special field name *PLACE	x		x			x

Feature	MRX		Support System/3		Mod 20	
	Yes	No	Yes	No	Yes	No
SQRT calculations	x		x			x
Stacker select		x	x		x	
Sterling currency		x	x		x	
Suppression of 1P skip to channel 1	x			x	x	
Tapes	x			x	x	
Translation table	x		x			x
Unprintable character halt		x	x		x	
Variable-length records	x			x	x	
XFOOT operations	x		x			x

APPENDIX I. INDEXED FILES

STRUCTURE

Indexed files are divided into two portions: the data portion, and the index portion.

DATA PORTION

The data portion of an indexed file is composed of blocks of fixed-length data records which are scattered on the storage device. Each of these data records contains a unique record key which serves to identify the particular record. The length of this key and the record position in which the key field begins are specified in columns 29-30 (length of key field entry) and 35-38 (starting location of key field entry) of the RPG II File Description Specifications Sheet.

INDEX PORTION

The index portion of an indexed file contains a copy of each data record's key. The keys are arranged in sequence. Each record's key is followed by an address associated with that key; the address consists of the number of the data block in which the data record resides, and the number of the record. The keys and addresses thus form the control information used in finding data records in the data portion of the file. Every record in the data portion of an indexed file is identified in the index portion of that file by such control information. (See Appendix J, "Index Portion of Indexed File," for a more detailed explanation of the index portion's structure.)

INDEX BLOCKS

When an indexed file is created, the Data Management facility of the Memorex Operating System writes the index control information on disc in blocks; these blocks of information are referred to as index blocks. The size of index blocks can vary from file to file (see "Index Block Sizes" in this appendix). It is governed by such factors as record key lengths and the number of records in the data portion of the file.

INDEX BUFFERS

To find the location of a specific data record, the index portion of the indexed file must be searched for the block containing the right key and address. Searching the index portion involves reading index information from disc into main storage for analysis by the Data Management facility. A buffer must be defined to hold the index information; this buffer is referred to as an index buffer. The index information is read into the index buffer by blocks until the proper key is found. Because the index is itself built with an index structure, no more than three blocks need to be read before the key is located; the address of the record is then extracted from the key to determine the record's location.

INDEXED FILE ACCESS

Indexed files may be accessed for data records either sequentially or randomly. The mode of processing affects the rate of retrieval.

SEQUENTIAL ACCESS

To access a data record in an indexed file sequentially, Data Management merely reads the index sequentially. The keys are already in sequential order. After accessing the first record, Data Management uses the address associated with the next key to obtain the next record. From the block number and record number which form the address, Data Management can calculate the storage location where the data resides. It reads in the data block, then readies the record for processing.

RANDOM ACCESS

Random access of an indexed file requires searching. The search begins with the high-level directory* in the index portion of the file. Because the high-level directory is accessed so frequently, it is often advantageous to have it in memory all the while the file is being used. This is accomplished by copying it into the high-level directory buffer rather than the usual index buffer. In this way the high-level directory is never copied over by other index information. When an RPG II user makes a high-level directory size entry in columns 47-52 of the File Description Specifications Sheet, he is specifying that the high-level directory is to stay in memory. The entry is valid only for random processing of indexed files.

INDEX BLOCK SIZES

When a user is planning the creation of indexed files, he must decide whether he wants to process the high-level directory in a main storage buffer. This option speeds up random processing, but requires extra space for the buffer. Once the user has made this decision, he must calculate the index block size; this he can do either by table or by formula (see "Calculating Index Block Size by Table" and "Calculating Index Block Size by Formula" in this appendix).

MINIMUM INDEX BLOCK SIZE

Every indexed file has a minimum index block size which depends on key size and file size. The user may utilize any index block size larger than the minimum if he has storage space for a larger index block; the larger the index block, the better retrieval becomes on random processing. If the user goes below the minimum index block size, the possibility exists that he may not be able to create the file size as planned.

---

* The high-level directory used in RPG II is referred to as the "directory-directory" in the Data Management Extended, COBOL, and Control Language Extended Reference Manuals.

OPTIMUM INDEX BLOCK SIZE

If the user chooses to process the file randomly with the high-level directory in a main storage buffer, a well defined optimum index block size may be used which minimizes storage space for the index buffer and high-level directory buffer. When using the optimum block size, however, he must be careful at file creation time not to exceed the file maximum number of records from which he calculated the keys per block. This restriction is further explained under "Index Buffer Size" in this appendix.

CALCULATING INDEX BLOCK SIZE BY TABLE

Once the user has determined his mode of processing, he may calculate the index block size by means of the tables on the following pages. In both tables I-1 (for minimum index block size) and I-3 (for optimum index block size), the larger of the two values in the file size is the determining factor. The user should also note that for consistency these two tables were computed for a maximum key size of 100 bytes, with one million records as the upper limit. Some index block sizes will be computed that exceed one track in number of bytes; since this size is more than the system limit of 7294 bytes for blocks, the user will have to choose a smaller key size or file size.

MINIMUM INDEX BLOCK SIZE

Table I-1 is used to determine the minimum number of keys per index block. The following procedure may be used to calculate the minimum block size using Table I-1 in conjunction with Table I-2:

1. In Table I-1, locate the number of records in the file and the key size. For example, if the number of records is 20,000 and the key size is 10, the minimum keys per index block is 24.

2. Enter the keys per block in the Control Language define (//DEF) statement, along with the key size.

3. Calculate the corresponding minimum index block size from Table I-2, using the minimum block size formula. With minimum keys per index block of 24 and a key size of 10, for example, the minimum index block size would be 384 bytes.

4. Enter the minimum index block size in the source program (see "Index Buffer Size" in this appendix).

Records in File	Key Size in Bytes														
	2	3	4	5	6	7	8	9	10	11 to 15	16 to 20	21 to 25	26 to 35	36 to 50	51 to 100
1 - 5,000	13	14	14	14	15	15	15	15	15	16	16	16	16	17	17
5,001 - 10,000	16	17	18	18	18	19	19	19	19	20	20	20	21	21	21
10,001 - 15,000	19	20	20	21	21	21	22	22	22	23	23	23	23	24	24
15,001 - 20,000	20	21	22	23	23	23	24	24	24	25	25	26	26	26	26
20,001 - 25,000	22	23	24	24	25	25	25	26	26	27	27	27	28	28	28
25,001 - 30,000	23	24	25	26	26	27	27	27	28	28	29	29	29	30	30
30,001 - 35,000	25	26	27	27	28	28	28	29	29	30	30	31	31	31	32
35,001 - 40,000	26	27	28	28	29	29	30	30	30	31	32	32	32	33	33
40,001 - 45,000	27	28	29	30	30	31	31	31	31	32	33	33	34	34	34
45,001 - 50,000	28	29	30	31	31	32	32	32	33	34	34	34	35	35	36
50,001 - 60,000	29	31	32	32	33	34	34	34	35	36	36	37	37	37	38
60,001 - 70,000	31	32	34	34	35	35	36	36	36	37	38	38	39	39	40
70,001 - 80,000	32	34	35	36	36	37	37	38	38	39	40	40	41	41	42
80,001 - 90,000	34	35	36	37	38	38	39	39	40	41	41	42	42	43	43
90,001 - 100,000	35	36	37	38	39	40	40	41	41	42	43	43	44	44	45
100,001 - 125,000	37	39	40	41	42	43	43	44	44	45	46	47	47	48	48
125,001 - 150,000	40	41	43	44	45	45	46	46	47	48	49	49	50	51	51
150,001 - 175,000	42	44	45	46	47	48	48	49	49	50	51	52	53	53	54
175,001 - 200,000	44	46	47	48	49	50	50	51	51	53	54	54	55	56	56
200,001 - 250,000	47	49	51	52	53	54	54	55	55	57	58	58	59	60	61
250,001 - 300,000	50	52	54	55	56	57	58	58	59	61	61	62	63	64	64
300,001 - 350,000	52	55	57	58	59	60	61	62	62	64	65	65	66	67	68
350,001 - 400,000	55	57	59	61	62	63	63	64	65	67	68	68	69	70	71
400,001 - 450,000	57	60	62	63	64	65	66	67	67	69	70	71	72	73	74
450,001 - 500,000	59	62	64	65	67	68	68	69	70	72	73	74	74	75	76
500,001 - 600,000	63	66	68	69	71	72	73	73	74	76	77	78	79	80	81
600,001 - 700,000	66	69	71	73	74	75	76	77	78	80	81	82	83	84	85
700,001 - 800,000	69	72	74	76	78	79	80	81	81	84	85	86	87	88	89
800,001 - 900,000	72	75	77	79	81	82	83	84	85	87	88	89	91	91	93
900,001 - 1,000,000	74	78	80	82	84	85	86	87	88	90	92	93	94	95	96

Table I-1. Minimum Keys Per Block.

$$\text{Optimum block size} = 10 + \left\{ \frac{(10)\,(0KB)\,(KS+4)}{9} \right\}$$

0KB = Optimum keys/block

KS = Key size

$$\text{Minimum block size} = 10 + \left\{ \frac{(10)\,(MKB)\,(KS+4)}{9} \right\}$$

MKB = Minimum keys/block

KS = Key size

NOTE: $\{\ \}$ = Round up if result not whole integer.

Table I-2. Optimum or Minimum Index Block Size.

LARGER THAN MINIMUM INDEX BLOCK SIZE

An index block size that is larger than the minimum may be calculated using the steps shown for calculating the minimum index block size; however, the keys per block figure must be greater than or equal to the minimum keys per block number selected.

OPTIMUM INDEX BLOCK SIZE

The optimum index block size can be calculated with the following procedure:

1. In Table I-3, locate the number of records in the file and the key size. For example, if the number of records is 20,000 and key size is 10, the optimum keys per block is 30.

2. Enter the keys per block in the Control Language define (//DEF) statement, along with the key size.

3. Calculate the corresponding optimum index block size from Table I-2, using the optimum block size formula. With optimum keys per index block of 30 and a key size of 10, for example, the optimum index block size would be 477 bytes.

4. Enter the optimum index block size in the source program.

Records in File	Key Size in Bytes														
	2	3	4	5	6	7	8	9	10	11 to 15	16 to 20	21 to 25	26 to 35	36 to 50	51 to 100
1 - 5,000	16	17	18	18	18	19	19	19	19	20	20	20	21	21	21
5,001 - 10,000	20	21	22	23	23	23	24	24	24	25	25	25	26	26	26
10,001 - 15,000	23	24	25	26	26	27	27	27	27	28	29	29	29	30	30
15,001 - 20,000	26	27	28	28	29	29	29	30	30	31	32	32	32	33	33
20,001 - 25,000	28	29	30	31	31	32	32	32	33	34	34	34	35	35	36
25,001 - 30,000	29	31	32	32	33	34	34	34	35	36	36	37	37	37	38
30,001 - 35,000	31	32	33	34	35	35	36	36	37	37	38	38	39	39	40
35,001 - 40,000	32	34	35	36	36	37	37	38	38	39	40	40	41	41	42
40,001 - 45,000	34	35	36	37	38	38	39	39	40	41	41	42	42	43	43
45,001 - 50,000	35	36	37	38	39	40	40	41	41	42	43	43	44	44	45
50,001 - 60,000	37	39	40	41	42	42	43	43	43	45	45	46	46	47	48
60,001 - 70,000	39	41	42	43	44	44	45	45	46	47	48	48	49	49	50
70,001 - 80,000	40	42	44	45	46	46	47	47	48	49	50	50	51	52	52
80,001 - 90,000	42	44	45	47	47	48	49	49	50	51	52	52	53	54	54
90,001 - 100,000	44	46	47	48	49	50	50	51	51	53	54	54	55	56	56
100,001 - 125,000	47	49	51	52	53	54	54	55	55	57	58	58	59	60	61
125,001 - 150,000	50	52	54	55	56	57	58	58	59	61	61	62	63	64	64
150,001 - 175,000	52	55	57	58	59	60	61	61	62	64	65	65	66	67	68
175,001 - 200,000	55	57	59	61	62	63	63	64	65	67	68	68	69	70	71
200,001 - 250,000	59	62	64	65	67	68	68	69	70	72	73	74	74	75	76
250,001 - 300,000	63	66	68	69	71	72	73	73	74	76	77	78	79	80	81
300,001 - 350,000	66	69	71	73	74	75	76	77	78	80	81	82	83	84	85
350,001 - 400,000	69	72	74	76	78	79	80	81	81	84	85	86	87	88	89
400,001 - 450,000	72	75	77	79	81	82	83	84	85	87	88	89	91	91	93
450,001 - 500,000	74	78	80	82	84	85	86	87	88	90	92	93	94	95	96
500,001 - 600,000	79	82	84	87	89	90	91	92	93	96	97	98	100	101	102
600,001 - 700,000	83	87	90	92	94	95	96	97	98	101	102	103	105	106	107
700,001 - 800,000	87	91	94	96	98	99	100	102	102	105	107	108	110	111	112
800,001 - 900,000	90	94	97	100	102	103	105	106	106	110	111	112	114	115	116
900,001 - 1,000,000	93	98	101	103	105	107	108	109	110	113	115	116	118	120	121

Table I-3. Optimum Keys Per Block.

INDEX BUFFER SIZE

When the keys per block and key size are entered in the Control Language define statement and the corresponding minimum or optimum index block size has been calculated from Table I-2, the index block size is then entered in the RPG II source program via the index buffer size specification of the file definition card (columns 60-65 of the File Description Specifications Sheet).

HIGH-LEVEL DIRECTORY SIZE

If the user has elected to calculate the optimum keys per index block and optimum index block size, he then uses Table I-4 to calculate the number of bytes for the main storage buffer for the high-level directory entries, and enters the resulting figure in columns 47-52 of the file definition card (high-level directory size entry). Using an optimum index block size of 477 bytes, a key size of 10, and a file size of 20,000, for example, the number of bytes required for the buffer for the high-level directory would be 250.

Usage $= \left[\frac{9(OBS-10)}{10}\right] =$	US
Number keys/primary index block $= \left[\frac{US}{KS+4}\right] =$	NKP
Number keys/directory block $= \left[\frac{US}{KS+2}\right] =$	NKD
Total number keys represented/ directory block	= (NKP) (NKD) = NKRD
Number entries in High-level directory block $= \left\{\frac{\text{file size}}{NKRD}\right\} =$	NKDD
Number of bytes required for buffer for high-level directory entries	= 10 + (KS+2) (NKDD)

NOTE: $\{\ \}$ = Round up if result not whole integer.

$[\ ]$ = Round down if result not whole integer.

Table I-4.
Bytes Required in Buffer for High-Level Directory Entries

The user must be careful at file creation time not to exceed the file maximum number of records from which he calculated the keys per index block when using the optimum block size. If this were to occur, the main storage buffer would not be large enough to hold the high-level directory entries for random processing, and the excess would be written over the user program. When an index block size other than the optimum is chosen, and the main storage buffer is used to process the high-level directory entries, the buffer size should be the size of the index block. The system checks for overflow of this value at file creation time.

CALCULATING INDEX BLOCK SIZE BY FORMULA

If the user wishes to calculate the keys per index block with a file maximum number of records different from those given in Tables I-1 and I-3, he may use the following algorithms, along with Table I-5, to compute minimum and optimum keys per index block. The constants Ko and Km, taken from Table I-5, are based on key size.

$$\text{Optimum (OKB)} = \left\{ \sqrt[3]{\frac{FS}{Ko}} \right\}$$

$$\text{Minimum (MKB)} = \left\{ \sqrt[3]{\frac{FS}{Km}} \right\}$$

FS = Maximum File Size

NOTE: $\{\ \}$ = Round to next higher integer if result not whole integer.

KS	K_o	K_m	KS	K_o	K_m
2	1.2500	2.5000	52	.5975	1.1950
3	1.0888	2.1776	53	.5967	1.1934
4	.9877	1.9753	54	.5959	1.1918
5	.9184	1.8367	55	.5952	1.1904
6	.8681	1.7361	56	.5945	1.1890
7	.8299	1.6598	57	.5939	1.1878
8	.8000	1.6000	58	.5932	1.1864
9	.7759	1.5518	59	.5926	1.1852
10	.7562	1.5124	60	.5920	1.1840
11	.7396	1.4792	61	.5914	1.1828
12	.7256	1.4512	62	.5908	1.1816
13	.7136	1.4272	63	.5903	1.1806
14	.7031	1.4062	64	.5897	1.1794
15	.6940	1.3880	65	.5892	1.1784
16	.6859	1.3718	66	.5887	1.1774
17	.6787	1.3574	67	.5882	1.1764
18	.6722	1.3444	68	.5878	1.1756
19	.6664	1.3328	69	.5873	1.1746
20	.6612	1.3224	70	.5868	1.1736
21	.6564	1.3128	71	.5864	1.1728
22	.6520	1.3040	72	.5860	1.1720
23	.6480	1.2960	73	.5856	1.1712
24	.6443	1.2886	74	.5852	1.1704
25	.6409	1.2818	75	.5848	1.1696
26	.6378	1.2756	76	.5844	1.1688
27	.6348	1.2696	77	.5840	1.1680
28	.6321	1.2642	78	.5837	1.1674
29	.6296	1.2592	79	.5833	1.1666
30	.6272	1.2544	80	.5830	1.1660
31	.6249	1.2498	81	.5827	1.1654
32	.6228	1.2456	82	.5823	1.1646
33	.6209	1.2418	83	.5820	1.1640
34	.6190	1.2380	84	.5817	1.1634
35	.6172	1.2344	85	.5814	1.1628
36	.6156	1.2312	86	.5811	1.1622
37	.6140	1.2280	87	.5808	1.1616
38	.6125	1.2250	88	.5805	1.1610
39	.6111	1.2222	89	.5802	1.1604
40	.6097	1.2194	90	.5800	1.1600
41	.6084	1.2168	91	.5797	1.1594
42	.6072	1.2144	92	.5794	1.1588
43	.6060	1.2120	93	.5792	1.1584
44	.6049	1.2098	94	.5789	1.1578
45	.6038	1.2076	95	.5787	1.1574
46	.6028	1.2056	96	.5785	1.1570
47	.6018	1.2036	97	.5782	1.1564
48	.6009	1.2018	98	.5780	1.1560
49	.6000	1.2000	99	.5778	1.1556
50	.5991	1.1982	100	.5776	1.1552
51	.5983	1.1966			

Table I-5. Constants for Alternate Algorithm

APPENDIX J. INDEX PORTION OF INDEXED FILE

Figure J-1 shows the layout of an indexed file's index portion and the relationship of each block to the other parts of the index portion.

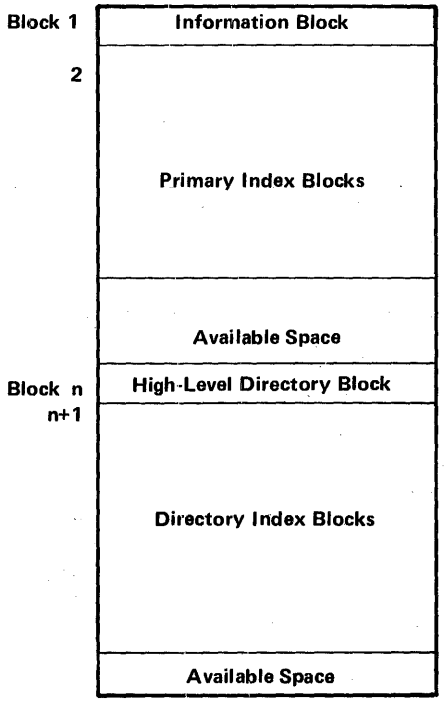


Figure J-1. Index Portion of an Indexed File

Each of the individual blocks shown in the diagram is discussed in the succeeding paragraphs of this appendix.

INFORMATION BLOCK

The information block of an indexed file's index portion supplies the addresses of the last blocks of the data and index portions of the file that were written. This enables the Data Management facility to make further additions to the file. Figure J-2 shows the fields contained in the information block.

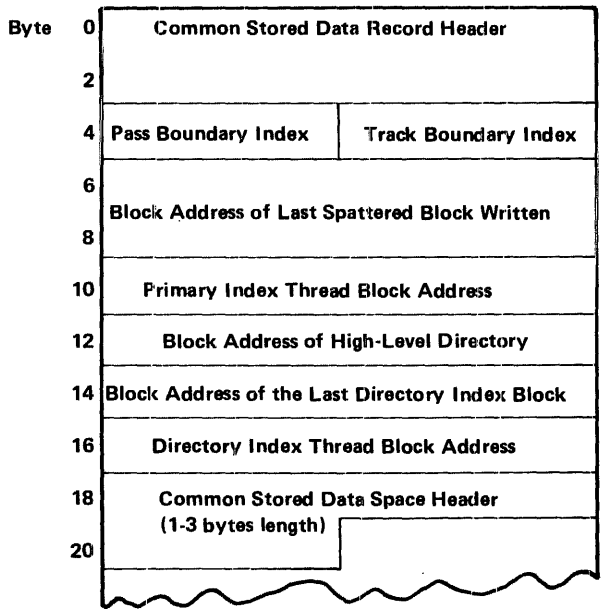


Figure J-2. Information Block

HIGH-LEVEL DIRECTORY BLOCK

Figure J-3 shows the layout of the high-level directory block. A directory index block address is associated with each key value.

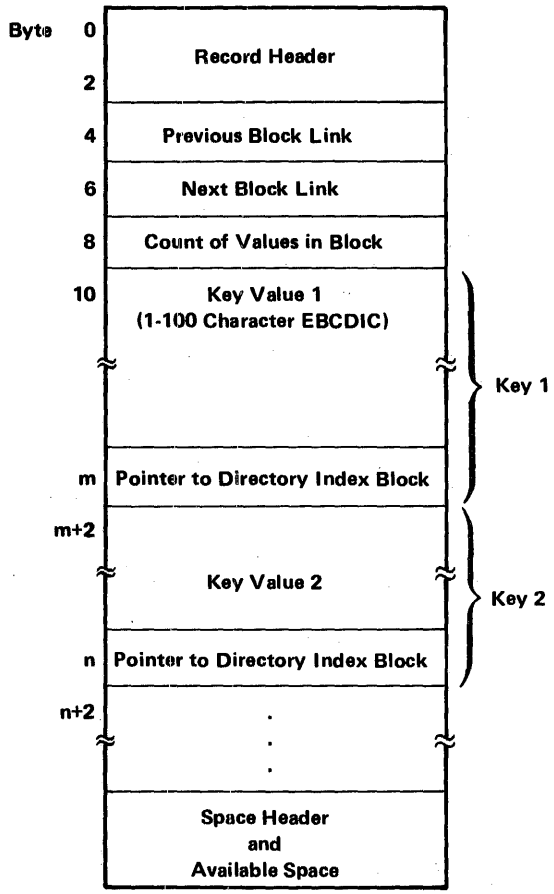


Figure J-3. High-Level Directory Block

The previous block link and next block link entries are not used in the directory blocks; they are reserved for future use.

DIRECTORY INDEX BLOCK

Figure J-4 illustrates the layout of the directory index block. Each key value has an associated primary index block address.

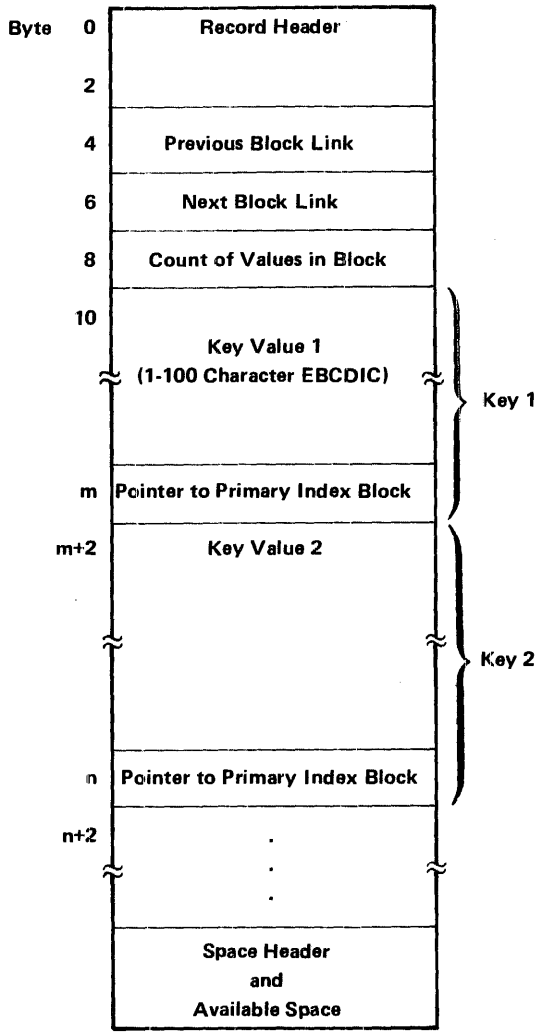


Figure J-4. Directory Index Block

PRIMARY INDEX BLOCK

Figure J-5 shows the layout of the primary index block. A block-record address is associated with each key value.

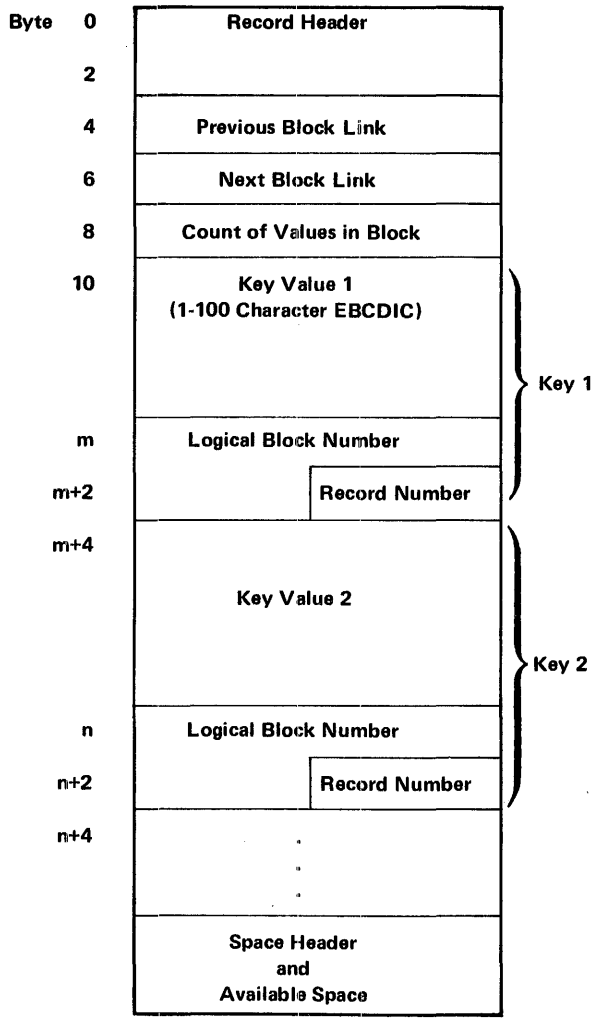


Figure J-5. Primary Index Block

Publications Bulletin No. 04

1/31/73

MEMOREX

Computer System Products

Update Package **2202.003-0002,** for:

MRX/OS RPG II
Reference Guide
2202.003

This Bulletin advises of changes which have occurred to the **RPG II Reference Guide** since the July 1972 preliminary edition was issued. New and replacement pages are provided where required.

Section	Pages	Action
F	1-10	Replace entire section

Changes to text, tables, and illustrations are marked with a vertical line to the left of the change.

Please file this Bulletin with the publication to retain a record of changes.

Publications Bulletin

2202.003-0003

3/30/73

MEMOREX

Computer System Products

Update Package for:

MRX/OS RPG II
Reference Guide
2202.003

This Bulletin advises of changes which have occurred to the **RPG II Reference Guide** since the July 1972 preliminary edition was issued, and is in addition to update packages 2202.003-0001 and 2202.003-0002.

New and replacement pages are provided as follows:

Section	Pages	Action
3	11-12	Replace
4	5-6	Replace
8	11-12	Replace
8	12.1-12.2	Insert
8	13-14	Replace
G	1-31	Replace entire section

Changes to text, tables, and illustrations are marked with a vertical line to the left of the change.

Please file this Bulletin with the publication to retain a record of changes.

Sequence Number: S 08

Publications Bulletin

2202.003-0004

4/30/73

MEMOREX

Computer System Products

Update Package for:

MRX/OS RPG II
Reference Guide
2202.003

This Bulletin advises of changes which have occurred to the **RPG II Reference Guide** since the July 1972 preliminary edition was issued, and is in addition to update packages 2202.003-0001 through 2202.003-0003.

New and replacement pages are provided as follows:

Section	Pages	Action
2	13-14	Replace
3	11-12	Replace
D	1-2	Replace
H	1-2	Replace
H	2.1	Insert

Changes to text, tables, and illustrations are marked with a vertical line to the left of the change.

Please file this Bulletin with the publication to retain a record of changes.

Sequence Number: S 11

Publications Bulletin

2202.003-0005

6/15/73

MEMOREX

Computer System Products

Update Package for:

MRX/OS RPG II
Reference Guide
2202.003

This Bulletin advises of changes which have occurred to the **RPG II Reference Guide** since the July 1972 preliminary edition was issued, and is in addition to update packages 2202.003-0001 through 2202.003-0004.

New and replacement pages are provided as follows:

Section	Pages	Action
1	15-17	Add
3	1-4	Replace
3	7-8	Replace
3	21-22	Replace
6	27-28	Replace
7	33-34	Replace
8	3-4	Replace
G	5-6	Replace
G	13-14	Replace
I	1-10	Add
J	1-5	Add

Changes to text, tables, and illustrations are marked with a vertical line to the left of the change.

Please file this Bulletin with the publication to retain a record of changes.

Publications Bulletin No. 01

11/30/72

MEMOREX

Computer System Products

Update Package **2202.003-0001**, for:

MRX/OS RPG II
Reference Guide
2202.003

This Bulletin advises of changes which have occurred to the **RPG II Reference Guide** since the July 1972 preliminary edition was issued. New and replacement pages are provided where required.

Section	Pages	Action
3	1-2	Replace
3	15-16	Replace
4	1-2	Replace
5	1-2	Replace
F	1-10	Replace entire section
G	1-31	Replace entire section
H	1-2	Replace

Changes to text, tables, and illustrations are marked with a vertical line to the left of the change.

Please file this Bulletin with the publication to retain a record of changes.

INDEX

COMMENTS FORM

MRX/OS RPG II Reference Guide (Preliminary)

Please send us your comments, to help us produce better publications. Use the space below to qualify your responses to the following questions, if you wish, or to comment on other aspects of the publication. Please use specific page and paragraph/line references where appropriate. All comments become the property of the Memorex Corporation.

- Is the material:

	Yes	No
Easy to understand?	☐	☐
Conveniently organized?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Accurate?	☐	☐
Suitable for its intended audience?	☐	☐
Adequately indexed?	☐	☐

- For what purpose did you use this publication (reference, general interest, etc.)?

  _____

- Please state your department's function: _____

  _____

- Please check specific criticism(s), give page number(s), and explain below:

  ☐ Clarification on page(s) _____

  ☐ Addition on page(s) _____

  ☐ Deletion on page(s) _____

  ☐ Error on page(s) _____

_____

_____

_____

MEMOREX

First Class

Permit No. 250
Santa Clara
California 95050

Business Reply Mail

No Postage Necessary if Mailed in the United States

Postage Will Be Paid By

Memorex Corporation

Santa Clara Software Publications
Department 9725 - M/S 00-21
1200 Memorex Drive
Santa Clara, California 95052

Thank you for your information.....

Our goal is to provide better, more useful manuals, and your comments will help us to do so.

.....Memorex Publications