96768710

CONTROL DATA CORPORATION

RPG II
VERSION 2
REFERENCE MANUAL

CDC® OPERATING SYSTEM:
INTERACTIVE TERMINAL-ORIENTED
SYSTEM (ITOS)

REVISION RECORD

REVISION	DESCRIPTION
A	Manual released.
(4/77)	
B	Manual revised; TRACER utility added.
(6/77)	
C	Revised to reflect ITOS 1.1 changes
(10/77)	

Publication No.
96768710

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning this manual to:

Control Data Corporation
Publications and Graphics Division
4455 Eastgate Mall
La Jolla, California 92037

or use Comment Sheet in the back of this manual.

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual, are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

PAGE	REV
Cover	--
Title page	--
ii	C
iii/iv	C
v/vi	B
vii	C
viii	C
ix	C
1-1 thru 1-3	A
2-1 thru 2-7	A
3-1	A
3-2	A
4-1	A
4-2	A
4-3	C
5-1 thru 5-3	A
5-4	C
5-5 thru 5-8	A
5-9	C
5-10	A
5-11	A
6-1 thru 6-6	A
7-1	A
8-1 thru 8-14	A
9-1 thru 9-8	A
9-9	C
9-10 thru 9-18	A
9-19	C
9-20	A
10-1 thru 10-10	A
10-11 thru 10-15	C
11-1 thru 11-3	A
12-1	A
12-2	A
12-3	C
12-4	C
12-5 thru 12-12	A
13-1	A
13-2 thru 13-9	B
13-10	C
A-1 thru A-4	A
B-1	A
C-1 thru C-9	A
D-1 thru D-11	A
D-12	C
D-13 thru D-17	A
E-1	C
E-2 thru E-6	A
E-7	C
E-8 thru E-12	A
E-13	C
E-14 thru E-28	A
E-29 thru E-32	B
E-33 thru E-36	A
F-1 thru F-6	A
G-1	A
G-2	A
H-1	A
H-2	A
H-3 thru H-6	C
H-7	A
I-1	C
I-2	C
I-3	A
I-4	A
I-5	C
I-6	C
J-1	C
J-2	C
J-3	A
Index-1	B
Index-2	B
Index-3	A
Index-4	B
Index-5	A
Index-6	B
Comment sheet	C
Cover	--

PREFACE

This is a reference manual for the CDC® Small Computer Report Program Generator (RPG II) programming language. It is intended for programmers having at least one year's programming experience and having some knowledge of file organizations.

The RPG II compiler operates directly under the Mass Storage Operating System (MSOS 5) which operates through the Interactive Terminal Oriented System (ITOS).

This RPG II compiler system is completely byte-oriented, meaning that word boundaries in computer memory are ignored during compilation and execution of an RPG II program. References to computer memory in this manual involve bytes, characters, and bits rather than words.

For users familiar with IBM's System/3 RPG II, appendix B lists the important differences between this version of RPG II and System/3 implementations.

The following publications are related to RPG II:

Publication	Publication Number
Mass-Storage Operating System (MSOS) Version 5 Reference Manual	96769400
(MSOS) Version 5 Diagnostic Handbook	96769450
Interactive Terminal-Oriented System (ITOS) Version 1 Reference Manual	96768290
Real-Time Operating System (RTOS) Version 3 Reference Manual	96769560
Sort/Merge Version 1 Reference Manual	96769260
Magnetic Tape Utility Processor Reference Manual	96768400

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or undefined parameters.

CONTENTS

APPENDIXES

INDEX

FIGURES

TABLES

Report Program Generator (RPG II) consists of a symbolic programming language and a compiler program. Through the symbolic programming language the programmer describes his program requirements on RPG II specification sheets. These specifications serve as input to the compiler program. The compiler also accepts additional data arranged in tables and arrays. This type of table and array data is fixed information that is unlikely to change for the life of the program. Compiler output includes an executable object program (stored by the compiler on disk) and/or a listing of the specifications plus any error messages and a core map. The programmer can leave the object program on disk to be processed later, or he can proceed directly to executing the object program.

INPUT/OUTPUT

The object program plus data stored in files make up the input to the execution phase. The execution phase can occur repeatedly with different data. The output of that phase is determined by the application. It may include reports, modifications to the input data files, completely new data files, or combinations of these functions.

Input to the compilation and execution phases may come from any input device listed in appendix D. During the execution phase, the object program can perform sequential access with files on any device. In addition, a disk file allows the following:

- Indexed and random access by key or record number
- Access according to a separate file called a record address file

Record address files list the locations of records to be processed and the order in which those records are to be processed.

The programmer has considerable flexibility in the processing of indexed files. An indexed file may be developed through an RPG II program or the RPG II disk utilities (appendix J).

RPG II CYCLE

RPG II operates on the premise that most business applications involving the processing of data files adhere to a fixed sequence of processing steps: namely, read a record, process it, and output some data. These three steps make up the basic RPG II cycle and are repeated for each record in the input files.

SINGLE INPUT FILE PROCESSING

In the simplest RPG II program, a single input file is processed. When an employee payroll report is to be produced, the input file may consist of a single card for each employee. This card might contain the employee's name, the hours worked, and the amount of pay per hour. As part of the desired output, the name and hours worked may be transferred from the card to the report. In addition, the employee's hours would be multiplied by the hourly pay rate to produce the gross pay; this figure would also be shown in the output report. RPG II specifications for this application would contain the following:

- A description of input and output files and records
- A single calculation statement in which the multiplication is performed

Records in the input file would be processed one at a time within the basic RPG II cycle.

With a single input file, at least three areas of complication may be added to the basic cycle:

- Different record types within the file
- Look-ahead fields
- Different report levels

In addition to the input record described above, the input file might contain records listing the home address of each employee as well as records containing any overtime hours worked by each employee. For an input file containing more than one record type, programmers may assign internal switches to be used as *indicators* to *condition* alternate calculation and/or output steps. The home address record could trigger a switch saying "bypass the calculation phase for this record" while the overtime record could set another indicator requesting a time-and-a-half calculation, the result of which would be added to the straight-time product.

With multiple input record types the basic cycle remains the same, but different activities may be programmed to occur within the steps of the cycle. For each employee in the input file a record type *group* is specified. In addition, the program can indicate that the home address record is required with each straight-time record, while the overtime record may be omitted if no overtime was worked this week. (In some applications, any number of a specified record type may be accepted by a program; for example, multiple sales records for a single salesman.)

Look-ahead fields permit the examination of specified fields in the next record during processing of the current record. This is useful in the determination of proper processing of the current record. For example, if employee records are grouped according to company department and if the last record belonging to a department is to be processed uniquely, the look-ahead feature can be used to trigger this unique processing.

The output report may show, in addition to each employee's gross pay, payroll totals for each department within each plant within the company. The breakdowns by department, plant, and company are termed *control levels*, department being the lowest level and company the highest. A change in control level can require different activities: when department number changes, the department total must be reset to zero while payroll accumulation continues at the plant level until the plant identification changes. RPG II allows an indicator to be set automatically when a change in control level (called a *control break*) occurs. The programmer specifies the *control level indicator* to be set on when a programmer-specified *control field* changes.

A change in plant identification implies a change in department. Thus, when a control break occurs at the plant level, that control level indicator, as well as all lower control level indicators (in this example, the department control level indicator), is turned on. This function is performed automatically by RPG II, provided control level indicators are properly specified in the program. When a change in plant identification does occur, the current plant and department totals in the output report should be shown before accumulating fresh totals for the new plant. Thus, when control break processing is specified in the program, the RPG II cycle is expanded to the following sequence:

1. Input a record.
2. Determine if a control break has occurred; if not, go to step 5.
3. Perform total calculations.
4. Perform total output.
5. Perform detail calculations.
6. Perform detail output.

MULTIPLE INPUT FILE PROCESSING

With multiple input files, RPG II processing can become quite complicated. Each file may contain the complexity of a single input file, and the input files may be interrelated as well. The programmer has complete control over the order in which files and each record within each file are processed. When a particular processing order is not specified in the program, the files are processed in the order in which they are specified on the File Description Specification sheets, as if they were strung end-to-end as a series of single input files.

When the files are interrelated, the program probably should not process the files on the end-to-end basis. Rather, the files should be processed on the basis of *matching records*. Record matching is accomplished through the comparison of fields from records in two or more files. When a match occurs, a *matching record indicator* is automatically set on, and a record is selected for processing based upon the order in which the files were specified in the File Description Specification sheets. With no match, a record is selected according to the program's collating sequence for the matching record fields. As with record type and control level indicators, the matching record indicator can be used to condition calculations and output in the program. Returning to the example, record matching can be used to incorporate this week's payroll information into an annual payroll file.

CYCLE SUMMARY

As described above, RPG II distinguishes between functions performed each cycle (detail operations) and functions performed when a control break occurs (total operations). Because the processing of a control group should be completed before starting the processing of a new control group, total operations are performed before detail operations in the RPG II cycle. All input records processed between one control break and the next comprise a control group.

In addition to detail and total operations, there are 1P (first page) and LR (last record) operations. 1P processing is unique to the first cycle. The 1P indicator can be used to condition report headings and is very useful in the proper positioning of preprinted report forms. RPG II automatically turns the LR indicator on when the last input record from the last input file has been reached. This indicator is commonly used to condition summary lines in a report.

The general operations involved in a single cycle are shown in figure 1-1. Appendix F contains detailed information about internal RPG II operation.

VALID RPG II NAMES

The following rules apply to names used in RPG II programs:

- RPG II filenames are from one to eight characters long; RPG II field names are from one to six characters long.
- Field names and filenames are left-justified in the specification fields in which they appear.
- The first character of either a filename or a field name must be alphabetic (see appendix A, Glossary, for a definition of alphabetic characters). The remaining characters are any combination of alphabetic and numeric characters (special characters are not allowed).
- Blanks cannot appear between characters in the name.

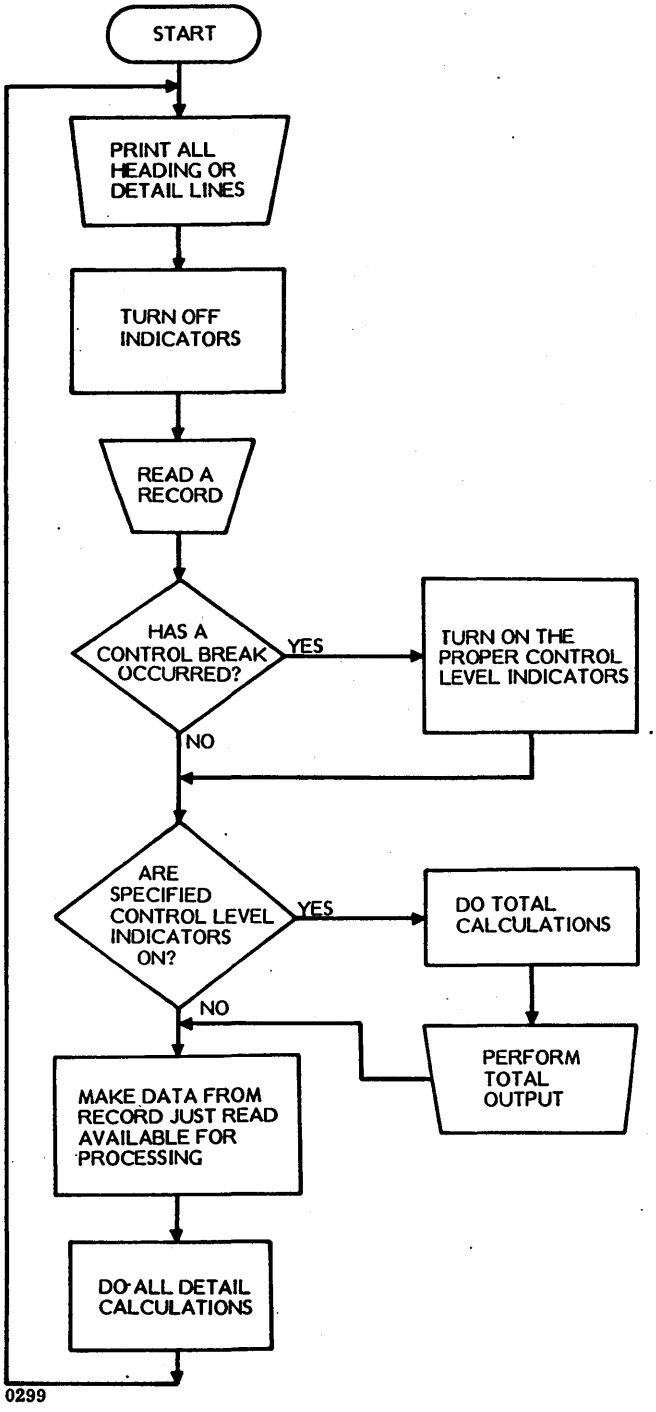


Figure 1-1. General Program Cycle

RPG II SPECIFICATIONS FORMS 2

After the programmer has determined his report requirements (input and output formats and calculations), he codes the information on RPG II specifications forms. Seven types of preprinted forms are available to the programmer. There is a control card for the whole job, and there are six form types for the input, calculation, and output logical specification areas:

- Control Card Specifications (form H) provide information pertaining to the compilation as a whole.
- File Description Specifications (form F) and Extension Specifications (form E) describe files, tables, and arrays to be used by the program.
- Line Counter Specifications (form L) provide information about line printer files produced by the program.
- Input Specifications (form I) and Output Specifications (form O) describe records in the files named in forms F and E.
- Calculation Specifications (form C) describe operations to be performed on the input data.

In addition, the programmer may specify:

- *Compilation time tables* and *arrays* (tables and arrays that do not change for the life of the program)
- Translation tables (according to which characters in the input and output files are changed)
- Alternate collating sequence tables (used to modify the collating sequence of the standard character set)

These three specification types do not have preprinted forms. Table and array formats are defined by the programmer within his program. Each record in the table or array must adhere to the format so defined. Translation and alternate collating sequence tables have fixed formats (see section 11), but they are not entered on preprinted forms because the number of table entries varies from program to program.

Each RPG II program has one optional control card specifications line. The numbers and types of other specifications vary with the application, but RPG II requires that the specifications be presented to the compiler in the order shown in figure 2-1. The ** and /* delimiting records illustrated in the figure are also required in the source program, as appropriate. For convenience, records are illustrated as punched cards. As the figure indicates, the minimum requirement for an RPG II source program is one file description specification record, one input specifications record, and the trailing /* record.

In the logical specification area of input, the programmer describes files, records, fields, and file devices.

In the calculation area, operations to be performed on the input data are described. These operations may be conditioned (for example, by MR, the matching record indicator or by a zero result in the last calculation operation). Calculation operation codes provide arithmetic functions, field movement, the definition of internal subroutines, usage of internal and external subroutines, setting and testing of flags, table lookup, and special input and output.

In the logical specification area of output, the user describes files, records, fields, and file devices. When a file is being updated, specifications in this area augment the input specifications for the file. When a new file is being created, the user also describes record organization and field positions. In addition, when the output file is a listing file, the user specifies any editing to be performed by RPG II.

Figures 2-2 through 2-6 contain copies of the preprinted specifications forms. To the experienced RPG II programmer these pages and the summary of features in appendix C should be a sufficient introduction to this version of RPG II. Sections 3 through 13 contain a detailed description of RPG II functions and usage.

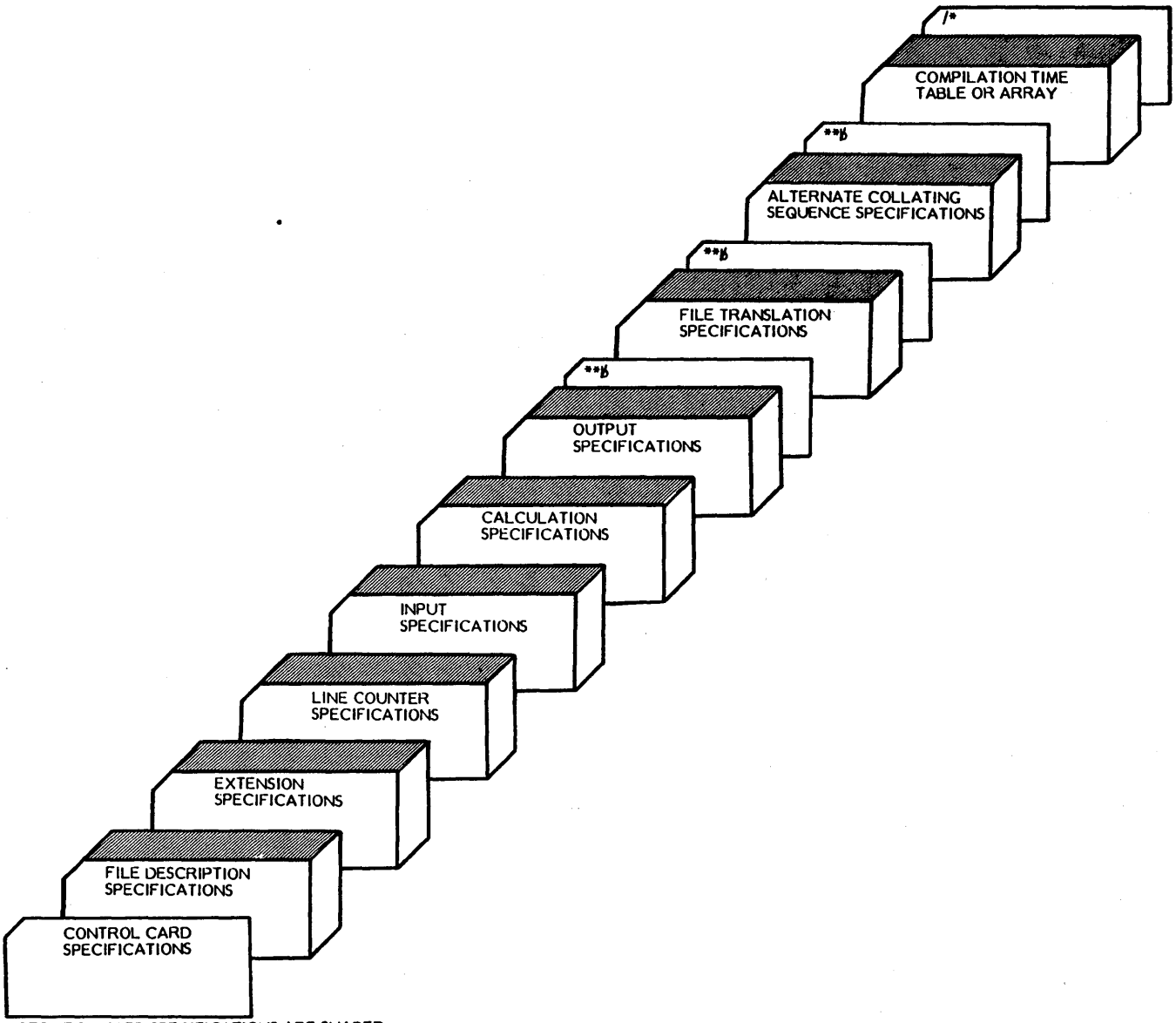


Figure 2-1. Arrangement of the RPG II Source Program

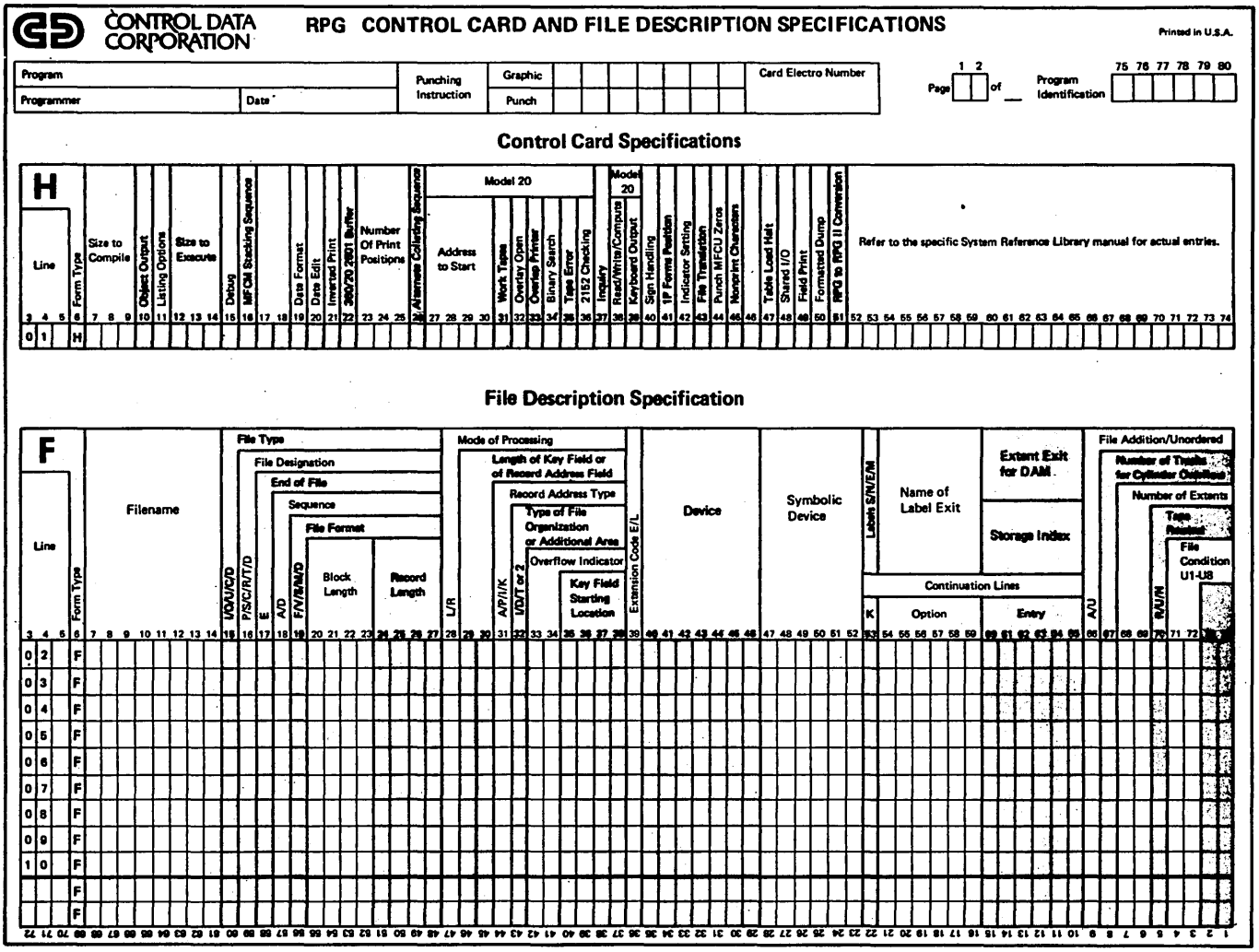


Figure 2-2. RPG Control Card and File Description Specifications Sheet

CONTROL DATA CORPORATION

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

Printed in U.S.A.

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

Extension Specifications

E

Line 3 4 5	Form Type 6	Record Sequence of the Chaining File 7 8	Number of the Chaining Field 9 10	From Filename 11-18	To Filename 19-26	Table or Array Name 27-32	Number of Entries Per Record 33-34	Number of Entries Per Table or Array 35-39	Length of Entry 40-42	P/B/L/R 43	Decimal Positions 44	Sequence (A/D) 45	Table or Array Name (Alternating Format) 46-51	Length of Entry 52-54	P/B/L/R 55	Decimal Positions 56	Sequence (A/D) 57	Comments 58-74
0 1	E																	
0 2	E																	
0 3	E																	
0 4	E																	
0 5	E																	
0 6	E																	
0 7	E																	
0 8	E																	
	E																	
	E																	

Line Counter Specifications

L

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

Figure 2-3. RPG Extension and Line Counter Specifications Sheet

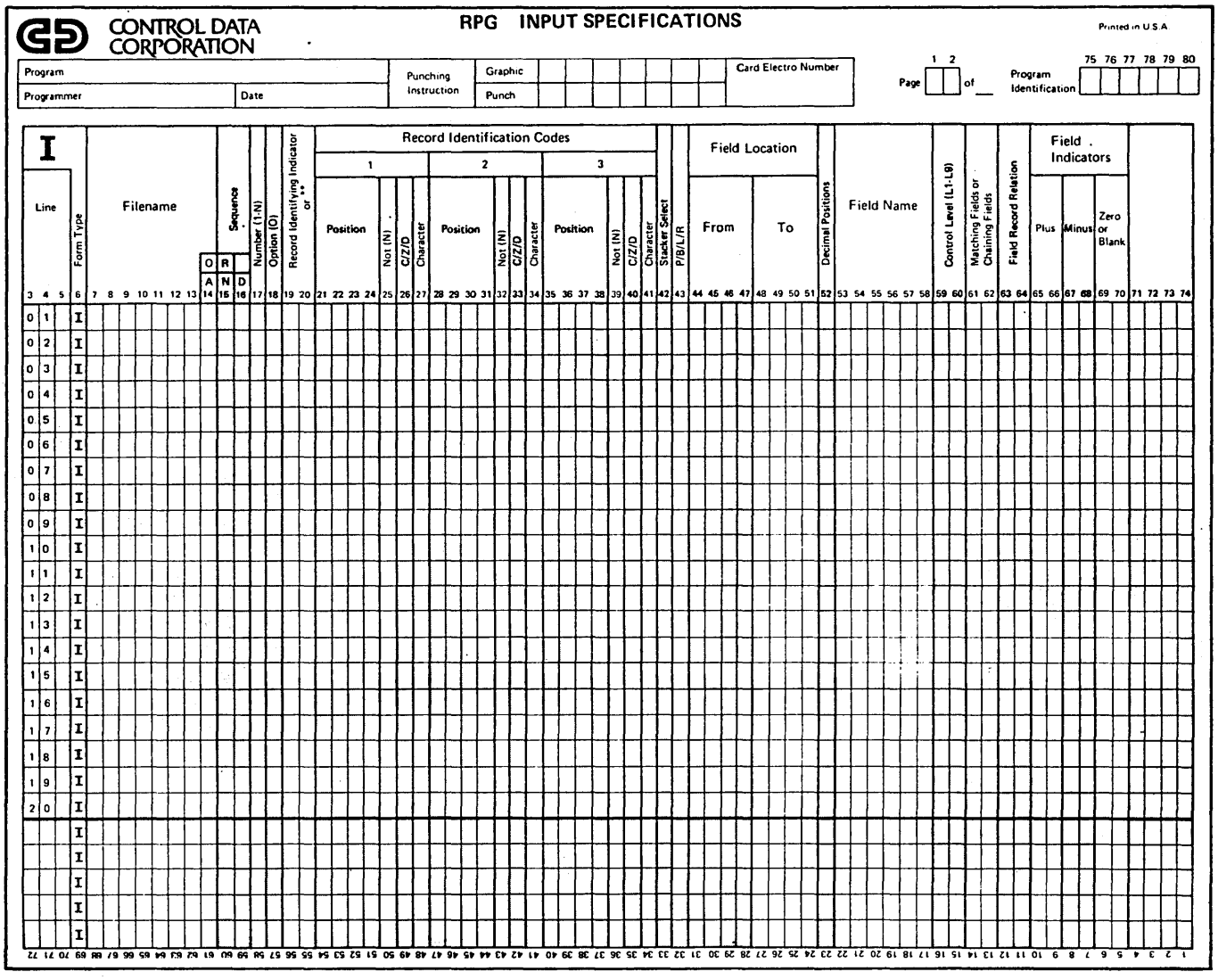


Figure 2-4. RPG Input Specifications Sheet

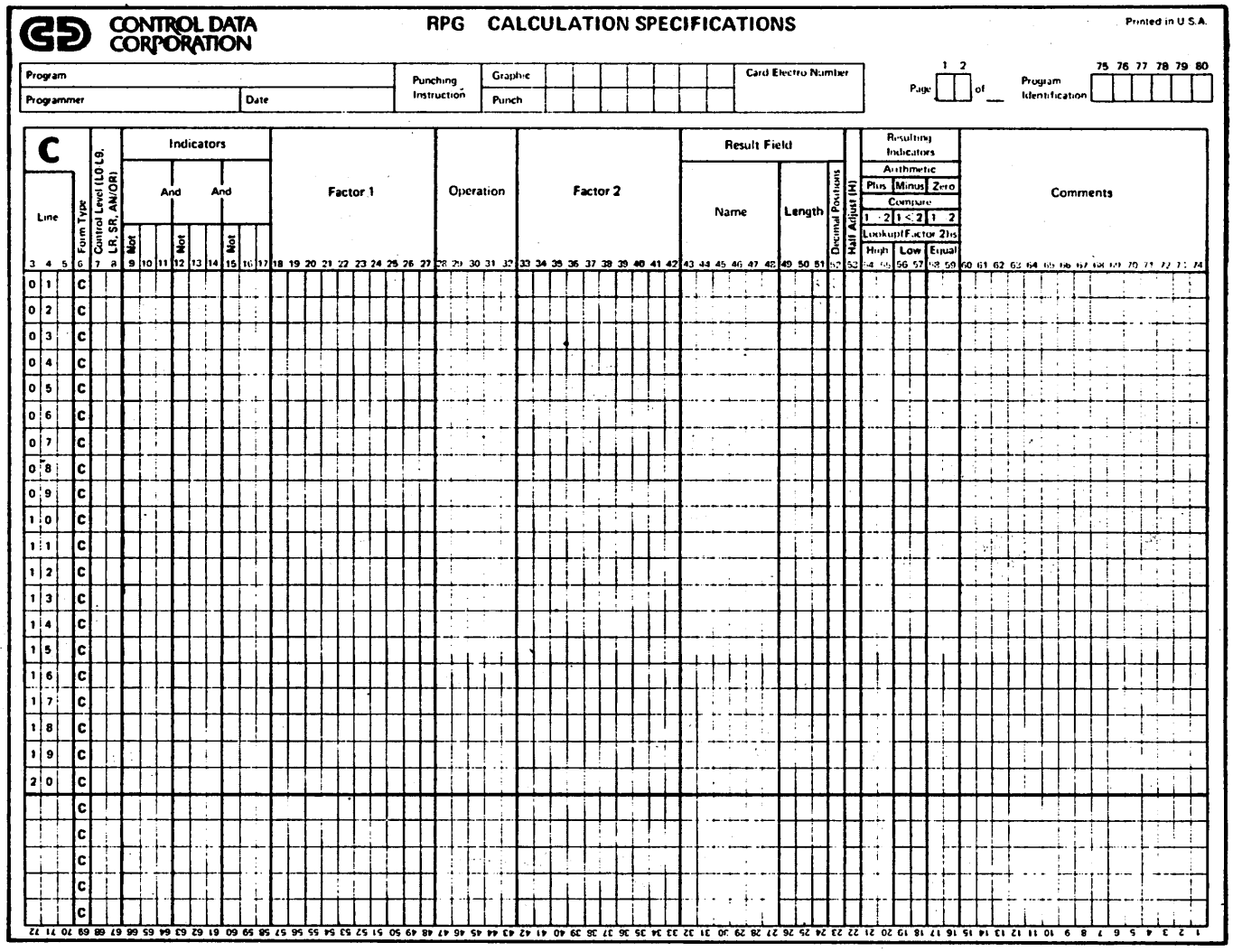


Figure 2-5. RPG Calculation Specifications Sheet

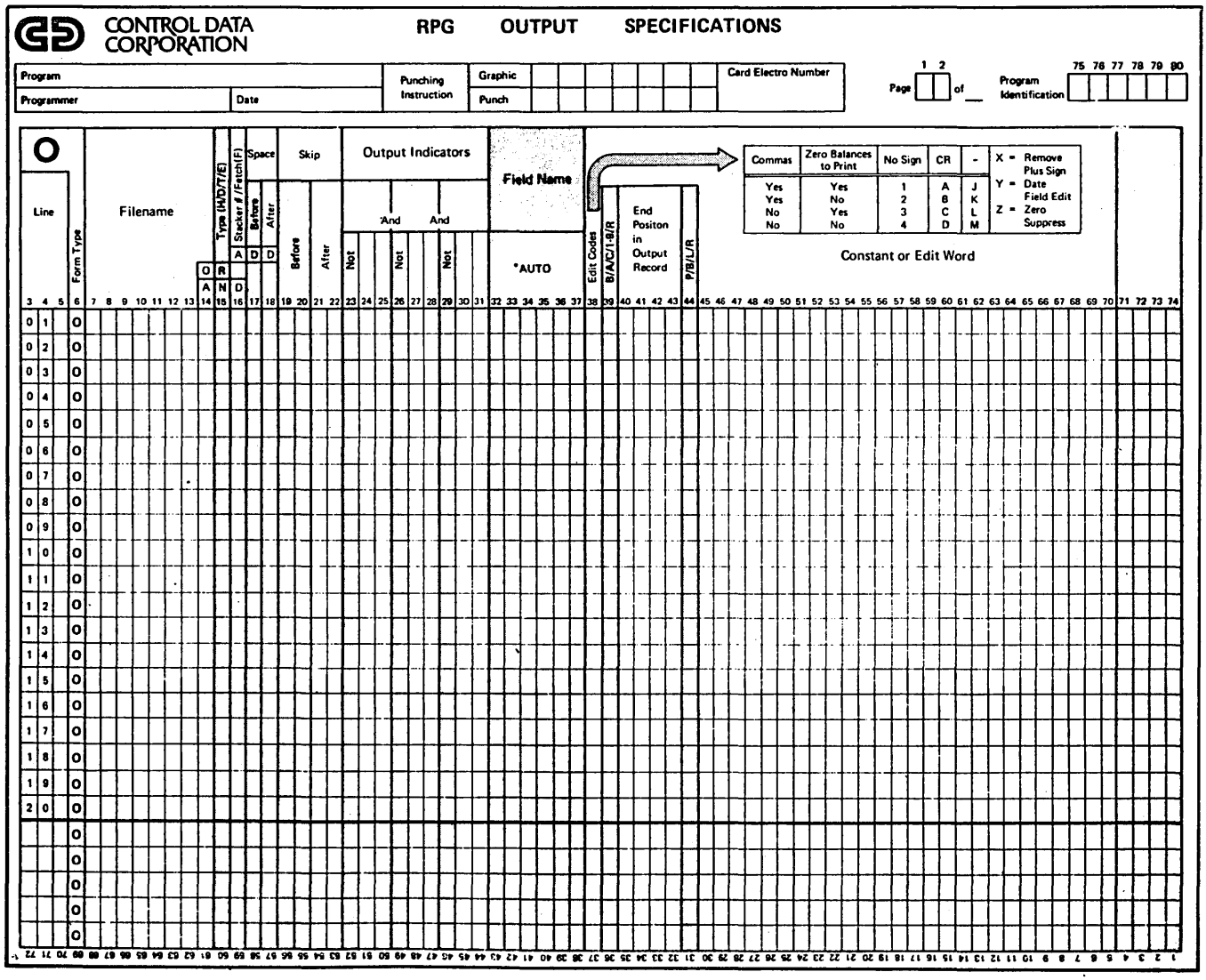


Figure 2-6. RPG Output Specifications Sheet

This section defines entries that are common to all RPG II coding sheet formats except file translation table and alternate collating sequence records, which are described in section 11. The common entries are:

Columns	Entries	
1-2	Page	Page and line together form a five-digit *sequence number.*
3-5	Line	
6	Form type	
7	Comments	
75-80	Program identification	

During processing of the specifications forms, the RPG II compiler checks for proper ordering of the sequence numbers. When proper order is violated, that fact is noted in the printout produced by the compiler but otherwise has no effect on the compilation. Blanks found in columns 1 through 5 are treated as zeros.

PAGE

Columns	Values
1-2	01-99

Columns 1 and 2 in the upper right corner of the specifications forms are used to number the pages of the RPG II program. Number the pages in ascending order. The forms are presented to the compiler in the following order:

1. Control Card Specifications
2. File Description Specifications
3. Extension Specifications
4. Line Counter Specifications
5. Input Specifications
6. Calculation Specifications
7. Output Specifications

A program may require more than one sheet or none of a particular specifications type, but in any case the above order must be maintained.

LINE

Columns	Values
3-5	000-999

The single control card specifications line, if used, is always line 010 of a program. Any other sequence numbers used need not be consecutive, but they should be in ascending order.

FORM TYPE

Column	Values	Meanings
6	H	Control Card Specifications sheet (or 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 Specifications sheet

The form type is preprinted in column 6 of each line of each specifications sheet. The letter identifies the type of specification of each line of coding.

The H in column 6 of the Control Card Specifications form stands for header record. Each RPG II source program has only one header record, and, if used, that record must be the first line of the program.

COMMENTS

Column	Value	Meaning
7	*	Comment line

Any specifications line containing an asterisk in column 7 is entirely commentary. Columns following the asterisk may contain any members of the character set. The compiler performs sequence checking on the line and includes the line in the source program listing, but aside from that the line is ignored.

PROGRAM IDENTIFICATION

Columns	Values	Meanings
75-80	Blanks	RPGOBJ
	Valid RPG II name	Program identification

Columns 75 through 80 of the Control Card Specifications sheet are used to name the object program. This name is used in a program directory to identify the location of a program on disk.

Any combination of alphabetic and numeric characters may be used in columns 75 through 80; however, the first character must be alphabetic. Blanks must not be used between characters in the name. The program name should be unique. If columns 75 through 80 are left blank, the compiler automatically assigns the name RPGOBJ to the object program.

Columns 75 through 80 of all source program specification sheets, except the Control Card Specifications sheet, may contain any characters or may be left blank. These entries are ignored by the compiler but are produced in the source program listing.

Only one control card may be submitted for each program. It provides information about the program and the system to the RPG II compiler. If the control card is omitted, a blank control card is assumed. For coding the control card, one specifications line is provided on the Control Card and File Description Specifications sheet (figure 4-1). The remainder of this section describes the columns of form H that are used. Any column that is not specifically described must be filled with a blank.

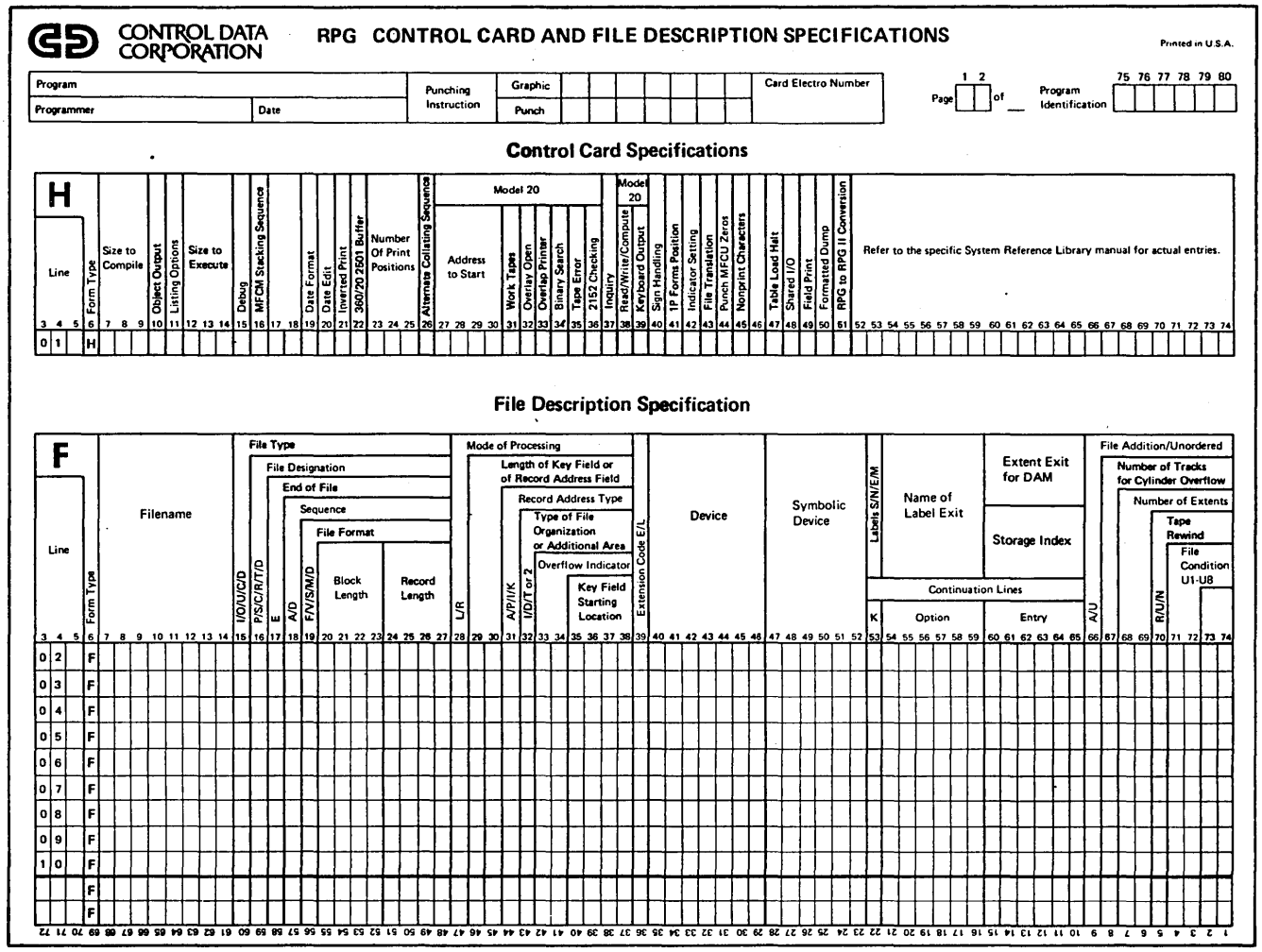


Figure 4-1. Control Card and File Description Specifications Sheet

DEBUG

Column	Values	Meanings
15	Blank	Do not perform DEBUG operations.
	1	Perform requested DEBUG operations.

This specification in column 15 indicates:

- Whether DEBUG operations are to be performed by the object program, and
- Whether lines containing the DEBUG operation code are diagnosed as errors in the program listing

As the name implies, the DEBUG operation code in the calculation specifications is a tool to be used in checkout of the object program. Intermediate debugging output is produced only when requested by this specification and a DEBUG operation code on form C.

When checkout of the object program has been completed, changing this entry to a blank eliminates the then extraneous debugging output; i.e., changes need not be made to the Calculation Specifications sheet.

INVERTED PRINT

Column	Values	Meanings
21	Blank	Domestic format
	I	Foreign format
	J	Foreign format (leading zero remains for zero balances)
	D	United Kingdom format

Column 21 is used to describe the format and punctuation used for numeric literals on the Calculation Specifications sheet, the order of the system date (referenced by UDATE) field, and edit codes used on output.

NOTE

The input for UDATE must be in the format expected as output. For example, if D (United Kingdom format) is specified in column 21, the input format must be dd/mm/yy (where dd=day; mm=month; yy=year).

Inverted print specifications and resulting formats are shown in table 4-1.

ALTERNATE COLLATING SEQUENCE

Column	Values	Meanings
26	Blank	Use normal collating sequence.
	S	Use alternate collating sequence.

Every alphabetic, numeric, or special character holds a special position in relation to all other characters. This special order is known as the *collating sequence*. The normal (default) collating sequence is based on the way characters are represented in the machine and is listed in appendix D.

If characters are to appear in a sequence other than the normal sequence or if two or more characters are to have the same position in the sequence (this means they are considered equal), an alternate collating sequence must be described.

TABLE 4-1. INVERTED PRINT SPECIFICATIONS

Inverted Print Option	Numeric Literal using Period/Comma as a Decimal Point	Edit Codes using Period/Comma as a Decimal Point	Zero Suppress to the Left/Right of the Decimal Point	UDATE Appears with a Slash/Period
Blank	1234.56	1,234.56	.10	mm/dd/yy
D	1234.56	1,234.56	.10	dd/mm/yy
I	1234,56	1.234,56	,10	dd.mm.yy
J	1234,56	1.234,56	0,10	dd.mm.yy
mm=Month dd =Day yy =Year				

An alternate collating sequence applies to matching fields, sequence checking of files, and alphanumeric compare operations (COMP operation code). The alternate sequence does not apply to numeric comparisons, lookup operations, and control levels. (Refer to section 11 for additional details on alternate collating sequences.)

An alternate collating sequence table is printed out with the compiled program. The order of an RPG II source program including an alternate collating sequence table is shown in figure 2-1.

1P FORMS POSITIONING

Column	Values	Meanings
41	Blank	First 1P line is printed only once.
	1	First 1P line can be printed repeatedly.

This entry in column 41 is used only when the first output line conditioned by the first page (1P) indicator is written to a printer file. Ordinarily it is used when the output is going to preprinted forms, such as check blanks, where the proper alignment of forms is very important. When requested, forms positioning allows the operator to correctly position the forms by first printing a test line.

FILE TRANSLATION

Column	Values	Meanings
43	Blank	No file translation is needed.
	F	Input, output, update, and/or combined files are to be translated.

An F character in column 43 indicates the information contained in an input, output, update, or combined file is in a form not usable by the object program. For example, this option is used when an input file was produced by a computer that has a different character set than the one defined in appendix D.

With an input file, translation occurs before the data is processed by the object program. For output files, translation occurs as the output data is produced. When an update or combined file is translated, the conversion is performed for both the input and output phases for the file.

Note that since an external medium is implied in file translation, the internal collating sequence of the object program is not affected by the translation.

Refer to section 11 for further information on file translation.

SHARED I-O

Column	Values	Meanings
48	1	All disk files share a single input/output area.
	Blank	Each disk file uses a separate input/output area.

Column 48 applies to disk files only. Normally an RPG II program uses one input/output area for each file. An entry in this column allows all disk files to use one input/output area. Specifying a shared input/output area reduces the amount of core storage needed by the object program. This is particularly important if a program is so large that it cannot run in the available core storage. However, the use of a shared input/output area increases the time required to execute the object program. Therefore, it is a good idea to make sure that the program exceeds the capacity of the system before specifying shared input/output.

NOTE

A shared input/output area cannot be specified for multivolume file (entry greater than 01 in columns 68 and 69 of the File Description Specifications sheet).

Additional input/output areas (entry in column 32 of the File Description Specifications sheet) cannot be specified for disk files using a shared input/output area.

INTERNAL CODE SPECIFICATION

Column	Values	Meanings
50	Blank	Internal code is EBCDIC
	Non-blank	Internal code is ASCII

The normal internal RPG II code is EBCDIC. If the ASCII option is selected, execution is not compatable with the IBM system 3.

EXECUTION IN ITOS ENVIRONMENT

Column	Values	Meanings
51	Blank	RPG II object code is provided in ITOS-compatible form.
	Non-blank	Object code is provided for MSOS batch-type execution.

MSOS batch type code should be used only for systems that lack any ITOS capability; this type of code should not be used in an ITOS environment under the job processor.

File description specifications are required for every file used by the object program. The File Description Specifications sheet is illustrated in figure 4-1. Generally, only one line is needed to describe a file.

Form F describes all files except compilation time tables and arrays; these are described only on form E. Records in form F files are described on other specification sheets:

- Input data records are described in the input specifications (form I).
- Output data records are described in the output specifications (form O).
- Update and combined files are described on both forms I and O.

Record address files and pre-execution and execution time tables and arrays are further described on form E, Extension Specifications. Display file records need no further descriptions.

FILENAME

Columns	Values
7 - 14	Any valid RPG II filename

Columns 7 through 14 are used to assign a unique filename to every file used in the object program. Every file must be named, with the following exceptions:

- Compilation time tables and arrays do not require a filename.
- If multiple tables or arrays are read in at pre-execution time from the same device, only one filename is required.

Filenames may duplicate field names.

FILE TYPE

Column	Values	Meanings
15	I	Input file
	O	Output file
	U	Update file
	C	Combined file
	D	Display file

The file type entry in column 15 indicates whether the object program is to read data from the file, write data to the file, or both.

INPUT FILES

Input files are records that a program uses as a source of data. When an input file is described in a program, it indicates that records are to be read from the file. All input files except table, array, and record address files must be further described on the Input Specifications sheet. Table/array and record address files must be further described in the extension specifications.

OUTPUT FILES

Output files are records written, punched, or printed by a program. All output files, except table and array output files, must be further described on the Output Specifications sheet.

UPDATE FILES

Update files are disk files from which a program reads a record, updates fields in the record, and writes the record back in the location from which it was read. Update files must be further described on both the Input and Output Specifications sheets; however, only the fields to be updated need be described on the Output Specifications sheet. A record in an update file can be updated only once during a cycle. A chained file or a demand file may be updated at detail time, total time, or at exception output time. All other disk files should be updated only at detail time (during the same program cycle in which a record is read) to avoid unpredictable results.

COMBINED FILES

A combined file is both an input and an output file. A program reads records from a combined file and includes output data on the records in the file. The result is one file that contains both input and output data.

An output record to a combined file is stored in a hold area until another record is read from the combined file. The output record is produced just before the new record is read. Each record stored in the hold area overlays and replaces any record previously stored in the hold area. Thus, only one record can be available as output to a combined file during any one RPG II cycle. This record is the last record that satisfied output conditions during the cycle. The RPG II

program should be written in such a way that only one record of a combined file satisfies output conditions in a single cycle. In addition, output should not be conditioned in such a way that output conditions are satisfied for more than one combined file in a single cycle.

DISPLAY FILES

A display file is a collection of information from fields used by a program. The DSPLY operation code must be used in the calculation specifications in order to print a field or record directly from storage and/or key data into a field or record in storage. Display files need only be described on the File Description Specifications sheet.

FILE DESIGNATION

Column	Values	Meanings
16	Blank	Display file or nonchained output file
	P	Primary file
	S	Secondary file
	C	Chained file
	R	Record address file
	T	Table or array file (pre-execution time)
	D	Demand file

Column 16 is used to further identify the use of input, update, and combined files. The column is left blank for display files and all output files except chained output files (direct load).

PRIMARY FILES

A primary file is the main file from which a program reads records. In multifile processing, the primary file is used to control the order in which records are selected for processing. (More information on record selection in primary files can be found in section 13.) A primary file can be an input, update, or combined file. In programs that read records from only one file, that file is the primary file. Every program must have only one primary file.

SECONDARY FILES

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

Note that table, array, chained, record address, and demand files are not involved in record selection in multifile processing.

CHAINED FILES

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

- Read records randomly
- Load a direct file

A chained file can be an input, output, or update file.

RECORD ADDRESS FILES

A record address file is an input file that indicates to the object program:

- Which records are to be read from a disk file
- The order in which the records are to be read from the disk file

Only one record address file can be specified in a program. A record address file must be further defined on the Extension Specifications sheet.

Record address files contain record key limits or relative record numbers in binary format. Files that contain limits are used with indexed files only. Record address files on disk that contain binary relative record numbers are called address output (ADDROUT) files. They are produced by disk sort programs and can be used with any type of disk file.

TABLE OR ARRAY FILES

A table or array file is a sequential input file that contains table or array entries. The entries can be read into the program during compilation or immediately before execution of the program. Only pre-execution time table or array files are defined on the File Description Specifications sheet. However, all tables and arrays must be described on the Extension Specifications sheet.

A table or array output file (produced after LR output) is defined as a normal output file and does not require an entry in column 16.

Entries read during compilation become a permanent part of the program. Both compilation and pre-execution time tables or arrays can be changed at execution time. Compilation time tables or arrays, however, can be permanently altered only by recompiling the program. Pre-execution time tables can be permanently altered each time the program is executed.

Table and array files are not involved in record selection and processing. They are only a means of supplying entries for tables and arrays used by the program. When pre-execution

time table or array files are read prior to the execution of the program, the program reads all the entries from the table and array files before it begins record processing.

DEMAND FILES

Demand files can be input, update, or combined files. A demand file is accessed only after a READ operation code in the calculation specifications is executed, not during the normal record selection process. Demand files can be processed either sequentially by key or consecutively. More than one demand file may be designated; the operand of the READ specifies which of several demand files is meant.

END-OF-FILE

Generally, execution of the object program continues until the end-of-file has been reached on all primary, secondary, and record address files for which an E has been specified in this column of the File Description Specifications sheet.

Column	Values	Meanings
17	Blank	a. If none of the primary, secondary, or record address files has an E in its specification, each such file is read to its end. b. If any has an E in its specification, those with a blank in column 17 need not be exhausted.
	E	All records from this file must be processed before the program can end.

This column applies only to input, update, and combined files that are used as primary, secondary, or record address files.

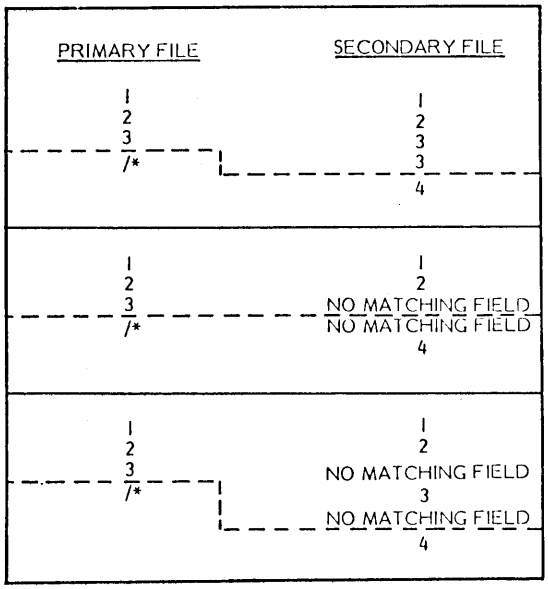
In single file processing, the primary file is read to its end regardless of the entry in column 17. Neither is column 17 significant to the data file associated with a record address file.

In multifile processing, all pertinent files are processed to end-of-file unless some files do and some files do not have an E in column 17. In that case only those files having the E specifications are necessarily processed to end-of-file. There is an important exception. When the primary file is the only file with an E in column 17 and matching records have been specified between that file and at least one secondary file, execution is terminated only after:

- All secondary records that match the last primary record have been processed, and/or
- The first secondary record without matching fields has been processed.

These situations are illustrated in figure 5-1.

In summary, if the records from all files must be processed, column 17 must be blank or contain Es for all files.



NOTES: 1. NUMERIC VALUES SHOW CONTENTS OF MATCH FIELDS.
2. RECORDS ABOVE DOTTED LINE ARE PROCESSED BEFORE END-OF-JOB.

0301

Figure 5-1. End-of-File Processing (Letter E Designated for Primary File Only)

SEQUENCE

Column	Values	Meanings
18	Blank	No sequence checking is to be done.
	A	Sequence checking is to be done. Records in the file are in ascending order.
	D	Sequence checking is to be done. Records in the file are in descending order.

Column 18 is used to indicate whether or not the object program is to check the sequence of records.

This column applies to input, update, or combined files used as primary or secondary sequential files. Sequence checking is required when matching fields are used in the records from the file (entry in columns 61 and 62 of form I). Files with matching fields must be either all ascending or all descending. An equal condition does not violate the sequence of a file.

This column does not apply to table, array, chained, demand, record address, or output files.

FILE FORMAT

Column	Values	Meanings
19	F	Fixed-length records
	V	Variable-length records (EBCDIC tape files only)
	D	Variable-length records (ASCII tape files only)

An F entry in column 19 indicates all records in the file are the same length (i.e., have the same number of characters). A V entry indicates records in a tape file are variable-length EBCDIC, and a D entry indicates records in a tape file are variable-length ASCII.

Enter an F in column 19 for any file that does not reside on magnetic tape.

BLOCK LENGTH

Columns	Values	Meanings
20-23	Blank	Block length for this file is the same as record length.
	1-9999	a. Record length or multiple of record length for disk file b. Record length for non-disk or nontape file
	18-9999	Tape file block length

The use of columns 20 through 23 is interrelated with the device assigned to the file. Charts of record and block lengths appropriate for each device in the system are listed in appendix D.

When a block length is specified, the length is right-justified in columns 20 through 23, and leading zeros may be omitted.

Criteria to be used in the specification of block lengths for disk and tape files are discussed below.

DISK FILES

A block length specifies the number of characters to be read from or written to a file at one time and must therefore be a multiple of the file's record length. The maximum block length for any file is 9999 characters.

The function of a block length entry is to specify the amount of core storage to use for an input/output area. Most users will leave this entry blank, as the RPG II compiler automatically assigns an efficient block length to all files.

TAPE FILES

The block length for EBCDIC tapes with fixed-length records (F in column 19) is a multiple of the record length. For ASCII tape files having fixed-length records, the block length is a multiple of the record length plus the length of the buffer offset (see Continuation Lines).

For tapes with variable length records, the block length specified need not be an exact multiple of record length. However, the following things should be remembered in the block length specification:

- Add four bytes (characters) per block and four bytes per record.
- The minimum specification is the length of the longest record plus eight (four for one block and four for one record).

RECORD LENGTH

Columns	Values	Meanings
24-27	1-8191	Number of characters per record
	18-8191	Number of characters per tape record

A record length in columns 24 through 27 is right-justified, and leading zeros may be omitted.

Except for tape files having variable length records, all records in one file must be the same length. The length of the longest record is specified for tapes having variable length records.

For an update file, record length must not vary between the input and output phases.

The maximum record length allowed for a file depends upon the device assigned to the file (appendix D). A specified record length can be shorter than the maximum length but no longer.

MODE OF PROCESSING

Column	Values	Meanings
28	Blank	a. Access is consecutive, or b. Access is sequential, in the order the records appear in an index (sequential by key).
	L	Access is sequential, within limits defined by a record address file with record key limits.

Column	Values	Meanings
28	R	a. Access is random by relative record number, or b. Access is random by key, or c. Access is by ADDROUT file, or d. This is a direct (random) file load.

Column 28 is used to indicate the method by which records are to be read from the file or to indicate that a direct file load (random load) is to take place.

An entry is made in this column only if:

- This is an input, update, or combined file.
- This is a primary, secondary, demand, or chained file.
- The file resides on disk.

All other types of files have sequential organization, and consecutive processing is the only possible method.

Users must coordinate the file type (column 16) with the file processing mode (column 28), the record address type (column 31), and the file organization (column 32) for all input, update, and combined files. Legal specifications for the retrieval of records from disk files are indicated in table 5-1.

TABLE 5-1. SPECIFICATIONS IDENTIFYING METHODS FOR RETRIEVING RECORDS

File Designation	File Organization	Possible Access Modes	Allowable Entries (Column 16 / 28 / 31 / 32)
Primary, secondary, demand	Sequential or direct	Consecutive	P ␣ ␣ ␣ or 1-9 S ␣ ␣ ␣ or 1-9 D ␣ ␣ ␣
		By ADDROUT file	P R I ␣ or 1-9 S R I ␣ or 1-9
	Indexed	By ADDROUT file	P R I I S R I I
		Sequential by key	P ␣ A/P I S ␣ A/P I D ␣ A/P I
		Sequential within limits	P L A/P I S L A/P I D L A/P I
Record address	Sequential or direct	Consecutive (record key limits)	R ␣ ␣ ␣
		Consecutive (ADDROUT)	R ␣ I T
Chained input	Sequential or direct	Random by relative record number	C R ␣ ␣
	Indexed	Random by key	C R A/P I

NOTE: ␣ = Blank

CONSECUTIVE PROCESSING

The consecutive method applies to sequential and direct files as well as indexed input files. During consecutive processing, records are read in the order they appear in the file; for example, one card after another. A direct organization is similar to sequential, but gaps may occur in the file to allow for future additions. (One might simulate direct organization by inserting blank cards every so often in a card deck.) The gaps are really records filled with blank characters. The user should check for these blank records in his program.

Consecutive processing takes place until the file ends or until the program ends for another reason (see End-of-File).

SEQUENTIAL PROCESSING BY KEY

Sequential by key access applies to indexed files that are used as primary, secondary, and demand files. Sequential by key access is similar to consecutive access in that the index entries are processed consecutively. However, the associated data portion of the file is processed according to the order designated by the index. The index portion of a file is similar to the index of a book; key words are scattered throughout the book but are listed alphabetically along with their locations in the index.

The program reads records until the file ends or the program ends due to end of another file (see End-of-File).

SEQUENTIAL PROCESSING WITHIN LIMITS

Sequential within limits access can be performed through the use of either:

- A record address file containing limits, or
- The SETLL operation code on the Calculation Specifications sheet

The first use applies to indexed disk files used as primary, secondary, or demand files. A limits record contains the lower and upper limits of key values for records to be processed. This organization effectively divides an indexed file into a series of subfiles, each of whose record keys fall within a set of limits. A record in the file is actually accessed sequentially by key but only if the record's key falls within the specified limits. Each set of limits is a record of a record address file.

The second use applies only to indexed files used as demand files and cannot be used with a file for which a record address file also sets limits. The SETLL operation code sets the lower limit only, but access of data records proceeds as described above, with the upper limit being defaulted to the address of the last record in the file. The lower limit may be reset before the end of the file is reached.

The following steps are involved in sequential access within limits:

1. A limits record is read from the record address file, or a lower limit is set by the SETLL operation code.
2. Data records having keys greater than or equal to the lower limit and less than or equal to the upper limit are read and processed (the upper limit being end-of-file when the SETLL operation code has been executed).

These two steps are repeated until either the end of the record address file is reached or the program is terminated due to the end of another file (see End-of-File).

A record address file containing limits must conform to these rules:

- Each record in the record address file contains only one set of limits.
- A record key can be from 1 to 29 characters in length. Both the lower and upper limits must be equal in length to the key field length specified in columns 29 and 30. Thus, the length of a record in the record address file must be twice the length of a record key.
- The lower limit must begin in character position 1 of the limits record. The higher limit immediately follows the lower limit in the record.
- Numeric limits are right-justified in their record fields, but leading zeros must not be suppressed.
- Alphanumeric record keys may contain blanks. A record key must not contain any X'FF' characters, however.

A record address file containing limits and the associated file of keys may have different formats. For example, one file may be in packed decimal format while the other is in unpacked decimal format. During execution RPG II automatically changes the format of the record address file containing limits to that of the object file, if necessary. The format of each file is designated by an A or P in column 31. The unpacked key length must be twice the packed key length minus one or two. Refer to Record Address Type and Packed Decimal Format, section 6, for further clarification of these specifications.

The record address file may contain duplicate limit sets. Thus, the data records can be processed more than once, with or without intervening record retrieval, in a single execution of the object program.

When an upper and lower limit in one set are equal, only one data record is processed via that set.

PROCESSING BY ADDROUT FILE

Address output (ADDROUT) files are record address files produced by disk sort programs. ADDROUT files are similar to indexes for indexed files, but they do not contain keys. A single object file (the data file pointed to) can be processed by one ADDROUT file in one execution time and a totally different ADDROUT file in another execution time.

An ADDROUT file may be a tape or disk file, but the object file must reside on disk. The ADDROUT file contains 3-byte disk records or 18-byte tape records, either of which contains binary *relative record numbers*† of the object file's records. During processing the relative record number is converted to a disk address, and the data record at that address is read.

Processing by ADDROUT file continues until either the end of the ADDROUT file is encountered or the program ends due to the end of another input file (see End-of-File).

RANDOM PROCESSING

Random processing applies to chained files and occurs only when a CHAIN operation code in the calculation specifications is executed. The operand of a CHAIN operation designates the chained file from or to which records are to be read or written.

For sequential and direct files, records are identified by relative record number.

With indexed files, record processing is based on information from the key field of a record. Indexed files may also be accessed by relative record number if they are input files.

Since records from chained files are read at the time of execution of a CHAIN operation code (in the calculation phase), it is simple to process chained update files. A record can be read at total calculation time, fields of the record can be modified in ensuing calculation steps, and the updated record can be returned to the file at total output time. The same sequence of update events can be programmed to occur during detail calculation and detail output time.

LENGTH OF KEY FIELD OR RECORD ADDRESS FIELD

Columns	Values	Meanings
29-30	Blank	Sequential or direct file or chained input file accessed by relative record number
	1-29	Length of record key
	3	Length of ADDROUT file record

Columns 29 and 30 apply only to indexed disk files and record address (including ADDROUT) files. Enter one of the following:

- The length of record keys in record address files that contain limits. Unpacked decimal format is implied, and the maximum entry in columns 29 and 30 is 29 (bytes). Remember that the length of records in the record address file is twice the length of the record keys.
- The length of record keys in indexed files. The maximum entry is 29 if the keys are in unpacked decimal format or 8 if the format is packed decimal. All of the key fields in the records of an indexed file must be the same length.
- The length of records in ADDROUT files. Three is the only possible entry here for ADDROUT files.

An entry is right-justified in these columns, and leading zeros may be omitted.

RECORD ADDRESS TYPE

Column	Values	Meanings
31	Blank	Records in this file are not accessed by key or record address file; i.e., sequential, direct, or consecutive processing requested.
	A	Records in this file are accessed by record keys having unpacked decimal or alphanumeric format.
	P	Records in this file are accessed by record keys having packed decimal format.
	I	This file is accessed via an ADDROUT file, or this is an ADDROUT file.

† Relative record numbers identify the positions of records relative to the beginning of the file; for example, the relative number of the second record of a file is 2.

Column 31 applies to indexed disk files specified as input, update, or chained output files. Columns 28 and 31 together indicate:

- The method through which records in a file are accessed
- Whether this is a direct file load

When a chained output file with packed keys is being created, packed format must also be specified for the keys in the output specifications.

FILE ORGANIZATION OR ADDITIONAL I/O AREA

Column	Values	Meanings
32	Blank	Sequential or direct file. Use one input/output area (buffer) for the file.
	I	Indexed file
	T	ADDROUT file
	1-9	Sequential or direct file. Use two buffers for the file.

Column 32 has two distinct functions:

- To declare the organization of a file as indexed, sequential, direct, or ADDROUT
- To request double buffering for a sequential or direct file

File organizations were described in Mode of Processing and are discussed further in section 13.

The use of two input/output areas, also known as double buffering, leads to faster execution of the object program. Any of the digits 1 through 9 requests double buffering, but 2 is preferred for a more readable source program.

OVERFLOW INDICATOR

Columns	Values	Meanings
33-34	Blank	a. This is not a printer file. b. An overflow indicator is not assigned to this printer file.
	OA-OG, OV	The specified overflow indicator is used to condition records in this printer file.

This entry allows the user to assign an overflow indicator to a printer file.

Columns 33 and 34 apply only to printer output files. Only one of the indicators OA, OB, . . ., OG, or OV can be assigned to a file, and an assigned indicator should be unique to its printer file.

An overflow indicator is commonly used to condition the production of heading lines (records), and it is also associated with the area of the printer page starting with the overflow line and ending with the bottom of the page. The indicator is turned on during execution of the object program when a space or skip terminates in the overflow area. It may also be turned on through the execution of a SETON operation code. See Overflow Line, section 7, and Stacker Select/Fetch Overflow, section 10, for further discussion of overflow indicators.

KEY FIELD STARTING LOCATION

Columns	Values	Meanings
35-38	Blank	This is not an indexed file.
	1-9999	This is an indexed file, and the key field begins in this character position of each record.

An entry is required in columns 35 through 38 for indexed files; the columns must be left blank for any other type of file.

The key field starting location identifies the first (high-order) position of the key field for every record in the file. That position must be the same for every record in the file. The key field contains the information that identifies a record and is used in the index portion of the indexed file.

A key field must not contain any X'FF' characters. This should be taken into consideration by the user when the key field has binary format or when an RPG II program is generating an indexed file.

A key field starting location is right-justified in these columns, and leading zeros may be omitted.

EXTENSION CODE

Column	Values	Meanings
39	Blank	Extension specifications and line counter specifications have not been prepared for this file.
	E	Extension specifications further describe this file.
	L	Line counter specificaitons further describe this printer file.

The Line Counter Specifications sheet can be used to further describe printer files.

All record address files and table and array files must have further definitions on the Extension Specifications sheet.

DEVICE

Columns	Values	Meanings
40-46	See appendix D	Device assigned to this file

Each file in the program is associated with a physical device. Appendix D lists the device mnemonics to be entered in these columns. The name is left-justified (starts in column 40) in this specification.

SYMBOLIC DEVICE

Columns	Values	Meanings
40-52	Blank	Use the logical unit for this device shown in appendix D.
	1-99	Use this logical unit.
	100-255	Use this address as the location containing this logical unit.

This entry allows the user to assign a logical unit other than that shown in appendix D.

CONTINUATION LINE

Column	Values	Meanings
53	Blank	This is not a continuation line; i.e., this is the first (or only) specifications line for this file
	K	This is a continuation record.

As the title of the entry implies, a continuation line is a line of additional specifications for a TAPE or SPECIAL file. (TAPE and SPECIAL are valid entries in columns 40 through 46.) One or two continuation lines can be supplied for a TAPE file, but a SPECIAL file can have only one continuation record. Continuation lines must immediately follow the initial specifications line for a file.

When a K is entered in column 53, an entry is required in columns 54 through 59 (Continuation Lines Option) and may be required in columns 60 through 65 (Buffer Offset Length).

CONTINUATION LINES OPTION

Columns	Values	Meanings
54-59	Blank	This is not a continuation line.
	ASCII	This tape file has ASCII format.
	BUFOFF	This ASCII tape input file contains a block prefix.
	Table or array name	This table or array is referenced by a user-written external subroutine. The table or array cannot be named ASCII or BUFOFF.
	EBCDIC	This tape file has EBCDIC format

BUFOFF is only specified for ASCII files. Therefore, if columns 54 through 59 of this continuation line contain BUFOFF, the same columns must say ASCII in the immediately preceding specifications line. The use of BUFOFF also requires an entry in columns 60 through 65 (Buffer Offset Length).

An entry in columns 54 through 59 is left-justified.

BUFFER OFFSET LENGTHS

Columns	Values	Meanings
60-65	Blank	This is not a continuation line having BUFOFF as the continuation lines option
	0-99	This is a continuation line, BUFOFF has been specified, and this is the length of the block prefix

An entry is required in columns 60 through 65 if BUFOFF has been specified in columns 54 through 59 of this continuation line.

BUFOFF and buffer offset length must not be specified for ASCII tape output files.

The offset length (the number of bytes by which records are offset within the buffer) is right-justified in these columns, and leading zeros may be suppressed Note that since an entry is required here if columns 54 through 59 say BUFOFF, a zero is required here if the block prefix is empty.

NAME OF LABEL EXIT

Columns	Values	Meanings
54-59	Blank	A SPECIAL device is not used by this file.
	Any valid RPG II name	Name of user-supplied external subroutine that performs input/output operations for this special device.

An entry is required in columns 54 through 59 if the file's declared device is SPECIAL. Name of Label Exit is a misnomer in that the subroutine performs all input/output operations for the SPECIAL device, including any required tape label processing.

The subroutine name is left-justified in columns 54 through 59.

FILE ADDITION/UNORDERED OUTPUT

Column	Values	Meanings
66	Blank	No file addition and no unordered output are required for this file.
	A	New records are to be added to this pre-existing file.
	U	New indexed output file, and records are to be written (loaded) in unordered sequence.

Column 66 applies to indexed, sequential, or direct disk files.

Records are added to the end of a sequential file. The file must be a declared output file (O in column 15 of this sheet).

With an indexed file, entries are also added to the ends of the index and data portions of the file. But then at the end of object program execution, RPG II reorganizes the entire index into ascending key sequence. This reorganization is more efficient when the key fields in added records are already in ascending sequence. File addition cannot be specified for an indexed file being accessed via the sequential-within-limits method.

When records are to be added randomly to an indexed file, the new records may have keys that are lower than, higher than, or between pre-existing keys in the file. When records are added to an indexed file sequentially:

- A new record must have a key that is lower than the key retrieved and higher than the preceding key, or
- The file must be single-volume and at end-of-file.

Direct files have gaps into which records may be added. With a direct file, file addition can take place through the specification of a file as an update file processed consecutively or through the execution of CHAIN operation codes. When the CHAIN operation code is used, records may replace the gaps. Otherwise, new records are added to the end of the direct file.

Records can be added at detail, total, and exception time. Records to be added are identified by ADD in columns 16 through 18 of the corresponding Output Specifications sheet (form O).

A U in column 66 indicates that an indexed file is to be created from records supplied in an unordered sequence. After the new file has been loaded and an index has been created, RPG II reorganizes the index into ascending sequence.

The various combinations of entries in column 15 (File Type) and column 66 and their meanings to indexed files are shown in table 5-2.

TABLE 5-2. FILE ADDITION/UNORDERED OUTPUT FOR INDEXED FILES

Column 15	Column 66	Meaning
I	Blank	Read records from the file without adding or updating records.
	A	Read records from the file and add new records to the file. Do not update records.
O	Blank	Write records to the file in ascending key sequence. (RPG II does not perform the ordering automatically.)
	A	Add records to the existing file.
	U	Write records to the file in an unordered key sequence.
U	Blank	Update records in the file without adding new records.
	A	Update records in the file and add new records to the file.

With indexed random processing, the records within a volume may be processed randomly, but the volumes are processed sequentially; i.e., processing cannot be switched back and forth between volumes.

The number of extents is right-justified, and a leading zero may be omitted.

NUMBER OF EXTENTS

Columns	Values	Meanings
68-69	Blank	Single volume file
	1-50	Number of volumes in the file

Columns 68 and 69 apply to disk files only and indicate the number of volumes (disks) that contain the file. Multivolume files cannot be specified with shared input/output (1 in column 48 of form H), sequential processing within limits (L in column 28 and A in column 31), or an unordered load (U in column 66).

A disk file must occupy consecutive cylinders of any individual disk.

TAPE REWIND

Column	Values	Meanings
70	Blank	Not a tape file or tape rewind information not supplied here
	R	Rewind this tape file at end-of-file.
	U	Rewind and unload this tape at end-of-file.
	N	Leave this tape at end-of-file.

Column 70 applies only to files assigned to magnetic tape.

FILE CONDITION

Columns	Values	Meanings
71-72	Blank	There is no external indicator for this file.
	U1-U8	The file is conditioned by this external indicator.

The entry in columns 71 and 72 allows a single program to function differently, depending upon conditions at run time. U1 through U8 are external indicators that can be set by the computer operator (see appendix B).

A typical application for the use of external indicators would be a sales report package including weekly reports and a month-end summary report. An external indicator could be used to condition month-end calculations and/or output.

Conditioning by external indicator applies to primary and secondary input (excluding table and array input files), update, output, display, chained, demand, and combined files. A record address file may be conditioned only if associated with a primary or secondary input, update, or combined file, and the associated file is conditioned by the same external indicator or no indicator.

When a specified external indicator is off, the file is treated as if it were at end-of-file (i.e., no records are read from or written to the file).

Since these indicators are only set externally, the SETON and SETOF operation codes have no effect on them.

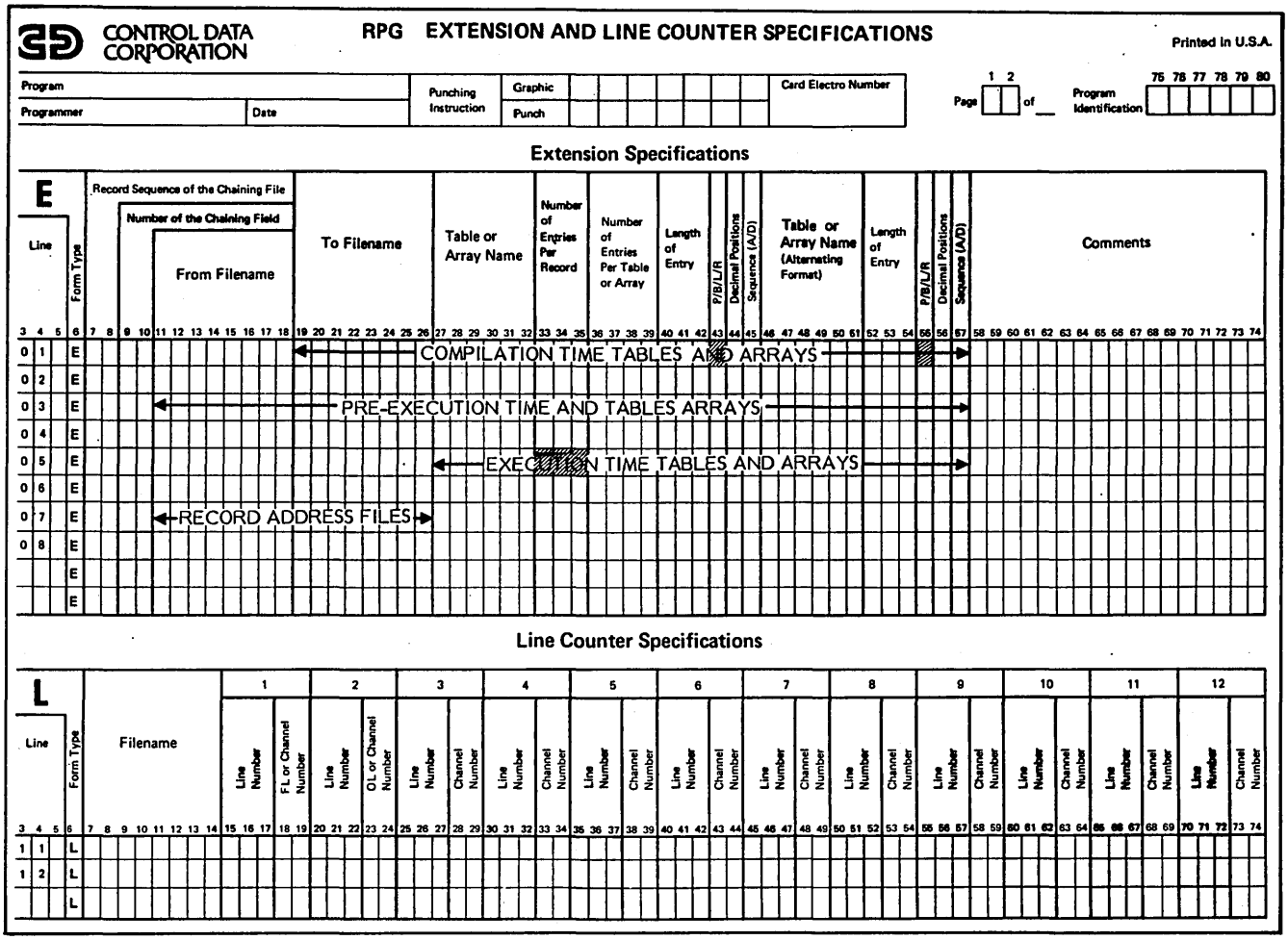
External indicators may be used:

- To condition calculation operations
- To condition output operations
- In a field record relationship (see Field Record Relation, section 8)

RPG II requires more information about record address files and pre-execution time table and array files than space permits on the File Description Specifications sheet. The Extension Specifications sheet provides this extra space and allows a complete description of compilation time tables and arrays. Another type of table or array, execution time, is partially described on this sheet. Previously unmentioned, an *execution time table* or *array* may be loaded at execution time via input or calculation specifications and/or output at execution time through calculation or output specifications.

The Extension and Line Counter Specifications sheet is shown in figure 6-1. Also shown in the figure are columns used by the four types of data sets described in Extension Specifications:

- Record address files (including ADDROUT files)
- Compilation time tables and arrays
- Pre-execution time tables and arrays
- Execution time tables and arrays



NOTE: SHADED COLUMNS MUST BE BLANK FOR THE FILE NAMED.

Figure 6-1. Extension and Line Counter Specifications Sheet

Columns applicable to each type of data set are shaded in figure 6-1.

Comprehensive descriptions of the use of tables, arrays, and record address files are located in sections 12 and 13.

FROM FILENAME

Columns	Values	Meanings
11-18	Blank	a. Compilation time table or array if columns 33 through 35 (Number of Entries Per Record) contain an entry, or b. Execution time table or array if columns 33 through 35 are blank.
	Filename	a. Pre-execution time table or array file, or b. Record address file

An entry is required in columns 11 through 18 for record address files and pre-execution time table and array files. For the latter use, this entry specifies the input file containing the table or array. Since one file may contain more than one table or array, the same filename may appear in the From Filename column more than once on the sheet.

The filename is left-justified in this field and must be identical to the corresponding filename on the File Description Specifications sheet.

TO FILENAME

Columns	Values	Meanings
19-26	Blank	a. Execution time table or array, or b. Compilation or pre-execution time table or array that is not to be output at end-of-job
	Name of output file	The name of the file to which this compilation or pre-execution time table or array is to be output at end-of-job
	Name of input or update	The name of the data file associated with the record address file

When columns 11 through 18 name a record address file, columns 19 through 26 designate the primary or secondary input or update data file that is processed via that record address file.

If a table or array file is named in columns 11 through 18, the entry here specifies an output file to which the table or array is written at the end of execution of the object program. An execution time table or array cannot be written out at end-of-job under RPG II control; it can be output earlier, however, through calculation or output specifications.

The filename is left-justified and must be identical to the appropriate filename in the file description specifications.

TABLE OR ARRAY NAME

Columns	Values	Meanings
27-32	Blank	Record address file
	Valid RPG II name	Name of primary table or array

This is the first opportunity to name a table or array in the program; all previous references to tables and arrays have been to the files containing such data sets. A table name must begin with the three characters TAB; any other name in these columns belongs to an array. No two tables or arrays may share the same name.

If this line of coding describes alternate (or paired) tables or arrays, columns 27 through 32 designate the first or primary one; and the alternate is defined in ensuing columns.

RPG II processes tables and arrays in the order in which they are defined on the Extension Specifications sheet. In particular, when more than one table or array is assigned to a single file, the tables and arrays are read from or written to the file in the order stated here.

A table or array name is left-justified in columns 27 through 32.

More information on tables and arrays can be found in section 12.

NUMBER OF ENTRIES PER RECORD

Columns	Values	Meanings
33-35	Blank	Record address file or execution time table or array
	1-999	Number of entries per record in a compilation time or pre-execution time table or array file

Each physical record of a compilation time or pre-execution time table or array file must have the same number of entries, except the last, which may have fewer entries.

Entries must begin in the first character position of a record, and gaps are not allowed between entries in the record. The absence of gaps, however, does not preclude the use of leading blanks within an entry. Furthermore, a record need not be filled to the end, and in fact cannot be if the record length is not a multiple of the length of an entry. An entry must be totally contained within a record; i.e., an entry may not start in one record and be continued into the next record.

When this line of coding describes a pair of alternate tables or arrays, the entry in columns 33 through 35 defines the number of *pairs* of entries in a record. A pair of entries may not be split between two records.

The number of entries per record is numeric and right-justified, and leading zeros may be omitted.

An example of a nonrelated table is shown in figure 6-2, and related tables are illustrated in figure 6-3. In each figure, the input medium is shown as punched cards for simplicity.

NUMBER OF ENTRIES PER TABLE OR ARRAY

Columns	Values	Meanings
36-39	Blanks	Record address file
	1-9999	Maximum number of table or array entries

Columns 36 through 39 specify one of two things:

- The exact number of entries or pairs of entries in this table or array
- The maximum number of entries or pairs of entries in this table or array

When the table or array is full, these columns indicate the exact number of entries. It is permissible, however, to leave space in a table or array for entries to be added at a later

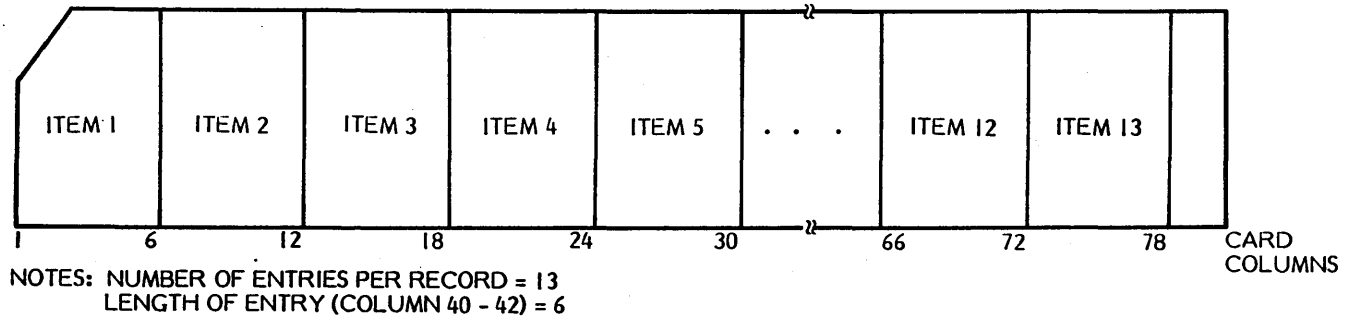
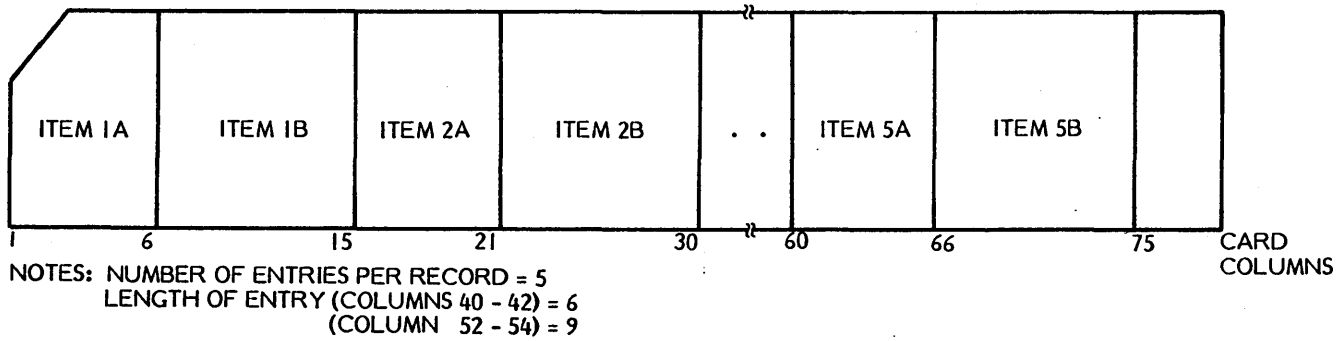


Figure 6-2. Nonrelated Tables



0302

Figure 6-3. Related Tables

time. This is called a *short table* or *array*, and for this case columns 36 through 39 describe the number of entries the table or array will have when filled.

As stated above, if related tables or arrays are being described in this coding line, columns 36 through 39 indicate the maximum number of pairs of entries in the tables or arrays.

Since there must be a one-to-one correspondence between items in related tables, the entry here applies to the table or array named in columns 46 through 51 as well as that named in columns 27 through 32.

The number of entries per table or array is right-justified, and leading zeros may be omitted.

LENGTH OF ENTRY

Columns	Values	Meanings
40-42	Blank	Record address file
	1-256	Number of characters in alphanumeric table or array entry
	1-15	Number of characters in nonbinary numeric table or array entry
	1-9	Number of characters in binary table or array entry

For a numeric table or array having packed decimal format, enter the unpacked length of an entry in columns 40 through 42. For a numeric table or array having binary format, specify here the number of digits of memory required to contain one entry. A two-character binary field requires four digits of memory; a four-character binary field requires nine. In other words, for any format table or array entry, columns 40 through 42 indicate the amount of space in memory required for a single entry.

A numeric table or array entry that is shorter than the length specified here may be padded with leading zeros or blanks. A short alphanumeric entry may be padded with blanks at either end.

If related tables or arrays are being described on this coding line, columns 40 through 42 specify the entry length for the table or array named in columns 27 through 32.

The entry length is right-justified in these columns. Leading zeros may be omitted.

PACKED OR BINARY FIELD

Column	Values	Meanings
43	Blank	a. Record address file
		b. Compilation time table or array
		c. Pre-execution or execution time table or array having unpacked decimal or alphanumeric format
	P	Pre-execution or execution time table or array with packed decimal format
	B	Pre-execution or execution time table or array with binary format

Numeric data is stored in external media in one of three formats – unpacked decimal, packed decimal, or binary. Unpacked decimal format requires the most external space for the representation of data. RPG II uses data only in unpacked decimal format and automatically converts packed decimal and binary fields to that representation before the data is used. For nontable and array data, the conversion takes place when an input record is selected for processing. RPG II converts an unpacked or binary table or array element just before it is moved to a *hold area*, from which it is used, and restores the element to its original format before moving it back to the table or array.

The amount of space required to store a table or array and the amount of time required to perform conversions on unpacked and binary data should be taken into consideration in determining the format for tables and arrays.

UNPACKED DECIMAL FORMAT

In unpacked decimal format every byte (eight bits) of storage can contain one character. (The character can be a decimal number as well as any alphabetic or special character.) Each character is divided into a high-order four-bit zone (or sign) and a low-order four-bit digit. In an unpacked decimal field the sign of the rightmost digit determines the sign of the entire field; other zone positions in the field are ignored. In binary notation a positive sign is 1111, and a negative sign is 1101. RPG II does not check the validity of numeric fields; the digit portion of any character is assumed to be the numeric value of the character.

Unpacked and packed representations of the decimal number 1,492 are shown in figure 6-4. Notice that the unpacked decimal representation of a positive number is identical to the alphanumeric representation of the same number (appendix D).

PACKED DECIMAL FORMAT

In packed decimal format, zones are stripped from characters so that digits are packed two per byte. The sign stripped from the low-order character is then placed in the low-order four bits of the field. (Signs or zones for higher-order digits are again ignored.) Packed decimal representation cuts the amount of space required by unpacked decimal format nearly in half.

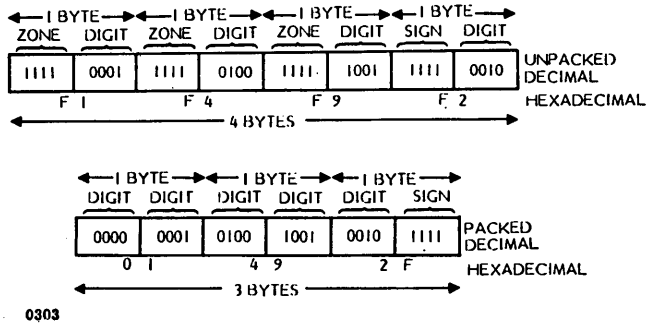


Figure 6-4. Number 1,492 in Unpacked and Packed Decimal Formats

BINARY FORMAT

In twos complement binary notation the high-order bit of the field designates the sign of the number (0 is positive, 1 is negative).

A binary field is either two or four bytes long. In the first case, the binary field consists of the 1-bit sign and a 15-bit numeric value that can range up to 9,999. RPG II reserves a 4-digit hold area for a 2-byte binary element. For a 4-byte binary field RPG II sets aside nine digits of core storage to accommodate the field when it is unpacked. Clearly, binary format requires the least storage, but the conversion process takes longer. Packed decimal format is thus a compromise between unpacked decimal and binary notations.

A 2-byte binary field having a value greater than 9,999 or a 4-byte binary field having a value greater than 999,999,999 cannot be converted to unpacked decimal without a loss of data. Data is lost from the high-order or most significant end in this case.

DECIMAL POSITIONS

Column	Values	Meanings
44	Blank	Record address file or alphanumeric table or array
	0-9	Numeric table or array, with this number of places after the decimal point

An entry in column 44 distinguishes a numeric table or array from an alphanumeric table or array. The digits 0 through 9 indicate the number of positions to the right of the (assumed) decimal point in a numeric table or array element. A zero is entered for integer data.

If this line of coding describes alternate tables or arrays, the specification in column 44 applies to the table or array named in columns 27 through 32.

SEQUENCE

Column	Values	Meanings
45	Blank	Record address file, or table or array entries are unordered
	A	Entries have ascending order
	D	Entries have descending order

Column 45 is used to describe the sequence of the entries in a table or array. Ascending sequence means that the items are ordered with the lowest entry (according to the collating sequence) preceding all higher entries in the table or array. With descending sequence the highest entry precedes all lower entries.

When an entry is made in column 45, the table or array is checked for the specified sequence. When order is violated, an error occurs; recovery procedures are outlined in appendix B.

The order of a table or array determines how RPG II performs LOKUP operations (section 9). An entry is required in column 45 for execution time tables and arrays only if high or low LOKUP is used on the tables or arrays; elements of this type of table or array are not sequence checked, however.

Consecutive equal elements are considered to be in sequence. This means that a short table or array can be padded with elements of all blanks or zeros at the front end and all nines at the back end if ascending sequence is specified.

If this line of coding describes alternate tables or arrays, the specification in column 45 applies to the table or array named in columns 27 through 32.

ALTERNATE TABLE OR ARRAY

Columns 46 through 57 are used to describe one of two things:

- If columns 27 through 32 name a compilation time or pre-execution time table or array, these columns specify a table or array that is paired with that table or array.
- If columns 27 through 32 name an execution time table or array, these columns may describe another table or array that is not related to the former and is not loaded in alternating format with the former. In other words, with execution time tables and arrays, each form E line may be used to describe two distinct tables or arrays.

Ordinarily, columns 46 through 57 are used for an alternate table or array. The rules listed for describing the primary table or array apply to the alternate. Remember that the two names must be unique. Corresponding specification fields are listed in table 6-1.

TABLE 6-1. ALTERNATE TABLE AND ARRAY SPECIFICATION CODES

Field	First Table Or Array	Second Table Or Array
Table or array name	27-32	46-51
Length of entry	40-42	52-54
Packed or binary field	43	55
Decimal positions	44	56
Sequence	45	57

COMMENTS

Columns	Values
58-74	Any characters

Columns 58 through 74 may contain any characters. Typically they are used for documentation of the RPG II source program.

A comments field does not affect compilation or execution of the program, but it is produced in the program listing by the RPG II compiler.

The Line Counter Specifications form should be used for each printer file in the RPG II program and must be provided for each printer file having an L in column 39 of the File Description Specifications sheet.

Line counter specifications designate the length of the form used in the printer and the line at which overflow occurs. When a printer file does not have line counter specifications, the form length defaults to that listed in appendix B, and the overflow line is assumed to be six lines less than that length.

Both the form length and the overflow line must be specified on any form L line. The Extension and Line Counter Specifications sheet is shown in figure 6-1.

FILENAME

Columns	Values	Meaning
7-14	Filename	Filename entered on corresponding form F

Columns 7 through 14 are used to identify the output file to be written on the printer. The name must be identical to the appropriate File Description Specifications entry, and that form F line must contain an L in column 39. The output device assigned to the file must be a printer.

The filename is left-justified in this field.

LINE NUMBER - NUMBER OF LINES PER PAGE

Columns	Values	Meanings
15-17	12-112	Exact number of available printing lines on a page

Columns 15 through 17 indicate the exact number of lines available on the printer page or form (measured from one top-of-form to the next). If a number less than 12 is specified, RPG II indicates an error.

The entry in these columns must be right-justified, and leading zeros may be omitted.

FORM LENGTH

Columns	Value	Meaning
18-19	FL	Form length

The entry of FL in columns 18 and 19 indicates that the preceding field specifies the number of lines per page (form length).

LINE NUMBER - OVERFLOW

Columns	Values	Meanings
20-22	1-112	This line number is the overflow line.

When the destination of a space, skip, or print operation falls beyond the overflow line (but before end-of-form), overflow occurs and the overflow indicator specified for this printer file on the File Description Specifications sheet turns on. When the overflow indicator is turned on, the following occurs before forms advance to the next page:

- Detail lines are printed, if this is the detail time of the cycle.
- Total lines are printed.
- Total lines conditioned by this overflow indicator are printed.

Use of overflow indicators with other indicators can cause a variable number of lines to be printed when overflow occurs. The user should allow for the maximum number to be printed, keeping pleasing spacing in mind, when specifying the overflow line.

Different overflow lines may be specified for different printer files. This may be necessary for a varying number of lines to be printed in overflow conditions.

The overflow line number is right-justified, and leading zeros may be omitted.

OVERFLOW LINE

Columns	Value	Meaning
23-24	OL	Overflow line

The entry of OL in columns 23 and 24 indicates that the preceding field specifies the overflow line number.

Input specifications describe the files, records, and fields that serve as input to the object program. The files described on form I are input, update, and combined files. Each file consists of one or more records, and each record contains one or more fields.

The specifications serve two purposes:

- To describe input records and their relationships to other records in the file
- To describe fields of each record

The columns used by each category of description are shown in figure 8-1. For each file there is at least one coding line of record description followed by as many field description lines as are required to completely describe the record. If appropriate, another record description line, followed by field description lines, is supplied until the one file is completely described. The combination of record and field specification lines are provided for each input file in the program (except record address and table and array files).

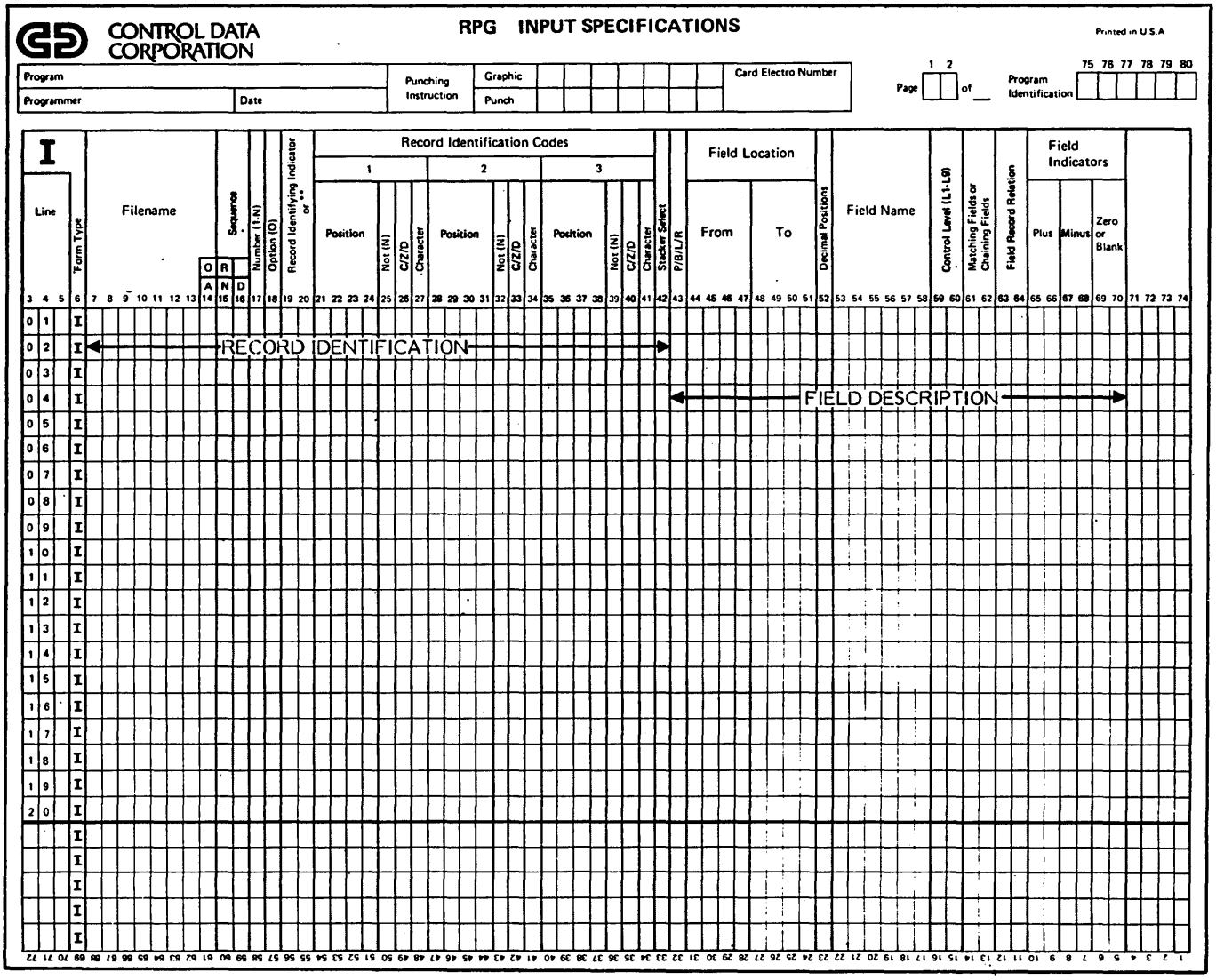


Figure 8-1. Input Specifications Sheet

FILENAME

Columns	Values	Meanings
7-14	Blank	a. Field description b. Filename is the same as that on the previous record description line
	Valid RPG II filename	The name of the file associated with this input specification

When an entry is made in columns 7 through 14, this is a record description line, and the entry names the file with which the record is associated. The filename goes on the first line describing records in the file for a particular record group. A record group must be completely described before another filename entry is made in these columns.

The filename specified here must be identical to the filename used on the File Description Specifications sheet for this file. The name is left-justified in this field.

SEQUENCE

Columns	Values	Meanings
15-16	Blank	Field description
	01-99	Check for special sequence
	Any two alphabetic characters	No check for special sequence

NOTE

The words AND or OR may be left-justified in columns 14 through 16. See Additional Record Identifications for a description of these entries.

This entry allows the user to request a check of the sequence of *record types*; a sequence specification on form F requests sequence checking based on matching record specifications.

An alphabetic entry is any two alphabetic characters (for example, AA, OK, ZT) and indicates that this record type is not part of a predetermined sequence of records. The record type may appear any time in the file, so long as it does not interrupt another sequence of records. Alphabetic sequence specifications are required for chained files and look-ahead records.

A numeric entry requests sequence checking. This could be required when a salesman's name record must precede his records of sales in a file. Each type of record would need a record identification code, and the record description lines would be sequence numbered in the order in which they should appear in the file.

The first record in a sequence must have sequence number 01. (Note that the leading zero must be used in column 15.) Ensuing sequence numbers need not be consecutive, but they must be in ascending order. In other words, records in a sequence are described on the sheet in the order in which they appear in the file. With a numeric sequence entry, an entry must be made in column 17, and column 18 should be considered.

Records within a single file may have alphabetic and numeric sequence entries, but those having alphabetic entries must be listed first on the sheet. Also, only one record sequence is allowed per file. AND or OR lines (description continuation lines) cannot have sequence entries; presumably, the line preceding the AND or OR line specified any required sequence checking for the record.

When RPG II detects an error in record type sequence, the indicator H0 is turned on. If H0 is not turned off by a SETOF operation, execution is stopped before the next record is read. Recovery from this type of sequence error is described in appendix B.

NUMBER

Column	Values	Meanings
17	Blank	Columns 15 and 16 contain alphabetic characters or blanks.
	1	Columns 15 and 16 are numeric, and only one record of this type is allowed in the sequence.
	N	Columns 15 and 16 are numeric, and any number of records of this type may be present before record type changes.

An entry is made in column 17 only if the entry in columns 15 and 16 is numeric. In the example of the sales and salesmen records, the record description for salesmen would contain a 1 in column 17, while the record description for sales would contain N in column 17. The N would indicate that any number of sales records could follow a salesman's record (including zero). Note that the sequence of salesman plus sales records may be repeated in the file; the file is not limited to one occurrence of the sequence, but it is limited to one record type sequence. AND and OR lines cannot contain entries in column 17. As with entries in the Sequence column, the line preceding the AND or OR line should contain entries appropriate for the record.

OPTION

Column	Values	Meanings
18	Blank	a. Columns 15 and 16 contain alphabetic characters or blanks.
		b. Columns 15 and 16 are numeric, and this record type is required in the sequence.
	0	Columns 15 and 16 are numeric, and this record type is optional in each group.

Column 18 is used in conjunction with columns 15 through 17 for sequence checking of record type. If all record types are declared optional, sequence errors are never found.

AND and OR lines should not contain entries in column 18 for the same reason cited for Sequence and Number.

The usage of columns 15 through 18 can be illustrated using the salesman/sales file as an example. That file description can be expanded to include the following record types:

Salesman's name

Salesman's street address

[Salesman's apartment number]

Salesman's city/state/zip code

[Sale]

Bracketed records are optional. If the salesman has an apartment, he has only one at the stated address. Hopefully, he will have zero sales records only while he is on vacation, but any number of optional sales records are allowed for. The appropriate encoding of columns 15 through 18 is shown in figure 8-2. Also shown in the figure are record sequences for salesmen A and B in a card file. Note that one of B's sales was credited to A; record type sequence checking does not detect that type of error.

RECORD IDENTIFYING INDICATOR, LOOK-AHEAD FIELDS, OR SPREAD CARDS

Columns	Values	Meanings
19-20	Blank	a. Field description, or
		b. No distinction need be made between this record type and the preceding record type.
	01-99	General record identifying indicator
	L1-L9	Control level indicator used as a record identifying indicator when a record type rather than a control field triggers a control break (see section 1)
	LR	Last record indicator (see section 1)
	H1-H9	Halt indicator used as record identifying indicator when this record type causes an error condition
	**	Look-ahead fields
	TR	Spread cards

RECORD IDENTIFYING INDICATORS

When a record has been selected for processing, the indicator specified by this entry is turned on by RPG II. The indicator is commonly used to condition calculation and output functions appropriate for the record type within the cycle. All other record identifying indicators are off during the cycle, unless chained files or demand files are being processed, at which time pertinent indicators are on simultaneously. A record identifying indicator is on for the duration of a program cycle.

The specified indicator may be unique for each record type, but sometimes it is useful to assign the same one to different record types – to condition identical calculation and output processing for more than one type of record. This type of assignment can be made by using the OR relationship.

A record identifying indicator need not be assigned if the program is not concerned with different record types.

When a control level indicator is used as a record identifying indicator, lower level indicators are not automatically turned on, as they are when a control field triggers a control break. No record identifying indicators are on during last record time (see appendix F). Record identifying indicators need not be assigned in any particular order.

Record identifying indicators may not be specified in the AND line of an AND relationship. However, different identifying indicators may be assigned to the record types in an OR relationship. A record identifier in an OR line that is not specified anywhere else in the program may be used to bypass unwanted records (for example, blank records).

Record Type	Sequence (Column 15-16)	Number (Column 17)	Option (Column 18)
Name	01	1	Ø
Street address	02	1	Ø
Apartment number	03	1	O
City/state/zip code	04	1	Ø
Sale	05	N	O

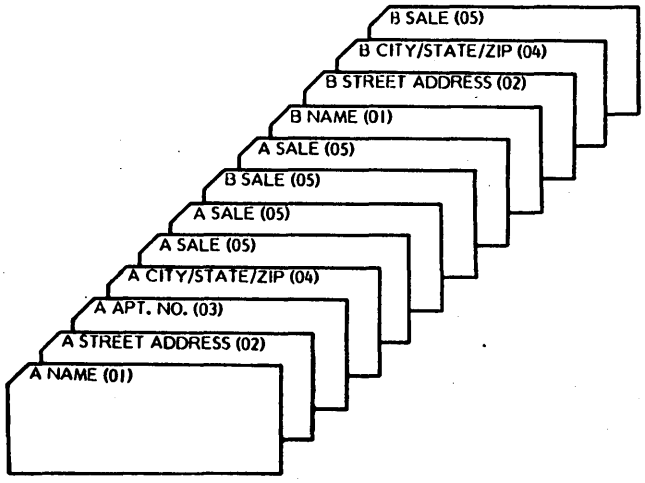


Figure 8-2. Record Type Sequence Checking

LOOK-AHEAD FIELDS

When columns 19 and 20 contain a pair of asterisks, fields contained on the current line and succeeding lines are look-ahead fields. A look-ahead field can be examined in the next record of the file, while the current record is still being processed. Look-ahead fields are used for two purposes:

- To determine when the last record of a control group is being processed
- To extend the matching record capability

Look-ahead fields apply to input, update, and combined files. In multifile processing, one record from each file is read into a read area, and record selection occurs in core from that read area. When a record is selected for processing, that record is moved to a process area, and the next record from the file is read into the read area in preparation for the next selection. In this way, fields of the next record are available for look-ahead while the current record is being processed. The exceptions to this procedure are update and combined files. In these cases, a second record is not available while the current record is being processed from the same file. Thus, in the case of update and combined files, look-ahead fields apply to the current record unless the current record is still in the read area (i.e., has not been selected yet). In single file processing, look-ahead for update and combined files always applies to the current record. For other single input files, look-ahead applies to the next record, as RPG II reserves two memory buffers for the file when look-ahead is specified.

Look-ahead fields cannot be specified for chained or demand files or files that contain header/trailer records (spread cards).

Only one set of look-ahead fields can be described per file, and the set of fields applies to every record in the file, regardless of record type. Look-ahead fields cannot be altered by the program, either by calculations or by blank-after operations.

Look-ahead field specification lines require alphabetic sequence codes, and columns 17, 18, and 21 through 74 must be blank.

If a field is to be available before and after its record is selected, the field must be described on two specifications lines – as a look-ahead field and as a normal field.

After the last record has been processed from a file, every character position of each look-ahead field for the file is automatically filled with nines.

Record selection is described in detail in section 13.

SPREAD CARDS

Some applications require that a data file be structured in such a way that a set of records in the file consists of a header record and some number of item or transaction records. An example of this could be a file organized by salesman number in which the salesman's identification would be the header record and that record would be followed by a variable number of his sales records, which could be called trailer records.

With the spread card capability of RPG II, trailer card information can be included on the header record and following records so as to conserve space in the input file.

An example of the usage of spread cards is shown in figure 8-3. In the example, each job classification contains a variable number of pay codes, and a pay rate is associated with each code. Through the use of spread cards, the 14-record file is reduced to three cards.

The spread card option is available for any sequential file that is defined to be a primary or secondary input file. Each spread card consists of a header portion followed by trailer portions, the maximum number of which is determined by the file's record length. The trailer portion may contain more than one field; however, each trailer portion must contain the same number of fields. Fields of a trailer portion may not be split between records. Look-ahead fields cannot be specified for spread cards.

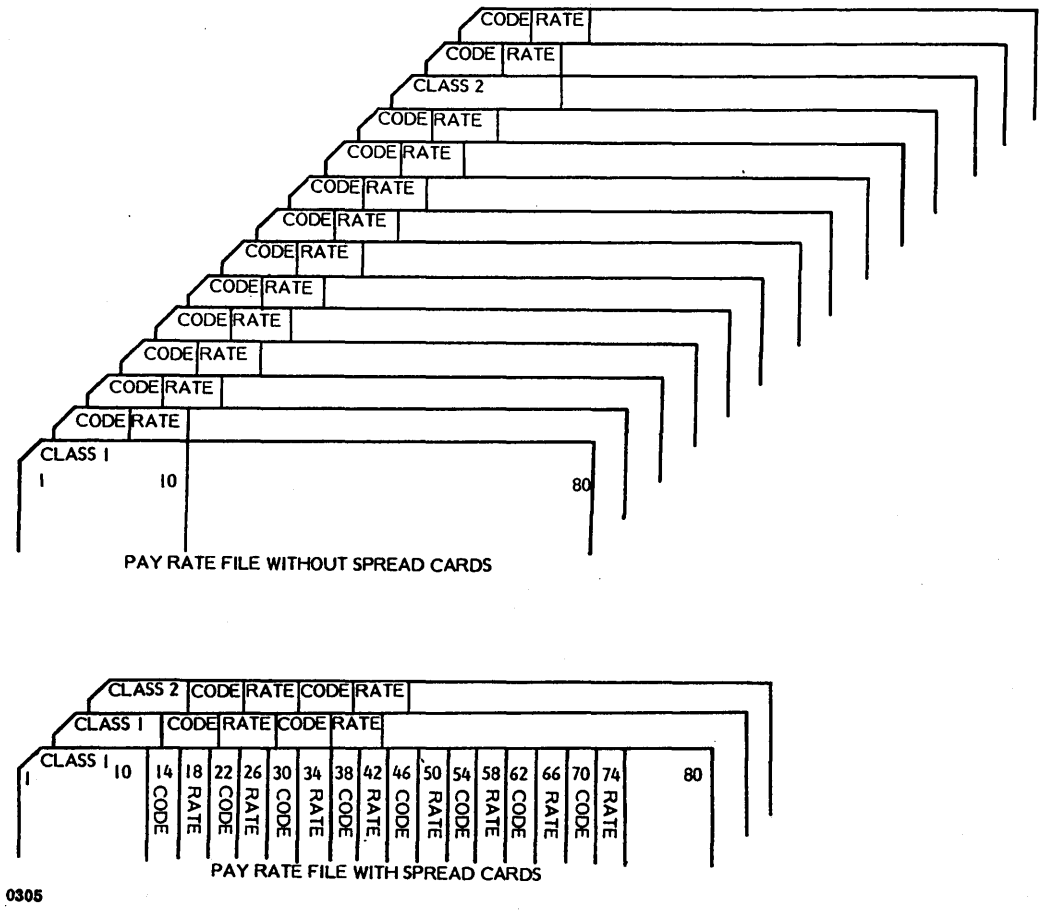


Figure 8-3. Example of Spread Cards

Spread cards are specified in the following fashion:

1. Enter a proper file and record type specifications line for the file. If the spread card record has a numeric record identifying indicator, an N must be specified in column 17, indicating a variable number of records.

2. Describe the fields of the header portion as ordinary fields on separate specifications lines immediately following the file and record type line. Any character positions up to the first trailer portion are considered to be part of the header. Any record identification codes for the record must be contained within the header; i.e., they must not occur in any trailer portion.

3. Enter TR in columns 19 and 20 of the next line to indicate that fields of the trailer follow on ensuing lines. All of the other columns of the TR line after column 6 must be left blank.

4. Describe the fields of the trailer on separate lines following the TR line. The fields are described as ordinary fields, except that columns 59 through 62 (Control Level and Matching Fields) must be left blank. Only fields that are to be used elsewhere in the RPG II program need be described, although the fields at the start and end positions of the trailer portion must be described to indicate the length of the trailer. In addition, only the first trailer portion need be described, as each trailer portion must have fields identical to those of the first trailer.

One trailer portion is processed per program cycle; RPG II treats a trailer portion combined with the header portion as one logical record. During processing, a physical record consisting of only a header portion is bypassed.

RPG II calculates the number of trailer portions per physical record according to record length and the lengths of the header and trailer portions. During spread card processing, a new physical record is read when:

- The calculated number of trailer portions has been processed, or
- An all-blank trailer portion has been encountered.

Proper input specifications for the spread card example are shown in figure 8-4.

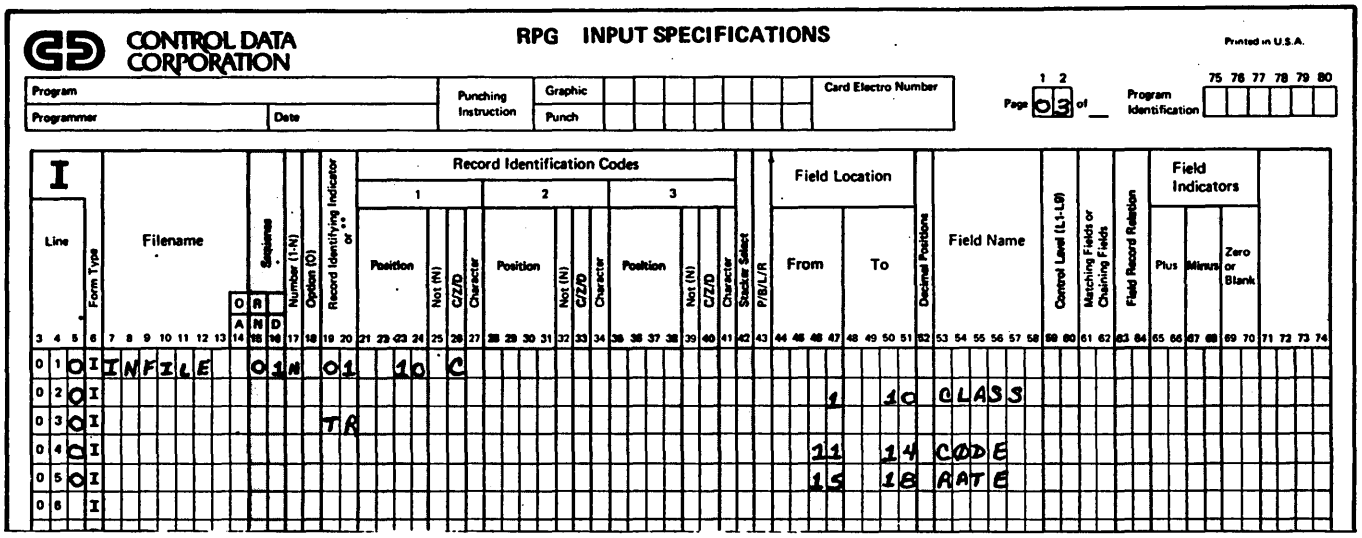


Figure 8-4. Input Specifications for Spread Card Example

RECORD IDENTIFICATION CODES

These entries tell RPG II how to distinguish between record types in a file. Up to three identifying codes may be specified per line of input specifications. When more than three codes are required to identify a record, additional codes are described on following lines – in either an AND or an OR relationship. Codes on a single line are in an AND relationship: code 1 AND code 2 AND code 3.

When a file contains only one kind of record or when it is not important to distinguish between records in the file, columns 21 through 41 may be left blank.

Only one input record is processed per cycle. When a record is selected for processing, its record identifying indicator, if any, is turned on. If a record fits the description of more than one record type, only the record identifying indicator for the first specification is turned on.

When RPG II encounters a record that cannot be identified, it turns on indicator H0. Execution is terminated before the next record is read unless H0 is turned off by a SETOF calculation operation. Recovery from this type of error is described in appendix B.

POSITION

Columns	Values	Meanings
21-24, 28-31, or 35-38	Blanks	Record identification code not needed
	1-9999	Record position of the record identification code

Each record identification code is one character in size. The record position (number of characters from the beginning of the record) is entered here. The position is right-justified, and leading zeros may be dropped.

NOT (N)

Columns	Values	Meanings
25, 32, or 39	Blank	a. Record identification code not needed, or b. Record identification code must be present.
	N	Record identification code must not be present.

The NOT specification is used to indicate that the record identification character must not be present in the specified record position.

C/Z/D

Columns	Values	Meanings
26, 33, or 40	Blank	Record identification code not needed
	C	Full character comparison
	Z	Zone comparison
	D	Digit comparison

These columns indicate what portion of the record identification character is used as part of the record identifying code: the zone portion (upper four bits), the digit portion (lower four bits), or both (full 8-bit character).

Many characters have either the same zone or same digit portion. The character set is grouped according to equal zone and equal digit in appendix D.

RECORD IDENTIFICATION CHARACTER

Columns	Values
27, 34, or 41	Blank or any alphanumeric character

Use any alphabetic character, numeric character, or special character to designate the character in the record that serves to identify the record. A blank is a valid record identification character.

ADDITIONAL RECORD IDENTIFICATIONS

Columns	Values	Meanings
14-16	AND	The record identification codes on this line are in an AND relationship with codes on the previous line.
14-15	OR	The record identification codes on this line are in an OR relationship with codes on the previous line.

More than three record identification codes can be designated for a record type by entering the word AND in columns 14 through 16 of the next line and using columns 21 through 41 as described in the previous paragraphs.

RPG II permits an OR relationship between record identification codes for one record type or between record types for efficient field description when:

- One record type can be identified by different combinations of record identification codes. For example, one record type can be identified by the character A or B or C in column x.
- Two or more record types have identical field descriptions but must be considered as different records.
- Two or more record types may have nearly identical field descriptions, but a small number of fields have different descriptions or may occur in one record type and not in another.

Grouping record types in OR relationships can simplify coding by allowing identical fields to be specified only once. The few fields that are unique can then be linked to their corresponding record types through entries in columns 63 and 64 (Field Record Relation).

STACKER SELECT

Column	Values	Meanings
42	Blank	a. Input device is not a card reader, or b. Cards automatically fall into primary card stacker, or c. This card reader does not have multiple stackers.
	1-4	Stacker into which this record type is stacked

An entry in column 42 indicates that certain input record types must be stacked in a specific stacker. (An entry of 1 designates the primary stacker.) Stacker select applies to input and combined files. Stacker select may also be specified for the output phase of combined files on the Output Specifications sheet.

Any record type that is stacker-selected on the Input Specifications sheet should not have output operations specified for it. If output operations are specified for a record type, the input stacker select specification is ignored if the output is performed.

Ordinarily, if the same stacker is used for both input and output files, an output record precedes an input or combined file record in the stacker. However, if look-ahead fields or dual input/output areas are specified for the input file, an input record precedes an output record in the stacker. Furthermore, stacker select may not be specified for input files having dual input/output areas.

A record type specified on an OR line may have a stacker select specification. If the OR line does not have a stacker selection, the records automatically fall in the primary stacker. A stacker selection may not be specified for a record type described on an AND line.

PACKED OR BINARY FIELD

Column	Values	Meanings
43	Blank	a. Record description line b. Alphanumeric or unpacked decimal format field
	P	Packed decimal format field
	B	Binary format field

Field descriptions begin in column 43.

Data may be stored externally in packed or binary format to conserve space. RPG II converts these fields to unpacked decimal format for internal storage. An entry of P or B in column 43 indicates to RPG II that reformatting of a field is required.

Any table or array that was read in packed or binary format should have an entry in column 43 of the Input Specifications sheet. Field positions indicated on form I should define the positions the table or array occupies in the record in packed or binary format. The unpacked decimal length of each table or array element is defined on the Extension Specifications sheet. When whole arrays are being used, each array element should be treated as a field.

Complete discussions of unpacked decimal, packed decimal, and binary formats are found in section 6.

FIELD LOCATION

Columns	Values	Meanings
44-51	Two one-to-four-digit numbers	Beginning (From column) and ending (To column) character positions of the field

The field location is the starting and ending character positions of the field in the record. The starting position is entered in the From column (columns 44 through 47), and the ending position is entered in the To column (columns 48 through 51). A one-character field is designated by encoding the same position in both the From and To columns; otherwise, the entry in the From column must be smaller than the entry in the To column. The positions are right-justified, and leading zeros may be omitted.

The entries indicate both the position and the size of the field. The maximum size for numeric data is 15 characters (eight if packed, four if binary); for alphanumeric data, 256 characters. To determine the length in bytes of a packed field, divide the unpacked length by 2, add 1, and ignore the remainder. (The 1 is added to account for the sign.) A binary field of four or fewer digits occupies two bytes; otherwise, a binary field requires four bytes. A field that is read from a card is limited to the length of a single card.

Fields within a record may be defined in any order on the Input Specifications sheet. Fields may overlap, and the same positions may be given different field names.

The From and To columns need not allow space for a whole array. When an array is read in, it is read from element 1 up to as many elements as will fit into the Field Location specifications.

DECIMAL POSITIONS

Column	Values	Meanings
52	Blank	Alphanumeric field
	0-9	Number of decimal places in numeric field

Column 52 is used to specify the number of positions to the right of the (assumed) decimal point in a numeric field. An entry must be made here only if the field is numeric. A zero is entered for an integer field. The number entered here cannot be larger than the effective field length designated under Field Location.

If a field is to be used in arithmetic operations, or if it is to be edited, the field must be numeric (i.e., column 52 must have an entry).

FIELD NAME

Columns	Values	Meanings
53-58	Valid RPG II field name	Field name, array name, or array element name
	PAGE, PAGE1, or PAGE2	Special words

Columns 53 through 58 are used to name the fields found in input records. The name listed here is used throughout the RPG II source program to reference this particular field. The names of fields in all the record types must be indicated; however, only the fields actually used elsewhere in the RPG II program need be listed. If the field is a table or array, the name must have already appeared on the Extension Specifications sheet.

FIELD NAMES

Different fields in the same record should have different names; thus, each field of a record is defined on a separate coding line. If two or more fields in the same record type have the same name, only the last definition of the field is used.

Fields in different record types may have the same name if they have the same format and length. Such fields may occupy different record positions, however. To avoid duplicate coding when describing the same field in two or more records, an OR relationship may be used (see Additional Record Identifications).

When a field is to be used in arithmetic operations, or if field contents are edited or zero suppressed, the field must be declared to be numeric. If a field is to be used as both an alphanumeric and numeric field, it must be defined twice by designating two different field names for the same record position.

SPECIAL WORDS (PAGE, PAGE1, PAGE2)

A reference to PAGE on the Output Specifications sheet causes pages of the affected output file to be numbered. RPG II will automatically number the pages of the file, starting with page 1.

To start the page numbering with a number other than one, a field with the name PAGE can be input and an alternate starting page number minus 1 in the field can be given. (Page number minus 1 is given because RPG II increments page counters just before they are output.)

The format specified for a special word must be numeric, up to 15 positions in length, and there must be zero decimal positions. The contents of a PAGE field should be right-justified, such as 0012.

Page numbering is restarted during execution of the program whenever the PAGE field is read from an input record. Because a page counter has a field name, the page number can enter into calculations like other input fields.

The special words PAGE1 and PAGE2 provide three page counters for a single RPG II program. The definitions of PAGE1 and PAGE2 are identical to the definition of PAGE. Use great care when using the same page counter for more than one output file.

CONTROL LEVEL

Columns	Values	Meanings
59-60	Blank	A control level indicator is not associated with this input field.
	L1-L9	Control level indicator to be associated with this input field

These columns allow the assignment of control level indicators to input fields. Control level indicators are used to specify the point at which certain calculation and output operations are to be done.

A control level indicator can be assigned to any field except a binary field. Furthermore, control level indicators cannot be associated with chained or demand files.

A field having an assigned control level indicator is known as a control field. When information in a control field changes, a control break occurs. All records having the same information in the control field are known as a control group.

Control level indicators are ranked from L9 (highest) to L1 (lowest), but may appear in any order in the input specifications. The L0 indicator is always on and thus may not be assigned as a control level indicator. L0 can be used as a conditioning indicator, however, as if it were a true control level indicator.

When a control field enters processing, the contents of that field are compared with the contents of the same field from the previous record. If they are not equal, the specified control level indicator is turned on along with all lower control level indicators. For example, if indicator L3 has been assigned to the field in question and a control break occurs, indicators L3, L2, and L1 are turned on. These indicators would then be used to condition calculation and output functions – typically summary operations. The assignment of control level indicators should be determined by the importance of a control field in relation to other control fields. For example, a control field that triggers a regional total calculation should have a higher control level indicator than a salesman's control field.

Several fields in an input record can be used as one control field by assigning the same control level indicator to all of the fields. This is known as *a split control field.* Fields of a split control field must be described on adjacent specification lines. The first field specified is considered the high-order part of the control field, and the last is the low-order part. Any fields in between have the significance of the order in which they are defined. Split control fields must adhere to the following rules:

- For a single control level indicator, control fields may be split in one record type and not in another if the field names are different. However, the total length of a control field, split or not, must not vary between record types.
- The length of the portions of a split control field may vary between record types, but again, the total length of the control field must be constant between record types.
- No other specification lines may come between lines specifying a single split control field.
- A split control field is considered to be numeric if any portion of it is defined to be numeric.
- A numeric split control field may exceed 15 characters as long as no portion of the field exceeds 15 characters and as long as the sum of control fields is not greater than 256 characters.
- A split control field cannot be made up of a mixture of packed and unpacked decimal portions.

General rules for the specification of control fields are the following:

- If a single control level indicator is used with different record types or even different files, the control fields associated with that indicator must be the same type (alphanumeric or numeric) and the same length.
- In one record type, record positions in control fields may overlap. However, the total number of character positions specified for control fields in the record type must not exceed 256.
- Fields from different record types that have been assigned the same control level indicator may have the same name.
- Except for elements of a split control field, control fields may be specified in any order.
- Decimal positions are ignored when numeric control fields are being compared with one another.
- Signs are ignored when numeric control fields are being compared with one another. Therefore, a numeric control field is always considered to be positive, and -4 is considered equal to +4.

- All control fields assigned a single control level indicator are considered numeric if any one of the fields is numeric. In this case, the sign portion of each of the control fields is ignored, even the sign portion of alphanumeric fields.

- The control field area in memory is initialized to hexadecimal zeros for numeric control fields. With alphanumeric control fields, the control field is initialized to the lowest collating (or alternate collating) sequence value given.

- It is very likely that a control break occurs when the first record containing a control field is read. For this reason, total calculations and output are bypassed for the first input record containing a control field.

- Ordinarily, all types of input records in a program contain the same number of control fields. However, some applications do not permit an equal number of control fields in all record types, and the result could be some unwanted control breaks. This situation is illuatrated in figure 8-5. The input file has two record types, one having the configuration:

Column 1	1
Columns 2 – 4	Project number
Columns 5 – 15	Project name

and the other containing these fields:

Column 1	Blank
Columns 2–4	Project number
Columns 5–6	Pay rate code
Columns 7–10	Pay rate
Columns 11–15	Employee number
Columns 16–19	Hours worked this week

Without special handling, in addition to the expected control breaks, a control break occurs when the first employee record following a project identification record is read. Figure 8-5 contains excerpts of an RPG II program that inhibits the unwanted control break and demonstrates the effect of the unwanted control break with and without the special handling (using indicator 15).

MATCHING FIELDS

Columns	Values	Meanings
61-62	Blanks	No matching is to be performed on this input field.
	M1-M9	Matching level assigned to this input field

Columns 61 and 62 are used for two purposes:

- Sequence checking of a single input, update, or combined file, or

- Record selection based upon matching records and sequence checking of multiple input, update, and combined files

In record matching, the file sequences must be consistent – all ascending or all descending, as encoded on form F. The order in which fields are matched depends upon the value (M1 through M9) assigned. M9 designates the highest order field, and M1 the lowest.

In single file processing, the assignment of M1 through M9 determines the order in which fields are checked for sequence. As in record matching, the higher order code must be assigned to the higher order field.

Sequence checking occurs automatically when files are being matched. When a sequence error is detected, indicator H0 is turned on. If H0 is not turned off during the calculation phase, the object program halts before the next record is read. Recovery from a sequence error is outlined in appendix B.

M1 through M9 are not indicators but cause the MR indicator to turn on when a match occurs.

The following rules apply to the specification of matching fields:

- Matching is allowed only with primary and secondary files.

- A given record may have as many as nine matching field codes.

- The number and lengths of fields to be matched (those with corresponding Mn codes) must be equal. The length of the combined matching fields of a record must be the same as for any record against which matching takes place. The sum of the lengths of all matching fields of a record must not exceed 256 characters.

- Matching fields within a record may overlap, but the total number of characters may not exceed 256 (i.e., overlapped character positions are counted twice).

- Fields to be matched may be either alphanumeric or numeric (but not binary). All matching fields having the same matching level are considered numeric if any one is declared numeric. Or, an alphanumeric field may be considered numeric for matching or sequence checking by defining it again with a different field name and designating it as numeric. The matching level is then used with the redefined field.

- Decimal positions are ignored when numeric fields are being matched. Also, a numeric field with leading zeros matches a field having the same value and suppressed leading zeros (i.e., 0123 equals ƀ123).

- Signs are ignored when numeric fields are matched. Thus, a negative number can match a positive number.

- Multiple matching fields are in an AND relationship. That is, matches must occur in all matching fields specified for a record before MR turns on.

```
100 PROJECT A                                                    SUBTOTAL

    02  00652          40.0          320.00
    02  10321           3.5           28.00
    02  56781          32.0          256.00
                                      604.00                     SUBTOTAL
                                      604.00                     TOTAL

110 PROJECT B

    01  10321          36.5          365.00
    01  45098          35.0          350.00
                                      715.00                     SUBTOTAL
    02  45098           5.0           40.00
                                       40.00                     SUBTOTAL
                                      755.00                     TOTAL

                                     1359.00                     GRAND TOTAL
```

OUTPUT SHOWING UNWANTED CONTROL BREAK

```
100 PROJECT A

    02  00652          40.0          320.00
    02  10321           3.5           28.00
    02  56781          32.0          256.00
                                      604.00                     SUBTOTAL
                                      604.00                     TOTAL

100 PROJECT B

    01  10321          36.5          365.00
    01  45098          35.0          350.00
                                      715.00                     SUBTOTAL
    02  45098           5.0           40.00
                                       40.00                     SUBTOTAL
                                      755.00                     TOTAL

                                     1359.00                     GRAND TOTAL
```

OUTPUT WHERE UNWANTED CONTROL BREAK IS SUPPRESSED

CONTROL DATA CORPORATION

RPG INPUT SPECIFICATIONS

Printed in U.S.A.

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

Programmer | Date

Page 03 of __ | Program Identification

I

Line	Form Type	Filename	Sequence (OR AND)	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank
01	O	MANPWR	AA			10		1	C	1																				
02	O																				2	4	0	PNUM	L2					
03	O																				5	15		PNAME						
04	O		BB			20		1	C																					
05	O																				2	4	0	PNUM	L2					
06	O																				5	6		RATCOD	L1					
07	O																				7	10	2	RATE						
08	O																				11	15		EMPNO						
09	O																				16	19	1	HRS						
10	I																													

Figure 8-5. Specifying Around Unwanted Control Breaks (Sheet 1 of 2)

CONTROL DATA CORPORATION

RPG CALCULATION SPECIFICATIONS

Printed in U.S.A.

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 02 of __ Program Identification

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / And	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus / Minus / Zero); Compare (1>2 / 1<2 / 1=2); Lookup(Factor 2)is (High / Low / Equal)	Comments
010	C		10		SETON						15	
020	C		20		SETOF						15	
030	C		20	HRS	MULT	RATE	WKPAY	62	H			
040	C		20	WKPAY	ADD	SUBTOT	SUBTOT	82				
050	C	L1		SUBTOT	ADD	TOT	TOT	82				
060	C	L2		TOT	ADD	GRDTOT	GRDTOT	92				
070	C											

CONTROL DATA CORPORATION

RPG OUTPUT SPECIFICATIONS

Program		Punching Instruction	Graphic		Card Electro Number
Programmer	Date		Punch		

Page 03 of __ Program Identification

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

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

Constant or Edit Word

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # (Fetch) (F)	Space: Before	Space: After	Skip: Before	Skip: After	Output Indicators: And / And	Field Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R	Constant or Edit Word
010	O	PRINT	D	1	1				10						
020	O									PNUM			4		
030	O									PNAME			16		
040	O		D	1					20						
050	O									RATCOD			7		
060	O									EMPNO			13		
070	O									HRS			19		
080	O									WKPAY M			28		
090	O		T	1					L1N15						
100	O									SUBTOTMB			28		
110	O												38		'SUBTOTAL'
120	O		T	1					L2						
130	O									TOT MB			28		
140	O												35		'TOTAL'
150	O		T	11					LR						
160	O									GRDTOTMB			28		
170	O												41		'GRAND TOTAL'
180	O														

Figure 8-5. Specifying Around Unwanted Control Breaks (Sheet 2 of 2)

- Alphanumeric matching fields are initialized to the lowest collating sequence (or alternate collating sequence) value when ascending sequence checking is specified, or the highest sequence value when descending sequence is specified. Numeric matching fields are initialized to zero.
- Not all files used in a program must have matching fields. Neither must all record types in a file for which matching has been specified have matching fields.
- When packed decimal fields are being matched, the unpacked length; i.e., (2 x packed length) -1, is regarded as the length of the match field.
- Match fields may not be split. This means that the same matching level cannot be used with two fields within a record type.
- Field names are ignored in matching record operations. Thus, fields from different record types which have been assigned the same matching level may have the same name.
- Matching is not allowed for chained or demand files.

When a successful match causes MR to turn on, the indicator is turned on before detail calculations and retains this setting for one complete cycle. A record selected by FORCE causes the MR indicator to go off for one complete cycle, while the forced record is being processed.

Record selection through record matching is discussed further in section 13.

FIELD RECORD RELATION

Columns	Values	Meanings
63-64	Blank	No particular field-record relationship
	01-99	Previously assigned record identifying indicator
	L1-L9	Previously assigned control level indicator
	MR	Matching record indicator
	U1-U8	Previously set external indicator
	H1-H9	Previously used halt indicator

Columns 63 and 64 provide a means of minimizing the number of field definition entries to be written. They also provide a convenient means for selective control of processing. The specification of an indicator in these columns means that the field described on this line is used only when that indicator is on.

The following general rules apply to this entry:

- All fields without a field record relation should be specified before fields with field record relations.
- All fields with the same field record relation (i.e., related to one record type) should be described on consecutive specification lines for more efficient use of core storage.
- All portions of a split control field must have the same field record relation entry, if any, and must be described on consecutive specification lines.
- When used with match or control fields, the field record relation indicator must be the same as a record identifying indicator for this file, and the match or control fields must be grouped in the specifications according to the field record relation indicator.
- When any match value (M1 through M9) is specified without field record relation, all match values must be declared once without field record relation. Conversely, if any match field in a set of match fields has a field record relation, all fields in the set should be declared with the same relation, using dummy (unreferenced) match fields for fields that are not used with all records.

As described under Additional Record Identifications, a record identifying indicator (01 through 99) can be specified here for records in an OR relationship having fields that occupy different positions in the records. In this case, the indicator specifies the record for which this field definition applies. Fields with no field record relation are associated with all the record types in the OR relationship.

Control fields and match fields can also be related to particular records in an OR relationship. Control fields and match fields without a specified field record relation are associated with every record type in the OR relationship. When two control fields have the same control level indicator or two match fields have the same matching level indicator, it is valid to assign a field record relationship to just one of the control or match fields. In this case, the control or match field definition having the field record relation is used when the indicator is on, and the field definition not having a field record relation is used when the indicator is off. Indicators L1 through L9, U1 through U8, and MR cannot be specified in columns 63 and 64 for control and match fields.

A control level indicator (L1 through L9) or the matching record indicator (MR) is specified as a field record relation when the defined field is to be used only when the control break or matching records occurs. The particular condition under which the field is accepted is indicated by L1 through L9 or MR, and the field is used only when the specified indicator is on.

When U1 through U8 have been encoded in columns 63 and 64, the normal processing for this field is carried out only when the specified external indicator has been set on prior to processing of the object program. If the indicator is off, all input processing of the field is bypassed. External indicators are more commonly used to condition entire files in the file description specifications. An external indicator is specified in columns 63 and 64 to condition only a field of a file – whether or not the file is conditioned externally.

A halt indicator (H1 through H9) is used to relate a field to a record in an OR relationship where the record has the same halt indicator coded in columns 19 and 20.

FIELD INDICATORS

Columns	Values	Meanings
65-70	Blank	Field examination not required
	01-99	Field indicator
	H1-H9	Halt indicator

Columns 65 through 70 are used to specify that the contents of this field are to be examined when its record has been selected for processing. The data contained in the field can be examined for the following conditions:

- Plus (columns 65 and 66). The field must be numeric. The specified indicator is turned on if the contents of the input field are greater than zero.
- Minus (columns 67 and 68). The field must be numeric. The specified indicator is turned on if the contents of the input field are less than zero.
- Zero or blank (columns 69 and 70). The specified indicator is turned on if this numeric field contains zeros or this alphanumeric field contains blanks. An all blank numeric field will also cause the specified indicator to turn on; however, an alphanumeric field containing zeros does not cause the indicator to go on.

Field indicators are defined in the input specifications; indicators are used in calculation and output specifications.

The following facts about field indicators should influence their proper usage.

- Field indicators are off at the beginning of program execution and remain off until a field satisfies one of the conditions. At that time, the appropriate indicator is turned on.
- A single numeric input field can be assigned one, two, or three field indicators, and the indicators need not be unique from one another. When the record enters processing, only the indicator appropriate to the field contents is turned on; the others are turned off.
- If the same indicator is assigned to fields in different record types, the status of the indicator reflects the record being processed.
- When different field indicators are assigned to fields in different record types, a field indicator that goes on stays on until another record of that type is read; i.e., it remains on throughout the processing of intervening records.
- When a single field indicator is assigned to several fields in a record type, the status of the indicator always reflects the contents of the field defined last in the specifications.
- The status of field indicators can be altered by SETON and SETOF calculation operations.

Halt indicators are typically used as field indicators when a plus, minus, zero, or blank field is an error. Unless a halt indicator is set off during calculations, program execution is halted at the end of the current cycle. Recovery procedures are outlined in appendix B.

Calculation specifications tell RPG II which run-time operations to perform. General rules for these specifications are as follows:

- Each operation is entered on a single line of the coding form. Generally, operations are entered in the order they are to be executed.
- Detail operations are specified first, followed by total operations, and finally, lines of subroutines.

Each calculation specification line can be divided into three parts.

- Columns 7 through 17 control when the operation is to be performed.
- The kind of operation and the data for which the operation is performed are defined in columns 18 through 53.
- Columns 54 through 59 specify the tests to be performed on the results of the calculation.

The Calculation Specifications sheet is shown in figure 9-1.

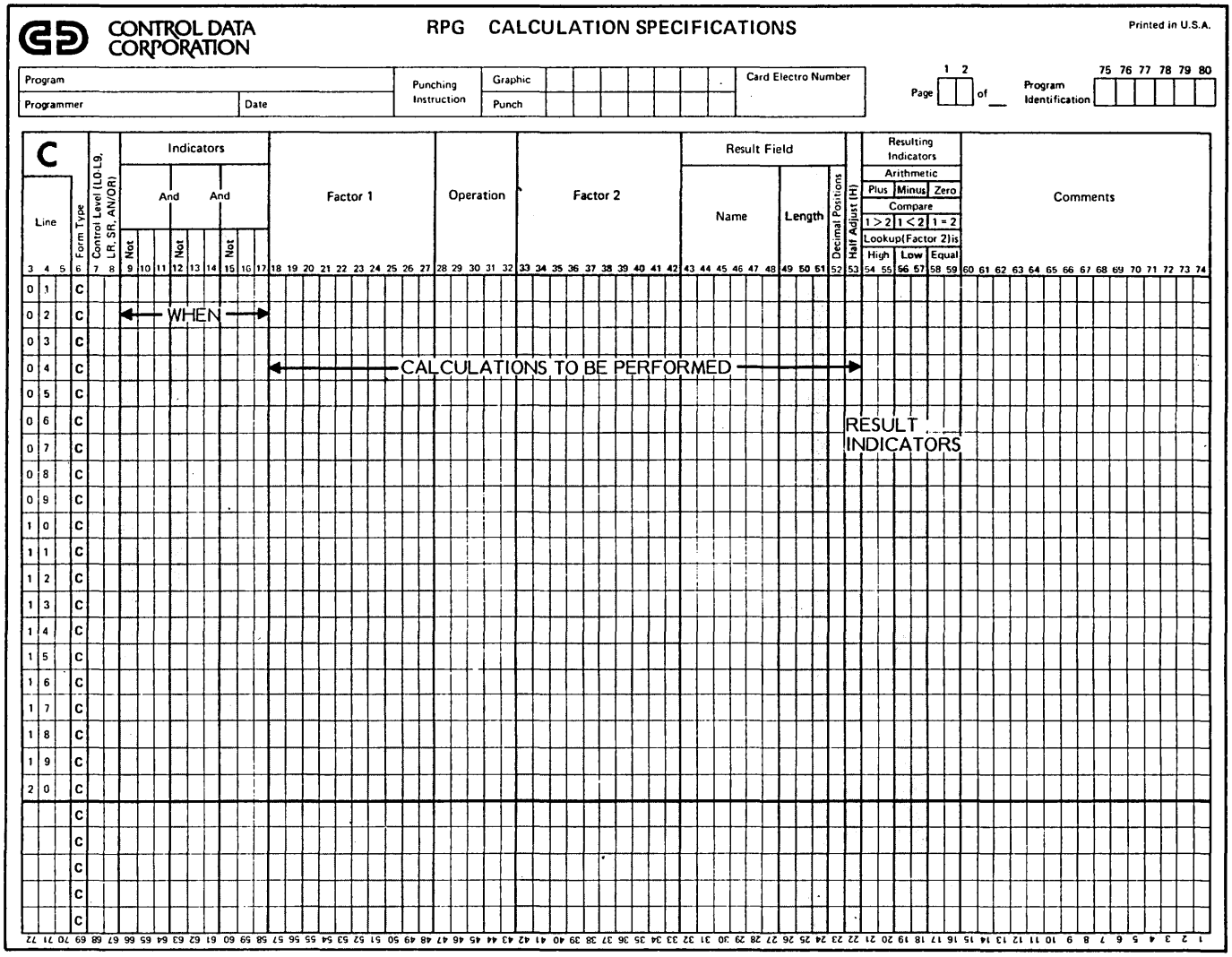


Figure 9-1. Calculation Specifications Sheet

CONTROL LEVEL

Columns	Values	Meanings
7-8	Blank	Calculation operation is not part of subroutine and is performed at detail time.
	L0-L9	Calculation operation is performed at total time when this indicator is on (L0 is always on).
	LR	Calculation operation is performed when the last record has been processed or when LR has been set on.
	SR	Calculation operation is part of a subroutine.
	AN, OR	Indicators listed on the current line are in an AND or OR relationship with indicators on the preceding line(s).

Calculation specification lines are presented to the compiler in the above order, with the exception that AN/OR lines are specified with their associated lines. More information is given about AN/OR lines under Indicators.

Calculation operations are performed at either detail time or total time. Blanks in columns 7 and 8 indicate performance at detail time. L0 and control level indicators L1 through L9 designate calculations to be performed at total time. As described under Control Level, section 8, control level indicators are set by control breaks. A change in the control field at any level causes the indicator associated with that level and all lower control levels to turn on. However, a lower level indicator is not turned on when:

- The higher level indicator is set by a SETON operation, or
- The higher level indicator is set on as a result of being used as a record identifying indicator.

When a control level indicator is specified in columns 7 and 8, the operation is performed only when the indicator is on. When a control break occurs, all operations conditioned by control level indicators are done before those that are not conditioned.

The L0 indicator is always on and is used for calculations that are to be done at total time whether or not a control break has occurred.

In one program cycle, all operations conditioned by control level indicators in columns 7 and 8 are done at total time. When control level indicators are specified in columns 9 through 17 of a line having blanks in columns 7 and 8, the operation is performed at detail time immediately following the control break.

The last record indicator, LR, is set on after the last input record has been read and processed. Control level indicators L1 through L9 are also turned on at the same time. When LR is set on by a SETON operation code, L1 through L9 are not affected. (LR cannot be set off.) When LR is on, the job ends after all total operations have been performed.

An SR in columns 7 and 8 identifies the line as part of a subroutine. Subroutines can be entered from either a detail or a total calculation sequence. Individual operations within the subroutine may be conditioned by any valid indicators, including L0 through L9, in columns 9 through 17. Subroutine lines must be specified last on the calculation sheet(s).

INDICATORS

Any entries in columns 9 through 17 specify the additional conditions that must be met after those in columns 7 and 8 are satisfied. These conditions are specified in the form of indicators that must be on and/or off.

Columns	Values	Meanings
9-17	Blank	Columns 7 and 8 list the only conditions. If 7 and 8 are blank, do the operation for every record read.
	01-99	Resulting indicators or record identification indicators defined elsewhere in the program
	L1-L9	Previously assigned control level indicators
	LR	Last record indicator
	MR	Matching record indicator
	H1-H9	Previously assigned halt indicators
	U1-U8	Previously assigned external indicators
	0A-0G, 0V	Previously assigned overflow indicators

Indicators are set or cleared according to input conditions or previous calculations. Up to three indicators may be specified (one each in columns 10 and 11, 13 and 14, and 16 and 17); if two or more are specified, they are considered to be in an AND relationship. If the desired indicator state is off, encode an N in the preceding column (column 9, 12, or 15).

Indicators are used in the following ways in columns 9 through 17:

- Use any record identifying indicator to condition an operation that is to be done only for a certain type of record.
- Use any previously defined field indicator to condition an operation that is to be done only when a field has a specified status.

- Use any resulting indicator specified in columns 54 through 59 of the Calculations Specifications sheet to condition operations according to the results of previous operations.

- When an input data error causes a halt indicator to go on, program execution is not stopped until the end of the cycle, after calculations have been performed. Halt indicators are commonly used in these columns to bypass operations on erroneous data. Conversely, halt indicators can be specified here to condition operations to be performed only when an error occurs.

- The matching record indicator is specified to condition operations to be done only when matching records have been found.

- An external indicator previously assigned on the File Description Specifications sheet is used to condition operations to be done only when a certain input file is available.

- The last record indicator is specified in these columns only when LR is set on during calculations to simulate end-of-file conditions. LR should be specified in columns 7 and 8 instead for operations to be performed at true end-of-file.

NOTE

When LR is used as a resulting indicator (columns 54 through 59), NLR should be specified in columns 9 through 17 of the same line in order to avoid the clearing of LR in the true end-of-file situation.

- When a control level indicator is specified in columns 9 through 17 and columns 7 and 8 are blank, the operation is performed on only the first record of a new control group at detail calculations time.

- Previously assigned overflow indicators are used to condition operations to be done when overflow occurs.

Points to remember when encoding columns 7 through 17 are:

- When columns 7 and 8 contain a control level indicator and columns 9 through 17 specify MR, the matching record description applies to the last record processed – not the record causing the control break. MR will not reflect the current record until all the total operations specified for the control break have been performed.

- A control level indicator in columns 7 and 8 designates a total calculation. A control level entry in columns 9 through 17 specifies a detail operation to be performed after the control break. The organization of the RPG II program cycle causes total operations for a control break to be executed before detail operations are performed on the record causing the control break.

AN/OR LINES

Up to three indicators may be specified on each AN/OR line. The first line of an AN/OR group contains blanks, L0 through L9, LR, or SR in columns 7 and 8. Other lines of the group contain AN or OR in columns 7 and 8, as appropriate. The last line of an AN/OR group lists the operation, operand(s), and resulting indicator(s) to be used when all of the conditions are met. The preceding lines should not specify operations, operands, or resulting indicators.

The indicators on each individual line of the AN/OR group are in an AND relationship with one another. It is not necessary to have three indicators on each AN/OR line, but each line should contain at least one indicator.

AN/OR lines give flexibility for conditioning. If more than three indicators must be on or off for a particular operation, AN lines allow a virtually unlimited number of indicators. OR lines allow alternate conditions. When there are combinations of AN and OR lines, AN has the higher binding power.

FACTOR 1 AND FACTOR 2

Many RPG II calculation operations require the specification of factors: factor 1 or factor 2 or both or neither, depending upon the particular operation. Information about the calculation operations and their factors can be found under Operation, and a summary of the specification of operation codes is contained in appendix D.

Columns	Values
18-27 and 33-42	Factor 1 and Factor 2 columns may contain the following entries: • Field name • Alphanumeric or numeric literal • Array element name, or name of table, array, or subroutine • Date field name (UDATE, UMONTH, UDAY, UYEAR) • Special name (PAGE, PAGE1, PAGE2) • Label for a TAG, BEGSR, or ENDSR operation (Factor 1 column only). Label for a GOTO or EXSR operation (Factor 2 column only) • Filename for a CHAIN, DEBUG, DSPLY, READ, SETLL, or FORCE operation (Factor 2 column only)

Entries are left-justified in the two fields.

LITERALS

Literals are self-defining data – either numeric or alphanumeric.

Numeric

A numeric literal is a combination of digits (0 through 9), an (optional) decimal point, and an (optional) arithmetic sign

(example: -12.3). Numeric literals must have the following characteristics:

- Not exceed 10 positions
- Not contain embedded blanks
- Be left-justified
- Have the sign, if encoded, in the leftmost column (18 or 33); if unsigned, the literal is assumed to be positive
- Not be enclosed in apostrophes

The decimal point, if encoded, may appear anywhere in the literal. When a decimal point is not encoded, it is assumed to follow the rightmost digit (i.e., the literal is integer).

Alphanumeric

An alphanumeric literal is a combination of any alphanumeric characters enclosed in apostrophes (examples: 'DON' 'T', '-12.3'). Alphanumeric literals must adhere to the following rules:

- The literal must not exceed eight characters plus two apostrophes.
- The literal may contain any alphanumeric characters, including blanks.
- The literal must be left-justified.
- An apostrophe within the literal is encoded by entering two consecutive apostrophes.

NAMES AND ELEMENTS

Table and array names were described briefly in section 6 and are discussed more fully in section 12. A subroutine name, a table or array name, or an array element name must:

- Be left-justified
- Start with an alphabetic character
- Not exceed six characters
- Not be the same as any field name
- Not include special characters or embedded blanks

DATE FIELD NAMES

The date field names (UDATE, UMONTH, UDAY, and UYEAR) refer to the current date that the host computer supplies to the object program. These values cannot be changed by calculation operations, so they must not be specified as Result Fields. They may be used as Factor 1 and Factor 2, for example, if one wants to calculate the next month's billing date based upon the current date.

SPECIAL NAMES

The special names are the three page counter names, PAGE, PAGE1, and PAGE2, discussed under Field Name in section 8. The page counters can be incremented explicitly in the calculation specifications or implicitly in the output specifications.

LABELS

Rules for establishing Factor 1 labels (used with TAG, BEGSR, and ENDSR) and Factor 2 labels (used with GOTO and EXSR) are discussed with those operations.

FILENAMES

Any valid RPG II filename can be used as Factor 2 for CHAIN, DEBUG, DSPLY, READ, SETLL, and FORCE operation codes, subject to the specific requirements of each operation.

OPERATION

Columns	Values
28-32	Operation code listed in table 9-1

The operation code is the mnemonic for the operation to be performed with Factor 1, Factor 2, and the Result Field. Operation codes are left-justified.

Generally, operations are performed in the order specified in the calculation sheets. All operations conditioned by control level indicators in columns 7 and 8 must follow those that are not conditioned by control level indicators. All operations that are part of a subroutine (SR in columns 7 and 8) must be listed after all other calculations in a program,

ARITHMETIC OPERATIONS

The fields and literals used in arithmetic operations must be numeric. Factor 1, Factor 2, and the Result Field may all be the same field or three different fields or any combination thereof. None of the fields specified as Factor 1, Factor 2, or Result Field may exceed 15 digits. Decimal points are aligned by RPG II before the operation is performed. If the result exceeds 15 characters, characters may be dropped from either or both ends, depending upon the location of the decimal point. The results of all arithmetic operations are signed. Any data placed in the Result Field replaces any data that was previously there, and the previous contents are lost. Factor 1 and 2 fields are not altered by the operations unless one or both of the names are the same as the Result Field name.

The arithmetic operation codes and their factors are shown in table 9-2.

For the operations ADD, Z-ADD, Z-SUB, MULT, DIV, and SQRT, whole arrays may be specified in the factor and

TABLE 9-1. OPERATIONS FOR CALCULATION SPECIFICATIONS

Class	Code	Operation
Arithmetic	ADD	Add
	Z-ADD	Zero and add
	SUB	Subtract
	Z-SUB	Zero and subtract
	MULT	Multiply
	DIV	Divide
	MVR	Move remainder
	SQRT	Square root
	XFOOT	Crossfoot – sum elements of array
Move	MOVE	Move
	MOVEL	Move left
	MOVEA	Move array
Move zone	MHHZO	Move high to high zone
	MHLZO	Move high to low zone
	MLLZO	Move low to low zone
	MLHZO	Move low to high zone
Compare and testing	COMP	Compare
	TESTZ	Test zone
Bit	BITON	Set bit on
	BITOF	Set bit off
	TESTB	Test bit
Setting indicators	SETON	Set indicator on
	SETOF	Set indicator off
Branching	GOTO	Go to
	TAG	Provide a branching label
Lookup	LOKUP	Table or array lookup
Subroutine	BEGSR	Begin an RPG II subroutine
	ENDSR	End an RPG II subroutine
	EXSR	Execute an RPG II subroutine
	EXIT	Exit to external subroutine
	RLABL	Define label for external subroutine
Input/Output	EXCPT	Force output record
	FORCE	Force record to be read
	DSPLY	Print and/or accept data at console device
	READ	Read from demand file
	CHAIN	Retrieve defined record
	SETLL	Set lower limits
Debug	DEBUG	Aid in finding programming errors
Time	TIME	Get time-of-day

TABLE 9-2. ARITHMETIC OPERATION CODES AND FACTORS

Factor 1	Operation	Factor 2	Result Field
Required	ADD	Required	Required
Blanks	Z-ADD	Required	Required
Required	SUB	Required	Required
Blanks	Z-SUB	Required	Required
Required	MULT	Required	Required
Required	DIV	Required	Required
Blanks	MVR	Blanks	Required
Blanks	SQRT	Required	Required
Blanks	XFOOT	Required	Required

result fields. In this case, corresponding elements are used to effect element-by-element operations (i.e., element 1 from the factor arrays is operated on to produce element 1 of the result array. This is followed by the use of element 2, etc.).

Add (ADD)

Factor 2 is added to Factor 1. The sum is placed in the Result Field.

Zero and Add (A-ADD)

Factor 2 is added to a field of zeros, and the sum is placed in the Result Field.

Subtract (SUB)

Factor 2 is subtracted from Factor 1. The difference is placed in the Result Field.

NOTE

When Factor 1 and Factor 2 have the same name, the Result Field is effectively set to zero.

Zero and Subtract (Z-SUB)

Factor 2 is subtracted from a field of zeros, and the difference is placed in the Result Field. This operation can be used to change the sign of a field, when Factor 2 and Result Field have the same name.

Multiply (MULT)

Factor 1 is multiplied by Factor 2. The product is placed in the Result Field. Be sure the specified Result Field is large enough to hold the product.

Divide (DIV)

Factor 1 (dividend) is divided by Factor 2 (divisor). The quotient is placed in the Result Field.

If Factor 1 contains zero, the quotient is zero. Factor 2 cannot contain zero. If it does, program execution halts immediately. If the operator elects to continue execution, the quotient and remainder are set to zero, as shown in appendix B.

Any remainder resulting from a divide operation is lost unless the next operation code is MVR. If the next operation is Move Remainder, half adjust (round) should not be specified for the divide operation.

Move Remainder (MVR)

This operation moves the remainder from the immediately preceding divide operation to Result Field. Move Remainder should not be used following a divide operation for arrays. A Move Remainder operation should be conditioned by the same indicators, if any, that conditioned the divide operation. Half adjust cannot be specified for the Move Remainder operation. The maximum length of the remainder is 15 digits, including decimal positions. The number of significant decimal positions in the remainder is the greater of:

- The number of decimal positions in Factor 1 of the preceding divide operation
- The sum of the decimal positions in Factor 2 and Result Field of the preceding divide operation

The maximum number of whole number positions the remainder has is equal to the whole number positions in Factor 2 of the preceding divide operation.

Square Root (SQRT)

The square root of the contents of Factor 2 is placed under Result Field.

As with other arithmetic operations, Factor 2 and the Result Field are numeric fields that can be up to 15 digits long, and each field can have up to nine decimal positions. The correspondence between Factor 2 and Result Field lengths is as follows:

- For every digit left of the decimal place in the Result Field, there should be two digits 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 right of the decimal place in Factor 2.

A whole array can be used in a SQRT operation if both Factor 2 and the Result Field have array names. In this case, the square root of every element in the Factor 2 array is calculated, and the results are stored in the corresponding element positions of the Result Field array.

The following statements apply to the SQRT operation:

- The root is automatically half-adjusted in the Result Field.
- The length of the Result Field must be greater than or equal to the decimal positions entry for the field.
- Factor 2 cannot contain a negative number. If it does, program execution is halted immediately; recovery is outlined in appendix B.
- Resulting indicators are not allowed on the SQRT specifications line.

Crossfoot (XFOOT)

This operation is used with arrays having numeric elements. The elements of the array named as Factor 2 are added together, and the sum is placed in the single-element Result Field. If the Result Field name is an element of the Factor 2 array, the sum reflects the value of the element before storing of the result took place.

Half adjust and resulting indicators can be specified with the XFOOT operation code.

MOVE OPERATIONS

Move operations move part or all of the data in the Factor 2 field to the Result Field. The contents of Factor 2 are not changed. Move operations must not have Factor 1 specifications, and there must be no resulting indicators.

The form of the data may be changed from alphanumeric to numeric and vice versa. This is accomplished by moving an alphanumeric field to a numeric field or a numeric field to an alphanumeric field. When an alphanumeric field is moved to a numeric field, blanks are converted to zeros. Zeros are not converted to blanks when a numeric field is moved to an alphanumeric field, however.

Decimal positions are ignored when numeric fields are moved. Thus, if the numeric data 1.23 is moved to a field having one decimal position, the result is 12.3.

Move (MOVE)

Characters of the data field or literal specified in Factor 2, starting with the rightmost character, are moved to the rightmost positions of the Result Field.

If the Result Field length is shorter than the length of the Factor 2 field, the excess leftmost characters of the Factor 2 field or literal are not moved. If the Result Field is longer than the Factor 2 field, all available characters are moved, and excess characters of the Result Field are not

changed. A numeric Result Field assumes the sign associated with Factor 2.

When data is moved into an indexed array, data is moved into only one element of the array. This is true even if the field being moved is larger than the array element.

Move Left (MOVEL)

This operation is similar to MOVE, except that characters are moved from the leftmost position of the Factor 2 data field to the leftmost character position(s) of the Result Field.

If a numeric Result Field is the same length as the Factor 2 field, it assumes the sign associated with Factor 2. If a numeric Result Field is longer than Factor 2, the Factor 2 sign is lost, and the excess rightmost Result Field characters remain intact, including the sign position. When the Result Field is shorter than Factor 2, the sign of the Factor 2 field becomes the sign of the Result Field.

Examples of the MOVEL operation are shown in figure 9-2.

Move Array (MOVEA)

With MOVEA, characters of Factor 2, starting with the leftmost character, are moved to the leftmost positions of the Result Field.

Arrays and fields specified with MOVEA must be alphanumeric, and Factor 2 and the Result Field must not reference the same array.

The number of characters moved is equal to the length of the shorter field (Factor 2 or the Result Field). If the Factor 2 field is longer than the Result Field, the excess rightmost characters of Factor 2 are not moved; when Factor 2 is shorter than the Result Field, the rightmost characters of the Result Field are unchanged.

MOVEA makes possible the following field movements:

- Several contiguous array elements can be moved to a single field.
- A single field can be moved into contiguous elements of an array.
- Contiguous elements of one array can be moved into contiguous elements of a second array.

Character movement starts with the first character of a field or unindexed array. When an array is indexed, movement starts with the first character of the referenced array element. Character movement continues until:

- The last array element has been moved or filled, or
- The last character of the field specified as Factor 2 or the Result Field has been moved or filled (this situation can result in the partial movement or filling of an array element).

Factor 2 and Result Field Same Length

	Factor 2		Result Field	
Numeric (2 decimal positions)	0,0,1,2,3	Before MOVEL	4,5,6,7,8	Numeric (1 decimal position)
	0,0,1,2,3	After MOVEL	0,0,1,2,3	
Numeric (2 decimal positions)	0,0,1,2,3	Before MOVEL	Z O W I E	Alphanumeric
	0,0,1,2,3	After MOVEL	0,0,1,2,L	
Alphanumeric	3,b,M,E,N,	Before MOVEL	4,5,6,7,8	Numeric (1 decimal position)
	3,b,M,E,N,	After MOVEL	3,0,4,5,5	
Alphanumeric	3,b,M,E,N,	Before MOVEL	Z,O,W,I,E,	Alphanumeric
	3,b,M,E,N,	After MOVEL	3,b,M,E,N,	

Factor 2 Longer Than Result Field

	Factor 2		Result Field	
Numeric (2 decimal position	0,0,0,0,0,1,2	Before MOVEL	4,5,6,7,8	Numeric (1 decimal position)
	0,0,0,0,0,1,2	After MOVEL	0,0,0,0,0	
Numeric (2 decimal positions)	0,0,0,0,1,2,3	Before MOVEL	Z,O,W,I,E,	Alphanumeric
	0,0,0,0,1,2,3	After MOVEL	0,0,0,0,1	
Alphanumeric	3,b,W,O,M,E,N,	Before MOVEL	4,5,6,7,8	Numeric (1 decimal positions)
	3,b,W,O,M,E,N,	After MOVEL	3,0,6,6,4	
Alphanumeric	3,b,W,O,M,E,N,	Before MOVEL	Z,O,W,I,E,	Alphanumeric
	3,b,W,O,M,E,N,	After MOVEL	3,b,W,O,M,	

Factor 2 Shorter Than Result Field

	Factor 2		Result Field	
Numeric (2 decimal positions)	0,0,1,2,3	Before MOVEL	2,3,4,5,6,7,8	Numeric (1 decimal position)
	0,0,1,2,3	After MOVEL	0,0,1,2,3,7,8	
Numeric (2 decimal postions)	0,0,1,2,3	Before MOVEL	O,b,Z,O,W,I,E,	Alphanumeric
	0,0,1,2,3	After MOVEL	0,0,1,2,L,I,E,	
Alphanumeric	3,b,M,E,N,	Before MOVEL	2,3,4,5,6,7,8	Numeric (1 decimal postion)
	3,b,M,E,N,	After MOVEL	3,0,4,5,5,7,8	
Alphanumeric	3,b,M,E,N,	After MOVEL	O,b,Z,O,W,I,E,	Alphanumeric
	3,b,M,E,N,	After MOVEL	3,b,M,E,N,I,E,	

Figure 9-2. Examples of MOVEL Operations

MOVE ZONE OPERATIONS

The four move zone operations are used to move only the zone portion of a character. Using a minus sign character (-) in a move zone operation does not result in a negative Result Field: the minus sign has an internal representation of X'60', while an X'D' zone designates a negative character. Moving one of the characters J through R, however, can produce a negative Result Field, as these characters have hexadecimal representations of X'D1' through X'D9'.

The four move zone operations are listed below and in table 9-3. In general, the word *high* involves alphanumeric fields only, while the word *low* can be used with alphanumeric and numeric fields.

Move-High-to-High Zone (MHHZO)

The zone of the leftmost character of the Factor 2 field is moved to the leftmost zone position of the Result Field. Both fields must be alphanumeric.

TABLE 9-3. MOVE ZONE OPERATIONS

Operation	Allowable Field Formats	Function
MLLZO		
Factor 2	Alphanumeric	Zone of rightmost byte of factor 2 is moved to zone of rightmost byte of result field.
Result Field	Alphanumeric	
Factor 2	Alphanumeric	
Result Field	Numeric	
Factor 2	Numeric	
Result Field	Alphanumeric	
Factor 2	Numeric	
Result Field	Numeric	
MHLZO		
Factor 2	Alphanumeric	Zone of leftmost byte of factor 2 is moved to zone of rightmost byte of result field.
Result Field	Numeric	
Factor 2	Alphanumeric	
Result Field	Alphanumeric	
MLHZO		
Factor 2	Alphanumeric	Zone of rightmost byte of factor 2 is moved to zone of leftmost byte of result field.
Result Field	Alphanumeric	
Factor 2	Numeric	
Result Field	Alphanumeric	
MHHZO		
Factor 2	Alphanumeric	Zone of leftmost byte of factor 2 is moved to zone of leftmost byte of result field.
Result Field	Alphanumeric	

Move-High-to-Low Zone (MHLZO)

The zone of the leftmost character of the Factor 2 field is moved to the rightmost zone position of the Result Field. The Factor 2 field must be alphanumeric, but the Result Field may be either alphanumeric or numeric.

Move-Low-to-Low Zone (MLLZO)

The zone of the rightmost character of the Factor 2 field is moved to the leftmost zone position of the Result Field. The Factor 2 field and the Result Field may be either alphanumeric or numeric.

Move-Low-to-High Zone (MLHAO)

The zone of the rightmost character of the Factor 2 field is moved to the leftmost zone position of the Result Field. The Factor 2 field may be either alphanumeric or numeric, but the Result Field must be alphanumeric.

COMPARE AND TESTING OPERATIONS

The compare (COMP) and test zone (TESTZ) operations test fields for certain conditions. The results of the tests are shown in resulting indicators; contents of the specified fields are not changed by the operations.

Compare (COMP)

The operation COMP compares the literal or the contents of the field in Factor 1 against the literal or contents of the field in Factor 2. The Factor 1 and Factor 2 fields must have identical formats: both numeric or both alphanumeric. An entire array cannot be specified as either factor. The Result Field must be left blank.

At least one resulting indicator (columns 54 through 59) must be specified. Depending upon the results of the comparison, the appropriate indicator is turned on as follows:

High – Factor 1 is greater than Factor 2.

Low – Factor 1 is less than Factor 2.

Equal – Factor 1 equals Factor 2.

A single indicator may be specified in two resulting indicator positions but should not be specified in all three resulting indicator positions.

Fields and literals are automatically aligned by RPG II before the comparison. Alphanumeric fields are aligned to their leftmost characters, and numeric fields are aligned by decimal points.

The maximum length of numeric fields is 15 digits. When one field is shorter than the other, the shorter field is padded with zeros until the lengths are equal.

The maximum length of alphanumeric fields is 256 characters. When one field is shorter than the other, the shorter field is padded with blanks on the right until the lengths are equal.

When an alternate collating sequence is supplied, the alphanumeric comparison is based upon that; otherwise, the alphanumeric comparison is according to the collating sequence shown in appendix D.

Test Zone (TESTZ)

This operation tests the zone of the leftmost character in the Result Field. The Result Field must be alphanumeric. The resulting indicators (columns 54 through 59) are set according to the following characters appearing in the leftmost position of the Result Field column:

High – & (ampersand) or A through I

Low –] (bracket), - (minus sign), or J through R

Equal – Any other character

Factor 1 and Factor 2 should be left blank for this operation.

An alternate collating sequence does not apply to the TESTZ operation.

BIT OPERATIONS

The bit operations allow users to set (BITON), clear (BITOF), and test (TESTB) bits that can be used for conditioning operations.

The Result Field specification must be a one-character alphanumeric field or element in a table or array.

The Factor 2 specification can have two formats:

- Bit numbers (0 through 7) can be specified. The numbers specify which bits are to be set on, set off, or tested. Up to eight bits can be specified in any order. Bits of the one-character Result Field are numbered 0 through 7 from left to right. Bits specified in Factor 2 are enclosed in apostrophes. When Factor 2 has a value of 0, the leftmost bit of the Result Field is to be set or tested. An entry of 7456 causes bits 4 through 7 (the rightmost four bits) to be set or tested.
- Factor 2 can also be a one-character field or element in an array or table. In this case, the contents of the Factor 2 specification (that is, those bits that are equal to one) indicate the bits to be set or tested.

No matter which format is used for Factor 2, unspecified bit positions cause the corresponding Result Field bits to be ignored.

Bit operations can be conditioned by entries in columns 7 through 17. If Field Length is specified, the value must be one. Factor 1, Decimal Positions, and Half Adjust columns must be blank for bit operations. Resulting Indicators (columns 54 through 59) are blank for BITON and BITOF, but must have at least one entry for TESTB operations.

All data fields are initialized by RPG II at the beginning of a job:

- Alphanumeric fields are initialized to blanks.
- Numeric fields are initialized to zeros with a positive sign.

If a field must be initialized to binary zero (hexadecimal X'00') for bit operations, the initialization must be specified in the RPG II source program.

Set Bit On (BITON)

The BITON operation causes bits specified in Factor 2 to be set equal to one in the corresponding Result Field bit positions. That is, the inclusive OR of Factor 2 and the Result Field are placed into the Result Field.

Set Bit Off (BITOF)

The BITOF operation causes bits specified in Factor 2 to be cleared to zero in the corresponding Result Field bit positions. The Result Field is ANDed with the complement of Factor 2.

Test Bit (TESTB)

Bits specified in Factor 2 of a TESTB operation cause corresponding bit positions in the Result Field specification to be tested for an on or off condition and resulting indicators to be set accordingly. (The Result Field is ANDed with Factor 2.)

At least one indicator must be specified in the resulting indicators field. A user may specify up to three, but a single indicator may not be specified in more than two positions. Resulting indicators are set according to the following conditions:

- Columns 54-55 – An indicator specified in these columns is set if each Result Field bit specified by the Factor 2 entry is off, with the exception of the anomaly listed with columns 58 and 59.
- Columns 56-57 – An indicator specified in these columns is set if the Result Field bits specified by the Factor 2 entry are in a mixed condition (some on, some off). If the Factor 2 entry causes only one bit to be tested, an indicator specified here is never set by the TESTB operation.
- Columns 58-59 – An indicator specified in these columns is set if each Result Field bit specified by the Factor 2 entry is on.

NOTE

If Factor 2 is a field that is all zeros, the indicator in columns 58 and 59 is turned on.

SETTING INDICATORS

RPG II provides two operation codes that are used to turn indicators on or off. Both may be conditioned by entries in columns 7 through 17.

Here are some points to remember when using SETON and SETOF:

- The following indicators may not be turned on by a SETON operation: 1P, MR, L0, and U1 through U8.
- The following indicators may not be turned off by a SETOF operation: 1P, MR, LR, L0, and U1 through U8.
- When LR is set on by a SETON operation conditioned by a control level indicator (columns 7 and 8), processing stops after all total output operations are completed. If the SETON is not conditioned by a control level indicator, processing stops after the next total output sequence is completed.
- If one or more halt indicators (H1 through H9) are turned on and not turned off before detail output operations are completed, processing stops at that time. Recovery from this type of halt is outlined in appendix B.
- Using SETON to set on a control level indicator (L1 through L9) does not automatically cause lower level indicators to go on.
- Indicators L1 through L9 and the record identifying indicators are always turned off after detail output time, regardless of the manner in which these indicators went on.
- Whenever a new record enters processing, the record identifying indicators and field indicators are set to reflect conditions on the new record. Previous settings of the indicators are lost.

Set On (SETON)

This operation turns on the indicator(s) specified in columns 54 and 55, 56 and 57, and 58 and 59.

Set Off (SETOF)

This operation turns off the indicator(s) specified in columns 54 and 55, 56 and 57, and 58 and 59.

BRANCHING OPERATIONS

Calculation operations are ordinarily performed in the order in which they are entered on specifications lines. Operations listed in this section, however, can alter this sequence for the following sample reasons:

- Some operations should be skipped as a result of certain conditions.
- Operations defined on preceding specifications lines should be repeated as a result of certain conditions.
- Some operations should be performed for some, but not all, record types.

Go To (GOTO)

If the conditions specified in columns 7 through 17 are met, control is transferred to the point in the program specified by the name in Factor 2. (See Tag section below.) Both forward and backward referencing is allowed. Be careful when transferring into a subroutine (SR in columns 7 and 8) that a transfer is made out of the subroutine before the ENDSR operation is encountered, since there is no program point available for the return.

When transferring from Detail calculation (blanks in columns 7 and 8) to Total calculation (L0 through L9 in columns 7 and 8), the RPG loop is altered. This may cause considerable confusion but may also be used to good advantage when used properly.

When conditions are not specified in columns 7 through 17 for the GOTO operation, the transfer always occurs.

Factor 1 and Result Field must be blank for a GOTO operation.

Tag (TAG)

The TAG operation defines names to which a GOTO operation may transfer control. The name goes in Factor 1 and may be one to six characters long. The first character must be alphabetic and placed in column 18. No special characters or embedded blanks may be used. The same label may not be used for more than one TAG operation code, but more than one GOTO is allowed to transfer control to a single TAG.

Columns 9 through 17 must be blank for a TAG operation code, but a control level indicator can be specified in columns 7 and 8 to signal that the branches will occur at total time.

LOOKUP OPERATIONS

RPG II provides a table lookup operation, encoded LOKUP, to be used for searching through a table or an array to find a specific element.

Lookup (LOKUP)

This operation code causes a search for a particular table or array element to be executed. Factor 1 specifies the search argument (data for which a match is to be found in the table or array), and Factor 2 specifies the name of the internally stored argument table or array to be searched. RPG II begins all table searches with the first element and stops searching when conditions have been satisfied. Array searches begin with the index specified in Factor 2; if none is given, the search begins with the first element. At least one indicator, but not three, must be designated in columns 54 through 59 to specify the type of search.

The search argument (Factor 1) may be:

- An alphanumeric or numeric constant
- A field name
- A table name
- An array element name

As described below, when a table is named as Factor 1, the table element last selected by a LOKUP operation is the search argument. Factor 1 and Factor 2 must have equal lengths and be of the same type (numeric or alphanumeric), but decimal positions of the two items are ignored.

The type of search is specified by entries in the resulting indicators field (columns 54 through 59). Before the search, indicators specified there are turned off. Indicators are turned on only when the conditions are met. Possible conditions are:

High (columns 54 and 55) – The search is terminated and this indicator turned on when the smallest (and first) entry greater than Factor 1 is found.

Low (columns 56 and 57) – The search is terminated and this indicator turned on when the largest (and last) entry less than Factor 1 is found.

Equal (columns 58 and 59) – The search is terminated and this indicator turned on when the first entry equal to Factor 1 is found.

Combinations of High and Equal and Low and Equal may be specified with Equal given precedence. That is, if an Equal cannot be found, the nearest higher or nearest lower entry is selected. High and Low may not be specified simultaneously.

Algebraic comparisons are made for numeric quantities. Alphanumeric quantities are compared according to the collating sequence specified in appendix D. An alternate collating sequence does not apply to LOKUP.

Tables and arrays to be searched by LOKUP must have an entry in column 45 (Sequence) of form E unless the search is for an equal condition only.

LOKUP operations may be conditioned by entries in columns 7 through 17.

LOKUP with One Table

If a single-table search is successful, RPG II retrieves the table value and stores it in a holding area set aside specifically for that table. Previous contents of the holding area (results of previous LOKUP operations) are destroyed. If the search is unsuccessful, the contents of the holding area are not altered.

References to a table name in later non-LOKUP operations actually refer to the table item in the holding area. Thus, later operations should be conditioned by the indicator that signals a successful search.

When a table name is specified as the search argument in Factor 1 of a LOKUP operation, RPG II uses the current value in the holding area as the search argument.

When a table name which has been Factor 2 in a successful search is specified as the Result Field of a later calculation operation, the holding area and the corresponding table element receive the result of the operation. Object programs can alter table elements in this manner.

LOKUP with Two Tables

Related tables can be specified as Factor 2 and the Result Field of a LOKUP operation. In this type of operation, Factor 1 designates the search argument, and Factor 2 specifies the table that is actually searched. When a search is successful, two items are moved to holding areas: the Factor 2 item resulting from the search and the corresponding item from the related table. Both holding area values remain available according to the rules outlined for a single-table search. When the related tables do not contain the same number of entries, the shorter number of elements determines the maximum number of items searched. Trailing elements in the longer table are ignored.

LOKUP with Arrays

When a LOKUP operation is performed with an array named as Factor 2, the Result Field must be blank. Searching an array is similar to searching a table except that RPG II does not move an array element to a holding area if the search is successful. However, RPG II does set appropriate resulting indicators after a successful array search.

When Factor 2 is an unindexed array, the search begins at the first element of the array.

When Factor 2 is an array name and index, the search begins with the element specified by the index. The Factor 2 format in this case is

[array name],[index]

where index can be a numeric field name or a numeric literal (with no decimal positions).

When index is a field name, a successful search causes RPG II to place the element number in the field designated as index. The array element can then be referenced in later calculation and output operations, using the array and index names.

When index is a field name and the search is unsuccessful, RPG II places a 1 in the index field unless the low indicator and a starting index greater than 1 are specified. In this case, the starting index is decremented by 1.

When index is a literal, only the indicators tell if the search conditions are satisfied. That is, RPG II does not change the value of the literal or otherwise identify the array element satisfying the search.

If a literal or field index is zero or greater than the number of elements in the array, the following results.

- With a literal index, the compiler detects the error and causes compilation to cease.
- With a field index, the error is not detected until run-time, and program execution is halted. If execution is continued (according to appendix B), the field index is given a value of one.

Examples of LOKUP operations

LOKUP operations with tables and arrays are illustrated in figure 9-3. At lines 010 through 040, a value stored at ACTNO is used to search the table TABACN, which has a related table, TABCUR. When the value at ACTNO matches an element of TABACN, the data item found at CHRGE is added to the related table element of TABCUR. When the value at ACTNO does not match any element of TABACN, indicator 11 is used to condition a branch to a TAG named NOMTCH.

At lines 110 through 140, there is a LOKUP of an array named ARRAY in which the purpose of the search is to find the array element that is nearest to but higher in value than the data item stored at ARG. A successful search causes indicator 20 to turn on and a branch to the TAG named NEXT to be taken. (Action taken for an unsuccessful search is not shown in the example.) Notice that this LOKUP indicates the presence or absence of the specified element in the array but does not show the element's location in the array or make the element available for future calculations.

Lines 210 through 250 show approximately the same LOKUP operation – indicator 20 is turned on when the first

CONTROL DATA CORPORATION — RPG CALCULATION SPECIFICATIONS — Printed in U.S.A.

Program | Punching Instruction | Graphic | Punch | Card Electro Number
Programmer | Date
Page 01 of __ | Program Identification

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus / Minus / Zero); Compare (1>2 / 1<2 / 1=2); Lookup(Factor 2)is (High / Low / Equal)	Comments
010	C			ACTNO	LOKUP	TABACN	TABCUR				11 10	RETRIEVE CURRNT BALANCE
020	C	*										
030	C		11		GOTO	NOMTCH						REPORT ERROR
040	C		10	CHRGE	ADD	TABCUR	TABCUR					UPDATE BALANCE
050	C											
060	C											
070	C											
080	C											
090	C			ARG	LOKUP	ARRAY					20	
100	C		20		GOTO	NEXT						BRANCH WHEN HIGHER ELEMENT FOUND
110	C	*										
120	C	*										
130	C											
140	C											
150	C											
160	C											
170	C				Z-ADD	1	X		20			
180	C			ARG	LOKUP	ARRAY,X					20	
190	C		N20		GOTO	FIN						
200	C				MOVE	ARRAY,X	HOLD		51			
210	C			NEXT	TAG							
220	C											
230	C											
240	C											
250	C											

Figure 9-3. Examples of LOKUP Operations

element higher in value than ARG is found. In this case, however, ARRAY is searched one element at a time, and an element satisfying the search is available through an index field named X. The index field is initialized to one (to point to the first element of the array) at line 210. If the desired element is found, X contains the index value of the element, and at line 240 the value of the element itself is moved to a location named HOLD for future reference. If the desired element is not found, X contains the value 1 at the conclusion of the LOKUP, and a branch to FIN is taken.

In any of the examples shown in figure 9-3, the search argument specified as Factor 1 in a LOKUP operation could have been a literal, rather than a constant, value.

SUBROUTINE OPERATIONS

A *subroutine* is a routine that is part of a main routine. A routine is a series of program steps that is executed over and over. An RPG II program is a main routine because the instructions are done over and over again (the program cycle). A subroutine is a series of instructions that may be performed several times within a single program cycle.

The purpose of a subroutine is to save lines of coding and the core memory space required for the object program. Instead of coding the same sequence of steps every time they are needed, the user will only need to code the group of steps once and can then refer to the group at every point in the program where it is needed. When the subroutine is contained in the main program, the series of steps is called an *internal subroutine*.

It may be that several main programs can use the same subroutine. Here again, the subroutine need be coded only once. References to the subroutine from programs that do not contain the subroutine are references to an *external subroutine*.

All subroutine operation codes are written in specification lines following all detail and total calculation lines. Lines of a subroutine (except AN or OR lines) are identified by SR in columns 7 and 8.

The first line of the subroutine must have a BEGSR operation code, and the last line must have an ENDSR operation code.

Even though the subroutine is entered last on Calculation Specifications, performance of the subroutine may occur at any point in the calculations portion of the program cycle.

All valid RPG II operations may be performed within the subroutine. Operations within the subroutine may be conditioned by any valid indicators in columns 9 through 17. Individual operations within a subroutine cannot be conditioned by control level indicators in columns 7 and 8, since those columns contain SR. However, execution of an entire subroutine can be conditioned by control level indicators in columns 7 and 8 of the referencing EXSR operation code.

Fields used in a subroutine may be defined either inside or outside the subroutine. In either case, a field may be used by both the main routine and the subroutine, and the name and characteristics of the field are identical in both usages.

An RPG II program may contain more than one subroutine. Each such subroutine is delimited by the BEGSR and ENDSR operation lines, and the subroutines may be specified in any order at the end of Calculation Specifications. A subroutine may not be contained within another subroutine; that is, each pair of BEGSR and ENDSR lines must be coded without intervening BEGSR/ENDSR lines. One subroutine may call another subroutine, however.

In other words, a subroutine may contain an EXSR statement. A subroutine may call itself or may call a subroutine that calls it.

A subroutine may contain a GOTO operation code, which branches either to a point within the subroutine or to any other point in the program. Program flow should make sense – the program should not be allowed to flow into an ENDSR operation if a corresponding EXSR has not been performed for the same subroutine.

Begin Subroutine (BEGSR)

The BEGSR operation defines the entry point of an RPG II subroutine and is the first physical statement of the subroutine. The left-justified name of the subroutine is defined in Factor 1. Columns 7 and 8 must contain SR; all other columns, except 60 through 74 (Comments), must be blank.

Every subroutine must have a unique name. The name may have one to six alphabetic or numeric characters, the first of which must be alphabetic. Embedded blanks are not allowed in the name.

End Subroutine (ENDSR)

An ENDSR operation is the last physical statement of an RPG II subroutine and causes an automatic branch back to the calling program. The Factor 1 field may contain a name referenced by GOTO operations within the subroutine. Columns 7 and 8 must contain SR.

Execute Internal Subroutine (EXSR)

The EXSR operation causes a transfer of control to an internal subroutine written in the RPG II language as a part of the same program. EXSR may appear anywhere in the program. After the subroutine has been performed, control returns to the next specification following the EXSR operation.

The name of the subroutine is left-justified in Factor 2. This name must be identical to the Factor 1 specification for a BEGSR operation code.

The EXSR operation code may be conditioned by any indicators; this means that the subroutine is executed only if all the conditions are met.

Exit-to-External Subroutine (EXIT or EXITF)

The EXIT operation transfers control to an external subroutine supplied by the user. EXITF is equivalent to EXIT but is used to exit to a FORTRAN subroutine (see

appendix G). The name of the subroutine is left-justified in factor 2. The EXIT operation can be conditioned by entries in columns 7 through 17. The resulting indicators field must be blank.

The EXIT operation code may be conditioned by any indicators; this means that the external subroutine is executed only if all the conditions are met.

Assembly language coding usage of external subroutines is described in appendix G.

Define-Label-for-External Subroutine (RLABL)

RLBABL defines a parameter that may be referenced by an external subroutine. The parameter may be a file name, field name, table, array, indicator, constant, or literal.

The RLABL calculation specification must contain RLABL in columns 28 through 32 and the parameter left-adjusted in Result Field Name (columns 43 through 48). Field length in columns 49 through 51 and decimal positions in column 52 are optional. All other columns must be blank.

The RLABL specifications must immediately follow the EXIT specification, which refers to the parameters. Indicators are referenced by coding INxx in Result Field Name where xx is the indicator code. For example, Matching Record Indicator is coded INMR.

PROGRAMMED I/O CONTROL

In the RPG II program logic outlined in appendix F, the normal sequence of events in a program cycle is:

1. Read a record.
2. Perform calculations.
3. Output a record.

Deviation from this pattern is possible via six operation codes that allow input or output at calculations time:

- EXCPT (exception)
- FORCE (force)
- DSPLY (display)
- READ (read)
- CHAIN (chain)
- SETLL (set lower limit)

Exception (EXCPT)

The EXCPT operation allows exception records to be written at calculations time. EXCPT is primarily used to produce a variable number of similar or identical records (either detail or total) in a single program cycle. One usage would be to write out the contents of a table in a program cycle.

An EXCPT operation may be conditioned by entries in columns 7 through 17. All other columns except 60 through 74 (Comments) must be blank. The exception output records are indicated by an E in column 15 of the Output Specifications sheet. Exception records may not be specified for a combined file.

Force (FORCE)

The FORCE operation overrides normal record selection and allows the user to select the file that will supply input for the next program cycle. The forced file can be a primary or secondary input, update, or combined file. Factor 2 names the file from which a record is to be forced.

The FORCE operation can be conditioned by columns 7 through 17, but all other columns except 60 through 74 (Comments) must be blank. FORCE should not be specified at total calculations time.

If a FORCE operation is executed, the next program cycle will process the next available record from the forced file. If several FORCE operations are executed in one calculations cycle, all but the last operation are ignored. The effect of a FORCE lasts only one cycle.

In multi-file processing, RPG II selects the first record to be processed in the normal fashion, but thereafter records can be forced in an abnormal sequence. At end-of-file on a forced file, normal record selection resumes.

Display (DSPLY)

Through a DSPLY operation, either or both of the following can occur:

- Up to 125 characters are printed from storage onto the console device without a halt in program execution.
- Up to 125 characters are printed from storage onto the console device. The program then halts, allowing the contents of the displayed item to be altered from the console.

With the first usage, the item to be displayed can be a field, a table element (holding area element), an array element, or a literal. The item is specified as Factor 1 with the DSPLY operation code.

For the second usage, the item to be displayed (and perhaps altered) can be a field, a table element, or an array element. A literal can be specified but cannot be altered during program execution. The item is specified as the Result Field with the DSPLY operation code.

Factor 2 of the DSPLY operation line designates the file on which Factor 1 and/or the Result Field is displayed. A display file is described completely on form F; the file needs no record descriptions. A single DSPLY operation may have specifications in both Factor 1 and the Result Field. In this case, contents of both Factor 1 and the Result Field are displayed. Program execution halts. The Result Field, but not Factor 1, can be changed.

A DSPLY operation may be conditioned by indicators in columns 7 through 17.

The rules for entering data during program execution are:

- Leading zeros need not be supplied for numeric data; RPG II automatically right-justifies numeric data after all characters have been keyed.
- A trailing minus sign is keyed for negative numeric data. Field length need not take the trailing minus sign into consideration.

- Alphanumeric fields are automatically left-justified after all characters have been keyed.
- Typing a single space causes a numeric field to be cleared to zeros and an alphanumeric field to be set to blanks.
- If no characters are entered and the space bar is not pressed, the Result Field item is not changed.

Specific information on the entering of characters and clearing the halt is provided in appendix B.

Read (READ)

The READ operation causes RPG II to read and process a record from a demand file during the calculations cycle. Unlike a record read by a FORCE operation, RPG II reads the record immediately, not at the start of the next program cycle. The READ operation is similar to the CHAIN operation, except that the READ file is processed sequentially and the CHAIN file is processed randomly.

A demand file is described on forms F and I, and the same filename is entered as Factor 2 of the READ operation. Columns 7 through 17 of the READ operation may contain entries. All other columns should be left blank except for an indicator specification in columns 58 and 59 and comments in columns 60 through 74. An indicator specified in columns 58 and 59 is turned on when end-of-file is reached on the demand file and is turned on thereafter every time a read from the demand file is attempted. Specification of an indicator in columns 58 and 59 is desirable, because otherwise the object program halts when the demand file is at end-of-file and every time thereafter that an attempt is made to read from the file.

A demand file's record identifying indicator remains on throughout a program cycle after a record has been read from the file. This is also true for multiple reads from one or more demand files in a single cycle.

A demand file is designated by a D in column 16 of form F. Demand file records may be accessed only through READ operations. Control levels, matching records, and look-ahead fields may not be specified for demand files. Furthermore, numeric sequence checking must not be requested for the file, and the MR indicator cannot be specified in columns 63 and 64 (Field Record Relation) of the input sheet.

When a demand file is conditioned by an external indicator (U1 through U8) that is not on, no records are read from the file, and the end-of-file indicator (columns 58 and 59) is not set on.

Demand files are input, update, or combined files that are accessed by the following methods:

- Sequential or direct disk files are processed consecutively.
- Indexed disk files are processed sequentially by key or within limits.
- Card, tape, or console files are processed sequentially.

Chain (CHAIN)

The CHAIN operation is used for two purposes.

- Random processing of an indexed, sequential, or direct disk file
- Loading (creating) a direct disk file

The operation requires Factor 1 and Factor 2 entries, but columns 43 through 53 and columns 56 through 59 must be blank. A CHAIN operation may be conditioned by entries in columns 7 through 17. A chained file may be conditioned by an external indicator (U1 through U8).

An indicator should be specified in columns 54 and 55 of the CHAIN operation. This indicator is set on when a record is not found; output cannot occur to a chained file when the specified record is not found unless the operation was intended for file addition. If columns 54 and 55 are blank and the record is not found, program execution halts.

When RPG II processes chained files, more than one record identifying indicator may be on at the same time in a single program cycle. This would happen if the program is chaining to one or more files during the same RPG II cycle. When chaining to the same file more than once during a cycle, only the last record processed is updated during output time unless an exception output operation is associated with each chain operation.

NOTE

If the same physical file is designated INPUT and UPDATE in the same program, successive chains to the same record may yield the old data after update, because the chained records are taken from the input buffer instead of the disk.

When chaining to a file having packed record keys, the Factor 1 entry must have a packed length equal to the key field length in the chained file. A packed key field is limited to eight bytes.

Random Input Processing

The entry in Factor 1 of the CHAIN operation identifies the record to be retrieved. The value is either a numeric relative record number (for sequential and direct files) or a record key (for indexed files). The relative record number or key can be contained in a field specified for that purpose. The entry in Factor 2 identifies the (chained) file. This file must be described on forms F and I.

Direct Output File Load

When a direct file is created (loaded), the file is described as a chained output file on forms F and O. Factor 1 names a relative record number, and Factor 2 names the file.

Relative record numbers define the record position of each record in the file. A relative record number can be all or part of a field in an input record, or it may be generated by the RPG II program.

When a direct file load is to occur, RPG II clears the entire area reserved for the file to blanks. As a record is being output, the relative record number is used to chain to the disk record to which the output occurs. Records not specified in the direct file load remain blanks.

A direct file load is very different from record addition to sequential or indexed files. In the latter cases, a data record is added to the first available position at the end of the file. Also, in the case of an indexed file, the record key and disk address are added to the file index. With a direct disk file, each record has a previously allocated position; each record is chained to its proper physical position at the time it is loaded.

Once the direct file has been created, the file can be modified in other RPG II program executions by defining the file as an update file processed consecutively or by the CHAIN operation.

It may be that duplicate record numbers will be entered as Factor 1. These record numbers are called *synonyms*. This could occur if a loaded record is composed of portions of several other records. However, only the last data loaded for one record number appears in the created file. Two methods are available for the satisfactory processing of synonyms:

- Define the file as a chained output file in order to clear the file to blanks in the first job. Then run one or more subsequent jobs using the update feature to read relative record locations, checking for synonyms, while loading the file.
- Define the file as a chained output file, clearing the file to blanks, and load all records not having synonyms in a first job. Then run a subsequent job using the update feature to select synonym records and load them into the file.

Set Lower Limits (SETLL)

The SETLL operation allows the lower limits (for indexed demand files being processed within limits) to be set during program execution.

Factor 1 contains a field name or literal that specifies the value of the lower limit being set. Factor 2 designates the name of the file for which the lower limit is set. The length of the field or literal must equal the length of the key for the file named as Factor 2.

If the file is accessed before a lower limit is set by SETLL, the record with the lowest key in the file is read.

When end-of-file is reached on the demand file, additional records may still be read from the file through the issuance of another SETLL operation. In other words, the object program is not limited to one SETLL operation for a demand file. However, if processing of the file continues after an end-of-file condition has been reached, the end-of-file indicator specified with the READ command is not turned off by RPG II.

DEBUG OPERATION

The DEBUG operation can assist in isolating programming errors in the object program. Whenever a DEBUG operation is executed, one or more records are written to an output file.

Debug (DEBUG)

The DEBUG operation may be placed at any place or several places in the calculation operations. Whenever the operation is executed, one or more records are written to an output file. The first record always lists the indicators that are currently on at the time of execution of the DEBUG operation. Following records show the contents of any one field.

A DEBUG operation is executed only if requested through a 1 in column 15 of the control card specifications (form H). When that column is blank, RPG II diagnoses each occurrence of a DEBUG operation with a warning. Both the DEBUG operation and the error message are shown on the source program listing.

On the DEBUG statement, Factor 1 is optional. It may contain a literal or field name to identify the particular DEBUG output. The literal or the value of the field named here is merely written in the first DEBUG output record. A Factor 2 entry is required and names the output file to which DEBUG records are written. All DEBUG statements in a program must refer to a single output file. The optional Result Field may name a field, a table or array element, or an entire array whose contents are to be written in the second DEBUG output record. Any valid indicators may be specified in columns 7 through 17. Columns 49 through 59 of the DEBUG operation must be blank.

Because of additional processing considerations, it is undesirable to name a direct or indexed file as the DEBUG output file.

The first DEBUG record is automatically produced when the operation is executed and is a list of all the indicators that are currently on in the object program. The format of the first record is:

Record Positions	Information
2-7	DEBUG-
8	Blank
9-16	Constant specified as Factor 1 or statement number of the DEBUG operation code in the program.
17	Blank
18-31	INDICATORS ON-
32-end of record	Names of all indicators that are on, separated by blanks or the word NONE

If the indicator list does not fit in one record, the list is continued in the next record, starting at position 2.

RPG II writes additional records to the output file if a Result Field is specified for the DEBUG operation. The format of these records is:

Record Positions	Information
2-12	FIELD VALUE or TABLE VALUE or ARRAY VALUE
13-14	Blank
15-end of record	Contents of the Result Field specification (up to 256 characters per item)

The record is continued starting in position 2 of the next record, if required. These additional records adhere to the following rules:

- Decimal points are not produced.
- A minus sign is appended to a negative item.
- Array elements are written in order, separated by blanks.
- An array element is not split between physical records if the element size is less than the size of the physical record.

An example of coding with the DEBUG operation is shown in figure 9-4. A search is made of ITMARY for an element matching ITMNO. If a match is not found, the following records are written to DBGOUT (assuming form H contains a one in column 15).

- A list of indicators currently on
- The word ITMNO and the value of that field
- A list of indicators currently on
- The contents of the entire array, ITMARY

When a match is not found, the MOVE operation is bypassed. When a match is found, DEBUG output is not produced, and the MOVE operation is executed.

Get Time-of-Day (TIME)

Column	Value
43-48	Name of field, table, array, or element to receive time

The TIME operation allows the current time of day to be read from the computer system during program execution. The Result Field specifies the field, table, array, or array element into which the time is to be stored. Time-of-day is maintained as a six-digit value in the form hhmmss (hours, minutes, and seconds).

Factor 1 and factor 2 must be blank.

RESULT FIELD

Columns	Values
43-48	Name of field, table, array, or array element.

This entry names a field, table, array, or array element used by the operation code specified. Generally, this field receives the result of the operation specified in columns 28 through 32. Other usages are described with the operation codes.

The name must conform to the rules for naming fields, tables, and arrays. The name may be previously defined in

CONTROL DATA CORPORATION — RPG CALCULATION SPECIFICATIONS — Printed in U.S.A.

Program / Programmer / Date / Punching Instruction / Graphic Punch / Card Electro Number / Page 01 of __ / Program Identification

Line	Form Type	Control Level (L0 L9, LR, SR, AN/OR)	Indicators: And / And / Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus / Minus / Zero; Compare 1>2 / 1<2 / 1=2; Lookup(Factor 2)is High / Low / Equal	Comments
01	C				Z-ADD1		INX					
02	C			ITMNØ	LØKUPITMARY,INX						99	
03	C		N99	'ITMNØ'	DEBUGDBFILE		ITMNØ					
04	C		N99		DEBUGDBFILE		ITMARY					
05	C		99		MØVE ITMARY,INX		SAVE					
06	C											

Figure 9-4. Example of DEBUG Operations

the extension specifications, input specifications, or elsewhere in the calculation specifications.

Otherwise, the Result Field name together with entries in columns 49 through 52 define the Result Field for further references. The new field may be either numeric or alphanumeric. A field used in arithmetic operations or numeric compare, or a field edited or zero suppressed in output specifications must be numeric. A numeric field is initialized to zeros by RPG II, and an alphanumeric field is initialized to blanks.

When the name has already been defined, entries in columns 49 through 52, if any, must agree with the previous definition.

FIELD LENGTH

Columns	Values	Meanings
49-51	Blanks	Field is defined elsewhere
	1-256	Result Field length

The length of the Result Field may be entered in columns 49 through 51, right-justified. Leading zeros may be omitted. When these columns contain an entry, the form of the data and its length after the operation has been performed must be considered. When the field is not long enough to contain a result, the most significant digits to the left of the decimal point are lost. (The least significant digits to the right of the decimal point may also be lost.) Unpacked lengths are specified here for numeric fields, and the maximum length allowed is 15 digits. Alphanumeric fields may be up to 256 characters long.

If the Result Field has been described in previous input or calculation specifications, any entry here must agree with the previous definition. If the Result Field contains a table or array name, an entry in these columns is optional, but if specified, must agree with the length described in the Extension Specifications.

DECIMAL POSITIONS

Column	Values	Meanings
52	Blank	a. Field is defined elsewhere, or b. Alphanumeric field
	0-9	Number of decimal positions in a numeric Result Field

This entry specifies the number of positions to the right of the decimal point in a numeric Result Field. No entry is made for alphanumeric fields. A zero is entered in column 52 if the numeric Result Field contains an integer.

An entry in column 52 requires an entry in columns 49 through 51 (Field Length).

A Result Field may not have more than nine decimal positions, and the number of decimal positions must never exceed the field length. The number may, however, be larger or smaller than the number of decimal positions that actually result from the operation. When the result of an operation produces fewer decimal positions than specified here, RPG II pads the low-order positions with zeros. If the specified number of decimal positions is smaller than the number that results from the operation, the rightmost digits are dropped (i.e., the result is not rounded).

HALF ADJUST

Column	Values	Meanings
53	Blank	a. Not an arithmetic operation, or b. Do not half-adjust.
	H	Half-adjust the result.

An H in this column specifies that the result of an arithmetic operation is to be half-adjusted (rounded). Half-adjusting is accomplished by adding a 5 to the position immediately to the right of the last decimal position specified for this field.

The addition is performed on the absolute value of the number to be stored. The correct sign is replaced before the rounded result is stored. All positions to the right of the decimal positions specified for the field are then dropped.

Half-adjust must not be specified with an MVR operation, nor may it be specified for a DIV operation followed by an MVR operation.

RESULTING INDICATORS

Columns	Values	Meanings
54-59	Blanks	No entry
	10-99	Numeric indicators
	H1-H9	Halt indicators
	L1-L9	Control level indicators
	LR	Last record indicator
	OA-OG, OV	Overflow indicators

Indicators specified in these columns serve four purposes:

- To describe the value of a Result Field after an arithmetic operation
- To indicate the outcome of a CHAIN, LOKUP, COMP, TESTB, or TESTZ operation
- To specify indicators for SETON and SETOF operations
- To signal end-of-file for the READ operation code

The specific uses of this field have been defined with the various operation codes, with one exception: When arithmetic operations are performed, the results are tested and indicators are turned on as follows:

- An indicator specified in columns 54 and 55 is turned on if the result is positive.
- An indicator specified in columns 56 and 57 is turned on if the result is negative.
- An indicator specified in columns 58 and 59 is turned on if the result is zero

A resulting indicator is typically used to condition following calculation or output operations.

NOTE

When LR is used as a resulting indicator, NLR should be used to condition the same operation (columns 9 through 17) to avoid setting off a valid LR indication.

If a single resulting indicator is specified on more than one calculations line, its condition reflects the result of the most recently executed operation. Thus, it is usually good practice to assign unique indicators as field indicators and/or resulting indicators in one program.

All halt indicators (H1 through H9) are off at the beginning of program execution. If more than one halt indicator is on when the program halts at the end of detail output, each halt indicator must be responded to separately. This is because it may be desirable to continue program execution under certain error conditions but terminate execution under others. By assigning unique halt indicators to various error conditions, the nature of an error condition can be readily identified.

COMMENTS

Columns	Values
60-74	Any characters

These columns are not interpreted by the RPG II compiler but are printed on the source program listing.

CALCULATION SPECIFICATIONS EXAMPLES

Highlights of some of the principles discussed in this section are illustrated in figure 9-5 and are discussed below.

Line 010 If halt indicator H2 is on, indicators H2 and L1 are set off.

Line 020 Bits 0 through 3 of the flag word named FLAGS are set off.

Line 030 This is a comments line (designated by the asterisk in column 7).

Line 040 If the matching record (MR) indicator is on, bit 0 in word FLAGS is set on.

Line 050 The value found at MINMUM becomes the lower limit to be used for processing the indexed demand file, RFILE, within limits.

Line 060 Control is transferred to the subroutine named SUB.

Line 070 The flag word FLAGS is interrogated. If both bit 0 and bit 1 of the word are on, indicator 19 is set on.

Line 080 If this is detail time in the cycle but control level indicator L1 is not on and indicator 19 is on, then the zone portion of the rightmost byte of ARG1 is moved to the zone portion of the rightmost byte or ARG2.

Line 090 If indicator 19 is on, the contents of FACT1 are divided by the contents of FACT2, and the result replaces FACT1.

Line 100 The divide operation at line 090 is immediately followed by an operation to move the remainder produced by the divide to a word called REM. REM is defined on this line to be six bytes long with room for two digits to the right of the decimal point. The remainder is rounded, if appropriate, before being stored. Notice that the MVR operation uses the same conditioning indicator (19) that the DIV operation used.

Line 120 In this operation, characters found at FIRST are moved, left-justified, into a newly-defined field, NAME, which is 16 alphanumeric characters long.

Line 130 If a level 1 control break has occurred, bit 3 of the word FLAGS is set on.

Line 140 When a level 1 control break has occurred, the last record (LR) indicator is on, the matching record (MR) indicator is not on, and control is transferred to label TAG.

Line 150 When a level 1 control break has occurred, a two-digit numeric integer field is initialized to 99.

Line 170 Defines the label TAG, to which control was transferred at line 140

Line 130 When a level 1 control break has occurred, the alphanumeric literal TOTAL is moved to a field named CONST.

Line 190 This entry names the beginning of a subroutine. Lines up through 240 are contained in the subroutine which is named SUB.

Line 200 A record is read from the demand file named RFILE. If the end of the file has been reached, halt indicator H2 is set.

Line 210 If H2 is on at this point, bit 1 of the word FLAGS is set on.

Line 220 Also if indicator H2 is on, control is transferred to SEXIT.

Line 240 SEXIT defines the label of the last instruction and the exit of the subroutine.

CONTROL DATA CORPORATION

RPG CALCULATION SPECIFICATIONS

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

Programmer | Date

Page 01 | Program Identification

C

Line	Form Type	Control Level (L0 L9, LR SR AN OR)	Indicators And Not	Indicators And Not	Indicators And Not	Factor 1	Operation	Factor 2	Result Field Name	Result Field Length	Result Field Decimal Positions	Result Field Half Adjust (H)	Resulting Indicators Arithmetic Plus	Resulting Indicators Arithmetic Minus	Resulting Indicators Arithmetic Zero	Resulting Indicators Compare 1 2	Resulting Indicators Compare 1 < 2	Resulting Indicators Compare 1 > 2	Resulting Indicators Lookup Factor 2 High	Resulting Indicators Lookup Factor 2 Low	Resulting Indicators Lookup Factor 2 Equal	Comments
010	C		H2				SETOF						H2	L1								
020	C						BITOF	'0123'	FLAGS													RESET FIRST
030	C	*																				FOUR FLAGS
040	C		MR				BITON	'0'	FLAGS													
050	C					MINMUM	SETLLR	FILE														
060	C						EXSR	SUB														READ FROM FILE
070	C						TESTB	'01'	FLAGS												19	
080	C		19	NL1			MLLZO	ARG1	ARG2													
090	C		19			FACT1	DIV	FACT2	FACT1													
100	C		19				MVR		REM	6	2	H										
110	C						)															
120	C						MOVEL	FIRST	NAME	16												LAST DETAIL OP
130	C	L1					BITON	'3'	FLAGS													
140	C	L1	NMR	LR			GOTO	TAG														
150	C	L1					Z-ADD	99	HINO	2	0											
160	C	L1					)															
170	C	L1				TAG	TAG															
180	C	L1					MOVE	'TOTAL'	CONST													LAST TOTAL OP
190	C	SR				SUB	BEGSR															
200	C	SR					READ	RFILE													H2	
210	C	SR	H2				BITON	'1'	FLAGS													
220	C	SR	H2				GOTO	SEXIT														
230	C	SR					)															
240	C	SR				SEXIT	ENDSR															END SUBROUTINE
	C																					

Figure 9-5. Calculation Specifications Examples

Output specifications are provided for files designated as output, update, or combined files in the File Description Specifications (form F). Output specifications also apply to input files into which records are added (I in column 15 and A in column 66 of form F).

Form O logically subdivides into two general categories:

- Record description entries (columns 7 through 31) describing records or lines of print
- Field description entries (columns 23 through 70) that indicate the fields and constants comprising a record or line

The indicators portion of the form may condition output of a field or an entire record. These logical subdivisions are shown in figure 10-1.

Field descriptions are entered on lines immediately following the associated record description.

All output fields are initialized to blanks before each cycle.

FILENAME

Columns	Values	Meanings
7-14	Blanks	This is not the first line of description of this output file.
	Valid RPG II filename	This is the first line of description of this output file, and this is the name of the file.

The filename entered in columns 7 through 14 must be identical to a filename listed on the File Description Specifications sheet. The name is left-justified in these columns.

The filename needs to be specified only once on the Output Specifications sheet, but it must be entered on the first descriptive line of the file. All subsequent record and field description lines up to a different filename pertain to this file.

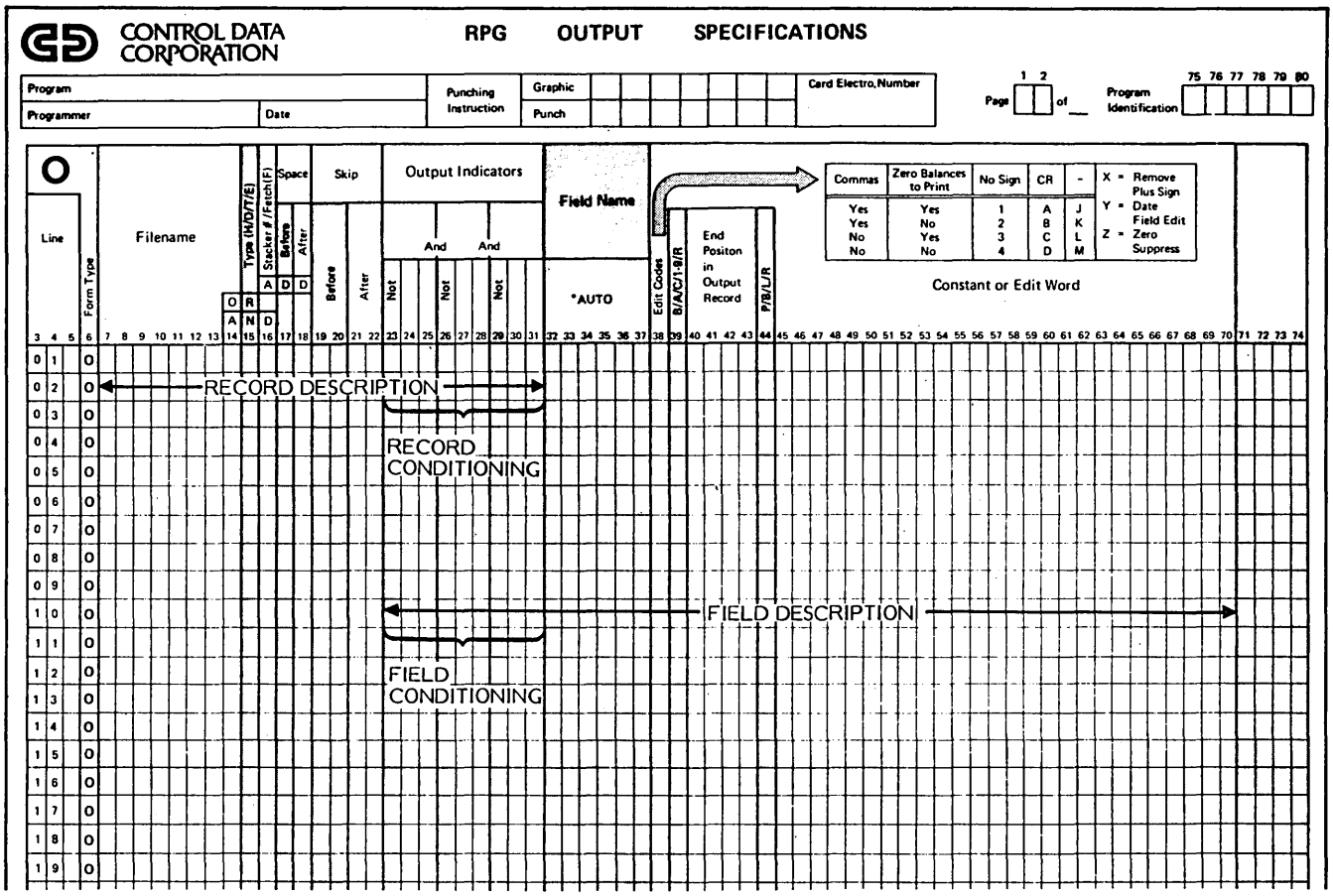


Figure 10-1. Output Specifications Sheet

AND/OR RELATIONSHIPS

Columns	Values	Meanings
14-16	Blanks	Field description line
	AND	Record description line. Output indicators on this line are in an AND relationship with those on the previous line.
14-15	OR	Record description line. Output indicators on this line are in an OR relationship with those on the previous line.

Form O allows up to three indicators per line for conditioning record output. These indicators are in an AND relationship. If more than three indicators are required to condition a record, an AND in columns 14 through 16 of the next line allows the specification of three more indicators. Then the condition of all indicators in an AND relationship must be satisfied before the output operation can be done.

An OR in columns 14 and 15 of the next line allows the user to specify up to three alternate output indicators for a record. As an example, the user may want to eject a printer page and print a report heading line (record) at the time of a control break (L2, for example) and when the last record has been read and is being processed. Thus, the user can request the page eject and the printing of the report heading when L2 OR LR is on. Simply specifying L2 OR LR, however, can yield duplicate page ejects and report headings when LR is on. This is avoided by encoding L2 and NLR (not last record) on one line and LR on the other.

Spacing and skipping specifications need not be specified on an OR line unless alternate spacing and skipping are desired.

When spacing and skipping are not specified on an OR line, any space and skip specifications on the previous line are used.

AND/OR lines cannot be used to condition fields of a record.

TYPE

Column	Values	Meanings
15	Blank	Field description line
	H	Heading record
	D	Detail record
	T	Total record
	E	Exception record

The specification in column 15 determines when a record should be output.

Heading records usually contain constant information such as report and column headings as well as page numbers. Conditioning heading records with the first page (1P) indicator is a common way to produce the records at the beginning of program execution. Subsequently, heading lines are usually printed as a result of overflow processing and skipping to a new page.

Detail records are closely connected to input data as they are ordinarily produced after each input record has been processed.

Total records are usually output after control breaks have occurred and total calculations have been performed and at end-of-job, when the LR indicator is on. Total records commonly contain the results of calculations on several input records. Unpredictable output may occur when total records are specified for primary or secondary update files.

Exception records are output when directed by the calculation operation EXCPT. Exception output cannot be specified for a combined file. Furthermore, exception output conditioned by control level indicators should not be specified for primary or secondary update files, as the results of the output may be unpredictable.

It is a customary programming procedure to prepare form O lines for each file in the following order: all heading records, followed by all detail records, followed by all total records, followed by exception records. This order is not mandatory, however.

ADD A RECORD

Columns	Values	Meanings
16-18	Blanks	a. Field description line, or b. This record is not being added.
	ADD	Add this record to this disk file.

This entry allows records to be added to existing input, update, and output files. The designated file must contain an A specification in column 66 of the File Description Specifications sheet.

The ADD entry must be on the first form O line describing the record to be added.

STACKER SELECT/FETCH OVERFLOW

Column	Values	Meanings
16	Blank	a. Use standard overflow routine for this printer file, or b. Use the predetermined stacker on this card device, or c. This card device does not have multiple stackers, or d. This is neither a printer nor a card file.

Column	Values	Meanings
16	1-4	Designates one of four stackers in which to place cards after they have been processed. Standard CYBER 18 devices do not provide for more than one stacker.
	F	Fetch overflow (printer files only)

Column 16 may be used for two entirely different purposes:

- To specify a stacker into which certain cards are to go after processing
- To request that the overflow routine be used at this point for a printer file

STACKER SELECT

Through column 16 one may indicate that certain cards are to fall into the designated stacker. When a stacker is not selected in column 16, all records of the file fall into the stacker designated in appendix B.

Only output and combined files may be stacker-selected through the output specifications. When the program is to perform output on a combined file, any stacker selection should be made in the output specifications rather than the input specifications. However, form O stacker selection overrides form I stacker selection.

When stacker selection is based upon matching records, the selection should be indicated for detail time (D in column 15), as it is only at this time that the MR indicator reflects the matching status of the record about to be sent to a stacker.

A stacker select entry may be made on an OR line but not an AND line. When column 16 is blank on an OR line, the record falls into the predetermined stacker; unlike spacing and skipping options, the OR line does not assume the stacker selection indicated for the preceding line.

FETCH OVERFLOW

Fetch overflow may be specified for any total, detail, or exception record that is not conditioned by an overflow indicator. An F in column 16 can cause overflow lines to be printed and the forms to advance before the printing of detail and total lines appropriate for the cycle is completed. Overflow can be fetched only if all the conditions specified in columns 23 through 31 are met and overflow has been sensed.

NOTE

Forms advance is not automatic when overflow is fetched. Forms advance takes place only by explicit request (probably on the first heading line for a new page).

When fetch overflow has not been specified, all detail and total lines conditioned for output in the cycle are printed. If any of those lines is printed on or below the overflow line, all such lines are printed before output lines conditioned by the overflow indicator are printed. If the overflow area is not large enough to hold all of these lines, unattractive output results. Use of the fetch overflow feature ensures overflow processing before the end of the page. Overflow indicator usage is summarized in table 10-1.

Fetch overflow must be designated for each line in an OR relationship if the overflow routine is to be fetched for each record in the relationship.

Partial specifications for fetching overflow are shown in figure 10-2. Lines 030, 050, and 060 can fetch overflow. This happens only when indicators OV and L5 and both set. Should overflow be fetched, the OV indicator is reset after the forms advance at line 010 is executed.

At the time that the record specified at line 030 is to be produced, indicators OV and L5 and interrogated. If they are both on, the forms are advanced (per line 010), the heading record specified at line 010 is written, and the records specified at lines 030 through 070 are then written on the new page. If overflow has not been sensed before the printing of line 030 but is set on by the printing of that line, overflow is fetched at the time of execution of line 050. If line 050 fetches overflow, the forms are advanced, the heading record is produced, and records corresponding to lines 050 through 070 are written on the new page. (Notice that if line 050 fetches overflow, line 060 does not, as indicator OV is reset at the time the heading record is printed.)

OVERFLOW PRINTING WITH EXCPT OPERATION CODE

An overflow indicator cannot condition an exception line, but it may condition fields within an exception line. The use of the EXCPT operation code with exception lines (E in column 15) causes exception lines to be printed during calculations time rather than output time. If the overflow line is sensed when an exception line is printed, an assigned overflow indicator is turned on, as expected, but overflow processing does not occur until another exception line specifying fetch overflow and having satisfied conditions is ready to be printed.

The actual overflow output lines must be type H, D, or T. The use of fetch overflow causes the H, D, or T overflow output lines to be printed if the overflow indicator is on. The overflow output lines are produced prior to the printing of the line on which fetch overflow is specified. Overflow may also be forced by issuance of a SETON of the appropriate overflow indicator prior to the EXCPT operation code, provided fetch overflow has been specified.

FORMS CONTROL-SPACE/SKIP

Through the space/skip feature, RPG II offers relative and absolute forms control to the user. If columns 17 through 22 are blank, single spacing occurs automatically after each line is printed.

TABLE 10-1. OVERFLOW INDICATOR USAGE

Conditions	Effect at Overflow Time
This file does not have an assigned overflow indicator.	1. The following lines are printed in the overflow area. • Any detail lines to be printed as a part of the current detail output. • Any total lines to be printed as a part of the current total output. 2. The form is automatically advanced to the top of the next page.
This file has an assigned overflow indicator, but the indicator is not used to condition any output lines.	1. The overflow indicator is turned on. 2. Detail and total printing continues in the overflow area. 3. The overflow indicator is turned off at the top of the next page.
This file has an assigned overflow indicator used to condition at least one output line.	1. The overflow indicator is turned on. 2. The following lines are printed in the overflow area. • Any detail lines to be printed as a part of the current detail output. • Any total lines to be printed as a part of the current total output. • Any total lines conditioned by the overflow indicator. • Any header or detail lines conditioned by the overflow indicator. 3. The overflow indicator is turned off after all lines conditioned by the indicator have been printed and top of form has been reached. 4. Regular detail cycles resume.
This file has an assigned overflow indicator, the overflow line has been passed, and the overflow routine has been fetched.	1. The overflow indicator has been turned on. 2. Total lines conditioned by the overflow indicator are printed. 3. Any header or detail lines conditioned by the overflow indicator are printed. 4. The line that fetched overflow is printed.† 5. Remaining detail and total lines are printed.
†The overflow indicator is turned off after top of form has been requested.	

Columns 17 and 18 allow up to three lines of spacing, before and/or after a print line, relative to adjacent lines. Columns 19 through 22 allow absolute spacing to a particular line on a page, before and/or after a line is printed. If both spacing and skipping are specified on the same line, the skip always precedes the space. Thus, if columns 17 through 22 all contain entries, spacing, skipping, and printing is executed in the following order:

1. Skip before
2. Space before
3. The line is printed
4. Skip after
5. Space after

Different spacing and skipping may be specified for lines in an OR relationship. When columns 17 through 22 of an OR line are blank, spacing and skipping are performed according to the entries on the line preceding the OR line.

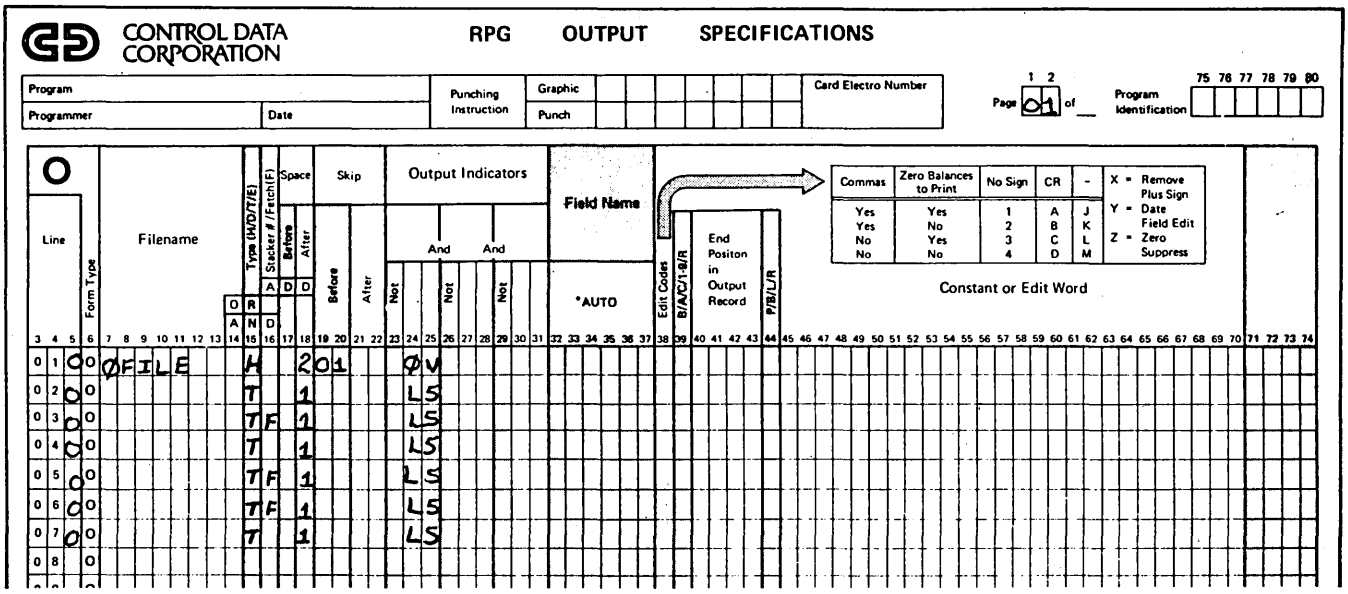


Figure 10-2. Specifications of Fetch Overflow Feature

SPACE FORMS CONTROL

Columns	Values	Meanings
17, 18	Blank	a. This is not a printer or console file, or b. Default to single space after line is printed.
	0	No spacing
	1	Single spacing
	2	Double spacing
	3	Triple spacing

Column 17 specifies the spacing before a line is printed, and column 18 specifies the spacing after a line is printed. If the destination of a space operation is a line in the overflow area, the overflow indicator, if assigned, turns on and remains on until the top of a new page is reached.

Because of the mechanism of a console device, the default of a single space after printing is actually a carriage return that occurs before the line is printed. Thus, a space specification of blank, zero, or one always yields a single space before printing at the console device.

SKIP FORMS CONTROL

Columns	Values	Meanings
19-20 21-22	Blanks	a. This is not a printer or console file, or b. Skipping is not requested.
19-20 21-22	01-99	Lines 1-99
	AO-A9	Lines 100-109
	BO-B2	Lines 110-112

Columns 19 and 20 specify the skipping before and columns 21 and 22 the skipping after a line is printed. The leading zero must be supplied for lines 1 through 9.

The codes designate the forms location of the next line to be printed. When the skip is to a lower line number, the forms are advanced. No action occurs when the forms are already positioned at the specified line.

If the destination of a skip is beyond the overflow line but before top of form, an overflow indicator assigned to the file is set on and remains on until forms advance occurs.

The destination line of a skip operation must not be beyond the form length defined on the Line Counter sheet.

The skip feature is useful in moving form stock more than the number of lines possible through spacing or when the number of lines to the destination line varies. The skip function is also useful for positioning subtotal, total, and heading lines, and is the most common way to reach a top of form.

OUTPUT INDICATORS

Columns	Values
23-31	Any RPG II indicators

Output indicator entries are used to condition output of entire records and/or fields of records. Any of the indicators associated with RPG II may be specified. Numerical indicators (01 through 99), control level indicators (L1 through L9), halt indicators (H1 through H9), and external indicators (U1 through U8) require previous specification in the RPG II program. When an overflow indicator (OA through OG or OV) is used to condition a record and/or field, that indicator must have been previously assigned to the same file in the file description specifications.

An indicator that conditions a record is entered on the pertinent record description line; similarly, an indicator that conditions a field is entered on the pertinent field description line.

The indicator codes are entered in columns 24 and 25, 27 and 28, and 30 and 31; the codes in a single line are in an AND relationship. If output is to occur when the indicator is off, enter an N in the preceding column (column 23, 26, or 29). It is a poor programming practice to specify only negative indicators on a line, but if a heading or detail record is conditioned in such a manner, the record is output at the beginning of the program cycle, when 1P lines are written.

Overflow indicators may not be used to condition exception records (E in column 15). Overflow indicators may condition fields of exception records, however.

Records conditioned by overflow indicators are produced only when overflow has been sensed for the file. An overflow indicator may have an OR relationship with the 1P indicator if identical heading records go on every page of the file. The 1P indicator, however, may be specified only for heading and detail records.

When a control level indicator (L1 through L9) conditions a total record without an overflow indicator, the record is written only after the last record in the control group has been processed. A detail record conditioned by a control level indicator but not an overflow indicator is produced only after the first record of the new control group has been processed. When columns 23 through 31 contain both a control level indicator and an overflow indicator, the record is produced only when overflow has been sensed and the control break has occurred.

All records conditioned by the last record indicator (LR) are produced last.

NOTE

Only one record is written to an update or combined file per cycle. Unpredictable results may occur if multiple records are designated for an update or combined file in a single cycle.

FIELD NAME

Columns	Values
32-37	a. Any previously defined field name
	b. The special words PAGE, PAGE1, PAGE2, *PLACE, UDATE, UDAY, UMONTH, or UYEAR
	c. The name of a table, array, or array element

Entries in columns 32 through 37 identify each field to be included in the output record or print line last specified. The columns are left blank on a record description line or if this line specifies in columns 45 through 70 a constant to be output. When an entry is made in columns 32 through 37, columns 7 through 22 of the same line must be blank.

Entries in these columns may be listed in any order on the sheet, as the order in which they appear in the output record is determined by entries in columns 40 through 43. They are usually listed sequentially, however, for easy reference. If later fields overlap the first fields, overlapped data from the first fields is lost. Entries in these columns are left-justified.

FIELD NAMES

A specified field, table, array, or array element name must be previously defined as a field of an input record in columns 53 through 58 of the input specifications or as a result field of an operation in columns 43 through 48 of the calculation specifications.

According to the internal representation of numeric data, a negative number shows a letter as the rightmost digit unless the field is edited (see Edit Codes).

PAGE, PAGE1, PAGE2

RPG II automatically numbers print file pages if PAGE, PAGE1, or PAGE2 is entered in columns 32 through 37. The three page counters, PAGE, PAGE1, and PAGE2, are provided for use with three different output files. The same name should not be used with two different output files.

When a page counter is named in these columns without being defined elsewhere in the program, the page number is taken from a four-digit numeric counter with no decimal positions; RPG II increments the counter by one before using it on each page.

On the other hand, a page field may be defined in the input or calculation specifications and be up to 15 positions long. Zero decimal positions are still required for a program-defined page counter.

When a page counter is output, leading zeros are suppressed automatically, and a sign is not shown with the low-order digit unless an edit word or edit code is specified for the counter. Page numbering starts with page one unless otherwise specified, and RPG II automatically increments a page counter by one every time the counter is output; this is true for page counters that are defined internally and page counters that are defined in the program.

A page counter may be reset to zero during program execution in several ways. One way is to specify Blank

After (column 39) in the same line in which the page counter is referenced. Another way is to use an output indicator to condition the page counter field. An output indicator does not truly condition output of the page counter, as a counter is output any time its associated record is output. When the output indicator is on, however, the specified page counter is reset to zero and incremented by one before the record is printed. Other ways to change the value of a page counter during program execution are to use the Z-ADD and MOVE operation codes in the calculation specifications with the desired reset value entered as a numeric literal in Factor 2 and the appropriate page counter in the Result Field.

*PLACE

*PLACE is a special RPG II word allowing fields to occupy more than one position in an output record without being respecified. When RPG II encounters *PLACE, all fields previously specified for the record are repeated in the same relative positions, ending at the character position named in columns 40 through 43 of the *PLACE line. A set of fields may be produced in a single record more than twice by the use of multiple *PLACE lines. An example of using the *PLACE feature is shown in figure 10-3. Either method of coding produces the output record shown in figure 10-4. When several fields are being repeated, the *PLACE feature is a very convenient device for saving lines of coding.

These rules should be followed when using *PLACE:

- *PLACE is entered on the line immediately following the last field specification to be repeated.
- *PLACE entries may be made on successive lines
- An *PLACE specification causes all prior field specifications for the record to be repeated.
- The end position specified for *PLACE should be at least twice the previously named end position for the record (not including previous *PLACE specifications), but not greater than 256. If this is not done, overlapping of fields results, with the *PLACE fields overlapping prior characters. The specified end position must not be lower than the preceding end position specification.
- The starting (leftmost) character position for an *PLACE move is always position one of the record.

CONTROL DATA CORPORATION RPG OUTPUT SPECIFICATIONS

Line	Filename	Type (H/D/T/E)	Stacker # / Fetch(F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And Not	Field Name	Edit Codes	End Position in Output Record	Constant or Edit Word
020	* Without *PLACE Feature											
040	PRINT	D		1				01				
050									FLDA		6	
060									FLDB		12	
070									FLDA		18	
080									FLDB		24	
090									FLDA		30	
100									FLDB		36	
130	* With *PLACE Feature											
150	PRINT	D		1				01				
160									FLDA		6	
170									FLDB		12	
180									*PLACE		24	
190									*PLACE		36	

Figure 10-3. Example of Coding with the *PLACE Feature

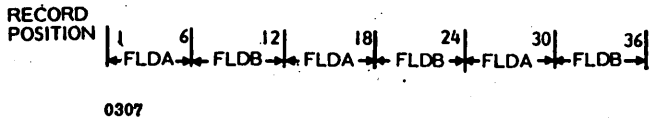


Figure 10-4. Example of *PLACE Output

DATE FIELD

The special words UDATE, UMONTH, UDAY, and UYEAR refer to the runtime-supplied current date. A reference to UDATE yields a six-character numeric date field in one of the formats:

- Domestic: mmddyy
- Foreign (including United Kingdom): ddmmyy

The format desired is specified in column 21 of form H, Control Card Specifications. When a UDATE field is edited, the eight-character result is one of the following:

- Domestic: mm/dd/yy
- United Kingdom: dd/mm/yy
- Foreign: dd.mm.yy

UMONTH refers to the two-character mm, UDAY to dd, and UYEAR to yy.

None of these special words may be specified in the result field of a calculation operation. Any of the words, however, may be used as Factor 1 or Factor 2 of a calculation operation.

EDIT CODES

Column	Values	Meanings
38	Blank	a. Alphanumeric field, or b. Field editing not desired.
	1-4, A-D, J-M, X-Z	See table 10-2.

An entry is made in column 38 to edit numeric fields in the following ways:

- To suppress leading zeros
- To omit signs from low-order digits of fields
- To punctuate fields without setting up edit words

An entry should not be made in column 38 for alphanumeric fields or when an edit word is entered in columns 45 through 70. Some edit codes allow the entry of edit code modifiers in columns 45 through 47, however.

TABLE 10-2. EDIT CODES

Category	Code	Meaning
One	X	Suppress positive sign. Insert two blanks in front of each array element.
	Y	Edit date field according to inverted print specification (column 21 of form H): Blank - mm/dd/yy D - dd/mm/yy I - dd.mm.yy J - dd.mm.yy Suppress first digit of date if zero.
	Z	Suppress zero balance and suppress sign.
Two	1	Print with commas, print zero balance, suppress sign.
	2	Print with commas, suppress zero balance, suppress sign.
	3	Print without commas, print zero balance, suppress sign.
	4	Print without commas, suppress zero balance, suppress sign.
	A	Print with commas, print zero balance, print sign as CR.
	B	Print with commas, suppress zero balance, print sign as CR.
	C	Print without commas, print zero balance, print sign as CR.
	D	Print without commas, suppress zero balance, print sign as CR.
	J	Print with commas, print zero balance, print sign as -.
	K	Print with commas, suppress zero balance, print sign as -.
	L	Print without commas, print zero balance, print sign as -.
	M	Print without commas, suppress zero balance, print sign as -.

There are two categories of edit codes. The first category (consisting of codes X, Y, and Z) generates punctuation only for a date field with code Y and prohibits entries in columns 45 through 47. All of the other edit codes belong to the second category, causing punctuation of the output field with decimal points/commas and the affixing of negative

balance indicators, and allowing entries in columns 45 through 47. The function of each edit code is listed in table 10-2.

NOTE

A positive sign is suppressed when any edit code is specified.

When an edit code is used to punctuate a whole array, two high-order blanks are provided with each array element.

An edit code can be specified only for unpacked numeric data. When an edit code is not specified for a negative field, a non-numeric character is produced in the low-order position of the field.

Edit codes are used in conjunction with entries in column 21 (Inverted Print) of the Control Card Specifications sheet. All edit codes suppress leading zeros, except when a J is specified in column 21 of form H. In that one case, a zero is supplied to the left of the decimal comma for zero and fractional balances.

Characters added to a field through edit codes should be taken into consideration in determining the ending record position of an output field. The effect of each edit code on numeric data is shown in table 10-3. (Data in the figure is six digits long and is aligned in the low order position to show edited field expansion.) Another illustration of the effects of entries in column 21 of form H is found in table 4-1.

The use of edit code modifiers with edit codes of the second category was mentioned above. Specifically, there are two allowable edit code modifiers:

- To replace leading zeros by asterisks, enter '*' in columns 45 through 47 of the line containing the edit code. This type of replacement is termed an *asterisk fill.*
- To generate a dollar sign immediately before the first digit of the field, enter '$' in columns 45 through 47 of the line containing the edit code. This type of editing is referred to as a *floating dollar sign.*

TABLE 10-3. EDIT CODE USAGE

Edit Code	Positive Number - Two Decimal Positions	Positive Number - No Decimal Positions	Negative Number - Two Decimal Positions	Negative Number - No Decimal Positions	Zero Balance - Two Decimal Positions			Zero Balance - No Decimal Positions
					Entry in Column 21 of Form H			
					blank, D	I	J	
Blank	123456	123456	00012L	00012L	000000	000000	000000	000000
1	1,234.56	123,456	1.23	123	.00	,00	0,00	0
2	1,234.56	123,456	1.23	123				
3	1234.56	123456	1.23	123	.00	,00	0,00	0
4	1234.56	123456	1.23	123				
A	1,234.56bb	123,456bb	1.23CR	123CR	.00	,00	0,00	0
B	1,234.56bb	123,456bb	1.23CR	123CR				
C	1234.56bb	123456bb	1.23CR	123CR	.00	,00	0,00	0
D	1234.56bb	123456bb	1.23CR	123CR				
J	1,234.56b	123,456b	1.23-	123-	.00	,00	0,00	0
K	1,234.56b	123,456b	1.23-	123-				
L	1234.56b	123456b	1.23-	123-	.00	,00	0,00	0
M	1234.56b	123456b	1.23-	123-				
X	123456	123456	00012L	00012L	000000	000000	000000	000000
Y	12/34/56	12/34/56	0/01/23	0/01/23	0/00/00	0.00.00	0.00.00	0/00/00
Z	123456	123456	123	123				

It is also possible to have a dollar sign precede the asterisk fill. In this case the field has a *fixed dollar sign*. A fixed dollar sign is requested in this manner:

- Encode a dollar sign constant in the record position immediately preceding the edited field.
- Enter '*' in columns 45 through 47 of the line containing the field name and the edit code.

BLANK AFTER

Column	Values	Meanings
39	Blank	a. Record description line, or b. Do not change this output field.
	B	Reset this field to zeros or blanks after output.

The entry of B in column 39 causes the source field to be reset to blanks (if alphanumeric) or zeros (if numeric) after the field has been placed in the output record. Blank after is a convenient means for resetting control break accumulators after they are output at total time.

Blank after cannot be specified for a date field name (for example, UDATE), a look-ahead field, or a constant.

When blank after is requested for a table name, both the holding area item and the equivalent table item are reset.

Care must be taken in specifying blank after for a field that is produced in more than one record. The blank after should be designated in the last record in which the field is output to avoid premature clearing of the field.

END POSITION IN OUTPUT RECORD

Columns	Values	Meanings
40-43	Blank	Record description line
	1-9999	Low-order character position of this field in the output record

The entry in columns 40 through 43 specifies the rightmost or least significant position of the designated field in the output record. The entry is numeric, right-justified, and leading zeros may be omitted. The end position must not exceed the record length.

If punctuation or other characters are added to the field through editing, the effective field length is increased. Be sure to allow sufficient room in the record for the source field and any additional characters; output results are unpredictable if edited fields overlap.

If the field name is PAGE, PAGE1, or PAGE2, the entry in columns 40 through 43 indicate the low-order position of the page number in the output record.

When the field name is *PLACE, columns 40 through 43 indicate the rightmost character position to which the move occurs.

PACKED OR BINARY FIELD

Column	Values	Meanings
44	Blank	a. Record description line, or b. Unpacked numeric or alphanumeric format field
	P	Packed decimal format field
	B	Binary format field

Column 44 indicates whether a numeric field is to be output in packed decimal or binary format. Packed and binary fields should not be printed; however, it is quite appropriate to produce a packed or binary field on disk, tape, or cards in order to conserve space on the external medium. When column 44 is blank, the data is output in unpacked format.

When format conversion is requested via column 44, four or fewer bytes are converted to two binary data bytes; five to nine unpacked data bytes become four binary bytes. Further discussion of unpacked, packed, and binary formats can be found in section 6.

CONSTANT OR EDIT WORD

Columns 45 through 70 are used for three purposes:

1. To define a constant to be placed in the output record
2. To indicate a modification to the edit function coded in column 38
3. To specify an edit word that formats the output field.

CONSTANTS

An alphanumeric constant may be defined by encoding an apostrophe in column 45 followed by up to 24 alphanumeric characters (including blanks) and terminated by another apostrophe. An apostrophe within the constant is encoded by entering two consecutive apostrophes. For example, the constant DON'T is encoded 'DON' 'T' in columns 45 through 52. Since a constant is actually a source field for the output record, columns 32 through 37 (field name) must be left blank in the same line.

When a constant exceeding 24 characters is required, subsets of the constant are entered on successive lines, as if they were independent. For each line of the constant, an appropriate entry must be made in columns 40 through 43 (End Position in Output Record).

EDIT CODE MODIFIERS

As described above, the two edit code modifiers, * and $, are entered with enclosing apostrophes in columns 45 through 47 for use in conjunction with the Edit Code in column 38. An entry of '*' shows that all leading zeros of the numeric field are replaced by the asterisk character (asterisk fill). '$' indicates that a dollar sign is placed just preceding the high-order nonzero digit of the numeric output field (floating dollar sign).

Asterisk fill and the floating dollar sign are commonly used for check protection.

EDIT WORDS

The edit codes in column 38 plus dollar signs and asterisks provide normal editing of typical numeric fields. An edit word, however, allows more unusual or complex editing of a numeric field. An edit word is like a mask or framework that allows insertion of characters from the source field named in columns 32 through 37. Through an edit word the operator may directly specify inclusion of decimal points, commas, zero suppression, negative balance indicators, dollar signs, asterisks, and even whole words between one digit and the next.

The following rules apply to using edit words:

- Edit words are used only with unpacked numeric source fields. Columns 32 through 37 name the field.
- Columns 38 (Edit Code) and 44 (Packed or Binary Field) must be blank on the line containing the edit word.
- Columns 40 through 43 (End Position in Output Record) must contain an entry.
- An edit word must be enclosed in apostrophes, with the leading apostrophe entered in column 45. Edit words are limited to 24 characters.
- Any printable character is valid in the edit word, but certain characters in certain positions have the meanings listed in the lines below.
- A blank in the edit word causes the digit from the corresponding position of the source field named in columns 32 through 37 to be placed in the output field. Thus, the leftmost blank receives the leftmost character, and so on. The number of replaceable characters in the edit word must be equal to the length of the field to be edited.
- An ampersand (&) in the edit word generates a blank in that position of the output field.
- A zero in the edit word causes leading zero suppression. The position containing the zero is the last (rightmost) digit position subject to suppression.
- An asterisk in the edit word causes zero suppression and replacement of suppressed zeros by asterisks. The position containing the asterisk is the last (rightmost) digit position subject to suppression and replacement.
- The first (highest position) zero or asterisk in the edit word stops zero suppression. Subsequent zeros and asterisks are treated as constant symbols and are interspersed where they occur with respect to the source field.
- A floating dollar sign is provided if the dollar sign is encoded immediately preceding the zero suppression code (zero or asterisk).
- A fixed dollar sign is provided if the dollar sign is encoded in column 46; i.e., the first character in the body of the edit word.
- Any editing characters other than a fixed dollar sign are replaced by blanks (or asterisks if asterisk fill is used) if they appear to the left of the most significant digit of the output field. If zero suppression is not specified by an asterisk or zero, all constant editing characters in the body of the edit word to the left of the most significant digit are suppressed by blanks.
- The sign of a negative field can be shown by a minus sign or the characters CR. These edit characters are replaced by blanks in the output field if the source field is positive.
- Asterisks or other characters in the edit word to the right of the characters CR or a minus sign always appear in the output field.
- The source field length must exactly match the number of replaceable characters in the edit word, with these exceptions:

  - An extra space must be left in the edit word for the floating dollar sign. This ensures an output record position for the dollar sign in the event that the output field is full.

  - An extra space may be left in the edit word if the first character in the edit word (column 46) is a zero. In this case, the source field is not zero suppressed, but all other specified editing is performed.

- If a minus sign or the characters CR are used to denote a negative field, the character position(s) used by these designations must be taken into consideration when entering the end position of the field in the field description line.

Examples of edit words and the output they produce are given in table 10-4.

OUTPUT SPECIFICATIONS EXAMPLES

Various examples of Output Specifications examples are shown in figure 10-5. The examples are discussed below.

LINES 010 THROUGH 190

Two heading records are to be printed if indicator 1P or OA is on; that is, the two records are to be printed on the first page (1P) and all overflow pages (OA). There is a skip to the sixth line of a new page before the heading records are

TABLE 10-4. EDIT WORD EXAMPLE

EXAMPLE NO.	45	50	60	70	SOURCE DATA	OUTPUT DATA
1	1	$ &	0 . & - & T Ø T A L '		1234567890-	$12,345,678.90 - TOTAL
2	1		. - '		0000000123-	1.23-
3					0000135678	0000135678
4					0000135678	0000135678
5					0000135678	0000135678
6	1		0 . & C R '		0000000008	.08 CR
7	1	0	- & A V A I L A B L E '		0000000000	0 AVAILABLE
8	1	$ 0	. - X '		0000000003	$0.03 *
9	1	$	0 . C R X X '		0098765432-	$987,654,32CR**
10	1	$	0 . '		0000000004	$ .04
11	1	*	. & - '		0000483726	*****4,837.26
12	1		'		0000000000	
13	1		'		0000432123	432123
14	1		'		0000432101-	432101
15	1		& C R & S U B T Ø T '		0000456789+	456789 SUB TOT
16	1		& C R & S U B T Ø T '		0000456789-	456789 CR SUB TOT
17	1		& - & & S U B T Ø T '		0000456789-	456789 - SUB TOT
18	1		& T Ø T & C R '		0000456789	456789
19	1		& T Ø T & C R '		0000456789-	456789 TOT CR
20	1		& & G A I N '		0000456789	456789 GAIN
21	1	$ 0	- & T Ø T A L '		0000456789	.0000456789 TOTAL
22	1	$	& - & T Ø T A L '		0000456789-	456789 - TOTAL

TABLE 10-4. EDIT WORD EXAMPLE (Contd)

EXAMPLE NO.	45	50	60	70	SOURCE DATA	OUTPUT DATA
23	/$		&-&TØTAL'		0000456789+	$ 456789 TOTAL
24	/	#0	&CR'		0000456789-	$456789 CR
25	/	#0	&CR'		4567890123-	$4567890123 CR
26	/$	0	&CR&TØTAL'		0000000000	$ 00 TOTAL
27	/		0'		0000456789-	456789
28	/0		'		0000456789	000456789
29	/*		'		0000456789-	*000456789
30	/*		'		3456789012	3456789012
31	/		*'		0000456789+	****456789
32	/	#0.	&TOTAL'		00000000 5	$0.05 TOTAL
33	/		.&CR&-TØTAL'		0000456789	4,567.89 -TOTAL
34	/		.&CR&&TØTAL'		0000456789-	4,567.89 CR TOTAL
35	/	*.	*CR** '		0000456789	*****4,567.89 ***
36	/	#0	.-SØH'		0000002345	$,023.45 SOH
37	/$&	0.	CR'		1234567890-	$ 12,345,678.90CR
38	/		-BAL&FWRD'		1234567890-	1,234,567,890-BAL FWRD
39	/	0	-BAL&FWRD'		0000000000	0 BAL FWRD
40	/	#0.	'		0000000006	$.06
41	/	$0.	-'		9876543210-	$98,765,432.10-
42	/	#0.	CR'		0001234567-	$12,345.67CR
43	/	DØLLARS	CENTS'		0000456789	4,567DOLLARS78CENTS
44	/	DØLLARS	CENTS'		000000	CENTS

TABLE 10-4. EDIT WORD EXAMPLE (Contd)

EXAMPLE NO.	45	50	60	70	SOURCE DATA	OUTPUT DATA
45	1	1, 10	DØLLARS CENTS&CR'		000000	ODOLLARS00
46	1	0 -	-		12345678	12-34-5678
47	1	0 HRS. MINS.&ØØ''	CLØCK'		0042	0HRS.42MINS. O'CLOCK

printed. Notice that the skip/space specifications are not repeated on the OR lines.

Lines 030 through 060 show what is actually printed in the first heading record: the constant SALES REPORT, the date, the constant PAGE, and the page number. A Z edit code is used to zero-suppress the page number. A Y edit code inserts slash punctuation in the date. The column headings SALE and AMOUNT are printed in the second heading record (lines 070 through 100). The absence of a skip specification indicates that the second record is to be printed on the same page as the first heading record.

Only one detail record format is shown. The record is produced whenever indicator 10 is on. However, the record contains an ITEM entry only when indicator L1 is also on (ordinarily when a level 1 control break occurs). When L1 is not on, the field AMOUNT is the only field in the record.

Two types of total records are shown. The record described in lines 140 through 160 is printed at total time when indicator L1 is on. The record consists of a subtotal of the amounts and an appended asterisk. The subtotal is reset to zero after the record is printed.

The grand total record described in lines 170 through 190 is produced only once: in the last total output time.

Note that all heading records for the file are described first, followed by all detail records, followed by all total records.

LINES 200 AND 210

A record is to be added to the disk file, DSKFIL, if indicator 77 is on at detail time. The record consists of a single 80-character alphanumeric or unpacked decimal numeric field named NEWREC.

CONTROL DATA CORPORATION — RPG OUTPUT SPECIFICATIONS

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # / Fetch Fl	Space Before	Space After	Skip Before	Skip After	Output Indicators And Not	Output Indicators And Not	Output Indicators And Not	Field Name	Edit Codes	End Position in Output Record	P/B/L/R	Constant or Edit Word
010	O	PRTFIL	H		2		06		1P					12		'SALES REPORT'
020	O		ØR						ØA							
030	O											UDATE	Y	21		
040	O													27		'PAGE'
050	O											PAGE	Z	32		
060	O		H		2				1P							
070	O		ØR						ØA							
080	O													4		'SALE'
090	O													25		'AMOUNT'
100	O		D	1					10							
110	O									L1		ITEM		4		
120	O											AMT	3	24		
130	O		T	21					L1							
140	O											SUBTØT	3B	24		
150	O													27		'*'
160	O		T	2					LR							
170	O											TØTAL	3	24		
180	O													28		'**'
190	O															
200	O	DSKFIL	D	ADD					77							
	O											NEWREC		80		
	O															

Figure 10-5. Output Specifications Examples

Section 4 references alternate collating sequences and file translation tables. File translation applies to input, output, update, and combined files where the external character set (used on the input/output device) differs from the internal character set (used inside the computer). With file translation, RPG II translates the input characters before using the data and the output characters before they are written out.

An alternate collating sequence may be defined to vary the collating hierarchy of the internal characters on an external medium.

Specifications of file translation tables and alternate collating sequences are submitted to the RPG II compiler with the source program. Preprinted forms are not provided for file translation tables and alternate collating sequences. However, this section describes these specifications as other RPG II input forms.

Proper placement of alternate collating sequence and/or file translation specifications in the RPG II source program is shown in figures 2-1 and 12-3.

FILE TRANSLATION TABLES

The internal character set used by this RPG II system is listed in appendix D. If a file referenced in a source program is produced by or will be used by another computer system, character translation of the entire file may be called for.

Another application of file translation occurs when, for security reasons, certain information in a file needs to be encoded into an unidentifiable character set on the external medium. As an example, a program may process an input file containing confidential wholesale prices paired with retail prices. File translation could apply to the wholesale price field so that only selected store personnel would be able to identify the store's price markup. Substituting a code word containing 10 distinct alphabetic characters (for example, PLAYGROUND) for the digits of wholesale prices would produce the desired encoding. Using the code word PLAYGROUND, the letter P represents the number zero, L represents one, and so on. A wholesale price of APP.UN in the input file would translate to 200.78, that is, two hundred dollars and seventy-eight cents. Typically, the code word is a combination of letters that is easily remembered by store personnel. Ten distinct letters are required to represent the ten digits 0 through 9. Hexadecimal equivalents of each letter in the word PLAYGROUND are listed along with the hexadecimal equivalents of the numbers 0 through 9 in table 11-1.

To indicate that there are files to be translated, enter an F in column 43 of the Control Card Specifications sheet. File translation table records must also be used to specify the exact nature of the translation. The format of each file translation record is as follows.

TABLE 11-1. PLAYGROUND EQUIVALENTS

Letter of Code Word	Hexadecimal Equivalent	Number	Hexadecimal Equivalent
P	D7	0	F0
L	D3	1	F1
A	C1	2	F2
Y	E8	3	F3
G	C7	4	F4
R	D9	5	F5
O	D6	6	F6
U	E4	7	F7
N	D5	8	F8
D	C4	9	F9

NOTE: Translation characters are specified in internal code (table D-5).

POSITIONS 1-8

The characters *FILESbb indicate that all input, output, update, and combined files referenced in the program are to undergo translation. (For update and combined files, both the input and output processing of the files include translation.)

A left-justified filename indicates that this individual input, output, update, or combined file is to be translated (in both the input and output phases, if this is an update or combined file). More than one individual file may be translated when translation records are entered for each filename.

POSITIONS 9-10

Enter the hexadecimal equivalent of the external character. This is the internal code corresponding to the character on the external medium.

POSITIONS 11-12

Enter the hexadecimal equivalent of the internal character. This is the form the character is to have during internal processing.

POSITIONS 13-16, 17-20, 21-24

Each set of four positions up to the end of the record specifies successive pairs of two-digit hexadecimal representations – first external, then internal.

The rules for including file translation tables with the source program are as follows:

- The file translation records immediately follow the RPG II source program specifications lines. The first translation record is preceded by a single record with the characters **␣ in positions 1 through 3. Other positions in the **␣ record may be used for comments.
- The order of characters in the translation records is not significant except that the pairing of character representations is mandatory.
- All records for a single file must have identical entries in columns 1 through 8.
- All translation table records for one file must be kept together.
- A file translation record is terminated by a blank character or end-of-record.

The file translation record for using PLAYGROUND as the code name for all files referenced in the program would be:

Column	Entry
1-8	*FILES␣␣
9-12	D7F0
13-16	D3F1
17-20	C1F2
21-24	E8F3
25-28	C7F4
29-32	D9F5
33-36	D6F6
37-40	E4F7
41-44	D5F8
45-48	C4F9

Only the letters in the example have been specified for translation. All other characters are processed in the normal manner. File translation is important to the example, as the characters in the code word PLAYGROUND would be nonsense in arithmetic calculations.

ALTERNATE COLLATING SEQUENCES

A collating sequence defines the relationship between one character and every other character used in a program. The normal (default) collating sequence of characters is defined in appendix D.

When column 26 of the Control Card Specifications sheet contains an S, alternate collating sequence records must be supplied to the compiler.

There are several ways to alter the normal collating sequence. One character may be inserted between two other characters; a character may be completely removed from the sequence; or the positions of two characters may be exchanged. A common application for an alternate collating sequence is to give the same hierarchy to zeros and blanks. An alteration of this kind causes an equal comparison for fields containing 00256 and ␣␣256.

NOTE

00256 should remain equal to 00256 and ␣␣256 should remain equal to ␣␣256.

In this application, only a single entry needs to be made in a single alternate collating sequence record.

Character insertion in a collating sequence, however, requires more than one entry. The number of entries required depends on the number of characters affected by the insertion. Thus, if the character zero is to occupy a collating hierarchy higher than nine, but all other hierarchies are to remain intact, ten entries are required in the collating sequence records.

The following sections describe the format for each alternate collating sequence record.

POSITIONS 1-8

Enter ALTSEQ␣␣ to indicate that this is an alternate collating sequence record.

POSITIONS 9-10

Enter the two-character hexadecimal representation of the first character whose hierarchy is being defined.

POSITIONS 11-12

Enter the two-character hexadecimal representation of the first character whose hierarchy is being used.

POSITIONS 13-16, 17-20, 21-24, . . .

These positions are used in the same way as positions 9 through 12. Enter as many pairs of characters as the record can contain.

As many records as are necessary to define the alternate collating sequence may be used; each record must conform to the above format.

The following are the rules for including an alternate collating sequence with the source program:

- Alternate collating sequence records follow the RPG II source program specifications lines and file translation records, if any. The first alternate collating sequence

record is immediately preceded by a single record with the characters **b in positions 1 through 3. Other positions in the **b record may be used for comments.

- The order of characters in a record is not significant, but the pairing of characters is mandatory.
- Positions 1 through 8 of continuation records contain ALTSEQbb.
- An alternate collating sequence record is terminated by a blank character or end of record.

The alternate collating sequence record for assigning the space character to the same hierarchy as zero is:

Column	Entry
1-8	ALTSEQbb
9-12	40F0 (Blank takes the 0's position.)

To cause zero to come after nine in the hierarchy, use:

Record 1

Column	Entry
1-8	ALTSEQbb
9-12	F1F0 (1 takes 0's position)
13-16	F2-F1 (2 takes 1's position)
17-20	F3F2 (3 takes 2's position)
21-24	F4F3 (4 takes 3's position)
25-28	F5F4 (5 takes 4's position)

Record 2

Column	Entry
1-8	ALTSEQbb
9-12	F6F5 (6 takes 5's position)
13-16	F7F6 (7 takes 6's position)
17-20	F8F7 (8 takes 7's position)
21-24	F9F8 (9 takes 8's position)
25-28	F0F9 (0 takes 9's position)

Two records were used to illustrate using more than one record. It is valid to enter all pairs of characters on one record. When the hierarchy of a nonprintable character is being used by a printable character, it is not necessary to assign another hierarchy to the nonprintable character. For example, if a character were being inserted so that the character 9 assumed the hierarchy of the (nonprintable) character represented by FA, it would not be necessary to include the pair FAFB, and so on.

It is valid to include the two cases described above in the same compilation.

Alternate collating sequences apply to matching fields, sequence checking of files, and the COMP operation code. An alternate collating sequence is ignored in numeric comparisons, look-up operations, and control levels.

USE OF TABLES AND ARRAYS IN RPG II PROGRAMMING

The specification of tables and arrays was described in section 6. This section describes the formation and usage of tables and arrays.

Tables are systematically arranged sets of information that are more limited in scope than files. RPG II treats tables in much the same way that clerks do in manual systems. A document (or record) provides a piece of known information such as an employee's identification number. The clerk (or RPG II) uses that figure to obtain another piece of information from a table by search or lookup. Employee number may be used to look up the corresponding tax withholding rate in a table of rates listed by employee number.

Tables are also used simply to provide verification of a piece of data. For example, a table of employee numbers may be used to validate an input employee number.

Table elements may be extracted and used in calculation operations. RPG II allows table updating, where entries may be changed during processing. For example, a new withholding rate may be substituted in the withholding rate table when an employee changes his W2 form.

Arrays are also systematically arranged sets of information that may be searched for a uniquely identified data item. Unlike tables, an array element may be referenced by position number (its array index). Another difference is that an unindexed reference to an array name is a reference to the entire set of data; a reference to a table name is a reference to the table item currently in the table's holding area. An array element or an entire array may be used as an operand in a calculation operation.

The advantages of using a table or array format instead of other file formats are that a table or array is compact and may be completely contained in computer memory during runtime for random accessing, and that table and array entries may be used repeatedly during the course of processing by an RPG II object program.

Table names begin with the characters TAB; array names do not. Both types of names follow the rules for naming fields. Each element in a single table or array has the same length, format (numeric or alphanumeric), and number of decimal positions.

Following is a review of some definitions related to tables and arrays.

- Compilation time table or array – A compilation time table or array is compiled with the source program and becomes a permanent part of the object program. This type of table or array can be permanently changed only through recompilation of the source program with the revised table or array, but the table or array may be output at end of job, if requested on form E, any time the object program is executed.
- Pre-execution time table or array – A pre-execution time table or array is loaded with the object program just prior to the actual execution of the RPG II object program; that is, before the first input file is read. A pre-execution time table or array may be modified by the object program and output at program termination.
- Execution time table or array – An execution time table or array is read or created during the input or calculations phase of an object program cycle but cannot be output at program termination by RPG II. (It can, however, be output by the object program.)
- Related tables and arrays – Related tables and arrays are tables and arrays that are used together. The items in each table or array are called corresponding entries; each item in the second table or array gives additional information about its corresponding item in the first table or array. Related tables and arrays may be specified separately or in alternating format. Where all items in one table or array must have the same characteristics (element length, etc.), related tables and arrays need not have identical characteristics, nor must they contain the same number of elements if they are specified separately. When a search of related tables is successful, corresponding items of the two tables are brought to the holding areas. A successful search of related indexed arrays causes the index field to point to the corresponding items in the related arrays.
- Short tables and arrays – A short table or array contains some null elements. For nonsequential (neither ascending or descending) tables and arrays, null elements equal zero if numeric, or blanks if alphanumeric. A short table or array may be created when all significant table or array items are not available when building the table or array, and the other significant items are to be added later. A short table or array must have at least one non-null element.
- Full table or array – Every element of a full table or array contains significant data.

An example of using related tables is illustrated in figure 12-1. A medical billing file (1) is sequenced by procedure code within record type within account number. There is a type 1 record for each patient and a variable number of type 2 records. The type 1 record shows a patient's name, address, and account number. Each type 2 record gives a treatment code, the date the treatment was administered, and the patient's account number. Treatment codes and associated treatment prices are maintained in related tables (2). The treatment code from the type 2 record is used as a search argument for treatment price from the price table (3). The prices extracted from that table are summed, and the patient is billed for the total. Two methods for specifying the related tables are shown in figure 12-2.

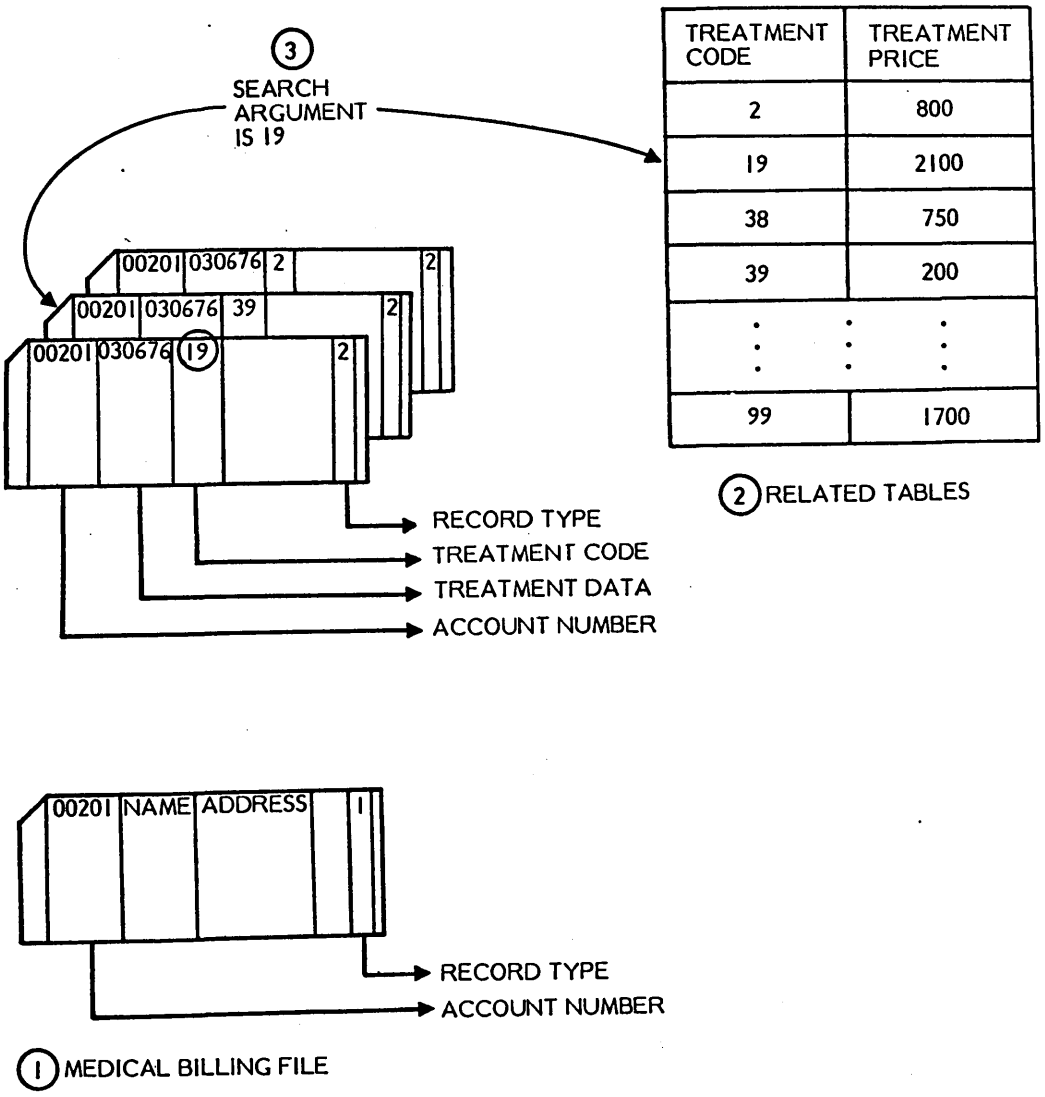


Figure 12-1. A Use for Related Tables

FORMING TABLES AND ARRAYS

Tables and arrays may be formed inside or outside the computer. Elements formed on the outside may be entered through any input device. Any computer program, including an RPG II program, may also form them, and they may be output to any sequential file medium. Tables and arrays have the same format and structure rules whether they are formed inside or outside the computer. The complete set of entries comprising a table or array is considered a sequentially organized file, although the entries need not be in sequence within a table or array.

A table or array file must have a unique name consistent with the file naming rules listed in section 1. The filename must not be confused with the table or array name described earlier in this section. Table and array files may be assigned to the same device and even have the same filename if they are read from the same device. Table and array files are read in the order the tables and arrays are specified on form E.

COMPILATION TIME

When tables and arrays are loaded at compilation time or pre-execution time, the entire table or array is loaded before the first input record is read by the object program.

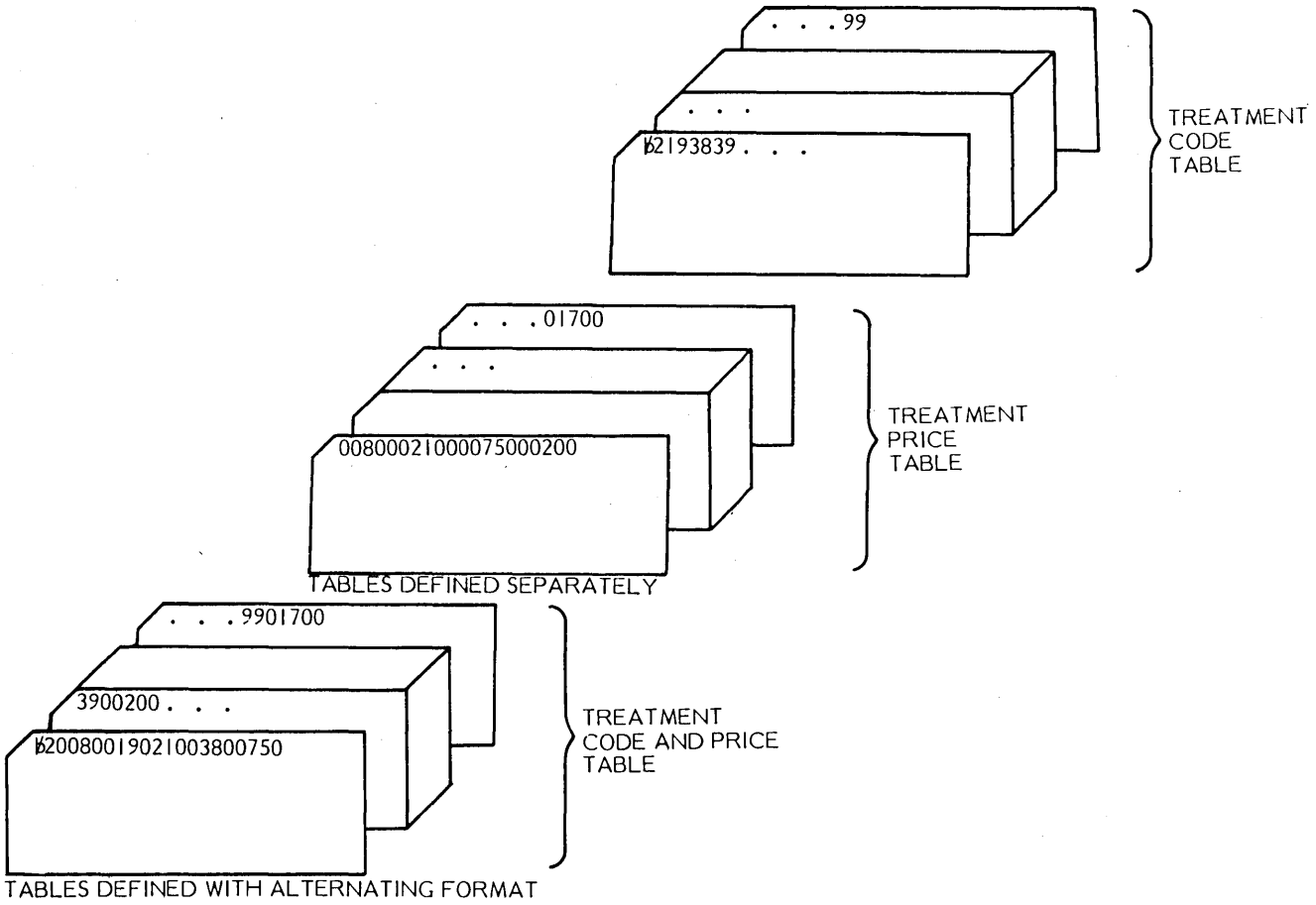
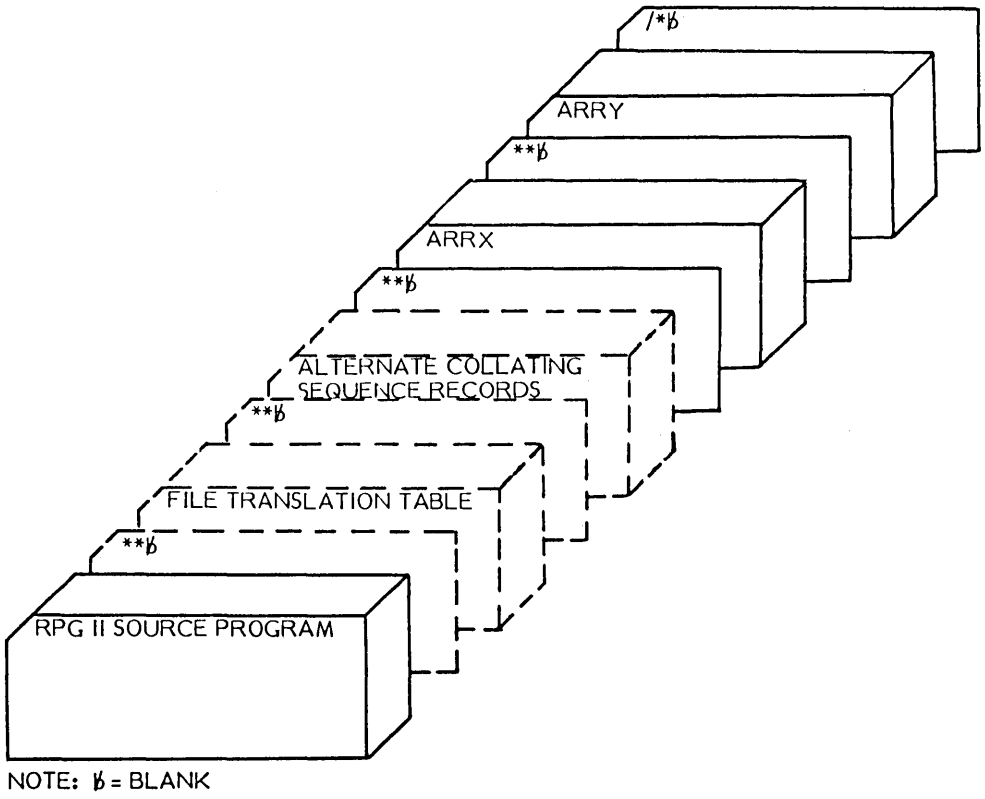


Figure 12-2. Two Methods for Defining Related Tables

Compilation time tables and arrays are compiled along with the source program, come from the same input stream as the source program, and, thus, do not have a From Filename column entry on the Extension Specifications sheet. Here are the rules for entering a compilation time table or array.

- The table or array must be on the same device that holds the source program and must follow the RPG II source program, file translation tables, and alternate collating sequence records, if any.
- Each distinct table or array must be preceded by a record having **b in positions 1, 2, and 3. As this record is a delimiter, the first element of a table or array record must not contain those characters in the first three positions.
- The last compilation time table or array is followed by a record reading /*b in positions 1, 2, and 3.
- The tables and arrays must be loaded in the same order as described on the Extension Specification sheet.
- Compilation time tables and arrays must have entries in columns 33 through 35 (Number of Entries Per Record) and must not have entries in columns 11 through 18 (From Filename) on the Extension Specification sheet.
- Numeric tables and arrays must be in unpacked decimal format.

The proper placement of compilation time tables and arrays in the compiler input stream is illustrated in figure 12-3. In this example, the user has a file translation table, alternate collating sequence records, and two of his own compilation time arrays, ARRX and ARRY. The file translation table and alternate collating sequence records are optional and because of that are illustrated with broken lines. If either a file translation table or an alternate collating sequence, but not both, is included, it must precede the user's tables and arrays. If both are included, the file translation table must precede the alternate collating sequence, as shown in the illustration.



0310

Figure 12-3. Placement of Compilation Time Tables and Arrays

PRE-EXECUTION TIME

Pre-execution tables and arrays are not part of the source program. They are used by the program like any other data file. They differ from execution time tables and arrays, however, because they are loaded by RPG II prior to execution of the object program.

Here are the rules for loading pre-execution time tables and arrays:

- A record containing /* in positions 1 and 2 must follow each pre-execution time table or array.
- Pre-execution time tables and arrays must be arranged in the same order they are described in the extension specifications.
- A pre-execution time table or array must have entries in columns 11 through 18 (From Filename) and 33 through 35 (Number of Entries Per Record) of the Extension Specifications, and may have entries in columns 43 and 55 (Packed or Binary Field), if appropriate.

EXECUTION TIME

Execution time table and array files are read by the object program and can only be described as any other input file on form I. The definition on form I specifies whether the file has a single record or many records, and whether or not the elements are contiguous in a record. If the unindexed array name appears as the field name on form I, then the array is contained in one record and the elements are contiguous. If the elements are not contiguous, each element must be specified on its own form I line, with the appropriate field positions specified.

Such a table or array may not be used until every element that is required for program execution is in core. Calculation and output operations, therefore, may have to be suspended for several program cycles until the object program has read all of the records from the file.

The extension and input specifications required to fill an execution time array from contiguous elements of a single input record are shown in figure 12-4. Using the same

CONTROL DATA CORPORATION

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

Printed in U.S.A.

Program | Programmer | Date | Punching Instruction | Graphic Punch | Card Electro Number | Page 01 of __ | Program Identification

Extension Specifications

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
01	E					ARY		8	6									
02	E																	

CONTROL DATA CORPORATION

RPG INPUT SPECIFICATIONS

Printed in U.S.A.

Program | Programmer | Date | Punching Instruction | Graphic Punch | Card Electro Number | Page 01 of __ | Program Identification

Line	Form Type	Filename	Sequence (OR / AND)	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank
01	I	ARFILE	AA			01																								
02	I																				1	48		ARY						
03	I																													

Figure 12-4. Input Specifications for Contiguous Array Elements

extension specifications, input specifications for array elements separated by one character are shown in figure 12-5. Finally, using the same extension specifications, input specifications for the array when the elements are separated by one-character index values for the array are shown in figure 12-6.

Execution time tables and arrays are not checked for sequence, but column 45 of form E (Sequence) must contain an A or D code if high or low LOKUP is specified for the table or array.

If the execution time table or array is in packed decimal or binary format, an appropriate entry must be specified in column 43 of form I. The length specified for tables and arrays in these formats reflects the packed or binary length of an element, in numbers of bytes. Columns 19 through 26 of form E must be blank for execution time tables and arrays; that is, the table or array cannot be output at end of job by RPG II.

An example of building an array during the calculations phase of execution is shown in figure 12-7. The array is to contain the square roots of the numeric values found in two input records of different types. The first input record has the character A in position 80. The second input record has any character except A, including blank, in position 80.

USING TABLES AND ARRAYS

This section further investigates how to use the specified tables and arrays.

CONTROL DATA CORPORATION

RPG INPUT SPECIFICATIONS

Printed in U.S.A.

Program | Programmer | Date | Punching Instruction | Graphic | Punch | Card Electro Number | Page 01 of __ | Program Identification

I

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank
01	O	ARFILE	AA			01																								
02	O																				1	6		ARY,1						
03	O																				8	13		ARY,2						
04	O																				15	20		ARY,3						
05	O																				22	27		ARY,4						
06	O																				29	34		ARY,5						
07	O																				36	41		ARY,6						
08	O																				43	48		ARY,7						
09	O																				50	55		ARY,8						
10																														

Figure 12-5. Input Specifications for Scattered Array Elements

CONTROL DATA CORPORATION

RPG INPUT SPECIFICATIONS

Printed in U.S.A.

Program | Programmer | Date | Punching Instruction | Graphic | Punch | Card Electro Number | Page 01 of __ | Program Identification

I

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank
01	O	ARFILE	AA			01																								
02	O																				1	1		0I1						
03	O																				2	7		ARY,I1						
04	O																				8	8		0I2						
05	O																				9	14		ARY,I2						
06	O																				15	15		0I3						
07	O																				16	21		ARY,I3						
08	O																				22	22		I4						
09	O																				23	28		ARY,I4						
10	O																				29	29		I5						
11	O																				30	35		ARY,I5						
12	O																				36	36		I6						
13	O																				37	42		ARY,I6						
14	O																				43	43		I7						
15	O																				44	49		ARY,I7						
16	O																				50	50		I8						
17	O																				51	56		ARY,I8						
18																														

Figure 12-6. Input Specifications for Array Elements and Indexes

CONTROL DATA CORPORATION

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

Printed in U.S.A.

Program | Programmer | Date | Punching Instruction | Graphic Punch | Card Electro Number | Page 01 of __ | Program Identification

Extension Specifications

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
01	E					ARY		11	7		0							
02	E																	

CONTROL DATA CORPORATION

RPG INPUT SPECIFICATIONS

Printed in U.S.A.

Program | Programmer | Date | Punching Instruction | Graphic Punch | Card Electro Number | Page 01 of __ | Program Identification

I

Line	Form Type	Filename	Sequence (O/A, R/N, D)	Number (1-N)	Option (O)	Record Identifying Indicator or *	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank
01	I	ARFILE	AA		01		80		CA																					
02	I																				1	7		CA						
03	I																				8	14		CB						
04	I																				15	21		CC						
05	I																				22	28		CD						
06	I																				29	35		CE						
07	I																				36	42		CF						
08	I		BB		C2		80	N	CA																					
09	I																				1	7		CG						
10	I																				8	14		CH						
11	I																				15	21		CI						
12	I																				22	28		CJ						
13	I																				29	35		CK						
14	I																													

Figure 12-7. Building an Execution Time Array Through Calculations (Sheet 1 of 2)

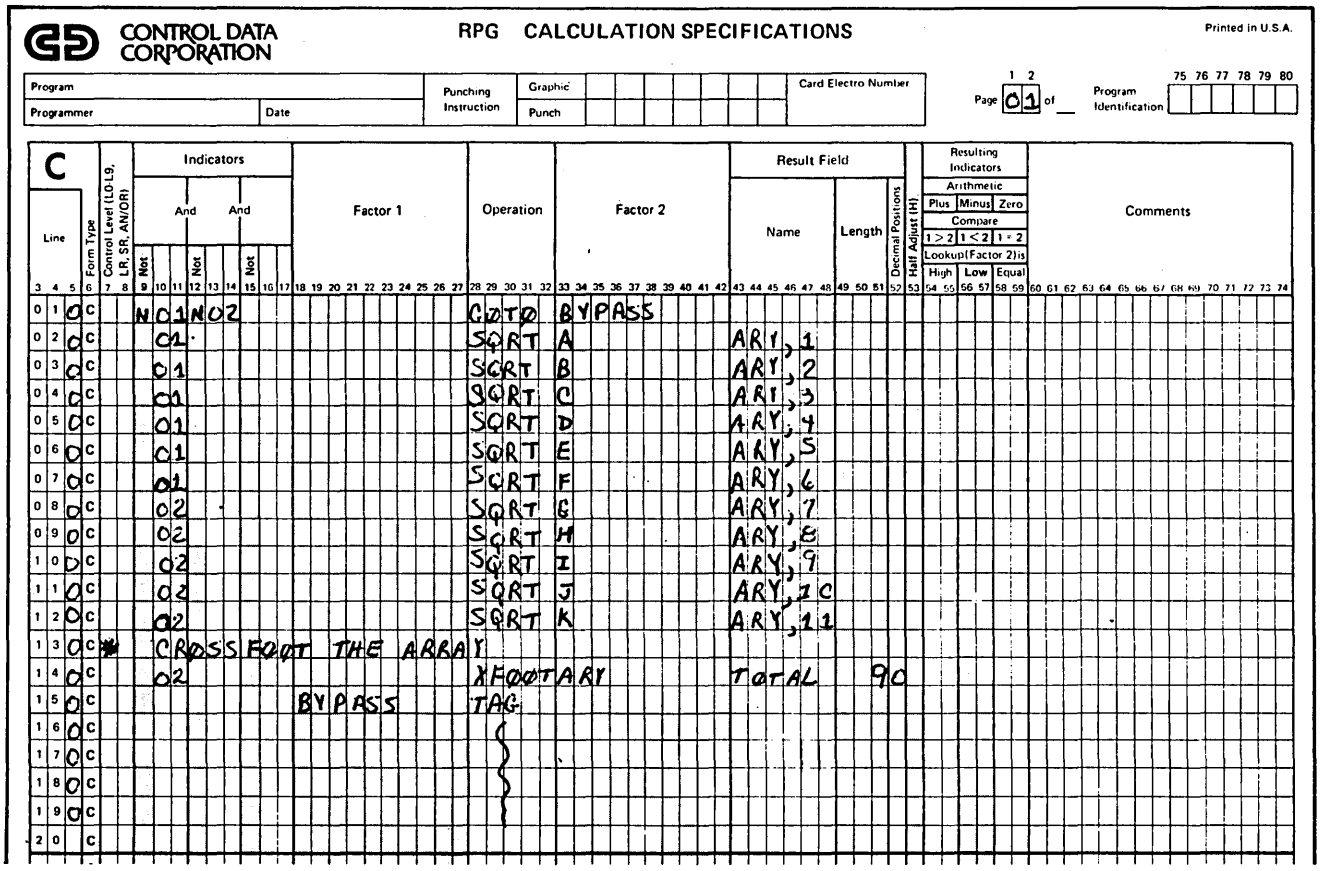


Figure 12-7. Building an Execution Time Array Through Calculations (Sheet 2 of 2)

SEARCHING TABLES AND ARRAYS

Tables and arrays may be searched using the LOKUP operation code described in section 9. The LOKUP operation is the key to a table, since it is this operation that brings a table item to the table's holding area for subsequent use in the object program.

When LOKUP is used to search an array, an array element satisfying the search is not brought to a holding area but, rather, the relative position of the element within the array is stored in a designated index field. The indexed array name may then be used in later calculation and output operations.

MODIFYING TABLE AND ARRAY ELEMENTS

When elements of a table or array are being deleted (replaced by zeros or blanks) or changed, or when elements are being added to a short table or array, the number of entries in the table or array is not altered; this number is fixed on the Extension Specifications sheet. Rather, the number of meaningful entries in a table or array may be altered. By the selection of proper search arguments and criteria, the null entries in a short table or array may be ignored during processing.

The simplest way to add elements to a short table or array is to include the new entries on the input records before execution of the object program. Entries may be added during execution, however, through calculation operations or via input records. An example of adding elements to related alphanumeric tables is shown in figure 12-8.

Any type of table or array may be temporarily modified during execution of the object program. This is done by using the table or array name in the Result Field of an arithmetic or move operation. This type of modification lasts through the duration of the job only; the next time the program is executed, the table or array has its original contents. Modifications are permanent only if the table or array is output as modified and the new records are used the next time the job is run, or if the table or array input records are manually altered before the next execution of the object program.

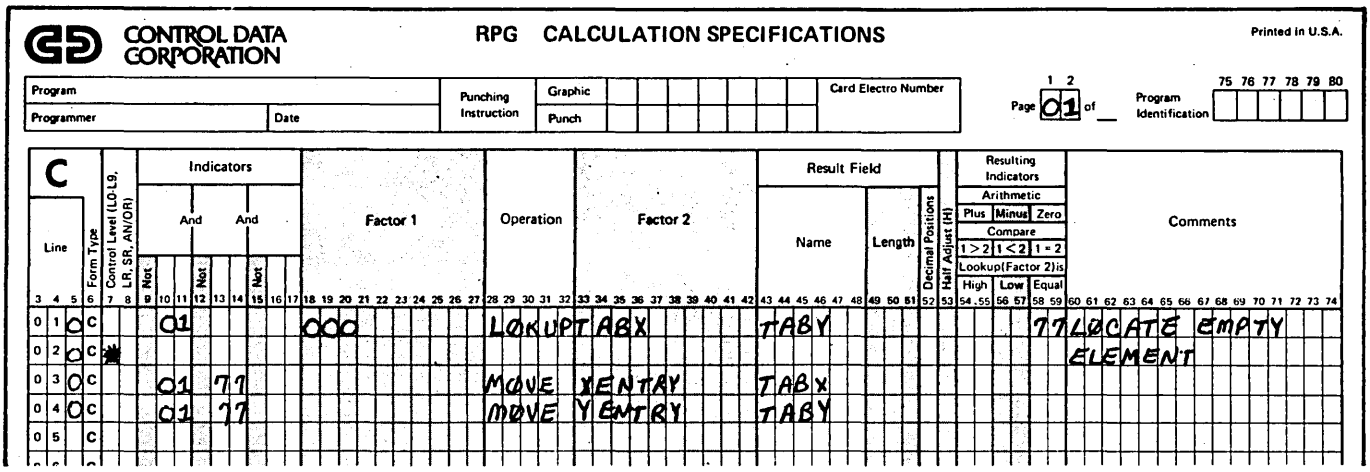


Figure 12-8. Adding Elements to Short Related Tables

An example of altering a table of pay rates is shown in figure 12-9. The table has to be modified to reflect an across-the-board 3 per cent cost-of-living increase. The pay rate table is related to a table of employee numbers, and table items are extracted via the input employee number. Notice that the new rates are rounded by a Half Adjust entry in column 53.

USING ARRAYS

An array name may be used in input, output, and calculation specifications. When an index is specified with the array name, the elements are referenced individually. A reference to the array name without an index is a reference to the entire array.

The maximum length an array name may have depends upon the way the name is to be used. If all references to the array are references to the whole array, the name may be six characters long. When an index is used with the array, however, the name may be no more than four characters long.

An index may be a numeric field with zero decimal positions or an integer literal. When used together, the array name and index are separated by a comma. But the space allocated to the array name, comma, and index is six positions in the input and output specifications and in the Result Field of the Calculation Specifications sheet. (Ten positions are allocated to Factor 1 and Factor 2 entries.) This is why an array name is limited to four characters if it is used with an index.

The index to an array may not be zero, negative, or greater than the number of declared entries in the array.

The following operation codes may reference an array name (with or without an index): ADD, Z-ADD, SUB, Z-SUB,

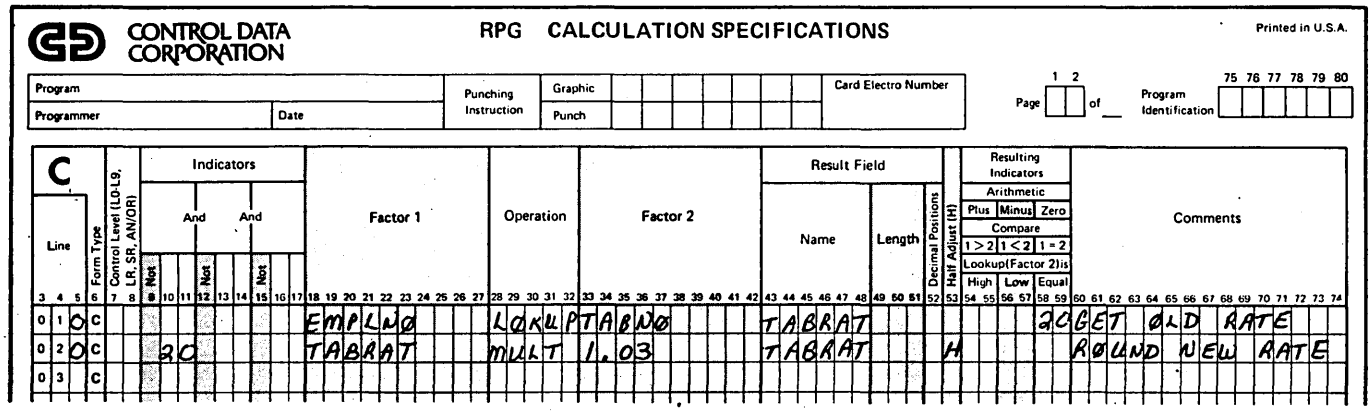


Figure 12-9. Modifying a Table During Calculation

MULT, DIV, SQRT, MOVE, MOVEL, MLLZO, MLHZO, MHLZO, MHHZO, MOVEA, DEBUG, XFOOT, and LOKUP. Factor 1 and Factor 2 may not be an array name unless the Result Field is also an array name, except when the operation code is LOKUP or XFOOT.

The following operations never reference a whole array but may reference an array element with an index: COMP, DSPLY, TESTZ, TESTB, BITQN, BITOF, and MVR.

The following rules apply to using an unindexed array name in calculations:

- When Factor 1, Factor 2, and the Result Field specifications are all unindexed array names, the operation is done using the first element of each array, then the second element of each array, and so on until the shortest array is exhausted. If the Result Field array is longer than at least one of the factors, unreferenced Result Field elements are not altered.
- When one of the factors is a field or constant and the other factor and the Result Field are unindexed arrays, the operation is done once for each element in the shorter array. The field or constant is used in each operation, and unreferenced Result Field array elements are not changed.
- When the operation does not use a Factor 1 entry (for example, Z-ADD), the Factor 2 entry is a field or constant, and the Result Field is an unindexed array, the operation is performed once for each array element using the same field or constant in each operation. The exception is the MOVEA operation, which moves the field into the array disregarding array element boundaries.
- Resulting indicators cannot be used due to the multiple operations being performed. The exceptions are the LOOKUP and XFOOT operations, where there is a single result.
- In arithmetic operations, arrays must be numeric. Half adjusting may be specified with these operation codes.

TABLE AND ARRAY OUTPUT

Tables and arrays may be output in two ways. One way is to have RPG II write out the table or array just before end-of-job. This is done by entering the desired output filename in columns 19 through 26 of the Extension Specifications sheet describing the table or array. The table or array is written out in its input format and includes any modifications made during execution of the object program. If the output file is a printer file, a skip to top-of-form should be requested following the last output line of any file that is also going to the same printer. This type of output may be requested for any table or array except an execution time table or array.

Any type of table or array may be output through output specifications. In these specifications, a table element and an indexed array element are treated like any other field specification. Blank-after codes may be used. Columns 40 through 43 of form O must contain the record position where the last field of the table or array is to end.

When an unindexed array name is specified as a field name on form O, the value specified as the end position in the output record is the position of the rightmost character of the last element in the array. Any editing specified for the array applies to each element in the array. Elements, therefore, must be individually specified as indexed elements if uniform editing cannot be used.

When an edit code (column 38 of form O) is specified for an unindexed array, two blanks are automatically inserted to the left of every element of the array. When an edit word is specified, blanks are not automatically inserted between elements. Edit words must contain explicit requests for blanks wherever they are desired.

EXAMPLE OF USING AN ARRAY

This example illustrates the use of arrays on the Output Specifications sheet and how arrays can reduce the number of coding lines required in a program. In this program, three levels of totals are maintained and output when appropriate control breaks occur.

A conventional method of accumulating the totals is shown in figure 12-10. The fields FDA, FDB, FDC, and FDD are added (as they are read from input records) to the first level totals, CL1A, CL1B, CL1C, and CL1D, respectively. At a level 1 control break, the level 1 subtotals are added to subtotals CL2A, CL2B, CL2C, and CL2D, respectively. At a level 2 control break, totals CL3A, CL3B, CL3C, and CL3D are updated. Total lines are output as control breaks occur, and totals are cleared to zeros after being output. Since CL1A through CL1D, CL2A through CL2D, and CL3A through CL3D are first defined as Result Fields on form C, the fields have automatically been initialized to zeros by RPG II. They do not, therefore, require initial clearing by the source program.

Figure 12-11 shows the same functions being performed with arrays. The execution time arrays TOT1, TOT2, and TOT3 are defined for accumulation of the various totals. The output produced by figures 12-10 and 12-11 is identical. Figure 12-11, however, eliminates most of the repetitious coding lines of figure 12-10. Since the array elements are defined on form E in figure 12-11, the definitions are not repeated on form C. Whereas in figure 12-10 the level 1, 2, and 3 fields are defined with lengths of five, six, and seven characters, respectively, each element of each array is defined as eight characters, for equal spacing on the output line. An RPG II supplies two blanks in front of each array element. The equal spacing occurs in figure 12-10 only as a result of individual end position specifications for each field.

CONTROL DATA CORPORATION

RPG CALCULATION SPECIFICATIONS

Printed in U.S.A.

Program | Programmer | Date | Punching Instruction | Graphic Punch | Card Electro Number | Page 01 of __ | Program Identification

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And/Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus/Minus/Zero; Compare 1>2/1<2/1=2; Lookup(Factor 2)is High/Low/Equal	Comments
01	C		10	FDA	ADD	CL1A	CL1A	5	2			
02	C		10	FDB	ADD	CL1B	CL1B	5	2			
03	C		10	FDC	ADD	CL1C	CL1C	5	2			
04	C		10	FDD	ADD	CL1D	CL1D	5	2			
05	C	L1		CL1A	ADD	CL2A	CL2A	6	2			
06	C	L1		CL1B	ADD	CL2B	CL2B	6	2			
07	C	L1		CL1C	ADD	CL2C	CL2C	6	2			
08	C	L1		CL1D	ADD	CL2D	CL2D	6	2			
09	C	L2		CL2A	ADD	CL3A	CL3A	7	2			
10	C	L2		CL2B	ADD	CL3B	CL3B	7	2			
11	C	L2		CL2C	ADD	CL3C	CL3C	7	2			
12	C	L2		CL2D	ADD	CL3D	CL3D	7	2			
13	C											

CONTROL DATA CORPORATION

RPG OUTPUT SPECIFICATIONS

Program | Programmer | Date | Punching Instruction | Graphic Punch | Card Electro Number | Page 02 of __ | Program Identification

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

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

Constant or Edit Word

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # / Fetch (F)	Space: Before / After	Skip: Before / After	Output Indicators: And/Not	Field Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R
01	O	PRINT	T		1 2		L1					
02	O							CL1A		J B	1 0	
03	O							CL1B		J B	2 0	
04	O							CL1C		J B	3 0	
05	O							CL1D		J B	4 0	
06	O		T		1 2		L2					
07	O							CL2A		J B	1 0	
08	O							CL2B		J B	2 0	
09	O							CL2C		J B	3 0	
10	O							CL2D		J B	4 0	
11	O		T		1 2		L3					
12	O							CL3A		J B	1 0	
13	O							CL3B		J B	2 0	
14	O							CL3C		J B	3 0	
15	O							CL3D		J B	4 0	
16	O											

Figure 12-10. Accumulating Totals Without Arrays

CONTROL DATA CORPORATION

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

Printed in U.S.A.

Program | Programmer | Date | Punching Instruction | Graphic Punch | Card Electro Number | Page 01 of __ | Program Identification

Extension Specifications

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
01	E					TØT1			4	E	2							
02	E					TØT2			4	E	2							
03	E					TØT3			4	E	2							
04	E																	

CONTROL DATA CORPORATION

RPG CALCULATION SPECIFICATIONS

Printed in U.S.A.

Program | Programmer | Date | Punching Instruction | Graphic Punch | Card Electro Number | Page 01 of __ | Program Identification

C

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators And	Indicators And	Factor 1	Operation	Factor 2	Result Field Name	Result Field Length	Result Field Decimal Positions	Half Adjust (H)	Resulting Indicators Arithmetic Plus	Arithmetic Minus	Arithmetic Zero	Compare 1>2	Compare 1<2	Compare 1=2	Lookup(Factor 2)is High	Lookup(Factor 2)is Low	Lookup(Factor 2)is Equal	Comments
01	C		10		FDA	ADD	TØT1,1	TØT1,1													
02	C		10		FDB	ADD	TØT1,2	TØT1,2													
03	C		10		FDC	ADD	TØT1,3	TØT1,3													
04	C		10		FDD	ADD	TØT1,4	TØT1,4													
05	C	L1			TØT1	ADD	TØT2	TØT2													
06	C	L2			TØT2	ADD	TØT3	TØT3													
07	C																				

CONTROL DATA CORPORATION

RPG OUTPUT SPECIFICATIONS

Program | Programmer | Date | Punching Instruction | Graphic Punch | Card Electro Number | Page C2 of __ | Program Identification

O

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # /Fetch(F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And Not	Output Indicators And Not	Output Indicators And Not	Field Name / *AUTO	Edit Codes	B/A/C/1-8/R	End Position in Output Record	P/B/L/R
01	O	PRINT	T	1	2				L1							
02	O											TØT1	J	B	40	
03	O		T	1	2				L2							
04	O											TØT2	J	B	40	
05	O		T	1	2				L3							
06	O											TØT3	J	B	40	
07	O															

Constant or Edit Word

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

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

Figure 12-11. Accumulating Totals With Arrays

The various file structures and modes of processing available in RPG II were described briefly in section 5. This section discusses in detail the subject of record selection and the significance of file structures to the modes of processing.

SEQUENTIAL FILE PROCESSING

All records in files with sequential organization are written to the storage medium with no intervening logical gaps; one logical record follows the next. Frequently, the records of a sequentially organized file are logically in sequence. For RPG II, logical sequence implies keys in the file that are in ascending or descending order. RPG II can process ordered and unordered sequentially organized files. An RPG II object program can create an ordered file if the input data is in the correct sequence. Alternately, a sort program can put existing sequential files in order.

All files on any medium can be created in sequential order by RPG II. Disk files may be created with indexed organization to permit random processing later. Other sequential disk files may be processed nonsequentially as well through the use of a record address file of tags (known as an ADDROUT file), which is prepared by a sort program.

RPG II object programs process records of sequential files as they appear from the beginning of the file. A sequential file may be processed to end-of-file, or processing may stop before end-of-file is reached (see End-of-File, section 5). As an example of the latter condition, consider a file of employee addresses that is referenced when mailing labels for the management newsletter are prepared. When the address corresponding to the highest management employee number has been extracted, there is no further use for the address file in this program, although the end of the address file may or may not have been reached.

An indexed file may be processed from beginning to end of the index or within index limits supplied by a record address file. In the above example, a record address file could supply the lowest and highest management employee numbers.

When there is only one input file, that file is the primary file (P in column 16 of form F). The entire file is processed beginning with the first record and continuing until the end of file has been reached.

With multiple input files, one is primary and the other(s) are secondary. If there is no relationship between the processing of the files, RPG II processes the primary file to its end before it processes the secondary file(s). In this case, two or more secondary files are processed in the order of their form F descriptions.

When a record matching relationship exists among multiple files, the file that controls processing is primary and the others are secondary. In the management newsletter example, the employee number input file is primary and the employee address input file is secondary. The application determines primary and secondary designations. In a billing program, sales transactions are accumulated for each customer record. In this case, the customer file is designated a secondary file, ensuring that individual sales transaction records (from the primary file) have already been read and accumulated before the next customer record is read.

In the following discussion of primary and secondary input files, it is important to keep in mind the fact that RPG II first reads an input record into a core area known as a read area and then moves the record to another core area known as a process area. A control break is detected while the record causing the control break is still in the read area. RPG II performs all appropriate control break processing before moving that record to the process area.

MATCHING RECORDS

Interrelated multiple input files are processed through the matching record feature of RPG II. The matching field entries in columns 61 and 62 of the Input Specifications sheet determine when records of a secondary file are to be processed. The following rules apply to matching field entries:

- Up to nine matching field entries (M1 through M9) may be specified for a record.
- The locations of matching fields in a record must be the same for all records of that type.
- The locations of matching fields in different record types may be different.
- Not all of the record types in a single file must have matching field specifications.
- Records may be matched by one field, many fields, or in their entirety.
- Records of two or more files being matched must have the same number of matching field specifications.
- The combined length of the matching fields for one record must equal the combined length of the matching fields of any record with which it is being matched.
- The combined length of matching fields within a record must not exceed 256 characters.
- When matching fields are specified for a single input file, the effect is sequence checking of the file.

Notice that matching fields may be specified for a file even if not all record types of the file are to enter into the matching. Matching field entries are not made for records that are not to be matched.

RPG II turns on the matching record indicator (MR) when a match occurs between records of the primary file and secondary file(s). The indicator may be used to condition calculation and output functions. The indicator remains on during the complete processing of the selected record and is turned off by RPG II after all calculations and output specified for the record have been executed.

If there is no match between records of the primary file and secondary file(s), the MR indicator is not turned on, and the off condition may be used to condition calculations for unmatched records or selection of unmatched records for output.

If there is more than one matching field entry for a record, the codes M1 through M9 must be assigned to the fields in the order of increasing significance of the match. If a matching field in any record is defined as numeric, all fields at that same Mn level are treated as numeric fields in the match. If all fields at a given level are defined as alphanumeric, the fields are compared logically.

During the matching of records of primary and secondary files, four situations can occur:

- There is a matching primary record.
- There is a matching secondary record.
- There is an unmatched primary record.
- There is an unmatched secondary record.

Given these four possible situations, here are the rules for the order of processing:

- Records having no designated matching fields are processed before records having matching field specifications, regardless of file.
- When records match, the primary file record is processed first.
- Matching records of additional secondary files are processed in the order of specification of the files on form F.
- When records with matching field specifications do not match, the record first in sequence according to the matching field value is processed first, regardless of file.

Record selection based on matching fields of one primary file and one secondary file is illustrated in figure 13-1. The result is shown as a merger of the two files. Record selection is performed according to the two-digit numeric field values shown.

Record types not having matching field specifications are indicated with no match values. The MR indicator is on during processing of the shaded records. The files shown are in ascending sequence of the match fields; the record selection would be inverted if the files were in descending sequence. Notice that for both files, or neither, an E must have been entered in column 17 of form F (end of file) since processing of the secondary file continues after the end of the primary file has been reached.

LOOK-AHEAD FIELDS

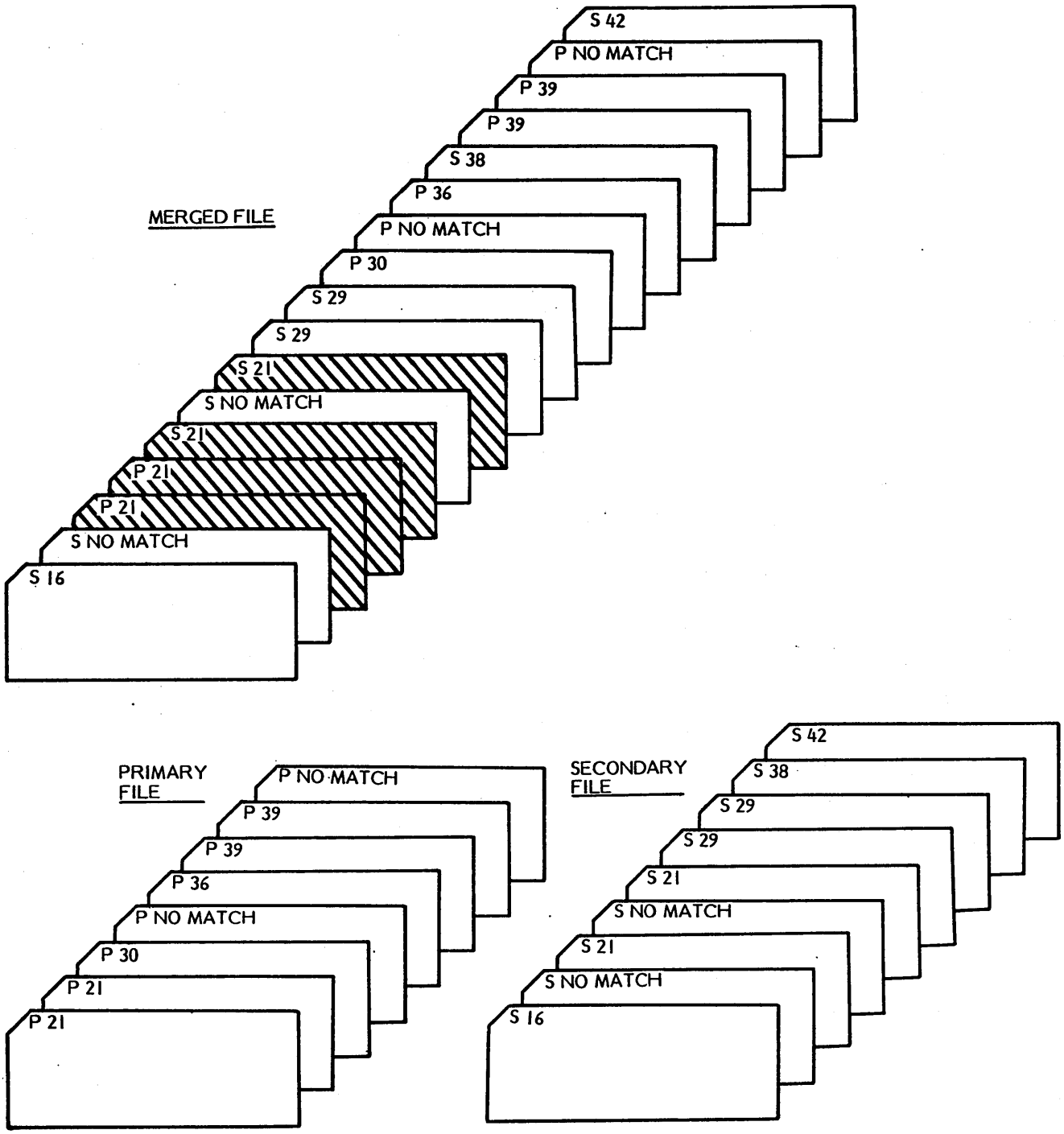
Look-ahead extends the matching record capability by allowing fields in the read area to be used before the record has been selected for processing. Look-ahead cannot be specified for chained or demand files, as records in these files are not held in a read area before processing.

Figure 13-2 illustrates the read and process areas as record selection in figure 13-1 is taking place. In picture 1, the first record of each file has been read. In picture 2, the secondary record with a matching field value of 16 has been selected for processing. Records selected for processing and records available for look-ahead are charted by picture number in table 13-1.

Look-ahead can be specified for update and combined files, but it applies to the current record of the file only. A copy of the current update or combined file record remains in the read area until the end of cycle; thus, the next record is not available for look-ahead until the next cycle.

If the secondary file in figure 13-1 were an update file, the read and process areas would take the configurations shown in figure 13-3. Records selected for processing and records available for look-ahead are shown in table 13-2.

An application for look-ahead fields is shown in figure 13-4. The job reads records from two files: a primary update file named PRIMARY and a secondary input file named SECONDARY. Changes for some items in the primary file are contained in the secondary file. Thus, if a record from the primary file matches one from the secondary file, the information in character positions 1 through 8 of the secondary file record is placed in positions 15 through 22 of the primary file record. When matching does not occur and a secondary record is being processed, the erroneous record is printed out. (A primary record without a matching secondary record is not an error condition and is written to the update file as is.)



NOTE: MR INDICATOR IS ON DURING PROCESSING OF THE SHADED RECORDS.

0311

Figure 13-1. Record Selection Based on Matching Fields

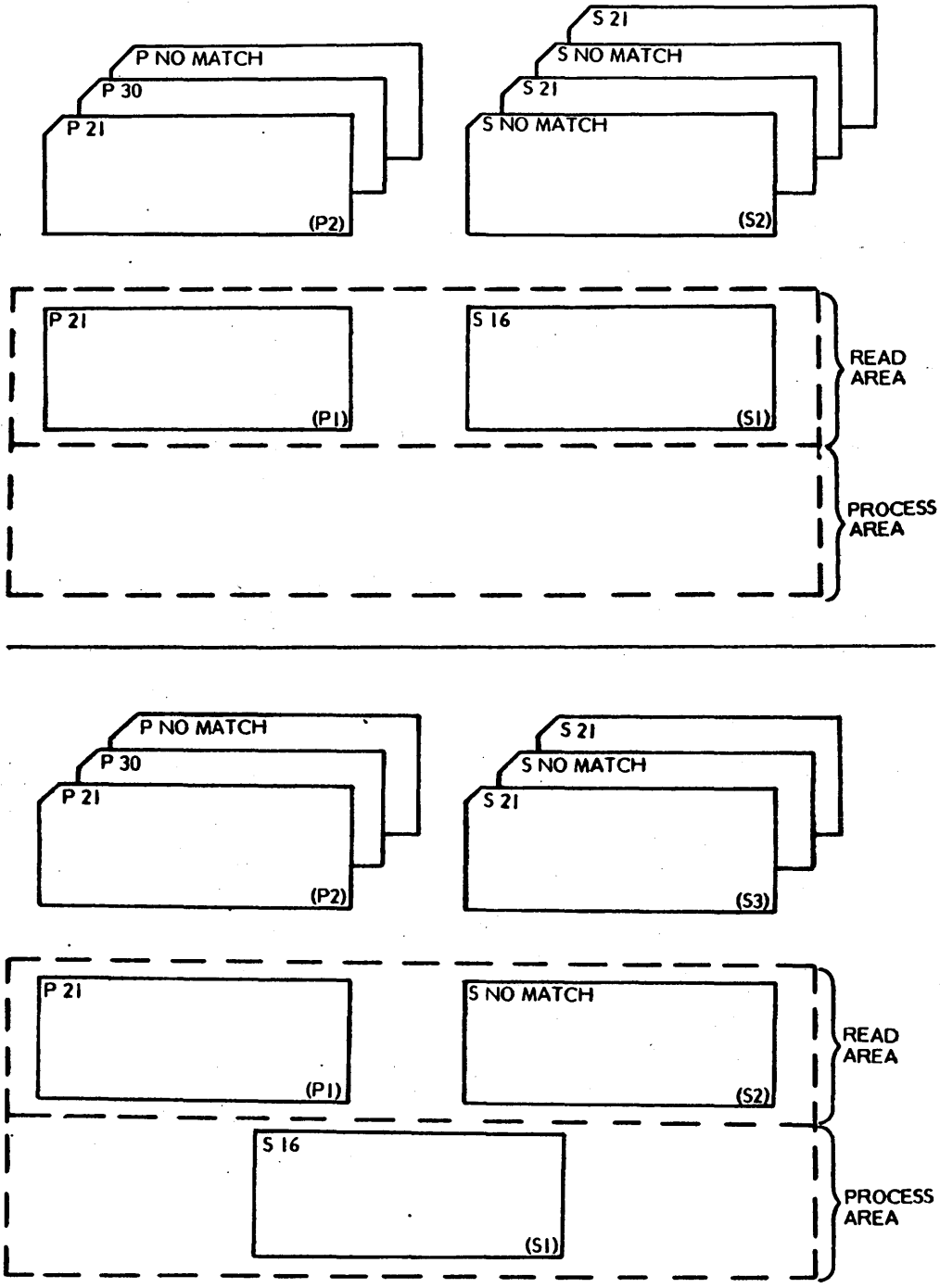


Figure 13-2. Read and Process Areas for Records in Figure 13-1 (Sheet 1 of 2)

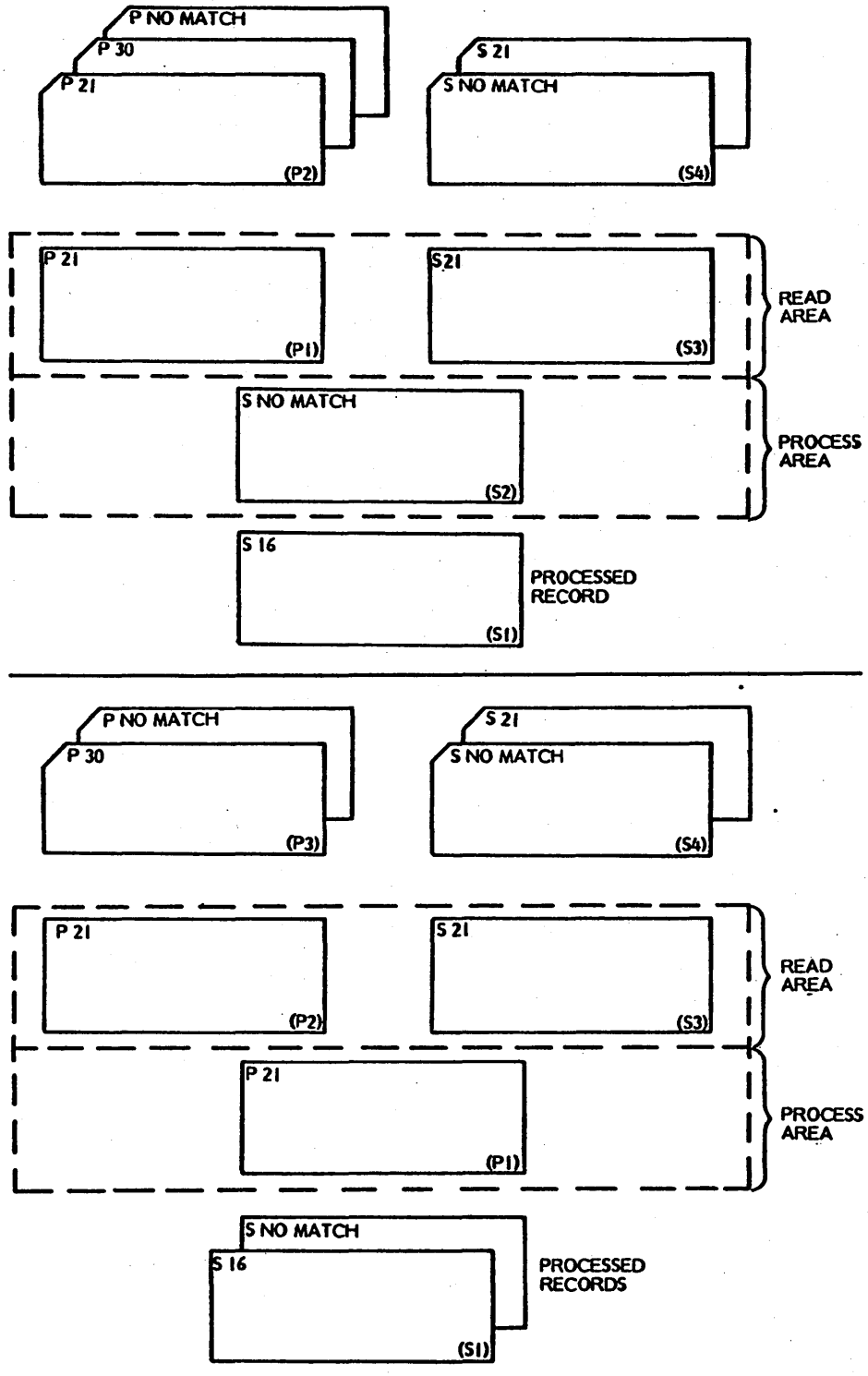


Figure 13-2. Read and Process Areas for Records in Figure 13-1 (Sheet 2 of 2)

TABLE 13-1. RECORDS AVAILABLE FOR LOOK-AHEAD

Picture	Record Being Processed	Records Available
2	S1	P1 and S2
3	S2	P1 and S3
4	P1	P2 and S3

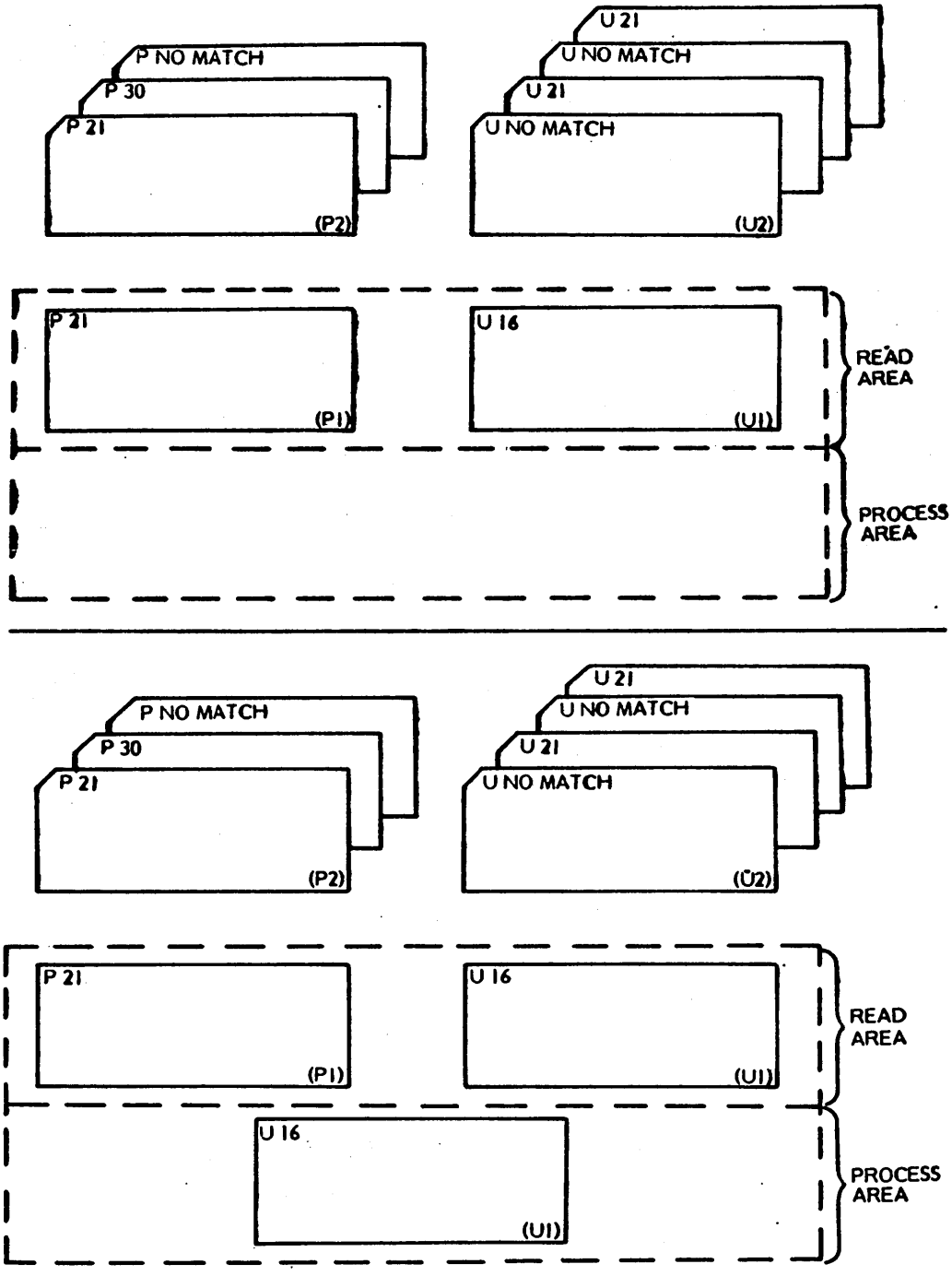


Figure 13-3. Look-Ahead With Update Files (Sheet 1 of 2)

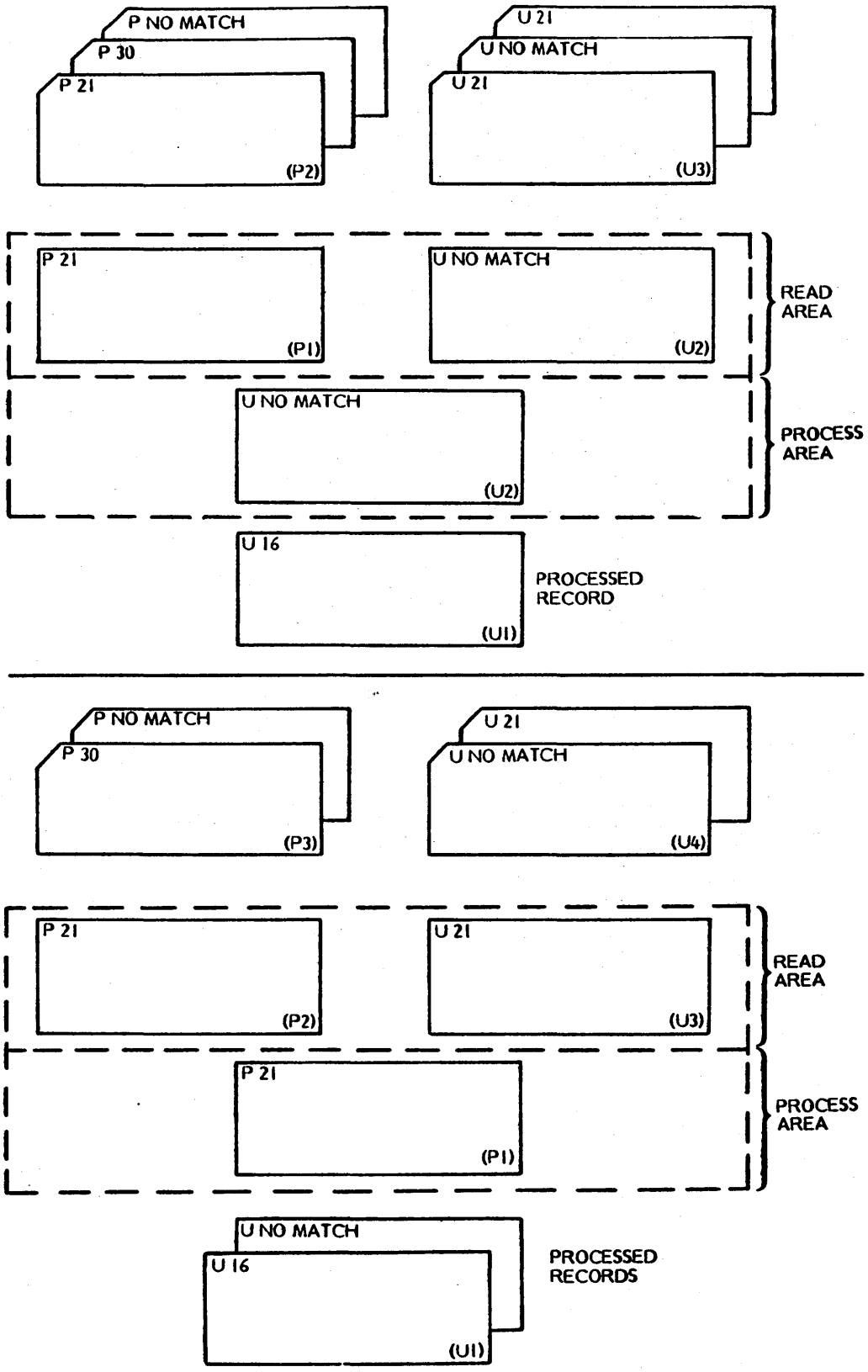


Figure 13-3. Look-Ahead With Update Files (Sheet 2 of 2)

TABLE 13-2. RECORDS AVAILABLE FOR LOOK-AHEAD WITH UPDATE FILE

Picture	Record Being Processed	Records Available
2	U1	P1 and U1
3	U2	P1 and U2
4	P1	P2 and U3

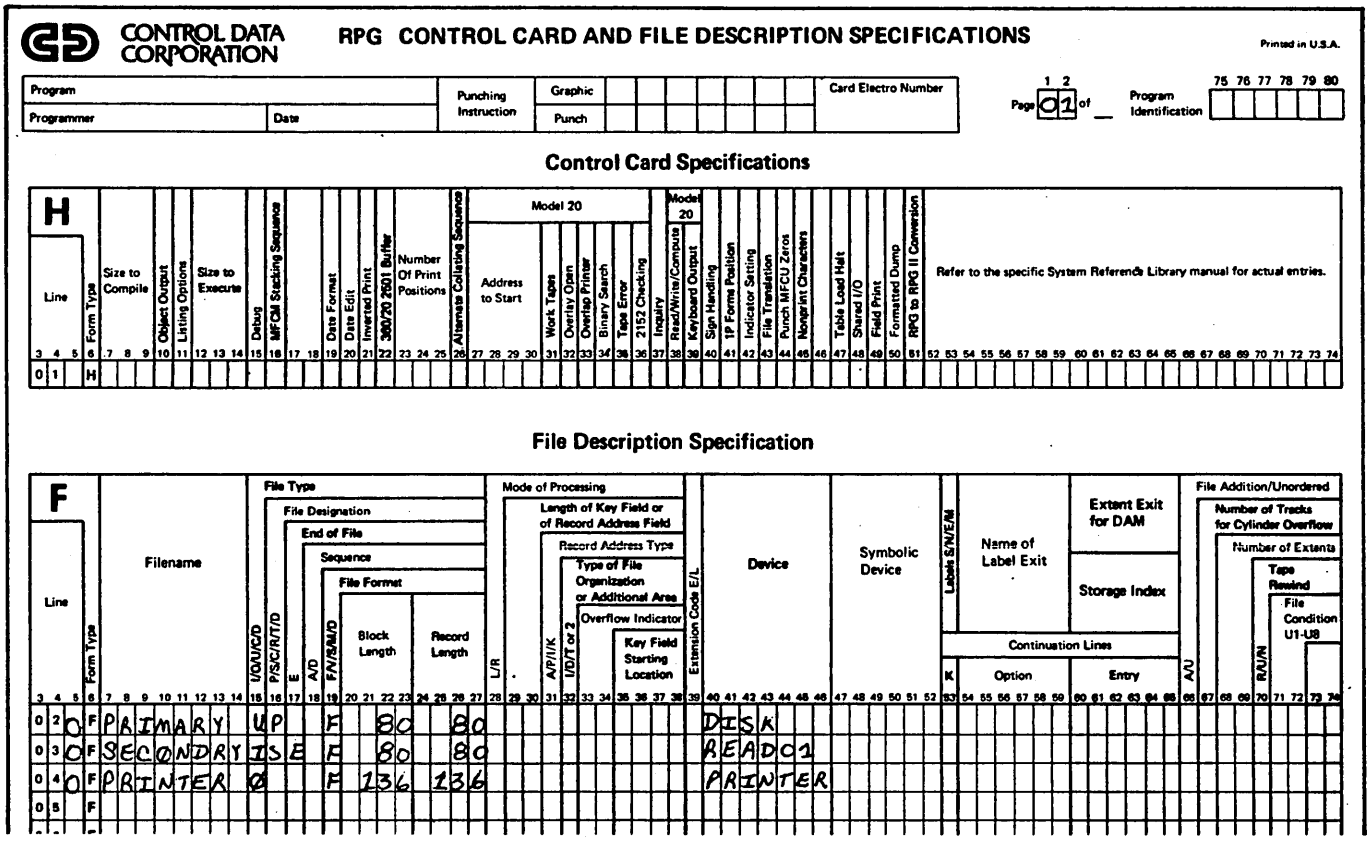


Figure 13-4. An Application for Look-Ahead Fields (Sheet 1 of 2)

CONTROL DATA CORPORATION

RPG INPUT SPECIFICATIONS

Program | Programmer | Date | Punching Instruction | Graphic Punch | Card Electro Number | Page 02 of __ | Program Identification

I

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Minus	Zero or Blank
01	I	PRIMARY	XX			C1																								
02	I																				1	8		ITEM1						
03	I																				9	14		MATCH		M1				
04	I	SECONDRY	YY			02																								
05	I																				1	8		ITEM2						
06	I																				9	14		MATCH		M1				
07	I		ZZ			**																								
08	I																				1	8		NXTITM						
09	I																													

CONTROL DATA CORPORATION

RPG OUTPUT SPECIFICATIONS

Program | Programmer | Date | Punching Instruction | Graphic Punch | Card Electro Number | Page 03 of __ | Program Identification

O

Line	Form Type	Filename	Type (H/D/T/E)	Stacker #/Fetch(F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And Not	And Not	And Not	Field Name *AUTO	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R
01	O	PRIMARY	D						01	MR						
02	O											NXTITM			22	
03	O	PRINTER	H		2	0	5		2P							
04	O														14	'INVALID RECORDS'
05	O		D		1				02	MR						
06	O											MATCH			6	
07	O															

Constant or Edit Word

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

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

Figure 13-4. An Application for Look-Ahead Fields (Sheet 2 of 2)

INDEXED FILE PROCESSING

Indexed files may be created (loaded) by RPG II on disk as output files. During the file creation, RPG II accumulates the record keys to form an index to the file. Once the file has been loaded, records of the file may be accessed by other RPG II programs on a random basis. RPG II might be given the record key of a single desired record and use this key to find the index that points to the right record. The record then may be used as a source of input information, or the record may be simply changed and returned to its previous position in the disk file.

The various methods of processing an indexed file provide great operational flexibility to the RPG II user, as keys to the indexed file, provided by record address files, can be changed each time the object program is executed. Consider a department store billing program which is to be run once a week. In order to bill customers on a monthly basis, it would be convenient to index the customer file alphabetically by name and to process approximately one quarter of the file each week. The processing limits could then have the following configuration:

Record Address File

	Lower level	Upper level
1st week	AARDAL	GYLLENSWAN
2nd week	HAACK	NYSTROM
3rd week	OAKLEY	TYSON
4th week	UEBELE	ZWELLING

Two other methods of processing an indexed file are:

- Random processing with the record key supplied in a CHAIN operation
- Processing by ADDROUT file

Processing an indexed file with specific keys using the CHAIN operation code allows even greater selectivity than processing within limits. Further, while processing within limits reduces the number of records to be handled, it still requires sequential processing within the limits. With CHAIN, the order of processing may be completely nonsequential. This mode of processing allows retrieval of specific unordered information in an indexed file.

As an example, suppose that certain customers in the department store billing file had very delinquent payment records. If the store manager had threatened legal action if a payment were not received by a certain date, on that date he could examine just those delinquent accounts by supplying the customer names through the CHAIN operation code.

An ADDROUT file is a special type of record address file created by sort programs. The indexed file is sorted by some field(s) in the records, and the ADDROUT file or binary relative record numbers is produced according to the sorting sequence. The key fields of the sort would ordinarily not be the same key fields that determine the order of the index for the file.

When the sequential by key or the random method is being used to access records of an indexed file, an entry of A in column 66 of form F (File Addition/Unordered Output) permits the addition of new records to a pre-existing file. New records are added to the end of the data file and new index information is added to the end of the index portion of the file. Then at the end of object program execution, RPG II reorganizes the entire index into ascending key sequence. File addition is performed more efficiently if the new records are presented in ascending key sequence.

DIRECT FILE PROCESSING

Direct access files are sequentially organized disk files that also may be processed randomly. Records of a direct file are assigned specific record positions on disk. Regardless of the order in which the records are put in the file, they always occupy specific disk addresses. A relative record number identifies the relative position of a record within the file.

A direct file is created in RPG II through the CHAIN operation code. The user supplies relative record numbers for the records to be loaded. Before the file is loaded, the entire disk area is cleared to blanks. Record positions remain blank and reserved for records not available at the time of file creation.

Records are retrieved from direct files through ADDROUT files or with the CHAIN operation.

Because a direct file is not organized according to a known key sequence, sequential processing of the file by key or limits is not possible.

Records may be added to a pre-existing direct file through an entry of A in column 66 of form F and by the execution of CHAIN operation codes, or through the declaration of the file as an update file processed consecutively. When CHAIN is used, new records replace the gaps. In the latter case, new records are added to the end of the direct file.

b – The character frequently used throughout this manual to denote a blank character

Access Method – A way of moving data between main storage and input/output devices

Alphabetic Characters – The 26 alphabetic EBCDIC characters and the three 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 forms. Alphanumeric fields contain alphabetic, numeric, or special characters

American National Standard Labels – Magnetic tape labels conforming to conventions established by the American National Standards Institute; also called ASCII label

Application Program – A program written for or by a user

Array Element Name – An array name followed by a comma and an index. The indexed array name designates a specific element in an array.

ASCII – American Standard Code for Information Interchange. The 128-character external code set used in this RPG II compiler.

ASCII Label – See American National Standard Label.

Assemble – To prepare a machine language program from a symbolic language program by the substitution of absolute operation codes for symbolic operation codes and absolute or relocatable addresses for symbolic addresses

Batch Processing – The technique of executing a set of computer programs such that each is completed before the next program of the set is started

Blocking – Two or more records combined into one block

Blocking Factor – The number of logical records combined into one block or physical record

Buffer – A temporarily reserved area of storage for use in performing an input/output operation and into which data is read or from which data is written; also called I/O area

Byte – Eight data bits; one character

COBOL – Stands for Common Business-Oriented Language; a business data processing language

Collate – Combining items from two or more ordered sets into one set having a specified order not necessarily the same as any of the original sets; contrast with merge

Collating Sequence – Any logical sequence used to put items of data into order

Combined File – A card file that both input and output operations are performed on. All cards in the file are read, but not all cards are punched, interpreted and/or stacker selected.

Compilation Time – The time during which the RPG II compiler accepts source program specifications, produces program and diagnostic listings, and generates the object program

Compile – To prepare a machine language program from a computer program written in another programming language through use of the overall logic structure of the program or by generating more than one machine instruction for each symbolic statement, or both, as well as performing the function of an assembler

Console – The computer part used for communication between the operator or maintenance personnel and the computer

Control Card – A punched card that contains input data or parameters for initializing or modifying a program

Control Field – A group of record bytes used in determining sequence in sorting or merging records

Conversion – The process of changing from one method of data processing to another or from one data processing system to another; also the processor of changing from one form of representation to another; e.g., to change from decimal representation to binary representation.

Data File – A collection of related data records organized in a certain manner. For example, a payroll file or an inventory file

Deblock – To make the first and each subsequent logical record of a block available for processing, one record at a time

Debug – To detect, locate, and remove mistakes from a routine or malfunctions from a computer; also called troubleshooting

Default Value – The choice the system makes among exclusive alternatives when no explicit choice is specified by the user

Detail Time – The part of execution time during which operations that occur every cycle are executed

Direct Access Storage Device – A device in which the access time is effectively independent of the location of the data

Direct Organization – A file organization containing, for purposes of storage and retrieval, a relationship between the contents of the records and their positions in the file. Contrast with indexed organization and sequential organization

Disk – A disk storage device

Disk Pack – A removable direct access storage volume containing magnetic disks on which data is stored. Disk packs are mounted on a disk storage drive.

Dump – To copy all or part of the contents of a storage, usually from an internal storage into an external storage; to process the contents or the data resulting from the process

EBCDIC – Extended Binary Code Decimal Interchange Code. The 256-character internal code used in this RPG II compiler

Execute – To carry out an instruction or group of instructions, as in a program

Execution Time – The time during which object program cycles are being executed

External Subroutine – A referenced subroutine that is not part of this RPG II source program

File – A group of related records treated as a unit

File Maintenance – To keep a file up to date by adding, changing, or deleting data

Fixed-Length Record – A record having the same length as all other records with which it is logically or physically connected. Contrast with variable-length record

Hexadecimal – Pertaining to a number system with a base of 16; valid digits range from 0 through F, where F represents the highest units position (15)

Index – An ordered list of the contents of a file or document, containing keys or reference notations for identification or location of those contents. Also, a table used to locate the records of an indexed file

Indexed Organization – A file organization containing records arranged in logical sequence by key. Indexes to these keys permit random processing of individual records. Contrast with direct organization and sequential organization

Input/Output – About either input or output, or both, a general term for equipment used to communicate with a computer (commonly called I/O), data involved in I/O communication, or media carrying the data for input/output

Installation – A particular computing system, considering the work it does and the people who manage it, operate it, apply it to problems, service it, and use the results it produces

Internal Subroutine – A subroutine coded in the RPG II language that is part of this RPG II source program

Interpret – To print on cards or to translate

Interrupt – To stop a process so that is can be resumed

I/O – See input/output

I/O Area – Same as buffer

ITOS – The Interactive Terminal-Oriented System for CYBER 18-20 computers

JCL – Job control language

Job – A specific group of tasks called out as a unit of work for a computer; a job usually includes all necessary computer programs, linkages, files, and instructions to the operating system. Also a collection of related problem programs identified in the job stream by a JOB statement. A job consists of one or more job steps.

Job Control Language – A programming language used to code job control statements. Abbreviated as JCL

Job Control Statement – A statement in a job or job step that is used to identify the job or describe its requirements to the operating system

Job Step – Work associated with one processing program or one cataloged procedure and related data

Job Stream – The sequence of operation control statements and data submitted to an operating system on an input unit activated for this purpose by the operator

K – 1024 bytes or words used in referring to storage capacity

Key – One or more characters within an item of data used to identify it or control its use

Label – An identification record for a tape or disk file

Language – A group of representations, conventions, and rules used to convey information

Language Translator – A general term for any compiler, assembler, or other routine that accepts statements in one language and produces equivalent statements in another

Library – A set of procedures, programs, routines, etc., on a direct access storage device. Also see source library and object library

Load – (1) To enter data or programs into storage (2) Same as (1) using the loader

Loader – See relocatable binary loader

Main Storage – A computer's general purpose storage. Contrast with auxiliary storage

Merge – To combine items from two or more similarly ordered sets into one set that is arranged in the same order (contrast with collate) or a program or routine that performs this function

MFCM – Multi-Function Card Machine. Same as IBM 2560 Multi-Function Card Machine

MFCU – Multi-Function Card Unit. Same as IBM 5424 Multi-Function Card Unit

MSOS – The Mass Storage Operating System for CYBER 18 systems

Multiprogramming – The concurrent execution of two or more programs by a computer

Nonstandard labels – Labels that do not conform to American National Standard or IBM Standard label conventions

Numeric Fields – All fields having a decimal positions specification in the appropriate columns of the specifications forms

Object Code – Output from a compiler or assembler that is an executable machine code or is suitable for processing to produce an executable machine code

Object Program – A fully compiled or assembled program ready to be loaded into the computer. Contrast with source program

Offline – Equipment or devices not under control of the CPU

Online – Equipment or devices under control of the CPU; a user's ability to ineract with a computer

Operating System – Software that controls the execution of computer programs and that may provide scheduling, debugging, input/output control, accounting, compilation, storage assignment, data management, and related services

Operator Message – A message from the operating system or a user program directing the operator to perform a specific function (i.e., mounting a disk pack or informing him of specific conditions within the system, such as an error condition)

Pack – Compressing data in a storage medium by taking advantage of known characteristics of the data so that the original data can be recovered (i.e., to compress data in a storage medium by making use of bit or byte locations that would otherwise go unused). See disk pack

Partition – A subdivision of main storage.

Permanent File – A disk file that is normally created for continuing use. Contrast with scratch file and temporary file

Pre-Execution Time – The time during which the object program is entering execution: after the object program has been loaded into core memory and just before the first program cycle is executed

Priority – A rank assigned to a job that determines its precedence in receiving system resources

Program Library – A group of available computer programs and routines

Prompting – A function that assists a user by requesting him to supply information necessary to continue processing

Queue – A waiting line or list formed by items in a system waiting for service (e.g., jobs to be performed), or to arrange in, or form, a queue

Random Processing – The treatment of data with respect to its location in external storage, and in an arbitrary sequence governed by the input against which it is to be processed. Contrast with consecutive processing and sequential processing

Record – A group of related items of data that are treated as a unit (e.g., one line of an invoice may form a record; a complete set of such records may form a file)

Record Key – The field or fields that uniquely identify a record in an indexed file

Relocatable Binary Loader – A program that prepares the output of language translators for execution. It combines separately produced object modules, resolves symbolic cross references among them, and produces executable code (a load module).

Reproduce – To prepare a duplicate of stored information, especially for punched cards

Routine – A set of coded instructions arranged in proper sequence to direct the computer to perform a desired operation or series of operations

RPG II – Report Program Generator. A business-oriented data processing language

Scratch File – A disk file that is created and deleted in the same job step. Contrast with permanent file and temporary file

Sector – A 96-word directly-addressable part of a disk storage device

Seek – To position the access mechanism of a direct access device at a specified location

Sequence – Items arranged according to a specified set of rules. In sorting, a collection of records whose control fields are in ascending or descending order according to the collating sequence.

Sequential organization – A file organization in which records are arranged in a physical sequence, but not necessarily in logical sequence. Contrast with direct organization and indexed organization

Sequential Processing – A method of treating data with respect to its location in external storage in a sequence governed by the logical order of the data in the file. Contrast with consecutive processing and random processing

Source Program – A computer program that is written in source language. Contrast with object program

Source Statement – A statement written in symbols of a programming language

Special Characters – The 217 EBCDIC characters not defined as alphabetic or numeric

Spooling – Reading and writing input and output streams on disk concurrently with job execution in a format convenient for later processing or output operations

Storage – A device into which data can be entered and held and from which it can be retrieved at a later time; also, any device that can store data

Storage Protection – An arrangement for preventing access to storage for either reading, writing, or both. Also called memory protection

System Generation – The process of tailoring the system to suit a user's requirements and of including the desired program products

Total Time – The part of execution time occurring after a control break in which operations for the prior control group are processed

Unit Record Devices – Card readers, card punches, and printers

Utility Program – A program designed to perform an everyday task, such as copying data from one storage device to another

Variable-Length Record – A record having a length independent of the length of other records with which it is logically or physically connected. Contrast with fixed-length record. Also, a file in which the records are not uniform in length

Volume – That portion of a single unit of storage that is accessible to a single read/write mechanism, such as a disk pack. Also, a recording medium that is mounted and demounted as a unit; e.g., a reel of magnetic tape or a disk pack

DEVIATIONS FROM IBM SYSTEM/3 RPG II

This product is designed to be nearly compatible with IBM System/3 Model 10 RPG II. Some of the differences are due to requirements of the operating environment; other changes represent system improvements.

Those IBM System/3 Model 10 RPG II features that are omitted are:

- A Telecommunications Specifications sheet is not supported.
- The P (punched card) option is not provided for Object Output on the Control Card Specifications sheet.
- A specification for Core Size to Execute is not supported on the Control Card Specifications sheet.
- An Inquiry specification on the Control Card Specifications sheet is not supported.
- A Punch MFCU Zeros specification is not supported for the Control Card Specifications sheet.
- The Nonprint Characters specification on the Control Card Specifications sheet is not supported.
- *PRINT as a special word in the output specifications is not provided.

Some features not applicable to current hardware configurations are supported. This implementation provides for the future addition of such features. An example of this is input/output stacker select. The compiler does not diagnose such a feature and defaults the specification to a logical alternative.

Other differences from System/3 RPG II signify the easing of restrictions. These features are listed in table B-1. In this table, the term unlimited means that the feature is limited only by the available computer memory.

TABLE B-1. DIFFERENCES BETWEEN SYSTEM/3 AND MSOS OPTIONS

Feature	System/3 Implementation	MSOS Implementation
Number of files allowed in one source program	Maximum of 20	Unlimited
Number of demand and/or chained files allowed in one source program	Maximum of 15	Unlimited
Name specified for Label Exit	Name must be SUBRxx or SRyzzz	Any name is allowed
Number of tables and arrays in one source program	Maximum of 63 and only 60 may be compile time	Unlimited
Number of spread card specification lines	Maximum of 128	Unlimited
Device used with spread cards	Device must be card reader	Any sequential input device is valid
Format of trailer fields on spread cards	Must be unpacked decimal	May be unpacked decimal, packed decimal, or binary
Number of AND/OR lines allowed on Input Specifications sheet	Maximum of 20	Unlimited
Number of characters in control fields	Maximum of 144	Maximum of 256
Number of characters in matching fields	Maximum of 144	Maximum of 256
Number of AND/OR lines on Calculation Specifications sheet	Maximum of 7	Unlimited
Number of AND/OR lines on Output Specifications sheet	Maximum of 20	Unlimited
Number of digits allowed with Y edit code	3 to 6	1 to 15

This appendix is provided as a brief summary of sections 3 through 10. It is intended as a quick reference for programmers who are acquainted with RPG II.

On any form, columns for which specifications are not listed are to be left blank.

COMMON ENTRIES

RPG II specifications forms should be ordered in ascending numeric sequence based on columns 1 through 5. Pages should be arranged in the following order and numbered accordingly:

1. Control Card Specifications (Form H)
2. File description Specifications (Form F)
3. Extension Specifications (Form E)
4. Line Counter Specifications (Form L)
5. Input Specifications (Form I)
6. Calculation Specifications (Form C)
7. Output Specifications (Form O)

Columns 1-2 (Page)

Numeric page number

Columns 3-5 (Line)

Numeric line number within page

Column 6 (Form Type)

Enter an H, F, E, L, I, C, or O to indicate the specifications form type.

Column 7 (comments)

An asterisk in this position indicates that the entire line is commentary.

Columns 75-80 (Program Identification)

Enter any valid characters into these columns of form H to identify the object program. If these columns are left blank, the name RPGOBJ is assigned to the program. Columns 75 through 80 on all other specifications lines may contain any entries.

CONTROL CARD SPECIFICATIONS - FORM H

Column 15 (DEBUG)

Blank	DEBUG operation not used
1	DEBUG operation is used

Column 21 (Inverted Print)

Blank	Domestic format
I	Foreign format
J	Foreign format (leading zero remains for zero balance)
D	United Kingdom format

Column 26 (Alternate Collating Sequence)

Blank	Normal collating sequence used
S	Alternate collating sequence is supplied at end of source program.

Column 41 (1P Forms Positioning)

Blank	First 1P line to be printed only once.
1	First 1P line may be printed repeatedly to allow forms positioning.

Column 43 (File Translation)

Blank	File translation not needed
F	Input, output, update, or combined files need translation (translation tables at end of source program).

Column 48 (Shared I/O)

Blank	Each disk file uses a separate input/output area
1	All disk files share a single input/output area.

Column 74 (Segmented Running)

Blank	The program is to be produced as a single entity.
1	The object program is to be segmented to conserve memory space.

FILE DESCRIPTION SPECIFICATIONS - FORM F

Columns 7-14 (Filename)

Enter a unique name for each file. The filename may be from one to eight characters long, must begin in column 7, and must be a valid RPG II filename.

Column 15 (File Type)

I	Input
O	Output
U	Update
C	Combined
D	Display

Column 16 (File Designation)

Blank	Display file or output file other than chained output file
P	Primary
S	Secondary
C	Chained
R	Record Address
T	Table or Array
D	Demand

Column 17 (End-of-File)

Blank	Program can end before end of this file is reached.
E	Program termination not possible before end of file on this file.

If column 17 is blank or E for all files, every record of every input, update, and combined file is processed before the program terminates. An E in this column requires an I, U, or C in column 15, and a P, S, or R in column 16.

Column 18 (Sequence)

Blank	Sequence checking not requested
A	Records are to be checked for ascending sequence.
D	Records are to be checked for descending sequence.

An entry is required here when matching fields are used. Column 18 applies to update and combined files and all input files except table, array, chained, demand, and record address files.

Column 19 (File Format)

F	Fixed length records
V	Variable length records (EBCDIC tape files only)
D	Variable length records (ASCII tape files only)

Columns 20-23 (Block Length)

Blank	Block length equals record length.
1-9999	Multiple of record length (tape and disk only) or record length (any device)

Columns 24-27 (Record Length)

1-9999	Number of characters per record
18-9999	Maximum number of characters per tape record

Column 28 (Mode of Processing)

Blank	a. Not a disk file b. Sequential by key c. Consecutive
L	Sequential within limits a. Random by relative record number b. Random by key c. By ADDROUT file d. Direct file load (random load)

This column applies only to disk files.

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

Blank	Sequential or direct file or chained input file accessed by relative record number
1-29	Length of indexed file's record key (entry made for indexed file or record address file containing limits)
3	Length of ADDROUT file record

Column 31 (Record Address Type)

Blank	Sequential or direct file
A	Indexed file, alphanumeric key
P	Indexed file, packed key
I	ADDROUT file, or processed by ADDROUT file

This column applies to disk files specified as input, update, or chained output files.

Column 32 (File Organization or Additional I/O Area)

Blank	Sequential or direct file; use one input/output area for the file
I	Indexed organization
T	ADDROUT file
1-9	Sequential or direct file; use two input/output areas for the file

Columns 33-34 (Overflow Indicator)

Blank	No overflow indicator is used.
OA-OG, OV	Overflow indicator used to condition records in the file

Columns 35-38 (Key Field Starting Location)

Blank	Not an indexed file
1-9999	Indexed file; record position in which the key field begins

Column 39 (Extension Code)

Blank	There are no file extension or line counter specifications for this file.
E	The file described on this line is a table file, array file, or record address file further described in extension specifications.
L	The file described on this line is a printer file further described in line counter specifications.

Columns 40-46 (Device)

Enter a device code from appendix D for the input/output unit used by the file specified in columns 7 through 14.

Column 53 (Continuation Line)

Blank	This is not a continuation line.
K	This is a continuation record.

Columns 54-59 (Continuation Lines Option)

Blank	This is not a continuation line.
ASCII	ASCII tape file specified
BUFOFF	ASCII tape input file contains a block prefix.
Table or array name	Name of table or array to be referenced by a user-supplied external subroutine. Name cannot be ASCII or BUFOFF.

Columns 60-65 (Buffer Offset Length)

Blank	Not a continuation line having BUFOFF as the Continuation Lines Option
0-99	Length of the block prefix in an ASCII tape input file that specifies BUFOFF

Columns 54-59 (Name of Label Exit)

Blank	No special device named in columns 40 through 46; no label exit is required.
Valid RPG II name	Name of user-supplied external subroutine that performs input/output operations for this special device.

Column 66 (File Addition/Unordered Output)

Blank	No file addition or unordered output for this file
A	New records are added to this preexisting file.
U	Records are to be loaded into this new indexed file in unordered sequence.

This position applies to sequential and indexed disk files.

Columns 68-69 (Number of Extents)

Blank	Single volume file
1-50	Number of volumes in the file

Column 70 (Tape Rewind)

Blank	Rewind information is specified at run time.
R	Rewind this tape at end of file.
U	Unload this tape at end of file.
N	Leave this tape at end of file

Column 70 applies only to magnetic tape files.

Columns 71-72 (File Condition)

Blank	File is to be processed unconditionally.
U1-U8	File is conditioned by the specified external indicator.

These positions apply to output files and primary and secondary input, update, and combined files, excepting table and array input 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.

EXTENSION SPECIFICATIONS – FORM E

Columns 11-18 (From Filename)

Blank	a. Compilation time table or array if columns 33 through 35 contain an entry b. Execution time table or array if columns 33 through 35 are blank
Filename	a. Pre-execution time table or array file b. Record address file named on form F

Columns 19-26 (To Filename)

Blank	This table or array is not output at end of execution.
Filename	a. The output file to which the compilation or pre-execution time table or array is to be written at job termination b. The name of the file for which the from file name record address file controls record selection

Columns 27-32 (Table or Array Name)

Enter the name of a table or array used in the program. Table names start with the three characters TAB; array names must not start with TAB. If alternating tables or arrays are specified on this line, enter here the name of the table or array to which the first element of the input record(s) belongs. Leave these columns blank if this line describes a record address file.

Columns 33-35 (Number of Entries Per Record)

Blank	Record address file or execution time table or array
1-999	Number of entries per record in a compilation time or pre-execution time table or array file

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

Blank	Record address file
1-9999	Maximum number of entries in this table or array

Columns 40-42 (Length of Entry)

Blank	Record address file
1-256	Number of characters in alphanumeric table or array entry
1-15	Number of characters in nonbinary numeric table or array entry
1-9	Number of characters in binary table or array entry

For packed or binary tables and arrays, enter the number of bytes of storage required to represent the data in unpacked format.

Column 43 (Packed or Binary Field)

Blank	a. Record address file b. Compilation time table or array c. Pre-execution or execution time table or array with alphanumeric or unpacked numeric format
P	Pre-execution or execution time table or array with packed numeric format
B	Pre-execution or execution time table or array with binary format

Column 44 (Decimal Positions)

Blank	a. Record address file b. Alphanumeric table or array
0-9	Number of positions to the right of the decimal point in this numeric table or array (zero if entries are integers)

Column 45 (Sequence)

Blank	a. Record address file b. Table or array entries are unordered.
A	Entries have ascending sequence.
D	Entries have descending sequence.

This position describes the sequence of data in a table or array. Column 45 must contain a nonblank entry if high or low look-up is to be used.

Columns 46-57 (Alternate Table or Array)

These positions, corresponding in definitions to columns 27 through 45, describe a table or array that is in alternating format with the table or array named in columns 27 through 32.

Columns 58-74 (Comments)

An entry in these columns is assumed by the compiler to be entirely commentary.

LINE COUNTER SPECIFICATIONS - FORM L

Columns 7-14 (Filename)

Enter the name of a printer file for which you wish to specify a form size and overflow line.

Columns 15-17 (Line Number - Number of Lines Per Page)

12-112	Number of lines available for printing on the printer form

Columns 18-19 (Form Length)

FL	FL indicates that the previous entry is the form length.

Columns 20-22 (Line Number - Overflow Line)

1-112	Number of the overflow line

Columns 23-24 (Overflow Line)

OL	OL indicates that the previous entry is the overflow line number.

INPUT SPECIFICATIONS - FORM I

Entries must be made on this form for every input, update, and combined file used in the program.

Columns 7-14 (Filename)

Blank	a. Field description line b. This line uses the previously listed filename.
Valid RPG II filename	The name of the input, update, or combined file associated with this input specification

Columns 15-16 (Sequence)

Blank	a. Field description line b. AND or OR line
01-99	Check for special sequence of record types. The order of these numbers determines the sequence.
Any two alphabetic characters	Record type sequence is not to be checked. An alphabetic entry is required for a chained file.

Within a file, record types with an alphabetic sequence entry must be described before record types with a numeric sequence entry.

Column 17 (Number)

Blank	Columns 15 and 16 contain alphabetic characters or blanks.
1	Positions 15 and 16 contain numeric characters; only one record of this type is present in each sequenced group.
N	Positions 15 and 16 contain numeric characters; one or more records of this type may be present in the sequenced group.

Column 18 (Option)

Blank	a. Columns 15 and 16 contain alphabetic characters or blanks. b. Columns 15 and 16 are numeric, and this record type is required in the sequence.
0	Columns 15 and 16 are numeric, and this record type is optional in each sequenced group.

Columns 19-20 (Record Identifying Indicator, Look-Ahead Fields, or Spread Cards)

Blank	a. Field description line b. No distinction need be made between this record type and the preceding record type.
01-99	Record identifying indicator
L1-L9	Control level indicator used as a record identifying indicator when record type rather than control field signals start of a new control group.
LR	Last record indicator
H1-H9	Halt indicator used as a record identifying indicator when checking for a record type that causes an error condition
**	Look-ahead fields
TR	Spread card

Columns 21-41 (Record Identification Codes)

This field is divided into three identical subfields:

- Columns 21 through 27
- Columns 28 through 34
- Columns 35 through 41

An AND relationship exists between the subfields.

Columns 21-24 (Position)

Blank	A record identification code is not specified.

1-9999	Character position of the record identification code

Column 25 (Not)

Blank	Either columns 21 through 24 are blank or the character (or zone or digit) must be present to identify the record.
N	The character, zone, or digit must not be present in the position being described.

Column 26 (C/Z/D)

Blank	Record identification code not needed
C	Full character comparison
Z	Zone comparison
D	Digit comparison

Column 27 (Record Identification Character)

Blank or any alphanumeric character	The alphabetic, numeric, or special character that identifies the record. Blank is a valid record identification character.

Additional record identification codes are specified as above in columns 28 through 34 and 35 through 41, as needed. If more than three codes are needed to identify a record, enter AND in columns 14 through 16 of the next line and enter additional codes. Enter OR in columns 14 and 15 of the next line if an OR relationship exists between the codes.

Column 42 (Stacker Select)

Blank	a. Input device is not a card reader b. Cards automatically fall into primary stacker. c. This card device does not have multiple stackers.
1-4	Stacker into which this record type is stacked

This column applies only to input and combined files assigned to card readers.

Column 43 (Packed or Binary Field)

Blank	a. Record description line b. Unpacked decimal or alphanumeric format field
P	Packed decimal format field
B	Binary format field

Columns 44-51 (Field Location)

Enter two one- to four-digit numbers to identify the beginning of a field (From Column) and the end of a field (To Column) in the input record. These entries are identical for a one-position field.

Column 52 (Decimal Positions)

Blank	Alphanumeric field
0-9	The number of decimal positions in the numeric field named in positions 53 through 58 (zero for integers)

Columns 53-58 (Field Name)

These positions name the field specified in columns 43 through 52. The following entries may be made here:

- o A valid RPG II field name
- o The name of an array or array element
- o The special words PAGE, PAGE1, or PAGE2.

Columns 59-60 (Control Level)

Blank	The field being described is not a control field.
L1-L9	The level of the control field being described

These columns must be blank for chained or demand files.

Columns 61-62 (Matching Fields)

Blank	No matching is to be performed on this input field.
M1-M9	Matching level assigned to this input field

Columns 63-64 (Field Record Relation)

Blank	No particular field-record relationship
01-99	Record identifying indicator assigned to a record type
L1-L9	Previously assigned control level indicator
MR	Matching record indicator
U1-U8	Previously assigned external indicator
H1-H9	Previously assigned halt indicator

The following general rules apply to this entry:

- All fields without field record relation should be specified before fields with field record relation.
- All fields with the same field record relation entry should be entered on consecutive lines.
- All parts of a split control field must have the same field record relation entry and must be described on consecutive specification lines.

Columns 65-70 (Field Indicators)

Blank	Field examination is not required.
01-99	Field indicator
H1-H9	Halt indicator (when checking for an error condition in the field data)

When indicators are specified in these positions, RPG II makes appropriate settings when the field is input. When the field is numeric, indicators are specified for positive, negative, and zero conditions in columns 65 and 66, 67 and 68, and 69 and 70, respectively. One, two, or three indicators may be specified, but only the indicator that reflects the result of the test is set on; other specified indicators are set off. An alphanumeric field may only be tested for all blanks (columns 69 through 70).

CALCULATION SPECIFICATIONS - FORM C

Columns 7-8 (Control Level)

Blank	Operation done at detail time
L0	Calculation is performed at total time (L0 is always on).
L1-L9	Calculation operation is done at total time when the appropriate control break occurs or the specified indicator is set on.
LR	Calculation operation is done after the last record has been processed or after LR has been set on.
SR	Calculation operation is part of a subroutine.

Calculation specifications must be ordered by the sequence listed above. AN or OR may be entered in these positions to indicate that indicators (columns 9-17) on the line are in an AND or OR relationship with indicators on the preceding line.

Columns 9-17 (Indicators)

Enter one, two, or three indicators. Any RPG II indicators except 1P and L0 can be used. Columns 9, 12, and 15 may contain blank or N. An AND relationship exists between indicators on a line. Additional lines may be used containing indicators in columns 9 through 17 that are in an AND or OR relationship with those on the first line by entering AND or OR in columns 7 and 8.

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

Factor 1 and Factor 2 may contain the following entries:

- Previously defined field name
- Alphanumeric or numeric literal
- Array element name, or name of a table, array, or subroutine
- Date field name (UDATE, UMONTH, UDAY, UYEAR)
- Special name (PAGE, PAGE1, or PAGE2)
- Label for a TAG, BEGSR, or ENDSR operation (factor 1 only) or a label for a GOTO or EXSR operation (factor 2 only)
- Filename (first six characters) for a CHAIN, DEBUG, DSPLY, READ, SETLL, or FORCE operation (factor 2 only)

Columns 28-32 (Operation)

Enter an operation code (appendix D), left-justified.

Columns 43-48 (Result Field)

Enter the name of the field, table, array, or array element that holds the result of the operation specified in positions 28 through 32. If the field named in Result Field has not been previously defined in extension, input, or preceding calculation specifications, it should be defined now by entries in positions 49 through 52 of this line.

Columns 49-51 (Field Length)

Blank	Result field is defined elsewhere.
1-256	Result field length (right-justified)

The maximum length of a numeric field is 15 digits; the maximum length of an alphanumeric field is 256 characters.

Column 52 (Decimal Positions)

Blank	Result field designated elsewhere or alphanumeric result field
0-9	Number of decimal places in a numeric result field

Column 53 (Half Adjust)

Blank	Do not half adjust the result field.
H	Half adjust the result field.

Half adjusting may be specified only with arithmetic operations.

Columns 54-59 (Resulting Indicators)

Enter any of the following indicators: 01 through 99, H1 through H9, L1 through L9, LR, OA through OG, and OV. Columns 54 and 59 are used:

- To test the value of the result field after an arithmetic operation. Enter up to three indicators to test for a positive result (columns 54 and 55), a negative result (columns 56 and 57), or a result of zero (columns 58 and 59).
- To check the outcome of a CHAIN, LOKUP, COMP, TESTB, or TESTZ operation.

  With the CHAIN operation, an indicator specified in columns 54 and 55 is set on if the record is not found. With the LOKUP operation, up to three indicators are specified to indicate that factor 2 is high (columns 54 and 55), low (columns 56 and 57), or equal (columns 58 and 59). For the COMP operation, up to three indicators are specified to indicate that factor 1 is greater than (columns 54 and 55), less than (columns 56 and 57), or equal to (columns 58 and 59) factor 2.

With the TESTB operation, resulting indicators have the following meanings:

- Positions 54 and 55: An indicator in these positions is turned on if each bit specified in factor 2 is off in the result field.
- Positions 56 and 57: An indicator in these positions is turned on if two or more bits were tested and of mixed status (some bits on and some bits off).
- Positions 58 and 59: An indicator in these positions is turned on if each bit specified in factor 2 is on in the Result Field.

With the TESTZ operation, resulting indicators reflect the zone of the leftmost character in the result field, as follows:

- Positions 54 and 55: Turned on by the zone portion of the characters & and A through I.
- Positions 56 and 57: Turned on by the zone portion of the characters (right bracket), - (minus sign or hyphen), and J through R
- Positions 58-59: Turned on by the zone portion of any other character
- To specify up to three indicators to SETON or SETOF.
- To indicate end-of-file for the READ operation code. An indicator specified in columns 58 and 59 is set on if an end-of-file condition is reached through a READ operation. Once end-of-file is reached, a halt occurs after each read operation if no indicator is entered.

Columns 60-74 (Comments)

The compiler treates entries in these columns as comments and produces them with the program listing.

OUTPUT SPECIFICATIONS - FORM O

Columns 7-14 (Filename)

Enter a valid RPG II filename for each output, update, and combined file used by the program. Each filename need be specified only once, on the first line describing that file.

Columns 14-16 (And/Or Relationships)

Enter AND in positions 14 through 16 or OR in positions 14 and 15 if output records are in an AND or OR relationship.

Column 15 (Type)

Blank	Field description
H	Heading record description
D	Detail record description
T	Total record description
E	Exception record description

Columns 16-18 (Add a Record)

Enter ADD in these columns if the record is to be added to an input, update, or output disk file. An A must also be encoded in position 66 of the File Description Specifications sheet for the file to which a record is added.

Column 16 (Stacker Select/Fetch Overflow)

Blank	a. Use standard overflow routine for this printer file. b. Use the predetermined stacker on this card device. c. This card device does not have multiple stackers. d. This is neither a printer nor a card file. e. Field description line
1-4	Indicates the stacker in which to place records
F	Fetch overflow (printer files only)

Only combined or output files may be stacker selected on form O. Output stacker selection overrides a specification for input stacker selection.

Columns 17-22 (Space/Skip)

If these columns are blank, single spacing occurs after each line is printed. Spacing and skipping are not allowed on the printer keyboard.

Columns 17-18 (Space)

Enter a number (0 through 3) to specify how many lines to skip before (column 17) and after (column 18) a line is printed.

Columns 19-22 (Skip)

Blank	No skipping
01-99	Lines 1-99
A0-A9	Lines 100-109
B0-B2	Lines 110-112

Enter one of the two-character specifications listed above to indicate where the current line will be printed. All lines between are bypassed. A skip before printing is specified in columns 19 and 20, a skip after in columns 21 and 22. If the skip line number is less than the current line number, a skip to a new page occurs.

Columns 23-31 (Output Indicators)

Enter one to three indicators to condition the output record. Any RPG II indicators may be specified. An N may be entered in columns 23, 26, and 29 to indicate negative conditioning. An AND relationship exists between indicators on a line. Additional lines of indicators in an AND or OR relationship may be used by entering AND in columns 14 through 16 or OR in columns 14 and 15 of each additional line.

Columns 32-37 (Field Name)

Enter one of the following to name every field written out:

- Any previously defined field name
- The special words PAGE, PAGE1, PAGE2, *PLACE, UDATE, UDAY, UMONTH, and UYEAR
- A previously defined table name, array name, or array element

These positions must be blank if a constant is entered on positions 45 through 70 of the line. If an entry is made in the Field Name column, positions 7 through 22 must be blank.

Column 38 (Edit Codes)

Enter an edit code to accomplish:

- Suppression of leading zeros for a numeric field
- Omission of a sign from the low-order position of a numeric field
- Punctuation of a numeric field without having to generate an edit word

A table summarizing the edit codes that can be entered in column 38 is found in appendix D.

Column 39 (Blank After)

Blank	a. Record description line b. Field is not reset after being written out.
B	Field is reset to blanks (if alphanumeric) or zeros (if numeric) after being output.

This column must be blank for look-ahead and update fields. If Field Name is a table name, the last table element retrieved is reset.

Columns 40-43 (End Position in Output Record)

Positions 40 through 43 indicate the location in the output record of the field or constant being written out. Enter the number of the position occupied by the rightmost character of the output field. The End Position entry must not be greater than the record length.

Column 44 (Packed or Binary Field)

Blank	a. Record description line b. Field has unpacked numeric or alphanumeric format.
P	Field has packed numeric format.
B	Field has binary format.

Packed and binary fields may be written to tape or disk but should not be printed. Position 44 must be blank with *PLACE fields.

Columns 45-70 (Constant or Edit Word)

Constant

- Field Name (positions 32 through 37) must be blank.
- A constant is enclosed in apostrophes. Enter the leading apostrophe in position 45.
- An apostrophe is represented in a constant by two apostrophes.
- Up to 24 characters of constant information may be placed in one line. Additional lines may be used, but each line must be treated as a separate line of constants. The end position of each line must appear in positions 40 through 43.

Edit Word

Enter any edit word to specify editing of numeric fields. Edit words must be enclosed by apostrophes. Constants are allowed within edit words. Edit words are not used with edit codes. However, when edit codes 1 through 4, A through D, and J through M are used, positions 45 through 47 may contain '*' (to denote asterisk fill) or '$' (to denote a floating dollar sign).

TABLE D-1. OPERATION CODES

Type of Operation	Function of Operation	Operation Code (Columns 28-32)	Indicators	Factor 1	Factor 2	Result Field	Half Adjust	Resulting Indicators
Arithmetic Operations	Add Factor 2 to Factor 1.	ADD	O	R	R	R	O	O
	Clear Result Field and add Factor 2.	Z-ADD	O	B	R	R	O	O
	Subtract Factor 2 from Factor 1.	SUB	O	R	R	R	O	O
	Clear Result Field and subtract Factor 2.	Z-SUB	O	B	R	R	O	O
	Multiply Factor 1 by Factor 2.	MULT	O	R	R	R	O	O
	Divide Factor 1 by Factor 2.	DIV	O	R	R	R	O	O
	Move remainder of preceding division to Result Field.	MVR	O	B	B	R	B	O
	Sum elements of an array and put sum in Result Field.	XFOOT	O	B	R	R	O	O
	Derive the square root of Factor 2.	SQRT	O	B	R	R	B	O
Move Operations	Move Factor 2 into Result Field, right-justified.	MOVE	O	B	R	R	B	O
	Move Factor 2 into Result Field, left-justified.	MOVEL	O	B	R	R	B	B
	Move Factor 2 array into Result Field, left-justified.	MOVEA	O	B	R	R	B	B
Move Zone Operations	Move low zone of Factor 2 to low zone of Result Field.	MLLZO	O	B	R	R	B	B
	Move high zone of Factor 2 to high zone of Result Field.	MHHZO	O	B	R	R	B	B
	Move low zone of Factor 2 to high zone of Result Field.	MLHZO	O	B	R	R	B	B
	Move high zone of Factor 2 to low zone of Result Field.	MHLZO	O	B	R	R	B	B
Compare and Zone Testing Operations	Compare Factor 1 to Factor 2.	COMP	O	R	R	B	B	R
	Identify the zone in the left-most position of the Result Field.	TESTZ	O	B	B	R	B	R

O = Optional
R = Required
B = Blank

TABLE D-1. OPERATION CODES (Continued)

Type of Operation	Function of Operation	Operation Code (Columns 28-32)	Indicators	Factor 1	Factor 2	Result Field	Half Adjust	Resulting Indicators
Binary Field Operations	Set on specified bits.	BITON	O	B	R	R	B	B
	Set off specified bits.	BITOF	O	B	R	R	B	B
	Test specified bits.	TESTB	O	B	R	R	B	R
Setting Indicators	Set one, two, or three specific indicators on.	SETON	O	B	B	B	B	R
	Set one, two, or three specific indicators off.	SETOF	O	B	B	B	B	R
Branching within RPG II	Branch to another RPG II calculation specification line.	GOTO	O	B	R	B	B	B
	Identify the name in Factor 1 as a destination label to which GOTO may branch.	TAG	B	R	B	B	B	B
Lookup Operations	Table lookup.	LOKUP	O	R	R	O	B	R
	Array lookup.	LOKUP	O	R	R	B	B	B
Internal Subroutines	Beginning of the subroutine.	BEGSR	B	R	B	B	B	B
	End of the subroutine.†	ENDSR	B	O	B	B	B	B
	Call to execute the subroutine.	EXSR	O	B	R	B	B	B
External Subroutines	Call to execute the subroutine.	EXIT	O	B	R	B	B	B
	Define argument for external subroutine.	RLABL	B	B	B	R	B	B
	Forcing record to be read next	FORCE	O	B	R	B	B	B
Program Control	Forcing output printing	EXCPT	O	B	B	B	B	B
	A field is printed on the console device and/or data is entered via the console device into a field.	DSPLY	O	O	R	O	B	B
	A record is read from a demand file.	READ	O	B	R	B	B	O
	A record is read from or written to a disk file.	CHAIN	O	R	R	B	B	O
	Sets lower limits for indexed files being processed within limits	SETLL	O	R	R	B	B	B
Debug Function	Aid in finding programming errors	DEBUG	O	O	R	O	B	B

O = Optional
R = Required
B = Blank

† Columns 7 and 8 must have an SR entry for all internal subroutine lines.

TABLE D-1. OPERATION CODES (Contd)

Type of Operation	Function of Operation	Operation Code (Columns 28-32)	Indicators	Factor 1	Factor 2	Result Field	Half Adjust	Resulting Indicators
Debug Function	Aid in finding programming errors	DEBUG	O	O	R	O	B	B
Time	Get time of day	TIME	O	B	B	R	B	B

O = Optional
R = Required
B = Blank

TABLE D-2. SUMMARY OF PROGRAM INDICATORS

Indicator	Where Defined	Where Used	Turned On	Turned Off	Notes
Field Indicators 01-99	Input Specifications sheet	Calculation Specifications sheet, Indicators column. Output Specifications sheet, Output Indicators column	By Blank or Zero (or Plus or Minus) in specified field	Before this field status is to be tested the next time	Turning these indicators on or off can also be accomplished by using the SETON and SETOF operation codes.
H1-H9	Input Specifications sheet	Calculation Specifications sheet, Indicators column; Output Specifications sheet, Output Indicators column	Whenever the specified field status or record identification condition is satisfied	Internally, at the end of the detail cycle	Turning these indicators on or off can also be accomplished by using the SETUP and SETOF operation codes.
LR	Internal	Calculation Specifications sheet, Control Level column; Output Specifications sheet, Output Indicators column	After the last record of the last input file has been processed	At the beginning of object program execution	Turning these indicators on can also be accomplished by using SETON operation codes. All control level indicators (L1-L9) are also turned on when LR is turned on internally.
L0	Internal	Calculation Specifications sheet, Control Level column; Output Specifications sheet, Output Indicators column	At the beginning of object program execution	Never turned off by RPG II	Cannot be SETON or SETOF.
Control level indicators L1-L9	Input Specifications sheet	Calculation Specifications sheet, Control Level or Indicators column; Output Specifications sheet, Output Indicators column	When the value in a control field changes. All lower level indicators are also turned on.	At the end of the following detail cycle	Turning these indicators on or off can also be accomplished by using the SETON and SETOF operation codes.
MR	Internal	Calculation Specifications sheet, Indicators column; Output Specifications sheet, Output Indicators column	When the matching fields of a secondary file match the matching fields of the primary file	When all total calculations and output are completed for the last record of the matching group	
1P	Internal	Output Specifications sheet, Output Indicators column	At the beginning of object program execution	Before the first detail record is read	

TABLE D-2. SUMMARY OF PROGRAM INDICATORS (Contd)

Indicator	Where Defined	Where Used	Turned On	Turned Off	Notes
Resulting Indicators 01-99 or H1-H9	Calculation Specifications sheet	Calculation Specifications sheet, Indicators column; Output Specification sheet, Output Indicators column		For 01-99, the next time the same indicator is a resulting indicator and the specified condition is not met.	Turning these indicators on or off can also be accomplished by using the SETON and SETOF operation codes. Indicators 01-99 and H1-H9 are the commonly used resulting indicators. However, L1-L9, LR, and previously defined indicators (0A-0G and 0V) are also acceptable to the compiler.
With arithmetic operations			By a positive (or negative or zero) balance in field		
Plus					
Minus					
Zero					
With COMP				For H1-H9, internally, at the end of the detail cycle	
High			If Factor 1 > Factor 2;		
Low			If Factor 1 < Factor 2;		
Equal			If Factor 1 = Factor 2		
With LOKUP					
High			If table > Factor 1;		
Low			If table < Factor 1;		
Equal			If table = Factor 1		
With TESTZ			If a C zone or & is present; if a D zone or minus sign is present; if a C or D zone is not present		
High					
Low					
Equal					
With CHAIN (random processing) Columns 54-55			By a no record found condition		
With READ Columns 58-59			By end-of-file on the demand file		
OA-OG, OV	File Description Specifications sheet	Calculation Specifications sheet, Indicators column; Output Specification sheet, Output Indicators column	When the destination of a space, skip, or print operation falls within the overflow area	At the end of the detail cycle	Turning these indicators on or off can also be accomplished by using the SETON and SETOF operation codes.
Record Identifying Indicator 01-99	Input Specifications sheet	Calculation Specification sheet, Indicators column; Output Specification sheet, Output Indicators column	When the specified record has been read and before total calculations are executed	Before the next record is read during the next cycle.	Turning these indicators on or off can also be accomplished by using the SETON and SETOF operation codes.

TABLE D-3. VALID INDICATORS

Indicators	File Description Form		Input Form				Calculations Form			Output Form
	Overflow Indicator (Columns 33-34)	File Conditioning (Columns 71-72)	Record ID Indicator (Columns 19-20)	Control Level (Columns 59-60)	Field Record Relation (Columns 63-64)[1]	Field Indicator (Columns 65-70)	Control Level Indicator (Columns 7-8)	Conditioning Indicator (Columns 9-17)	Resulting Indicator (Columns 54-59)	Conditioning Indicator (Columns 23-31)
01-99			X		X	X		X	X	X
H1-H9			X		X	X		X	X	X
1P										X[2]
MR					X[3]			X		X
OA-OG, OV	X							X	X	X[4]
LO							X			X
L1-L9			X	X	X[3]		X	X	X	X
LR			X				X	X	X	X
U1-U8		X[5]			X			X		X

X = Indicator valid in these columns

1. Invalid for look-ahead fields
2. Valid only for heading and detail lines
3. Invalid for match field or control field specification
4. Cannot condition an exception record, but may condition fields of an exception record
5. Invalid for table input files

TABLE D-4. EDIT CODES

Edit Code	Commas	Decimal Point	Sign for Negative Value			Zero Balance[1]			Zero Suppress
			No Sign	CR	-	Domestic or United Kingdom	Foreign (I)	Foreign (J)	
1	Yes	Yes	x			.00 or 0	,00 or 0	0,00 or 0	Yes
2	Yes	Yes	x			Blanks	Blanks	Blanks	Yes
3	No	Yes	x			.00 or 0	,00 or 0	0,00 or 0	Yes
4	No	Yes	x			Blanks	Blanks	Blanks	Yes
A	Yes	Yes		x		.00 or 0	,00 or 0	0,00 or 0	Yes
B	Yes	Yes		x		Blanks	Blanks	Blanks	Yes
C	No	Yes		x		.00 or 0	,00 or 0	0,00 or 0	Yes
D	No	Yes		x		Blanks	Blanks	Blanks	Yes
J	Yes	Yes			x	.00 or 0	,00 or 0	0,00 or 0	Yes
K	Yes	Yes			x	Blanks	Blanks	Blanks	Yes
L	No	Yes			x	.00 or 0	,00 or 0	0,00 or 0	Yes
M	No	Yes			x	Blanks	Blanks	Blanks	Yes
X[2]	No	No							No
Y	No	No							Yes[3]
Z	No	No							Yes

1. Zero balances for foreign formats are written in two ways, depending upon the entry of I or J in column 21 of the Control Card Specifications sheet.

2. The only editing performed with the X code is the insertion of two blank characters in front of each element of an array.

3. The leftmost zero only is suppressed. The source field can be up to 15 digits. Slash characters are inserted between every two characters. If the length of the field is an odd number of characters, a single digit follows the rightmost slash. For example, source fields having three, four, and five digits have the following edited patterns:

   nn/n
   nn/nn
   nn/nn/n

TABLE D-5. NORMAL COLLATING SEQUENCE AND HEXADECIMAL EQUIVALENTS OF CHARACTERS

Collating Sequence	Character	Hexadecimal Equivalent	Collating Sequence	Character	Hexadecimal Equivalent
1	Blank	40	33	F	C6
2	¢	4A	34	G	C7
3	.	4B	35	H	C8
4		4C	36	I	C9
5	(	4D	37		D0
6	+	4E	38	J	D1
7		4F	39	K	D2
8	&	50	40	L	D3
9		5A	41	M	D4
10	$	5B	42	N	D5
11	*	5C	43	O	D6
12	)	5D	44	P	D7
13	;	5E	45	Q	D8
14		5F	46	R	D9
15	- (minus)	60	47	S	E2
16	/	61	48	T	E3
17	,	6B	49	U	E4
18	%	6C	50	V	E5
19	_ (underscore)	6D	51	W	E6
20		6E	52	X	E7
21	?	6F	53	Y	E8
22	:	7A	54	Z	E9
23	#	7B	55	0	F0
24	@	7C	56	1	F1
25	'	7D	57	2	F2
26	=	7E	58	3	F3
27	"	7F	59	4	F4
28	A	C1	60	5	F5
29	B	C2	61	6	F6
30	C	C3	62	7	F7
31	D	C4	63	8	F8
32	E	C5	64	9	F9

TABLE D-6. PRINTABLE CHARACTERS GROUPED BY EQUAL ZONES

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

TABLE D-6. PRINTABLE CHARACTERS GROUPED BY EQUAL ZONES (Contd)

Group	Character
8	blank 0 1 2 3 4 5 6 7 8 9

TABLE D-7. PRINTABLE CHARACTERS GROUPED BY EQUAL DIGITS

Group	Character
1	blank & - (minus) } 0 (zero)
2	/ A J 1
3	B K S 2
4	C L T 3
5	D M U 4
6	E N V 5
7	F O (letter O) W 6
8	G P X 7

TABLE D-7. PRINTABLE CHARACTERS GROUPED BY EQUAL DIGITS (Contd)

Group	Character
9	H Q Y 8
10	I R Z 9
11	¢ | :
12	. $, (comma) #
13	< * % @
14	( ) _ (underscore) ' (apostrophe)
15	+ ; > =
16	| ¬ ? "

The 1963 American Standard Code for Information Interchange (ASCII) is used by MSOS. ASCII code uses eight bits: bit 8, which is always zero, is omitted in the table below. Bits 1 through 4 contain the low-order four bits of code for the character in that row. Bits 5 through 7 contain the high-order three bits of the code for the character in that column. The code is given in ascending sequence.

TABLE D-8. ASCII CHARACTER SET

ASCII Symbol	Bit Configuration	Hexadecimal Number	Meaning
NULL	000 0000	0	Null/idle
SOM	000 0001	1	Start of message
EOA	000 0010	2	End of address
EOM	000 0011	3	End of message
EOT	000 0100	4	End of transmission
WRU	000 0101	5	Who are you
RU	000 0110	6	Are you
BELL	000 0111	7	Audible signal
FE_0	000 1000	8	Format effector
HT/SK	000 1001	9	Horizontal tab skip (punched card)
LF	000 1010	A	Line feed
V_{TAB}	000 1011	B	Vertical tabulation
FF	000 1100	C	Form feed
CR	000 1101	D	Carriage return
SO	000 1110	E	Shift out
SI	000 1111	F	Shift in
DC_0	001 0000	10	Device control/data link escape
DC_1	001 0001	11	
DC_2	001 0010	12	Device controls
DC_3	001 0011	13	
DC_4 (STOP)	001 0100	14	Device control/stop
ERR	001 0101	15	Error
SYNC	001 0110	16	Synchronous idle
LEM	001 0111	17	Logical end of media
S_0	001 1000	18	
S_1	001 1001	19	
S_2	001 1010	1A	
S_3	001 1011	1B	Information separators
S_4	001 1100	1C	
S_5	001 1101	1D	
S_6	001 1110	1E	
S_7	001 1111	1F	

TABLE D-8. ASCII CHARACTER SET (Contd)

8-Bit ASCII Codes	171x-1 TTY Array	171x-2 TTY Array	026 Punches	029 Punches	6-Bit Ext. BCD Mag Tape	8-Bit ASCII Codes	171x-1 TTY Array	172x-2 TTY Array	026 Punches	029 Punches	6-Bit Ext. BCD Mag Tape
20_{16}	Space	Space	No Punch	No Punch	20_8	40_{16}	@	@	0-8-7	8-4	37
21			11-8-2	12-8-7	52	41	A	A	12-1	12-1	61
22	"	"	8-7	8-7	17	42	B	B	12-2	12-2	62
23	#	#	12-8-7	8-3	77	43	C	C	12-3	12-3	63
24	$	$	11-8-3	11-8-3	53	44	D	D	12-4	12-4	64
25	%	%	0-8-5	0-8-4	35	45	E	E	12-5	12-5	65
26	&	&	8-2	12	00 (35)	46	F	F	12-6	12-6	66
27	'	'	8-4	8-5	14	47	G	G	12-7	12-7	67
28	(	(	0-8-4	12-8-5	34	48	H	H	12-8	12-8	70
29	)	)	12-8-4	11-8-5	74	49	I	I	12-9	12-9	71
2A	*	*	11-8-4	11-8-4	54	4A	J	J	11-1	11-1	41
2B	+	+	12	12-8-6	60	4B	K	K	11-2	11-2	42
2C	,	,	0-8-3	0-8-3	33	4C	L	L	11-3	11-3	43
2D	-	-	11	11	40	4D	M	M	11-4	11-4	44
2E	.	.	12-8-3	12-8-3	73	4E	N	N	11-5	11-5	45
2F	/	/	0-1	0-1	21	4F	O	O	11-6	11-6	46
30	0	0	0	0	12	50	P	P	11-7	11-7	47
31	1	1	1	1	01	51	Q	Q	11-8	11-8	50
32	2	2	2	2	02	52	R	R	11-9	11-9	51
33	3	3	3	3	03	53	S	S	0-2	0-2	22
34	4	4	4	4	04	54	T	T	0-3	0-3	23
35	5	5	5	5	05	55	U	U	0-4	0-4	24
36	6	6	6	6	06	56	V	V	0-5	0-5	25
37	7	7	7	7	07	57	W	W	0-6	0-6	26
38	8	8	8	8	10	58	X	X	0-7	0-7	27
39	9	9	9	9	11	59	Y	Y	0-8	0-8	30
3A	:	:	8-5	8-2	15	5A	Z	Z	0-9	0-9	31
3B	;	;	11-8-6	11-8-6	56	5B			12-8-5	12-8-2	75
3C			12-8-6	12-8-4	76	5C			0-8-2	0-8-2	36
3D	=	=	8-3	8-6	13	5D			11-8-5	11-8-2	55
3E			8-6	0-8-6	16	5E			11-8-7	11-8-7	57
3F	?	?	12-8-2	0-8-7	72	5F			0-8-6	0-8-5	32

Refer to note 2.
Refer to note 4.

1. The 171x-2 teletypewriter (TTY) array is the ASCII 68, 64 character subset. This array is the same as used on the 171x-3 devices which receive from a 1774.
2. To operate in 026 punched card mode, ASCII 63 options are selected. To operate in 029 punched card mode, ASCII 68 options are selected. These options are assembly-time options for each driver affected.
3. The CDC Standard 1.10.003 is supported by an assembly option. For CDC ASCII mode of operation, the card punches 12-8-2 and 12-0 are stored internally as 7B. The card punches 11-8-2 and 11-0 are stored internally as 7D. For line printer operations, the internal codes 7B and 7D are converted to 5B and 5D to allow printing the hardware compatible graphic characters [(left bracket) and] (right bracket).
4. Since 173x magnetic tape controllers do not provide any code conversion, BCD code 00 is illegal and causes a noise record or BCD code 35 is substituted for the illegal 00 code to prevent tape errors.

   On tape write operations the ASCII codes 25_{16} (%) and 26_{16} (&) are written as BCD 35_8.

   On tape read operations the BCD code 35_8 is always translated to an ASCII $25 (%).

TABLE D-9. DEVICE CODES, RECORD LENGTHS, AND BLOCK LENGTHS

Device Mnemonic	Logical Unit	Maximum Record Length	Maximum Block Length
DISK	8	8191	9999
CRT77	4	80	80
TAPE0	6	8191	9999
TAPE1	16	8191	9999
TAPE2	17	8191	9999
PRINT84	12	131	131
READ01	10	80	80
MFCM1	10	80	80
MFCM2	10	80	80
DISK40	8	8191	9999
DISK45	8	8191	9999
MFCU1	10	80	80
MFCU2	10	80	80
PRINTER	12	131	131
PRINTR2	12	131	131
CONSOLE	4	80	80
TAPE	6	8191	9999
READER	10	80	80
READ42	10	80	80
PUNCH	11	80	80
SPECIAL		8191	9999
BLANK	†	8191	9999

†In batch mode the standard input/output unit is used. In terminal mode the user terminal is employed.

TABLE D-10. ASCII TO EBCDIC TRANSLATION TABLE AND ALTERNATE COLLATING SEQUENCE CODING SHEET (CYBER 18 GRAPHICS)

Code	CYBER 18 Graphic	ASCII Entry	Replaced By/Takes Place of	Code	CYBER 18 Graphic	ASCII Entry	Replaced By/Takes Place of
00000000	Null	00		00111000	8	38	F8
00000001	SDM	01		00111001	9	39	F9
00000010	STX	02		00111010	:	3A	7A
00000011	ETX	03		00111011	;	3B	5E
00000100	EOT	04		00111100	<	3C	4C
00000101	ENQ	05		00111101	=	3D	7E
00000110	ACK	06		00111110	>	3E	6E
00000111	BEL	07		00111111	?	3F	6F
00001000	BS	08		01000000	@	40	7C
00001001	HT	09		01000001	A	41	C1
00001010	LF	0A		01000010	B	42	C2
00001011	VT	0B		01000011	C	43	C3
00001100	FF	0C		01000100	D	44	C4
00001101	CR	0D		01000101	E	45	C5
00001110	SO	0E		01000110	F	46	C6
00001111	SI	0F		01000111	G	47	C7
00010000	DLE	10		01001000	H	48	C8
00010001	DC1	11		01001001	I	49	C9
00010010	DC2	12		01001010	J	4A	D1
00010011	DC3	13		01001011	K	4B	D2
00010100	DC4	14		01001100	L	4C	D3
00010101	ANK	15		01001101	M	4D	D4
00010110	SYN	16		01001110	N	4E	D5
00010111	ETB	17		01001111	O	4F	D6
00011000	CAN	18		01010000	P	50	D7
00011001	EM	19		01010001	Q	51	D8
00011010	SUB	1A		01010010	R	52	D9
00011011	ESC	1B		01010011	S	53	E2
00011100	FS	1C		01010100	T	54	E3
00011101	GS	1D		01010101	U	55	E4
00011110	RS	1E		01010110	V	56	E5
00011111	US	1F		01010111	W	57	E6
00100000	Blank	20	40	01011000	X	58	E7
00100001	!	21	5A	01011001	Y	59	E8
00100010	"	22	7F	01011010	Z	5A	E9
00100011	#	23	7B	01011011	[¢	5B	4A
00100100	$	24	5B	01011100	\ ¬	5C	5F
00100101	%	25	6C	01011101	]	5D	27
00100110	&	26	5O	01011110	^	5E	4F
00100111	'	27	7D	01011111	—	5F	6D
00101000	(	28	4D	01100000	`	60	2D
00101001	)	29	5D	01100001	a	61	2F
00101010	*	2A	5C	01100010	b	62	
00101011	+	2B	4E	01100011	c	63	
00101100	,	2C	6B	01100100	d	64	
00101101	-	2D	6O	01100101	e	65	
00101110	.	2E	4B	01100110	f	66	
00101111	/	2F	61	01100111	g	67	
00110000	0	30	F0	01101000	h	68	
00110001	1	31	F1	01101001	i	69	
00110010	2	32	F2	01101010	j	6A	
00110011	3	33	F3	01101011	k	6B	2C
00110100	4	34	F4	01101100	l	6C	25
00110101	5	35	F5	01101101	m	6D	3B
00110110	6	36	F6	01101110	n	6E	3E
00110111	7	37	F7	01101111	o	6F	3F

NOTE: Blank entries indicate that the code is not changed.

TABLE D-10. ASCII TO EBCDIC TRANSLATION TABLE AND ALTERNATE COLLATING SEQUENCE CODING SHEET (CYBER 18 GRAPHICS) (Contd)

Code	CYBER 18 Graphic	ASCII Entry	Replaced By/Takes Place of	Code	CYBER 18 Graphic	ASCII Entry	Replaced By/Takes Place of
01110000	p	70		10101001		A9	
01110001	q	71		10101010		AA	
01110010	r	72		10101011		AB	
01110011	s	73		10101100		AC	
01110100	t	74		10101101		AD	
01110101	u	75		10101110		AE	
01110110	v	76		10101111		AF	
01110111	w	77		10110000		B0	
01111000	x	78		10110001		B1	
01111001	y	79		10110010		B2	
01111010	z	7A	3A	10110011		B3	
01111011	⊕	7B	C0	10110100		B4	
01111100	:	7C	20	10110101		B5	
01111101	⊖	7D	D0	10110110		B6	
01111110	~	7E	3D	10110111		B7	
01111111	Delete	7F	22	10111000		B8	
10000000		80		10111001		B9	
10000001		81		10111010		BA	
10000010		82		10111011		BB	
10000011		83		10111100		BC	
10000100		84		10111101		BD	
10000101		85		10111110		BE	
10000110		86		10111111		BF	
10000111		87		11000000		C0	23
10001000		88		11000001		C1	41
10001001		89		11000010		C2	42
10001010		8A		11000011		C3	43
10001011		8B		11000100		C4	44
10001100		8C		11000101		C5	45
10001101		8D		11000110		C6	46
10001110		8E		11000111		C7	47
10001111		8F		11001000		C8	48
10010000		90		11001001		C9	49
10010001		91		11001010		CA	
10010010		92		11001011		CB	
10010011		93		11001100		CC	
10010100		94		11001101		CD	
10010101		95		11001110		CE	
10010110		96		11001111		CF	
10010111		97		11010000		D0	29
10011000		98		11010001		D1	24
10011001		99		11010010		D2	2E
10011010		9A		11010011		D3	3C
10011011		9B		11010100		D4	28
10011100		9C		11010101		D5	2B
10011101		9D		11010110		D6	2A
10011110		9E		11010111		D7	26
10011111		9F		11011000		D8	51
10100000		A0		11011001		D9	52
10100001		A1		11011010		DA	
10100010		A2		11011011		DB	
10100011		A3		11011100		DC	
10100100		A4		11011101		DD	
10100101		A5		11011110		DE	
10100110		A6		11011111		DF	
10100111		A7		11100000		E0	
10101000		A8		11100001		E1	

NOTE: Blank entries indicate that the code is not changed.

TABLE D-10. ASCII TO EBCDIC TRANSLATION TABLE AND ALTERNATE COLLATING SEQUENCE CODING SHEET (CYBER 18 GRAPHICS) (Contd)

Code	CYBER 18 Graphic	ASCII Entry	Replaced By/Takes Place of	Code	CYBER 18 Graphic	ASCII Entry	Replaced By/Takes Place of
11100010		E2	53	11110001		F1	31
11100011		E3	54	11110010		F2	32
11100100		E4	55	11110011		F3	33
11100101		E5	56	11110100		F4	34
11100110		E6	57	11110101		F5	35
11100111		E7	58	11101110		F6	36
11101000		E8	59	11110111		F7	37
11101001		E9	21	11111000		F8	38
11101010		EA		11111001		F9	39
11101011		EB		11111010		FA	
11101100		EC		11111011		FB	
11101101		ED		11111100		FC	
11101110		EE		11111101		FC	
11101111		EF		11111110		FE	
11110000		F0	30	11111111		FF	

NOTE: Blank entries indicate that the code is not changed.

TABLE D-11. EBCDIC TO ASCII TRANSLATION TABLE AND ALTERNATE COLLATING SEQUENCE CODING SHEET (SYSTEM/3 GRAPHICS)

Code	System/3 Graphic	Entry	Replaced By/Takes Place of	Code	System/3 Graphic	Entry	Replaced By/Takes Place of
00000000		00		00011011		1B	
00000001		01		00011100		1C	
00000010		02		00011101		1D	
00000011		03		00011110		1E	
00000100		04		00011111		1F	
00000101		05		00100000		20	7C
00000110		06		00100001		21	E9
00000111		07		00100010		22	7F
00001000		08		00100011		23	C0
00001001		09		00100100		24	D1
00001010		0A		00100101		25	6C
00001011		0B		00100110		26	D7
00001100		0C		00100111		27	5D
00001101		0D		00101000		28	D4
00001110		0E		00101001		29	D0
00001111		0F		00101010		2A	D6
00010000		10		00101011		2B	D5
00010001		11		00101100		2C	6B
00010010		12		00101101		2D	60
00010011		13		00101110		2E	D2
00010100		14		00101111		2F	61
00010101		15		00110000		30	F0
00010110		16		00110001		31	F1
00010111		17		00110010		32	F2
00011000		18		00110011		33	F3
00011001		19		00110100		34	F4
00011010		1A		00110101		35	F5

NOTE: Blank entries indicate that the code is not changed.

TABLE D-11. EBCDIC TO ASCII TRANSLATION TABLE AND ALTERNATE COLLATING SEQUENCE CODING SHEET (SYSTEM/3 GRAPHICS) (Contd)

Code	System/3 Graphic	Entry	Replaced By/Takes Place of	Code	System/3 Graphic	Entry	Replaced By/Takes Place of
00110110		36	F6	01101101	_	6D	5F
00110111		37	F7	01101110	>	6E	3E
00111000		38	F8	01101111	?	6F	3F
00111001		39	F9	01110000		70	
00111010		3A	7A	01110001		71	
00111011		3B	6D	01110010		72	
00111100		3C	D3	01110011		73	
00111101		3D	7E	01110100		74	
00111110		3E	6E	01110101		75	
00111111		3F	6F	01110110		76	
01000000	Blank	40	20	01110111		77	
01000001		41	C1	01111000		78	
01000010		42	C2	01111001		79	
01000011		43	C3	01111010	:	7A	3A
01000100		44	C4	01111011	#	7B	23
01000101		45	C5	01111100	@	7C	40
01000110		46	C6	01111101	'	7D	27
01000111		47	C7	01111110	=	7E	3D
01001000		48	C8	01111111	"	7F	22
01001001		49	C9	10000000		80	80
01001010	¢	4A	5B	10000000		81	
01001011	.	4B	2E	10000010		82	
01001100	<	4C	3C	10000011		83	
01001101	(	4D	28	10000100		84	
01001110	+	4E	2B	10000101		85	
01001111	!	4F	5E	10000110		86	
01010000	&	50	26	10000111		87	
01010001	¬	51	D8	10001000		88	
01010010		52	D9	10001001		89	
01010011		53	E2	10001010		8A	
01010100		54	E3	10001011		8B	
01010101		55	E4	10001100		8C	
01010110		56	E5	10001101		8D	
01010111		57	E6	10001110		8E	
01011000		58	E7	10001111		8F	
01011001		59	E8	10010000		90	
01011010	!	5A	21	10010001		91	
01011011	$	5B	24	10010010		92	
01011100	.	5C	2A	10010011		93	
01011101	)	5D	29	10010100		94	
01011110	;	5E	3B	10010101		95	
01011111		5F	5C	10010110		96	
01100000		60	2D	10010111		97	
01100001	/	61	2F	10011000		98	
01100010		62		10011001		99	
01100011		63		10011010		9A	
01100100		64		10011011		9B	
01100101		65		10011100		9C	
01100110		66		10011101		9D	
01100111		67		10011110		9E	
01101000		68		10011111		9F	
01101001		69		10100000		A0	
01101010		6A		10100001		A1	
01101011		6B	2C	10100010		A2	
01101100	%	6C	25	10100011		A3	

NOTE: Blank entries indicate that the code is not changed.

TABLE D-11. EBCDIC TO ASCII TRANSLATION TABLE AND ALTERNATE COLLATING SEQUENCE CODING SHEET (SYSTEM/3 GRAPHICS) (Contd)

Code	System/3 Graphic	Entry	Replaced By/Takes Place of	Code	System/3 Graphic	Entry	Replaced By/Takes Place of
10100100		A4		11010010	K	D2	4B
10100101		A5		11010011	L	D3	4C
10100110		A6		11010100	M	D4	4D
10100111		A7		11010101	N	D5	4E
10101000		A8		11010110	O	D6	4F
10101001		A9		11010111	P	D7	50
10101010		AA		11011000	Q	D8	51
10101011		AB		11011001	R	D9	52
10101100		AC		11011010		DA	
10101101		AD		11011011		DB	
10101110		AE		11011100		DC	
10101111		AF		11011101		DD	
10110000		B0		11011110		DE	
10110001		B1		11011111		DF	
10110010		B2		11100000		E0	
10110011		B3		11100001		E1	
10110100		B4		11100010	S	E2	53
10110101		B5		11100011	T	E3	54
10110110		B6		11100100	U	E4	55
10110111		B7		11100101	V	E5	56
10111000		B8		11100110	W	E6	57
10111001		B9		11100111	X	E7	58
10111010		BA		11101000	Y	E8	59
10111011		BB		11101001	Z	E9	5A
10111100		BC		11101010		EA	
10111101		BD		11101011		EB	
10111110		BE		11101100		EC	
10111111		BF		11101101		ED	
11000000	⊖	C0	7B	11101110		EE	
11000001	A	C1	41	11101111		EF	
11000010	B	C2	42	11110000	0	F0	30
11000011	C	C3	43	11110001	1	F1	31
11000100	D	C4	44	11110010	2	F2	32
11000101	E	C5	45	11110011	3	F3	33
11000110	F	C6	46	11110100	4	F4	34
11000111	G	C7	47	11110101	5	F5	35
11001000	H	C8	48	11110110	6	F6	36
11001001	I	C9	49	11110111	7	F7	37
11001010		CA	C	11111000	8	F8	38
11001011		CB		11111001	9	F9	39
11001100		CC		11111010		FA	
11001101		CD		11111011		FB	
11001110		CE		11111100		FC	
11001110		CF		11111101		FD	
11010000	{ ⊖	D0	7D	11111110		FE	
11010001	J	D1	4A	11111111		FF	

NOTE: Blank entries indicate that the code is not changed.

COMPILATION ERRORS

These messages are output during compilation of RPG II source programs with the source listing. The error message follows the source statement that contained the error and is of the form:

******** ERROR nnnnx

Where:

nnnn	is	the compilation error number.
x	is	the specifications form type H, F, E, L, I, C, O, or A. (A is for arrays.)

Error numbers of 0100 and above are not recoverable and object code is not generated. Error numbers are grouped by form type in tables E-1 through E-5.

TABLE E-1. CONTROL CARD SPECIFICATION ERRORS

Error Message Number	Error	Definition	Action
0001H	Illegal entry in columns 7-14	Entry must be blank.	No special action taken
0002H	Illegal Debug entry in column 15	Entry must be blank.	Blank assumed
0003H	Illegal entry in columns 16-20	Entry must be blank.	No special action taken
0004H	Illegal Inverted Print entry in column 21	Entry must be D, I, J, or blank.	Blank assumed
0005H	Illegal entry in columns 22-25	Entry must be blank.	No special action taken
0006H	Illegal Alternate Collating Sequence entry in column 26	Entry must be S or blank.	S (alternate collating set) assumed
0007H	Illegal entry in columns 27-39	Entry must be blank.	No special action taken
0008H	Illegal Sign Handling entry in column 40	Entry must be blank.	Blank assumed
0009H	Illegal 1P Forms Position Entry in column 41	Entry must be 1 or blank.	1 (forms positioning) assumed
0010H	Illegal Indicator Setting entry in column 42	Entry must be blank.	Blank assumed
0011H	Illegal File Translation entry in column 43	Entry must be F or blank.	F (file translation) assumed
0012H	Illegal entry in columns 44-49	Entry must be blank.	No special action taken
0013H	Illegal Formatted Dump entry in column 50	Entry must be blank.	Blank assumed
0014H	Illegal internal code entry in column 51	Entry must be E or blank.	Blank assumed
0015H	Illegal entry in columns 52-73	Entry must be blank.	No special action taken
0016H	Illegal Segmented Running entry in column 74	Entry must be 1 or blank.	Blank assumed
0017H	This entry in column 52 is invalid	Entry must be 1 or 6.	Blank assumed
0018H	Invalid entry in column 49	Entry must be blank.	Blank assumed
0019H	Invalid entry in column 48	Entry must be blank or 1.	Blank assumed
0020H	Invalid entry in column 52	Entry must be blank for this configuration	Blank assumed

TABLE E-1. CONTROL CARD SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
0100H	Abnormal end-of-file reached		Compiler exited
0101H	Illegal Form Type entry in column 6	Entry must be H or F.	Source line (record) bypassed; processing continues with next line.
0102H	Invalid Form Type entry of H in column 6	Header card encountered more than once.	Previous data ignored; processing continues with new data.
0103H	Illegal JCB string		
0001F	Illegal File Type entry in column 15	Entry must be D, C, U, O, or I.	I assumed
0002F	Illegal File Designation entry in column 16	Entry must be D, T, R, C, S, P, or blank.	Blank assumed
0003F	Illegal Sequence entry in column 18	Entry must be A, D, or blank.	Blank assumed
0004F	Not used		
0005F	Illegal Extension Code entry in column 39	Entry must be E, L, or blank.	Blank assumed
0006F	Illegal Filename entry in columns 7-14	Invalid character encountered in name or name included on embedded blank	Character accepted as part of file name
0007F	Illegal File Format entry in column 19	Entry must be F, V, or D.	F assumed
0008F	Invalid File Designation entry of P in column 16	Entry encountered more than once; primary file already defined	Secondary file assumed
0100F	Abnormal end-of-file reached	/* or ** encountered	Compiler exited
0101F	Illegal Form Type entry in column 6	Entry must be F, E, L, or I.	Source line bypassed; processing continues with next line.
0102F	Logical unit number in columns 50-52 is invalid.	Entry is not a numeric value less than 256.	Blank entry assumed
0103F	Illegal Filename entry in columns 7-14	Blank entry	A blank filename assumed
0104F	Invalid Filename entry in columns 7-14	Filename encountered more than once	No special action taken
0105F	Illegal Record Length entry in columns 24-27	Entry must be a numeric entry greater than zero	Record length of one assumed
0106F	Illegal Record Address Type entry in column 31	Entry must be A, P, I, or blank.	Blank assumed
0107F	Illegal End of File entry in column 17	Entry must be E or blank.	Blank assumed

TABLE E-1. CONTROL CARD SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
0109F	Illegal Block Length entry in columns 20-23	Entry must be a valid numeric entry or blank.	Blank assumed
0110F	Illegal Mode of Processing entry in column 28	Entry must be L, R, or blank.	Blank assumed
0111F	Illegal Length of Key Field or Record Address Field entry in columns 29-30	Entry must be a valid numeric entry or blank.	Blank assumed
0112F	Illegal Type of File Organization entry in column 32	Entry must be I, T, 1-9, or blank.	Blank assumed
0113F	Illegal Overflow Indicator in columns 33-34	Entry must be OA-OG, OV, or blanks.	OA assumed
0114F	Illegal Key Field Starting Location entry in columns 35-38	Entry must be a valid numeric entry or blank.	Blank assumed
0115F	Illegal Device entry in columns 40-46	Entry must be a valid device name.	No device assigned to file
0116F	Illegal Symbolic Device entry in columns 47-52	Entry must be a blank.	Blank assumed
0117F	Illegal Labels entry in column 53	Entry must be blank.	Blank assumed
0118F	Illegal File Condition entry in columns 71-72	Entry must be U1-U8 or blank.	U1 assumed
0119F	Illegal entry in column 67	Entry must be blank.	Blank assumed
0120F	K entry in column 53 invalid with entry of blanks in columns 7-52	Continuation line requires an entry of K.	K assumed; line processed as a continuation line.
0121F	Illegal Option entry in columns 54-59 of a continuation line	Entry must be a valid option name (ASCII, BUFOFF, EBCDIC, or table/array name).	Source line bypassed; processing continues with next line.
0122F	Illegal File Addition/Unordered entry in column 66	Entry must be A, U, or blank.	Blank assumed
0123F	Illegal Tape Rewind entry in column 70	Entry must be R, U, N, or blank.	Blank assumed
0124F	Entry in columns 60-65 invalid with Option entry of BUFOFF in columns 54-59	Entry must be numeric.	Zero assumed
0125F	Invalid Sequence entry in column 18	Different entry encountered previously	Previous entry used
0126F	No File Designation entry of P in column 16 encountered	No primary input file specified.	No special action taken
0127F	File Type entry of I, U, or C in column 15 invalid with File Designation entry of blank in column 16	Only output and display files do not require file designation.	No special action taken

TABLE E-1. CONTROL CARD SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
0128F	File Type entry of O or D in column 15 invalid with File Designation entry of D in column 16.	Only input, update, or combined files can be demand files.	No special action taken
0129F	File Type entry of O or D in column 15 invalid with File Designation entry of T in column 16.	Only input, update, or combined files can be table/array files.	No special action taken
0130F	File Type entry of O, U, C, or D in column 15 invalid with File Designation entry of R in column 16.	Only input files can be record address files.	No special action taken
0131F	File Type entry of C or D in column 15 invalid with File Designation entry of C in column 16.	Only input, output or update files can be chained files.	No special action taken
0132F	File Type entry of O or D in column 15 invalid with File Designation entry of S in column 16	Only input, update, or combined files can be secondary files.	No special action taken
0133F	File Type entry of O or D in column 15 invalid with File Designation entry of P in column 16	Only input, update, or combined files can be primary files.	No special action taken
0135F	File Type entry of O or D in column 15 invalid with End of File entry of E in column 17	Only input, update or combined files permit end-of-file designation.	No special action taken
0136F	File Designation entry of C, T, D, or blank in column 16 invalid with End of File entry of E in column 17	Only record address, primary, or secondary files permit end-of-file designation.	No special action taken
0137F	File Type entry of O or D in column 15 invalid with Sequence entry of A or D in column 18	Only input, update, or combined files permit sequence specification.	No special action taken
0140F	Block Length entry in columns 20-23 invalid with File Format entry of F in column 19 and Record Length entry in columns 24-27	Block Length must be record length or a multiple for fixed length records.	No special action taken
0141F	Block Length entry in columns 20-23 invalid with File Format entry of V in column 19 and Record Length entry in columns 24-27	Block Length must be at least eight bytes larger than record length for variable length records.	No special action taken
0144F	Record Address Type entry of A, P, or blank in column 31 invalid with File Organization entry of T in column 32	ADDROUT file organization requires an I in column 31.	No special action taken

TABLE E-1. CONTROL CARD SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
0145F	Mode of Processing entry of R or L in column 28 invalid with File Organization entry of T in column 32	ADDROUT file organization cannot be processed randomly or within limits.	No special action taken
0146F	Record Address Type Entry of A or P in column 31 invalid with File Organization entry of 1-9 or blank in column 32	Record address type processing illegal for sequential file	No special action taken
0147F	Mode of Processing entry of L in column 28 invalid with Type of File Organization entry of 1-9 or blank in column 32	Limit processing illegal for sequential or direct file	No special action taken
0149F	Record Address Type entry of blank in column 31 invalid with File Organization entry of I in column 32	Indexed sequential file must have alphanumeric or packed keys.	No special action taken
0151F	Record Address Type entry of A or P in column 31 invalid with direct file access (R in column 28 and blank or 1-9 in column 32)	Direct files must be retrieved by key or identification.	No special action taken
0152F	Invalid Length of Key Field or Record Address Field entry in columns 29-30	Columns 29-30 entry illegal with a non-record address file that does not use keys	No special action taken
0153F	When device is not a printer, Extension Code entry of E or blank in column 39 invalid with Overflow Indicator entry of OA-OG or OV in columns 33-34	Overflow indicator can only be defined for a line counter file.	No special action taken
0154F	Invalid Key Field Starting Location entry in columns 35-38	Location specification illegal with non-record address files not using keys	No special action taken
0155F	File Format entry of V or D in column 19 invalid with File Designation entry of T in column 16	Table files must be fixed format.	No special action taken
0156F	Display File (column 15) illegal with device specified in columns 40-46	Display files must use a console or similar device.	No special action taken
0157F	Combined File (column 15) illegal with device specified in columns 40-46	Combined files must use a card reader/punch device.	No special action taken
0158F	Update File (column 15) illegal with device specified in columns 40-46	Update file must use a disk device.	No special action taken
0159F	Output File (column 15) illegal with device specified in columns 40-46	Output files must use a device capable of output.	No special action taken

TABLE E-1. CONTROL CARD SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
0160F	Input File (column 15) illegal with device specified in columns 40-46	Input files must use a device capable of input.	No special action taken
0161F	For device specified (columns 40-46), file must be sequential (column 31).	For device specified, column 31 must be blank.	No special action taken
0162F	For device specified (columns 40-46), records must be fixed length (column 19).	Entry in column 19 must be F for device specified.	No special action taken
0163F	For device specified (columns 40-46), multivolume files (columns 68-69) not allowed	Number of extents must be blank for device specified.	Blank number of extents assumed
0164F	Block Length (columns 20-23) exceeds maximum record length for device specified in columns 40-46.	Block length must be less than or equal to maximum device length.	Maximum device length assumed
0165F	Record Length (columns 24-27) exceeds maximum record length for device specified in columns 40-46.	Record length must be less than or equal to maximum device length.	Maximum device length assumed
0170F	Core Index Entry in columns 60-65 invalid with File Organization entry of T, 1-9, or blank in column 32 and with Mode of Processing entry of L or blank in column 28	Core index legal only with indexed sequential files processed randomly.	No special action taken
0171F	File Addition entry of A in column 66 invalid with File Type entry of C or D in column 15	File addition legal only with input, output, and update files	No special action taken
0173F	Device entry in columns 40-46 invalid with Tape Rewind entry in column 70	Tape rewind can only be specified with tape files.	No special action taken
0174F	Illegal Filename entry in columns 7-14	First character in column 7 cannot be numeric or blank.	No special action taken
0178F	Illegal number of extents entry in columns 68-69	Entry must be numeric or blank.	Blanks assumed
0179F	Illegal entry in columns 73-74	Entry must be blank.	No special action taken
0180F	File Condition entry of U1-U8 in columns 71-72 invalid with File Type entry of D in column 15	File conditioning indicator illegal with display file	No special action taken
0181F	Extension Code entry of E in column 39 invalid with File Designation of D in column 16	Extension specifications not allowed with display file	No special action taken
0182F	Extension Code entry of E in column 39 invalid with Device entry of CONSOLE in columns 40-46	Extension specifications not allowed with console file	No special action taken

TABLE E-1. CONTROL CARD SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
0183F	Extension Code entry of E in column 39 invalid with File Organization entry of I in column 32	Extension specifications not allowed with indexed file	No special action taken
0184F	Extension Code entry of L in column 39 invalid with Device entry in columns 40-46	Line counter specifications not allowed with specified device	No special action taken
0185F	Sequence entry of A or D in column 18 invalid with File Designation entry of T, C, R, or D in column 16	Sequence specification illegal with demand or table file	No special action taken
0186F	File specification line with File Format entry of D in column 19 not followed by continuation line with Option entry of BUFOFF in columns 54-59	D-format file requires BUFOFF continuation specification.	No special action taken
0187F	File Format entry of F or V in column 19 invalid with Option entry of BUFOFF in columns 54-59	BUFOFF legal only with D-type records	No special action taken
0188F	Invalid Storage Index Entry entry in columns 60-65	Entry was not a valid numeric entry or blank.	Blank assumed (no core index)
0189F	Chained file (C in column 16) must be processed randomly (R in column 28)		No special action taken
0190F	Key starting position greater than record length		No special action taken
0191F	Additional I/O areas are not allowed within shared I/O	Column 32 must be I, T, or blank when shared I/O is specified.	No special action taken
0192F	Multivolume files may not be used with shared I/O	Columns 68-69 must be blank or 01 when shared I/O is used.	No special action taken
0913F	Randomly processed primary file must be processed by ADDROUT		No special action taken

TABLE E-2. EXTENSION SPECIFICATION ERRORS

Error Message Number	Error	Definition	Action
0001E	Chaining entries in columns 7-10 must be blank.		
0002E	From Filename entry in columns 11-18 encountered on file specifications; however, file specification did not contain an Extension Code entry of E in column 39.		No special action taken
0003E	File specification processed with an Extension Code of E in column 39. This requires extension specifications; however, no extension specification records with a Form Type entry of E in column 6 are encountered for the file. File name follows.		No special action taken
0100E	Abnormal end-of-file reached		Compiler exited.
0101E	Illegal Form Type entry in column 6	Card out of sequence	Source line bypassed; processing continues with next line.
0102E	Illegal Table or Array Name entry in columns 27-32	Blank entry	Zero length name assumed; processing continues.
0103E	Illegal Length of Entry in columns 40-42	Entry must be 1-256 for alpha, 1-15 for numeric, 1-14 binary.	Length of 1 assumed.
0104E	Illegal Packed/Binary entry in column 43.	Entry must be P, B, L, R, or blank	Blank assumed
0105E	Illegal Decimal Positions entry in column 44	Entry must be 0-9 or blank.	Zero assumed
0106E	Illegal Sequence (A/D) entry in column 45	Entry must be A, D, or blank.	Blank assumed.
0107E	Illegal Length of Entry entry in columns 52-54	Entry must be 1-256.	Length of 1 assumed
0108E	Illegal Packed/Binary entry in column 55	Entry must be P, B, L, R, or blank.	Blank assumed
0109E	Illegal Decimal Positions entry in column 56	Entry must be 0-9 or blank.	Zero assumed
0110E	Illegal Sequence (A/D) entry in column 57	Entry must be A, D, or blank.	Blank assumed

TABLE E-2. EXTENSION SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
0111E	Illegal Number of Entries per Record entry in columns 33-35	Entry must be 1-999 or blank.	1 assumed
0112E	Illegal Number of Entries per Table or Array entry in columns 36-39	Entry must be 1-9999.	1 assumed
0113E	Invalid To Filename entry in columns 19-26	Filename did not appear on file specifications.	Since no valid information is available for processing file name, multiple related errors can occur due to lack of information.
0114E	Invalid From Filename entry in columns 11-18	Filename did not appear on file specifications.	Since no valid information is available for processing file name, multiple related errors can occur due to lack of information.
0115E	Illegal Table or Array Name entry in columns 27-32	Name contains invalid character, embedded blanks, or numeric first character.	Name used as it exists up to first blank
0116E	Illegal Table or Array Name entry in columns 46-51	Name contains invalid character, embedded blanks, or numeric first character.	Name used as it exists up to first blank
0117E	Illegal Number of the Chaining Field entry in columns 9-10	Entry must be C1-C9.	C1 assumed
0118E	Illegal To Filename entry in columns 19-26	Name contains invalid characters or embedded blanks.	File name used as it exists up to first blank
0119E	Illegal From Filename entry in columns 11-18	Name contains invalid characters or embedded blanks.	File name used as it exists up to first blank
0123E	Invalid or missing To Filename entry of a chaining file in columns 19-26		No special action taken
0124E	Invalid or missing To Filename entry of a record address file in columns 19-26		No special action taken
0125E	Invalid or missing From Filename entry of a record address file in columns 11-18		No special action taken
0126E	Invalid Table or Array Name entry in columns 27-32 or 46-51	Name encountered previously	No special action taken
0127E	Illegal Length of Entry entry in columns 40-42 or 52-54	Entry length must be equal to or less than record length of entry .	No special action taken

TABLE E-2. EXTENSION SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
0128E	Number of Entries per Record entry of blanks in columns 33-35 invalid with entry in From Filename in columns 11-18	Entry must be 1-999.	1 assumed
0129E	Number of Entries per Record entry in columns 33-35 invalid with Number of Entries per Table/Array entry in columns 36-39	Number of entries per record cannot be greater than number of entries per table/array.	No special action taken
0130E	Decimal Positions entry in column 44 or 56 invalid with Length of Entry entry in columns 40-42 or 52-54	Decimal position cannot be greater than length of entry.	Zero decimal position assumed
0131E	Illegal Table or Array Name entry in columns 27-32	Name contains a blank as first character.	No further processing of line
0132E	Packed/Binary entry of P, B, L, or R in column 43 invalid with non-initialized tables/arrays		No special action taken
0133E	Packed/Binary entry of P, B, L, or R in column 55 invalid with non-initialized tables/arrays		No special action taken
0134E	Length of Entry entry in columns 40-42 invalid with From Filename entry in columns 11-18	Primary table/array record size cannot exceed from-file record size.	No special action taken
0135E	Length of Entry entry in columns 52-54 invalid with From Filename entry in columns 11-18	Alternate table/array record size cannot exceed from-file record size.	No special action taken
0136E	Length of Entry entry in columns 40-42 invalid with To Filename entry in columns 19-26	Primary table/array record size cannot exceed to-file record size.	No special action taken
0137E	Length of Entry in columns 52-54 invalid with To Filename entry in columns 19-26	Alternate table/array record size cannot exceed to-file record size.	No special action taken
0138E	Illegal Table or Array Name entry in columns 46-51	Name contains a blank as first character.	No further processing of line
0139E	Illegal P=Packed/B=Binary entry of P or B in column 43 or column 55	Tables/arrays loaded at compile time cannot be in binary or packed format.	Data assumed to be unpacked; additional error can occur when data is input.

TABLE E-2. EXTENSION SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
0140E	To Filename in columns 19-26 is specified for table or array output but is not an output file (0 in column 15 of file specifications).		No special action taken
0141E	From Filename in columns 11-18 is specified for table or array input but is not a table file (T in column 16 of file specifications).		
0142E	From Filename in columns 11-18 is not a record address file (R in column 16 of file specifications).		No special action taken
0143E	To Filename in columns 19-26 is not an input or update file (I or U in column 15 of file specifications).		No special action taken
0001L	File Description Specifications for Filename in columns 7-14 did not request line control specifications (L in column 39).		
0002L	Columns 18-19 did not contain FL (assumed ok).		
0003L	Columns 23-24 did not contain OL (assumed ok).		
0004L	File description specification for Filename in columns 7-14 requested line counter specification (L in column 39), but none were present.		
0100L	Abnormal end-of-file reached.		
0101L	Illegal form type entry in column G (not I or L).		
0102L	Filename in columns 7-14 not previously defined on File Description Specifications.		
0103L	Number of lines per page entry (columns 15-17) did not contain a valid numeric entry greater than 12 (60 assumed).		
0104L	Overflow line number entry (columns 20-22) did not contain a valid nonzero numeric entry (40 assumed).		

TABLE E-2. EXTENSION SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
0011	Illegal Sequence entry in columns 15-16	Alphabetic Sequence entry encountered; however, numeric Sequence entry encountered previously for same file	No special action taken
0002I	Illegal alphabetic Sequence entry in columns 15-16	Both characters in alphabetic sequence must be alphabetic.	No special action taken
0003I	Number entry in column 17 or Option entry in column 18 invalid with alphabetic Sequence entry in columns 15-16	Both must be blank with alphabetic Sequence.	No special action taken
0004I	Illegal Number entry in column 17	Entry must be 1 or N.	N assumed
0005I	Illegal Option entry in column 18	Entry must be 0 or blank.	0 assumed
0006I	Illegal Sequence entry in columns 15-16	First numeric Sequence entry must be 01.	No special action taken
0007I	Invalid Sequence entry in columns 15-16	Numeric Sequence entry out of sequence	No special action taken
0008I	Both record identification and field description specifications encountered on same input specification line		Both properly processed
0009I	Illegal Stacker Select entry in column 42		Blank assumed
0010I	Illegal sterling sign position entry in columns 71-74	Sterling sign position specification not allowed	No special action taken
0011I	Illegal packed/binary entry in column 43	Entry must be P, B, L, R, or blank.	Blank assumed
0012I	Illegal Field Location entry in columns 44-51	When a key field is defined, a K must be entered in both From and To entries.	K assumed in both
0013I	Illegal Decimal Positions entry in column 52	Entry must be 0-9.	Zero assumed
0015I	Illegal entry in columns 17-18	Entry must be blank on And/Or line.	No special action taken
0016I	Illegal entry in columns 19-20	Entry must be blank on And line.	Record identification indicator entry ignored
0017I	Illegal entry in column 16	Entry must be blank on Or line.	No special action taken
0018I	Illegal entry in columns 7-42 of field description entry	Entry must be blank.	No special action taken
0019I	Illegal Sequence entry in columns 15-16	Entry contains invalid characters.	No special action taken

TABLE E-2. EXTENSION SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
0020I	Illegal Stacker Select entry in column 42	Entry must be blank on And line.	No special action taken
0021I	Illegal Field Location entry in columns 44-51	Trailer field overlaps header field of spread card.	No special action taken
0022I	Invalid Record Identifying Indicator entry in columns 19-20	Blank entry	No indicator assumed
0023I	Unexpected end-of-file reached		Compiler exited through C specification processor.
0024I	Invalid Field Name entry in columns 53-58	Entry encountered previously as numeric	Alpha assumed for this line only
0025I	Column 15-16 contains 'OR'	If OR line, 'OR' should be in 14-15	No special action taken; treated as 'NS'
0100I	Illegal Record Identifying Indicator entry in columns 19-20		No special action taken
0101I	Illegal Form Type entry in column 6	Card out of order.	Source line bypassed; processing continues with next line.
0102I	No header fields specified for trailer fields		No special action taken
0103I	Illegal entry in columns 17-18 and 21-74	Entry must be blank on look-ahead field definition line.	No special action taken
0104I	Sequence entry of numeric or blank in columns 15-16 invalid with look ahead fields (**) entry in columns 19-20	Sequencing not allowed with look-ahead files	No special action taken
0105I	Illegal Filename entry in columns 7-14	Name contains invalid characters or embedded blanks	Name used as it exists up to first blank
0106I	Illegal Filename entry in columns 7-14	Blank entry or blank character in entry	Since no valid file can be related to the following input specifications, multiple diagnostic errors can occur.
0107I	Illegal From entry in columns 44-47		1 assumed
0108I	Illegal To entry in columns 48-51		From entry assumed
0109I	Illegal To entry in columns 48-51	To entry must be equal to or greater than from entry in columns 44-47.	To assumed equal to From.
0110I	Illegal To entry in columns 48-51	To entry must be less than or equal to record size.	Record size assumed
0111I	Illegal Field Location entry in columns 44-51	Packed numeric field exceeds length of 8.	No special action taken

TABLE E-2. EXTENSION SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
0112I	Illegal Field Location entry in columns 44-51	Binary input field must have length of either 2 or 4.	No special action taken
0113I	Illegal Field Location entry in columns 44-51	Alpha field exceeds length of 256.	No special action taken
0114I	Illegal Field Location entry in columns 44-51	Unpacked numeric field exceeds length of 15.	No special action taken
0116I	Invalid Field name entry in columns 53-58	Entry encountered previously as alpha	No special action taken
0117I	Illegal Field Name entry in columns 53-58	Name contains a comma with no index.	Comma ignored; name processed as non-indexed
0118I	Blank Field Name entry in columns 53-58 invalid with blank Sequence entry in columns 15-16. (Blank first character in field name causes compiler to assume blank name entry.)		Since neither entry is present, compiler is unable to determine whether line is a record identification or a field description. Additional diagnostic messages can occur.
0120I	Packed/Binary entry in column 43 invalid with a non-numeric field		No special action taken
0121I	Invalid Filename entry in columns 7-14	Filename encountered previously on input specifications.	No special action taken
0122I	Illegal Field Name entry in columns 53-58	Name contains invalid characters or embedded blanks.	Field name used as it exists up to first blank
0123I	Illegal Field Name entry in columns 53-58	First character of field name cannot be numeric.	No special action taken
0124I	Invalid Field Name entry in columns 53-58	Record address or table file cannot be defined on input specifications.	No special action taken
0125I	Field Name entry in columns 53-58 encountered previously with different length		No special action taken
0126I	Field Name entry in columns 53-58 encountered previously with different decimal positions entry in column 52		No special action taken
0127I	Field Name entry in columns 53-58 encountered previously but was not an array		No special action taken

TABLE E-2. EXTENSION SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
0022I	Invalid Record Identifying Indicator entry in columns 19-20	Blank entry	No indicator assumed
0129I	Valid Field Name entry of an array name in columns 53-58; however, array name not encountered on Extension Specifications		No special action taken
0130I	Invalid Field Name entry in columns 53-58	Table illegal as input field	No special action taken
0131I	Invalid Field Name entry in columns 53-58	Reserved word other than PAGE illegal for input	No special action taken
0132I	Illegal Control Level entry in columns 59-60	Entry must be L1-L9 or blank	Blank assumed
0133I	Illegal Matching Fields entry in columns 61-62	Entry must be M1-M9 or blank	Blank assumed
0134I	Illegal Field Record Relation entry in columns 63-64		Indicator LO assumed
0135I	Illegal Plus Field Indicator in columns 65-66		Indicator LO assumed
0136I	Illegal Minus Field Indicator entry in columns 67-68		Indicator LO assumed
0137I	Illegal Zero Field Indicator entry in columns 69-70		Indicator LO assumed
0138I	Invalid Control Level entry of L1-L9 in columns 59-60	Control fields invalid with chained files	No special action taken
0139I	Invalid Control Level entry of L1-L9 in columns 59-60	Control fields invalid with demand fields	No special action taken
0140I	Invalid Control Level entry of L1-L9 in columns 59-60	Control files invalid with binary fields	No special action taken
0141I	Invalid Control Level entry of L1-L9 in columns 59-60	Control fields invalid with look-ahead files	No special action taken
0142I	Invalid Control Level entry of L1-L9 in columns 59-60	Control fields invalid with trailer fields	No special action taken
0143I	Invalid Matching Fields entry of M1-M9 in columns 61-62	Matching fields invalid with chained files	No special action taken
0144I	Invalid Matching Fields entry of M1-M9 in columns 61-62	Matching fields invalid with demand files	No special action taken
0145I	Invalid Matching Fields entry of M1-M9 in columns 61-62	Matching fields invalid with binary fields	No special action taken
0146I	Invalid Matching Fields entry of M1-M9 in columns 61-62	Matching fields invalid with look-ahead fields	No special action taken

TABLE E-2. EXTENSION SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
0147I	Invalid Matching Fields entry of M1-M9 in columns 61-62	Matching fields invalid with trailer fields	No special action taken
0148I	Invalid Matching Fields entry of M1-M9 in columns 61-62	Matching fields invalid with nonindexed arrays	No special action taken
0153I	Illegal entry in columns 7-18 and 21-74	Entry must be blank on trailer definition line	No special action taken
0154I	Invalid Filename entry in columns 7-14	Name not defined on file specifications	First file name defined on file specification assumed. Additional diagnostic errors can occur.
0155I	Invalid Filename entry in columns 7-14	File must be input, update, or combined	No special action taken
0156I	Look-ahead fields (**) entry in columns 19-20 invalid with AND or OR specifications		No special action taken
0157I	First input specification line of record identification cannot be AND or OR.		No special action taken
0158I	Illegal Not entry in columns 25, 32, or 39.	Entry must be N or blank.	No special action taken
0159I	Illegal C/Z/D entry in columns 26, 33, or 40.	Entry must be C, Z, or D.	No special action taken
0160I	Illegal Position entry in columns 21-24 or 35-38	Entry must be 1-9999 or blank.	No special action taken
0161I	Illegal Field Name entry of an array index in columns 53-58		Invalid characters in numeric index processed
0162I	Control Level indicator L1 entered for different field types in a preceding record identification group		No special action taken
0163I	Control Level indicator L2 entered for different field types in a preceding record identification group		No special action taken
0164I	Control Level indicator L3 entered for different field types in a preceding record identification group		No special action taken
0165I	Control Level indicator L4 entered for different field types in a preceding record identification group		No special action taken
0166I	Control Level indicator L5 entered for different field types in a preceding record identification group		No special action taken

TABLE E-2. EXTENSION SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
0167I	Control Level indicator L6 entered for different field types in a preceding record identification group		No special action taken
0168I	Control Level indicator L7 entered for different field types in a preceding record identification group		No special action taken
0169I	Control Level indicator L8 entered for different field types in a preceding record identification group		No special action taken
0170I	Control Level indicator L9 entered for different field types in a preceding record identification group		No special action taken
0171I	Matching Field indicator M1 defined more than once in a preceding record identification group		No special action taken
0172I	Matching Field indicator M2 defined more than once in a preceding record identification group		No special action taken
0173I	Matching Field indicator M3 defined more than once in a preceding record identification group		No special action taken
0174I	Matching Field indicator M4 defined more than once in a preceding record identification group		No special action taken
0175I	Matching Field indicator M5 defined more than once in a preceding record identification group		No special action taken
0176I	Matching Field indicator M6 defined more than once in a preceding record identification group		No special action taken
0177I	Matching Field indicator M7 defined more than once in a preceding record identification group		No special action taken
0178I	Matching Field indicator M8 defined more than once in a preceding record identification group		No special action taken
0179I	Matching Field indicator M9 defined more than once in a preceding record identification group		No special action taken

TABLE E-2. EXTENSION SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
0180I	Matching Field indicator M1 entered for different field types in a preceding record identification group		No special action taken
0181I	Matching Field indicator M2 entered for different field types in a preceding record identification group		No special action taken
0182I	Matching Field indicator M3 entered for different field types in a preceding record identification group		No special action taken
0183I	Matching Field indicator M4 entered for different field types in a preceding record identification group		No special action taken
0184I	Matching Field indicator M5 entered for different field types in a preceding record identification group		No special action taken
0185I	Matching Field indicator M6 entered for different field types in a preceding record identification group		No special action taken
0186I	Matching Field indicator M7 entered for different field types in a preceding record identification group		No special action taken
0187I	Matching Field indicator M8 entered for different field types in a preceding record identification group		No special action taken
0188I	Matching Field indicator M9 entered for different field types in a preceding record identification group		No special action taken
0189I	Invalid Field Indicators entry in columns 65-68	Plus and Minus indicators illegal with alpha field	No special action taken
0190I	Invalid look-ahead fields (**) entry in columns 19-20	Look-ahead previously specified and active.	No special action taken
0191I	No valid input specifications present		No special action taken
0192I	Illegal Position entry in columns 21-24, 28-31, or 35-38.	Entry must be less than or equal to record length.	No special action taken
0193I	Invalid look-ahead fields (**) entry in columns 19-20	Look-ahead only valid with primary or secondary file.	No special action taken
0195I	Illegal Field Name entry in columns 53-58	Table/array indexes must be numeric.	No special action taken

TABLE E-2. EXTENSION SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
0196I	Decimal Positions entry in column 52 invalid with Field Length entry in columns 44-51	Decimal position cannot be greater than field length.	No special action taken
0197I	Illegal Field Name entry in columns 53-58	Array index has value of zero.	No special action taken
0198I	Invalid Field Record Relation entry in columns 63-64	Field record relation indicator must be equal to record identification indicator when specified with matching or control fields.	No special action taken
0199I	Invalid Field Length entry in columns 44-51	Field length is zero.	No special action taken
0200I	Length of array defined on input specifications greater than length defined on extension specifications		No special action taken
0201I	Illegal Field Name entry in columns 53-58	An array index cannot be a table or array name.	No special action taken
0202I	Illegal Field Name entry in columns 53-58	An array index can have no decimal positions.	No special action taken
0203I	Illegal Field location entry in columns 44-51	A nonindexed array must define a length that is devisible by the length of elements in the array.	No special action taken
0204I	Illegal trailer specified in columns 19-20	Trailer already present for this file.	No special action taken

TABLE E-3. CALCULATION SPECIFICATION ERRORS

Error Message Number	Error	Definition	Action
0001C	Illegal Factor 1 entry in columns 18-27 or Factor 2 entry in columns 33-42	Label name exceeds six characters.	No special action taken
0002C	Illegal Operation entry of DEBUG in columns 28-32.	DEBUG not specified on header line	Source line bypassed processing continues.
0003C	Invalid Field Length entry in columns 49-51	Entry must be 1-256 with maximum of 15 for numeric and 256 for alpha.	Blank assumed
0004C	Illegal Field Length entry in columns 49-51	Entry must be 1-256 or blank.	Blank assumed
0005C	Indicators entry in columns 9-17 invalid with Operation entry in columns 28-32	Conditioning indicators illegal with specified operation code	Indicator entries in columns 9-17 ignored.
0006C	Illegal Control Level Indicators entry in columns 7-8	Entry must be L0, L1, L9 or LR.	L0 assumed

TABLE E-3. CALCULATION SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
0007C	Illegal Operation entry in columns 28-32	Only BEGSR operation code can be specified following a line with ENDSR operation code.	No special action taken
0008C	Abnormal end-of-file reached		Compiler exited normally.
0100C	Illegal Factor 1 entry in columns 18-27	Entry must be blank for specified operation code.	Factor 1 assumed blank
0101C	Illegal Factor 1 entry of a numeric field in columns 18-27		No special action taken
0102C	Illegal Factor 1 entry of an alphabetic field in columns 18-27		No special action taken
0103C	Illegal Factor 1 entry in columns 18-27		No special action taken. If entry is blank, numeric field assumed.
0104C	Illegal Operation entry in columns 28-32		Source line bypassed; processing continues with next line.
0105C	Illegal Factor 1 entry in columns 18-27	Label previously defined	No special action taken
0106C	Illegal Factor 1 entry in columns 18-27	Invalid label entry	No special action taken
0107C	Illegal Control Level entry in columns 7-8	Entry must be AN or OR due to no Operation entry in columns 28-32 of preceding line.	No special action taken
0108C	Illegal Factor 2 entry in columns 33-42	Entry must be blank for specified operation code.	No special action taken
0109C	Illegal Factor 2 entry of a numeric field in columns 33-42		No special action taken
0110C	Illegal Factor 2 entry of an alpha field in columns 33-42		No special action taken
0111C	Illegal Factor 2 entry in columns 33-42		No special action taken. If entry is blank, numeric field assumed.
0113C	Illegal Factor 2 entry in columns 33-42	Invalid label entry	No special action taken
0114C	Illegal Factor 2 entry in columns 33-42	File name not defined in file specifications or blank entry	Factor 2 assumed blank
0115C	Factor 2 must be same type as Factor 1 for specified operation code		No special action taken
0116C	Illegal Result Field entry in columns 43-48	Entry must be blank for specified operation code.	Result field assumed blank
0117C	Illegal Result Field entry of a numeric field in columns 43-48		No special action taken

TABLE E-3. CALCULATION SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
0118C	Illegal Result Field entry of an alpha field in columns 43-48		No special action taken
0119C	Illegal Result Field entry in columns 43-48		No special action taken
0120C	Illegal Form Type entry in column 6	Card out of sequence	Source line bypassed; processing continues with next line
0121C	Illegal Half-Adjust (H) entry in column 53	Invalid entry for specified operation code	Half-adjust entry ignored
0122C	Illegal Half-Adjust (H) entry in column 53	Entry must be H or blank.	Blank assumed
0123C	Illegal Resulting Indicators entry in columns 54-59	Entry must be blank for specified operation code.	No special action taken
0124C	Illegal Resulting Indicators entry in columns 54-59	At least one resulting indicator required for specified operation code	No special action taken
0125C	Illegal Resulting Indicators entry in columns 54-55		LO assumed
0126C	Illegal Resulting Indicators entry in columns 56-57		LO assumed
0127C	Illegal Resulting Indicators entry in columns 58-59		LO assumed
0128C	Illegal Control Level entry in columns 7-8	Total line cannot be entered following a last record line.	No special action taken
0129C	Detail specifications must proceed total specifications		No special action taken
0130C	Illegal Control Level entry in columns 7-8	Total line cannot be entered following a subroutine line.	No special action taken
0131C	Illegal Control Level entry in columns 7-8		No special action taken
0132C	Illegal Not entry in column 9		No special action taken
0133C	Illegal Not entry in column 12		No special action taken
0134C	Illegal Not entry in column 15		No special action taken
0135C	Illegal Indicator entry in columns 10-11		No special action taken
0136C	Illegal Indicator entry in columns 13-14		No special action taken
0137C	Illegal Indicator entry in columns 16-17		No special action taken
0138C	Illegal Factor 1 entry in columns 18-27 or Factor 2 entry in columns 33-42	No closing apostrophe on literal entry	No special action taken

TABLE E-3. CALCULATION SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
0139C	Illegal Operation Code entry in columns 28-32	ENDSR operation code invalid outside of a subroutine	No special action taken
0140C	Illegal Operation Code entry in columns 28-32	BEGSR operation code specified prior to termination of preceding subroutine	No special action taken
0141C	Illegal Result Field entry in columns 43-48	Entry previously specified with different length	No special action taken
0142C	Illegal Result Field entry in columns 43-48	Entry previously specified with different decimal positions	No special action taken
0143C	Illegal Result Field entry in columns 43-48	Entry previously specified as alphabetic	No special action taken
0144C	Illegal Result Field entry in columns 43-48	Entry cannot be a numeric literal	Name processed with first character numeric
0145C	Illegal Result Field entry in columns 43-48	Entry cannot be an alphabetic literal	Name processed with opening apostrophe in literal included in alphabetic name
0146C	Illegal Result Field entry in columns 43-48	Entry previously specified as numeric	No further validity tests made on field
0147C	Invalid Decimal Positions entry in columns 52	Entry illegal with alphabetic field	
0148C	Illegal Factor 2 entry in columns 33-42	Bit numbers specified incorrectly	Invalid bit numbers ignored
0149C	Illegal Factor 2 entry in columns 33-42	Output file for DEBUG operation must be at least 80 bytes in length	No special action taken
0150C	Illegal Factor 2 entry in columns 33-42	File name for DEBUG operation code must be an output file.	No special action taken
0151C	Illegal Factor 2 entry in columns 33-42	File name for DSPLY operation code must be a display file.	No special action taken
0152C	The following label name was referenced and not defined.		No special action taken
0153C	Illegal Factor 2 entry in columns 33-42	Filename contains invalid characters or embedded blanks.	File name used as it exists up to first blank
0154C	Illegal Factor 1 or Factor 2 entry	Alphabetic literal contains characters following closing apostrophe.	No special action taken
0155C	Illegal Factor 1 or Factor 2 entry	Alphabetic literal has length of zero.	No special action taken
0156C	Illegal Factor 2 entry in columns 33-42	Bit number literal contains characters following closing apostrophe.	No special action taken

TABLE E-3. CALCULATION SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
0157C	Illegal Decimal Positions entry in column 52	Entry must be 0-9 or blank.	Zero assumed
0158C	Illegal Operation entry in columns 28-32	DIV operation code must be entered prior to specification of MVR operation code.	No special action taken
0159C	No ENDSR operation code entered for previous subroutine		No special action taken
0160C	Illegal Resulting Indicators entry in columns 54-59	LOKUP operation code cannot have both Plus and Minus indicators specified in columns 54-57.	No special action taken
0161C	Illegal Factor 1 or Factor 2 entry	More than one decimal point in numeric literal.	Leftmost decimal point used
0162C	Illegal Factor 1 entry in columns 18-27	Label name previously defined	No special action taken
0163C	Illegal Factor 1 or Factor 2 entry	Label name contains invalid characters or embedded blanks.	Label name used as it exists up to first blank
0164C	Illegal Factor 1 or Factor 2 entry of blanks	Operation code requires a label name.	No special action taken
0165C	Illegal Resulting Indicators entry in columns 54-57	High/Low indicators must not be specified with LOKUP operation on non-sequenced table/array.	No special action taken
0166C	Illegal Factor 1 or Factor 2 entry	Factor 1 and factor 2 of LOKUP operation must have equal length fields specified.	No special action taken
0167C	Illegal Factor 1 or Factor 2 entry	Table/array indexes must be numeric.	No special action taken
0168C	Illegal Factor 1 or Factor 2 entry.	Field name contains more than six characters.	Name used with first six characters
0169C	Illegal operation entry in columns 28-32	FORCE operation illegal with output or display files	No special action taken
0170C	Operation entry of READ in columns 28-32 invalid with High/Low resulting indicators entry in columns 54-57.	High/Low indicators not allowed with READ operation	No special action taken
0171C	Operation entry of READ in columns 28-32 invalid with equal Resulting Indicators entry of HO in columns 58-59.	Equal indicator HO not allowed with READ operation	No special action taken
0172C	Decimal Positions entry in columns 52 invalid with Field Length entry in columns 49-51	Decimal position cannot be greater than field length	No special action taken
0173C	Illegal Factor 1, Factor 2, or Result Field entry	Name contains blank as the first character	No further processing of line

TABLE E-3. CALCULATION SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
0174C	Illegal Operation entry in columns 28-32	FORCE cannot be specified on total or last record specification.	No special action taken
0175C	Illegal Operation entry in columns 28-32	BEGSR can only be specified on subroutine specification.	No special action taken
0176C	Illegal Factor 1, Factor 2, or Result Field entry	Array index is zero.	No special action taken
0177C	Entries in columns 18-59 invalid with no Operation entry in columns 28-32		Line assumed to be first line of AND/OR related indicators
0178C	Illegal Factor 2 entry in columns 33-42	Entry is not a one-byte alpha field.	No special action taken
0179C	Illegal Result Field entry in columns 43-48	Entry is not a one-byte alpha field.	No special action taken
0180C	Factor 2 entry of Result Field entry invalid with Operation entry of MOVEA in columns 28-32	Factor 2 or Result Field must contain an array or an array element with MOVEA operation.	No special action taken
0181C	Illegal Factor 1 entry in columns 18-27	Field length of Factor 1 is greater than record length.	No special action taken
0182C	Factor 1 entry or Result Field entry invalid with Operation entry of DSPLY in columns 28-32	DSPLY operation requires Factor 1 or Result Field.	No special action taken
0183C	Illegal Result Field entry in columns 43-48	Field length of Result Field is greater than record length.	No special action taken
0184C	Illegal Factor 1, Factor 2, or Result field entry	An index is entered for a field that is not defined as an array.	No special action taken
0185C	Illegal Factor 1, Factor 2, or Result Field entry	An array index cannot be a table or array name.	No special action taken
0186C	Illegal Factor 1, Factor 2, or Result Field entry	An array index cannot be a ULABL field.	No special action taken
0187C	Illegal Factor 1, Factor 2, or Result Field entry	An array index can have no decimal positions.	No special action taken
0188C	Illegal Result Field entry in columns 43-48	Entry must be a table name.	No special action taken
0189C	Last record calculation specification must precede SR specifications.		
0190C	Detail specifications must precede LR specifications.		
0191C	Detail specifications must precede SR specifications.		
0192C	RLABL operation must follow either an EXIT operation or another RLABL operation.		
0193C	Result field has been previously defined differently.		

TABLE E-4. OUTPUT FORMAT SPECIFICATION ERRORS

Error Message Number	Error	Definition	Action
00010	Illegal Packed/Binary Output entry in column 44	Entry must be blank for alphabetic field.	No special action taken
00020	Illegal Packed/Binary Output entry in column 44	Entry must be blank for alphabetic constant.	No special action taken
00030	Illegal blank after entry in column 39	Entry must be blank for alphabetic constant.	No special action taken
00040	Illegal Space and Skip entries entry in column 17-22	Entry must be blank on And/Or line.	No special action taken
00050	Illegal Space Before entry in column 17		Blank assumed
00060	Illegal Space After entry in column 18		Blank assumed
00070	Illegal Skip Before entry in columns 19-20		Blank assumed
00080	Illegal Skip After entry in columns 21-22		Blank assumed
00090	Illegal Not entry in column 23		Blank assumed
00100	Illegal indicator entry in columns 24-25		Blank assumed
00110	Illegal Not entry in column 26		Blank assumed
00120	Illegal indicator entry in columns 27-28		Blank assumed
00130	Illegal Not entry in column 29		Blank assumed
00140	Illegal indicator entry in columns 30-31		Blank assumed
00150	Illegal Constants or Edit Words entry in columns 45-70	Constant must not contain less digit positions than field to be edited.	No special action taken
00160	Illegal Packed/Binary Output entry in column 44		No special action taken
00170	Illegal entry in columns 71-74	Entry must be blank.	No special action taken
00180	Invalid Field Name entry of *PLACE in columns 32-37	Data repeated in original position	No special action taken
01000	Illegal Form Type entry in column 6	Entry must be O.	Source line bypassed, processing continues with next line
01010	Illegal Filename entry in columns 7-14	Blank entry or blank first character in entry	Since no valid file can be related to the following output specifications, multiple diagnostic errors can occur.
01020	Illegal Type entry in column 15		D assumed

TABLE E-4. OUTPUT FORMAT SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
01030	Invalid or missing End Position in Output Record entry in columns 40-43		No special action taken
01040	Illegal Filename entry in columns 7-14	Name not defined on file specifications	Since no valid file can be related to the following output specifications, multiple diagnostic errors can occur.
01050	Illegal And/Or entry in columns 14-16	Preceding line must be record selection.	No special action taken
01060	Illegal Field Name entry in columns 32-37	Field name must be previously defined in program.	Field name assumed blank; line processed as constant output.
01070	Illegal Field Name entry in columns 32-37	Index name must be previously defined.	No special action taken
01080	Illegal Field Name entry in columns 32-37	Numeric index contains non-numeric character.	Non-numeric character assumed to be zero
01090	Illegal Constants or Edit Words entry in columns 45-70	No terminal apostrophe in constant or edit word	No special action taken
01100	Illegal End Position in Output Record entry in columns 40-43	Entry cannot be greater than record length.	No special action taken
01110	Illegal End Position in Output Record entry in columns 40-43	Entry cannot be less than field length.	No special action taken
01120	Illegal Field Name entry in columns 32-37	*PRINT not available	No special action taken
01130	Field Name entry in columns 32-37 invalid with an Edit Codes entry in column 38 or with an Edit Words entry in columns 45-70	Only numeric fields can be edited.	No special action taken
01140	Packed/Binary Output entry in column 44 invalid with edited fields		No special action taken
01150	Illegal Filename entry in columns 7-14	Name must be output, update, or combined file.	No special action taken
01160	Illegal Field Name entry in columns 32-37	Name contains invalid characters or embedded blanks.	Name used as it exists up to first blank
01170	Illegal Constants or Edit Words entry in columns 45-70	No opening apostrophe in column 45	No special action taken
01180	Illegal Constants or Edit Words entry in columns 45-70	Constant contains characters following closing apostrophe.	No special action taken
01190	Illegal Constants or Edit Words entry in columns 45-70	Constant has length of zero.	No special action taken
01200	Illegal Filename entry in columns 7-14	Name contains invalid characters or embedded blanks.	Name used as it exists up to first blank

TABLE E-4. OUTPUT FORMAT SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
01210	Illegal Skip entries in columns 19-22	Entry must be blank for an Add file.	No special action taken
01220	Invalid Indexed Sequential Add entry in columns 16-18	Add must be defined on file specifications in column 66 when Add specified on output specifications	No special action taken
01230	Illegal entry in columns 32-74	Must be blank on record identification line	No special action taken
01240	Illegal Output Indicator entry in columns 23-31	Overflow indicator specified that was not defined on file specifications	No special action taken
01260	Constants or Edit Words entry of dollar sign ($) or asterisk (*) in columns 45-70 invalid with Edit Codes entry of X, Y, or Z in column 38		No special action taken
01270	Invalid Field Name entry of *PLACE in columns 32-37	No valid fields exist to place.	No special action taken
01280	Illegal Blank After entry in column 39		No special action taken
01290	Invalid Edit Codes entry in column 38		No special action taken
01300	Constants or Edit Words entry in columns 45-70 invalid with Edit Codes entry in column 38	Edit constant illegal with edit code	No special action taken
01310	Output Indicators entry of 1P in columns 23-31 invalid with Type entry of T in column 15	1P indicator not allowed to condition total output	No special action taken
01320	Illegal End Position in Output Record entry in columns 40-43	Field length must not be larger than nine for binary output.	No special action taken
01330	Illegal Field Name entry in columns 32-37	Table/array indexes must be numeric.	No special action taken
01340	Output Indicators entry of OA-OG or OV in columns 23-31 invalid with Type entry of E in column 15	Overflow indicator not allowed to condition exception line	No special action taken
01360	Illegal Field Name entry in columns 32-37	Array index in columns 34-37 is zero	No special action taken
01370	Illegal Field Name entry in columns 32-37	An index is specified in columns 34-37 for a field that is not defined as an array.	No special action taken
01380	Illegal Field Name entry in columns 32-37	An array index cannot be a table or array name.	No special action taken
01390	Illegal Field Name entry in columns 32-37	An array index cannot be a ULABL field.	No special action taken

TABLE E-4. OUTPUT FORMAT SPECIFICATION ERRORS (Contd)

Error Message Number	Error	Definition	Action
01400	Illegal Field Name entry in columns 32-37	An array index can have no decimal positions.	No special action taken
01410	Illegal Stacker Select/Fetch Overflow entry of F in column 16	Fetch overflow not allowed for exception output	No special action taken
01420	ADD must be specified (columns 16-18) for an output and add file.		ADD is assumed in columns 16-18.

TABLE E-5. COMPILE TIME ARRAY ERRORS

Error Message Number	Error	Definition	Action
0001A	Short table/array loaded at compile time	Less than the maximum number of elements read into table/array	Remaining table/array elements filled with blanks if alphabetic, zeros if numeric and not in ascending sequence, and nines (9) if numeric and in ascending sequence.
0002A	Illegal entry in columns 1-2 of record	Should be end-of-file (/*) line	End-of-file line assumed
0100A	Table/array entries loaded at compile time not preceded by **Ø line		First line assumed to be **Ø; processing begins with second source line.
0101A	File name encountered on file translation entries that was not defined on file specifications		Since no valid file information can be related to bad file name, additional random errors can occur. Assumed file information can vary between programs.
0102A	Illegal file name entry on file translation records	File names not specified in valid form	File names following the error are bypassed; file names entered correctly prior to the error are processed.
0103A	File translation entries not preceded by **Ø line		Line processed as first file translation line
0104A	Alternate collating sequence entries not preceded by **Ø line		Line assumed to be **Ø line; processing begins with next line.
0105A	Illegal entry in columns 1-6 of alternate collating sequence record	Line does not begin with ALTSEQ.	Line assumed to begin with ALTSEQ; therefore, first eight columns not processed as data

TABLE E-5. COMPILE TIME ARRAY ERRORS (Contd)

Error Message Number	Error	Definition	Action
0107A	Mode definition (alphabetic or numeric) not encountered for following field		No special action taken
0108A	Length definition not encountered for following field		No further processing of field
0109A	Invalid character in hexadecimal data		No special action taken
0110A	Illegal table/array entry	Alphanumeric field out of sequence	No special action taken
0111A	Illegal table/array entry	Non-numeric character in numeric input field	Character assumed to be zero
0112A	Illegal table/array entry	Numeric field out of sequence	No special action taken

RUN-TIME ERRORS

These messages are output during execution of RPG II object programs when an abnormal situation occurs. No operator intervention is possible to respond to the message. The message is output to the MSOS console device and is of the form:

RPG nnnx ffffffff text

Where: nnn is the run-time error message number.

x is the designator. It is D for messages output by the data manager; otherwise it is R.

ffffffff is the file name, or blank if not applicable.

text is the message text.

The following is a list of run-time error messages.

0101R HALT INDICATOR H1 IS ON

0102R HALT INDICATOR H2 IS ON

0103R HALT INDICATOR H3 IS ON

0104R HALT INDICATOR H4 IS ON

0105R HALT INDICATOR H5 IS ON

0106R HALT INDICATOR H6 IS ON

0107R HALT INDICATOR H7 IS ON

0108R HALT INDICATOR H8 IS ON

0109R HALT INDICATOR H9 IS ON

0110R NO PRIMARY OR SECONDARY FILES ARE OPENED

0111R SQUARE ROOT OF NEGATIVE NUMBER

0112R NOT USED

0113R DIVISION BY ZERO

0114R ARRAY INDEX OUT OF RANGE

0115R INPUT TABLE OR ARRAY OUT OF SEQUENCE

0116R NO TABLE OR ARRAY DATA IN FILE

0117R TOO MUCH DATA FOR TABLE OR ARRAY

0118R SUBROUTINE RETURN STACK IS FULL

0119R SUBROUTINE RETURN WITHOUT PREVIOUS EXSR

0120R END-OF-FILE FOR DEMAND FILE

0121R RECORD NOT FOUND, KEY NOT IN INDEX FILE

0122R RELATIVE RECORD NUMBER OUT OF RANGE

0123R RECORD IS OUT OF SEQUENCE

0124R FILE OUT OF MATCHING RECORD SEQUENCE

0125R UNIDENTIFIED RECORD IN FILE

0126R TOO MUCH DATA FOR RECORD

0127R Not used

0128R Not used

0129R Not used

0130R ATTEMPT TO OUTPUT TO FULL FILE

0131R DIRECT ACCESS RECORD NO. OUT OF RANGE

0132R ATTEMPT TO UPDATE FILE BEFORE INPUT

0133R RECORD NOT FOUND WITH SPECIFIED KEY

0134R DUPLICATE KEY FOR OUTPUT TO FILE

0135R DEVICE ERROR

0136R FILE IS NOT PRESENT

0137R FILE DOES NOT CORRESPOND TO DMPT DESCRIPTION

0138R SETLL NOT VALID FOR THIS FILE TYPE

0139R Not used

0140R UNABLE TO CONTINUE DUE TO PREVIOUS ERROR

0141R Not used

0142R Not used

0143R Not used

0144R Not used

0145R Not used

0146R Not used

0147R Not used

0148R Not used

0149R Not used

0150R ALL HALT INDICATORS HAVE BEEN DISPLAYED

0151R FIRST LINE WILL BE PRINTED TWICE FOR FORMS ALIGN

0152R Not used

0153R Not used

0154R Not used

0155R ERROR IN PARAMETER(S) PASSED TO SUBROUTINE SUBRFL

0156R ERROR IN PARAMETER(S) PASSED TO SUBROUTINE SUBRAJ

0157R ERROR IN PARAMETER(S) PASSED TO SUBROUTINE SUBRMV

0158R ERROR IN PARAMETER(S) PASSED TO SUBROUTINE SUBRIN

0159R ERROR IN PARAMETER(S) PASSED TO SUBROUTINE SUBRED

0160R Not used

DATA MANAGER ERRORS

These messages are similar to the run-time error messages and are also output during execution of RPG II object programs when an abnormal situation is detected by data management. The message is output to the MSOS console device and is of the form described for run-time errors.

The following is a list of data manager error messages.

001D FILE IS ALREADY OPENED

002D TRIED TO PROCESS FILE IN WRONG MODE

003D SPECIFIED RECORD LENGTH IS INCORRECT

004D FILE NOT DEFINED IN DIRECTORY

005D AN ADD FILE FOR SEQ PROC MUST BE DEF AS OUTPUT

006D ADD/LOAD SEQUENTIALLY NOT FOR DIRECT DEF FILE

007D FILE DEFINITION IS WRONG FOR SEQ. PROC.

008D RECORD LENGTH NOT COMPATIBLE WITH THE DIRECTORY

009D FILE MGR DID NOT HANDLE DIR REQ STATUS $$$$

010D TRIED TO ACCESS AN UNOPENED FILE

011D READ 09 FOUND FILE TO BE NON UNIT-RECORD DEFINED

012D RECORD ######## WAS SHORT-PADDED WITH BLANKS

013D READ 12 FOUND FILE DEFINED AS A NON-DISK FILE

014D RECORD ######## WAS TOO LONG - RECORD TRUNCATED

015D WRIT17 FND FILE DEFI AS A NON UNIT-RECORD FILE

016D WRIT20 FND FILE DEFI AS A NON DISK-SEQ FILE

017D UPDT26 FND FILE DEFI AS A NON DISK-SEQ FILE

018D MULTIPLE UPDATES W/O READ IN BETWEEN INVALID

019D ADDT30 FOUND FILE DEFINED AS A NON DISK-SEQ FILE

020D CLOS35 FOUND FILE DEF AS A NON UNIT-REC FILE

021D CLOS38 FOUND FILE DEFINED AS A NON DISK-SEQ FILE

022D MASS STORAGE ERROR

023D FILE MGR DID NOT HANDLE FILE REQUEST $$$$$

024D END OF TAPE ENCOUNTERED

025D INPUT DEVICE FAILURE

026D OUTPUT DEVICE FAILURE

027D SHARED BUFFERS NOT ALLOWED FOR TAPE FILES

028D BLOCK SIZE MUST BE INTEGER MULTIPLE OF REC SIZE

029D READ11 FOUND FILE NOT DEFINED AS TAPE

030D WRIT19 FOUND FILE NOT DEFINED AS TAPE

031D CLOS37 FOUND FILE NOT DEFINED AS TAPE

032D DATA OVERFLOW - NO MORE TAPE VOLUMES DECLARED

033D ADDING RECORDS IS NOT ALLOWED IN DIRECT PROC.

034D OUTPUT FILE FOR DIRECT PROC MUST BE DEF DIR

035D DIRECT PROC. ON INDEXED FILE ALLOWED ONLY FOR INP

036D FILE DEFINITION IS WRONG FOR DIRECT PROC.

037D READ13 FOUND FILE DEFI AS A NON DISK-DIRECT FILE

038D WRIT21 FOUND FILE DEFI AS A NON DISK-DIRECT FILE

039D UPDT27 FOUND FILE DEFI AS A NON DISK-DIRECT FILE

040D CLOS39 FOUND FILE DEFI AS A NON DISK-DIRECT FILE

041D RECORD NUMBER ######## OUT OF RANGE

042D DOUBLE BUFFERING NOT ALLOWED FOR INDEX FILE

043D INDEX PROC. IS ALLOWED ONLY FOR INDEXED-DEF FILES

044D KEY LENGTH/POSITION NOT COMPATIBLE WITH DIRECTORY

045D OPEN05 WAS CALLED FOR NON-RANDOM PROCESSING

046D READ15 FOUND FILE DEFINED AS A NON INDEX FILE

047D WRIT23 FOUND FILE DEFINED AS A NON INDEX FILE

048D ADDT32 FOUND FILE DEFINED AS A NON INDEX FILE

049D CLOS41 FOUND FILE DEFINED AS A NON INDEX FILE

050D UPDT29 FOUND FILE DEFINED AS A NON INDEX FILE

051D THE KEY IN RECORD ######## MODIFIED BY UPDATE

052D CLOS40 FOUND FILE DEFINED AS A NON INDEX FILE

053D READ 19 FOUND FILE DEFINED AS A NON INDEX FILE

054D WRIT22 FOUND FILE DEFINED AS A NON INDEX FILE

055D UPDT28 FOUND FILE DEFINED AS A NON INDEX FILE

056D ADDT31 FOUND FILE DEFINED AS A NON INDEX FILE

057D SETL33 FOUND FILE DEFINED AS A NON INDEX SEQ FILE

058D RECORD ######## OUT OF ORDER

059D SETL33 SET LOWER LIMIT NOT ALLOWED FOR ADD FILES

060D KEY OF RECORD ######## ALREADY EXISTS

061D SEQ. PROC. FOR INDEX-DEFINED FILE ONLY FOR INPUT

062D UNABLE TO SORT THE KEYS

063D SLCT34 CALL STACKER SELECT IS NOT SUPPORTED

064D ERROR CODE IN BUFFER IS WRONG FOR TAPE

065D NO VOL1 LABEL ON SL TAPE

066D WRONG DATA SET NAME

067D TAPE VOLUMES MOUNTED IN WRONG ORDER

068D TAPE FILE NOT DEFINED BY MOUNT UTILITY

069D WRONG VOLUME SERIAL NUMBER

070D INVALID EXPIRATION DATE

071D FILE MANAGER ERROR IN READING MOUNT FILE

072D UNEXPIRED DATE SET

073D Not used

074D COMBINED FILES ARE NOT SUPPORTED

075D OPEN02 FOUND FILE DEF AS NON SPECIAL DEV FILE

076D WRONG BUFFER SIZE CANNOT BE USED FOR FIXED BLK

077D BLOCKING IS IMPOSSIBLE FOR SHARED BUFFERS

078D ERROR RETURN FROM SPEC DEV DRIVER, STATUS = $$$$$

079D READ 10 FOUND FILE DEF AS NON SPECIAL DEV FILE

080D UPDT25 FOUND FILE DEF AS NON SPECIAL DEV FILE

081D RECORD ######## WAS TOO LONG - RECORD IGNORED

082D CLOS36 FOUND FILE DEF AS NON SPECIAL DEV FILE

083D WRIT18 FOUND FILE DEF AS NON SPECIAL DEV FILE

084D Not used

085D Not used

086D Not used

087D Not used

088D Not used

089D Not used

090D Not used

091D Not used

092D Not used

093D Not used

094D Not used

095D Not used

096D Not used

097D Not used

098D Not used

099D ERROR AT LOCATION = $$$$$

100D Not used

MAGNETIC TAPE UTILITY ERRORS

Tables E-6 through E-10 list the four types of magnetic tape utility processor (MTUP) error messages: action, descriptive, critical, serious, and warning error messages. See the Magnetic Tape Utility Processor Reference Manual for further information.

MOUNT UTILITY ERRORS

MOUNT utility error messages are shown in table E-11.

TABLE E-6. MAGNETIC TAPE UTILITY ACTION MESSAGES

Error Message	Definition	Action
*INVALID PARM= "XXX. . ." *RETYPE PARM:_	The characters within quotes are invalid and may be corrected.	Enter corrected parameter.
10 ERRORS *CONTINUE:	Verify function has located 10 consecutive records that contain errors.	Type carriage return to terminate or type one character followed by carriage return to continue.
*MOUNT,OUTPUT, SCRATCH:		Type carriage return, which implies tape is ready, or type any other character followed by a carriage return, to terminate the initialize function.
*VOLSER=nnnnnn:		
VOL NOT EXPIRED USE:	Label processing: Output volume header records are checked against the system date.	Carriage return implies do not use. U implies use, ignoring expiration date.
*DATA SET NAME:	Label processing: Output volumes require a data set name if not available from input.	DSN='XXXXX'
VOLSER=nnnnnn	Informative tape file just opened with the specified volume serial number.	None

TABLE E-7. MAGNETIC TAPE UTILITY DESCRIPTIVE ERROR MESSAGES

Error Message	Definition	Action
FILES(S) NOT OPEN	A required file is not open and the specified function cannot be executed.	Open file and reenter function.
*INVALID OPEN OR CLOSE	The file being opened or closed is already in that state.	Open or close the proper file.
*FUNCTION NOT AVAILABLE	An attempted function is not available in the system. The function is not invalid; rather, the system was configured without the requested module.	Use another function, if possible.
*PARM NOT AVAILABLE	A parameter is not available in the system. The parameter is not invalid; rather, the system was configured without the requested module.	Use another parameter, if possible.
*INCORRECT VOL MOUNT:	The volume mounted does not contain a volume label or the header label sequence is incorrect; i.e., the wrong volume of a multiple volume file is mounted.	Mount correct volume and type a carriage return.

TABLE E-8. MAGNETIC TAPE UTILITY CRITICAL ERRORS

Error Code	Definition	Action
****C000****	Data buffer linkage has been destroyed. Cause: I/O malfunction, CPU malfunction.	Reload utility.

TABLE E-9. MAGNETIC TAPE UTILITY SERIOUS ERRORS

Error Code	Definition	Action
****S000****	Available memory has been filled.	Free memory by closing a file.
****S001****	Attempt to close file already closed.	Close proper file.
****S002****	1. Read end-of-file 2. Attempt to write on file not opened for write. 3. I/O error; i.e., parity, read or write error, lost data, or alarm.	Retry the function.
****S003****	Variable length block does not match actual length read, or variable read length is greater than specified block size.	Close all files. Open input as undefined and dump records to locate the erroneous record. File cannot be processed as variable length.
****S004****	Blocking has been requested and specified block size is smaller than specified record size.	Reopen file with proper parameters.
****S005****	Variable size error detected prior to write	Attempt to re-execute function after closing and reopening all files. Possible hardware malfunction.
****S006****	Fixed block error detected prior to write. Record length is not specified.	Close file and reopen with proper record size or dump file to locate erroneous records.
****S007****	Labeled file sequence number in error. (File is not opened.)	Mount proper volume and reopen.
****S008****	Labeled file EOF1 trailer label contains invalid information which does not correspond to header label 1.	This file cannot be processed with standard labels.
****S009****	Labeled file is missing EOF trailer labels.	File cannot be processed as labeled.
****S010****	End-of-tape sensed on output file (unlabeled)	Close the file with EOV and reopen after mounting new tape. Re-enter function to complete processing.
****S011****	A double file mark has been sensed on an input file. Processing is terminated.	Close input file and mount next volume. Re-enter function to complete processing.
****S012****	Invalid date	Re-enter date function with proper date.

TABLE E-9. MAGNETIC TAPE UTILITY SERIOUS ERRORS (Contd)

Error Code	Definition	Action
****S013****	Labeled volume sequence number incorrect. (Occurs after OPEN file is not opened.)	Mount proper volume and reopen file.
****S014****	ZERO LENGTH block specified in OPEN FILE is not opened.	Reopen, specifying proper block length.
****S015****	Block or record length specified is not a multiple of two. FILE, is not opened.	Reopen, specifying even block and record length. If either block or record length is odd, the data cannot be processed by the system.

TABLE E-10. MAGNETIC TAPE UTILITY WARNING MESSAGES

Error Code	Definition	Action
xxxW000xxx	Blocking not specified but block size and record size have been specified differently in OPEN.	Open file with proper parameters, or continue statement.
xxxW001xxx	File count specified as zero	Re-enter function with proper parameters or continue statement.
xxxW002xxx	Record count specified as zero	Re-enter function with proper parameters or continue statement.
xxxW003xxx	Input and output record lengths have been specified differently for COPY.	Re-enter function with proper parameters or continue statement.

TABLE E-11. MOUNT UTILITY ERROR MESSAGES

Error Message	Definition	Error Message	Definition
XXXXXXXX MOUNTED	The information for this tape volume has been written to the Mount file. It does not imply that all the information was written, as there may have been other errors. It shows that as much as possible was written.	INPUT ERROR, CONTINUE?	There was some failure of the logical input device preventing the reading of Mount information. Remaining data will be flushed and the program terminated.
		FILE NAME MISSING	The F parameter was not supplied.
UNRECOGNIZED PARMS	Self-explanatory – A parameter beginning with an illegal letter was supplied. The parameter itself will be highlighted by a question mark (?) printed underneath it on the console.	FILE MANAGER ERROR	For some reasons, the Mount file could not be written.
		CODE NOT A OR E	C parameter must be CA or CE.

TABLE E-11. MOUNT UTILITY ERROR MESSAGES (Contd)

Error Message	Definition	Error Message	Definition
MISSING PARAMETER	A parameter, such as F, C, V, D, or E was supplied without any data.	PARAMETER TOO LONG	F was more than 8 characters long, D was more than 17 characters long, or V was more than 6 characters long.
EXPIRATION DATE WRONG LENGTH	Must be five characters long.		

CATALOG UTILITY ERRORS

CATALOG utility error messages are as follows:

CATLOG ***ERROR***/
CATLOG READY TO INSTALL/
MASS STORAGE I/O ERROR/
REQUIRED PROGRAM NOT FOUND/
LIBRARY PROGRAM NOT VALID/
PROGRAM ON LGO NOT VALID/
TOO MANY PROGRAMS ON LGO/
LAST LGO PROGRAM HAS NO XFR/
INVALID CONTROL STATEMENT/

BINARY PUNCH UTILITY ERRORS

Binary punch utility (RBDPCH) error messages are as follows:

RBDPCH O/P UNLINKED RBD PROG AND SUBPROGS

RBDPCH ****ERROR****
*T NOT FOUND IN LGO FILE
BINARY OUTPUT ERROR
NAM BLOCK MISSING
DUPLICATE ENTRY POINTS
ENT/EXT TABLE OVERFLOW
PROGRAM NOT FOUND IN LIB
RBDPCH* OUTPUT COMPLETED
E10 UNPATCHED EXTERNAL

SWITCH UTILITY ERRORS

SWITCH utility error messages are as follows:

Message	Meaning
SWITCH = $x_1x_2x_3x_4x_5x_6x_7x_8$	x_i is the status of RPG external indicator U_i for i = 1 to 8. x_i = 0 if U_i is OFF x_i = 1 of U_i is ON
INVALID SWITCH ENTRY	*SWITCH xxxxxxxx statement was invalid.

RPG II OBJECT PROGRAM LOGIC (DETAILED)

For each input record processed, an RPG II object program performs the same general sequence of operations. This sequence of steps is called the program cycle. Within each cycle there are two periods of time in which calculations are performed and output records are produced. At total time, all total calculation operations (operations conditioned by control level indicators in columns 7 and 8 of form C) and total output (records conditioned by control level indicators) are done. Calculation and output operations not conditioned by control level indicators are performed at detail time.

The specific steps taken during each cycle are shown in the following flowchart (figure F-1). Total time occurs at steps 18 and 19; detail time is shown in steps 25 and 3. A program cycle begins at step 3 and continues through step 25. The remainder of this appendix is a description of the RPG II object program logic referencing numbers in the flowchart (figure F-1).

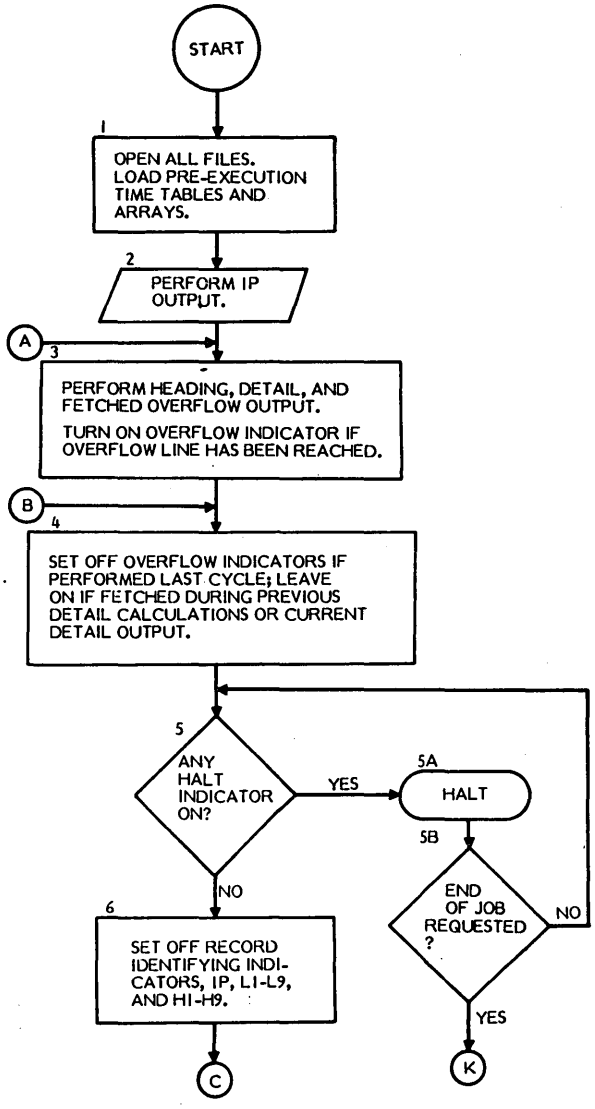


Figure F-1. Program Cycle (Sheet 1 of 4)

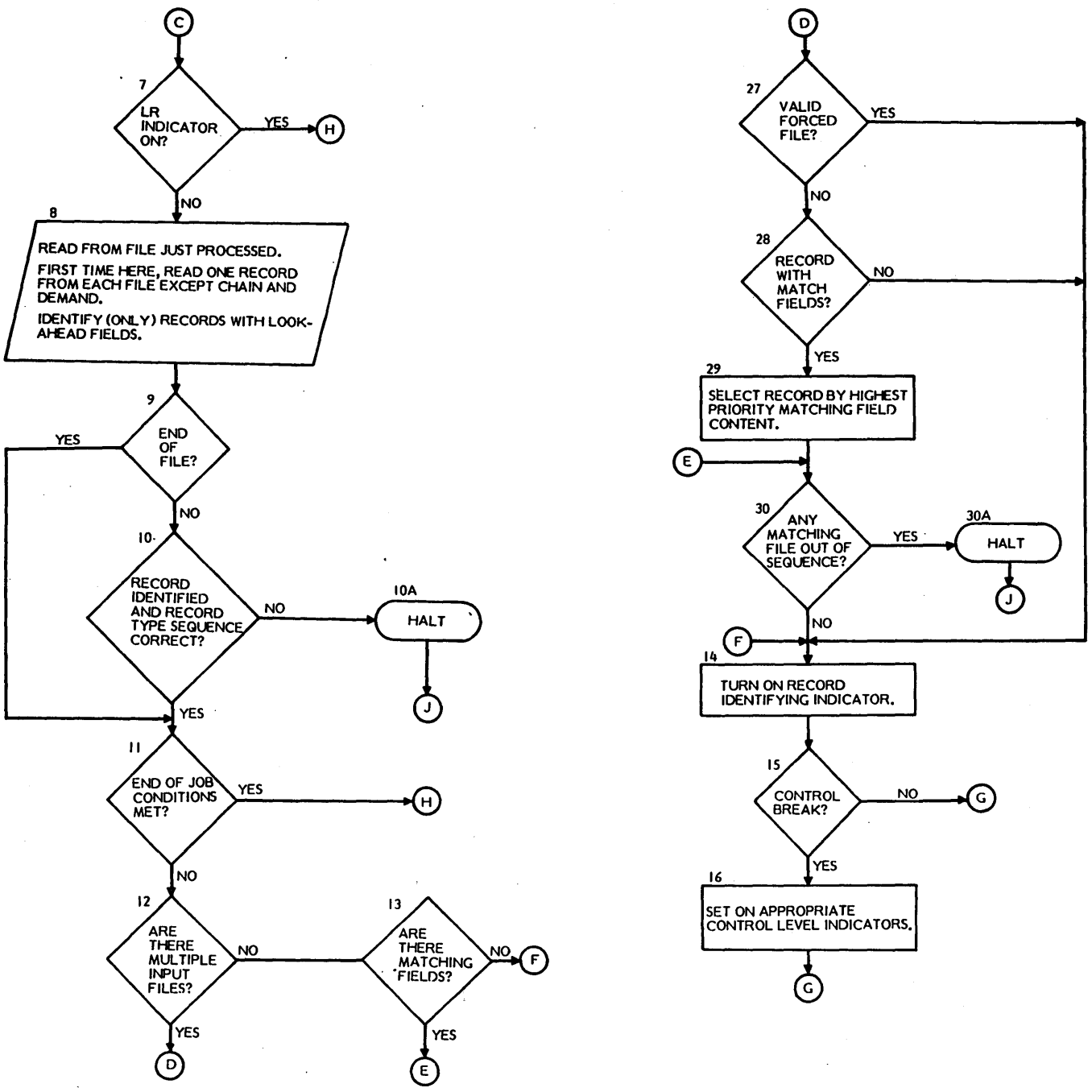


Figure F-1. Program Cycle (Sheet 2 of 4)

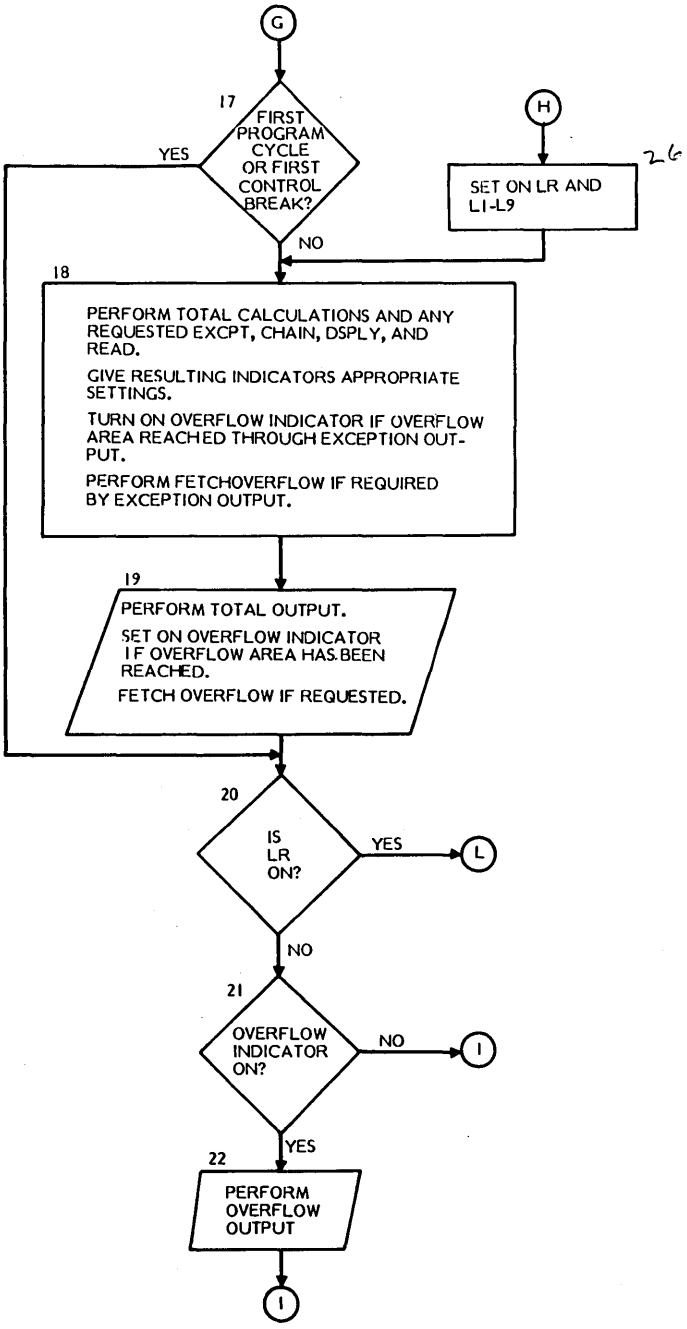


Figure F-1. Program Cycle (Sheet 3 of 4)

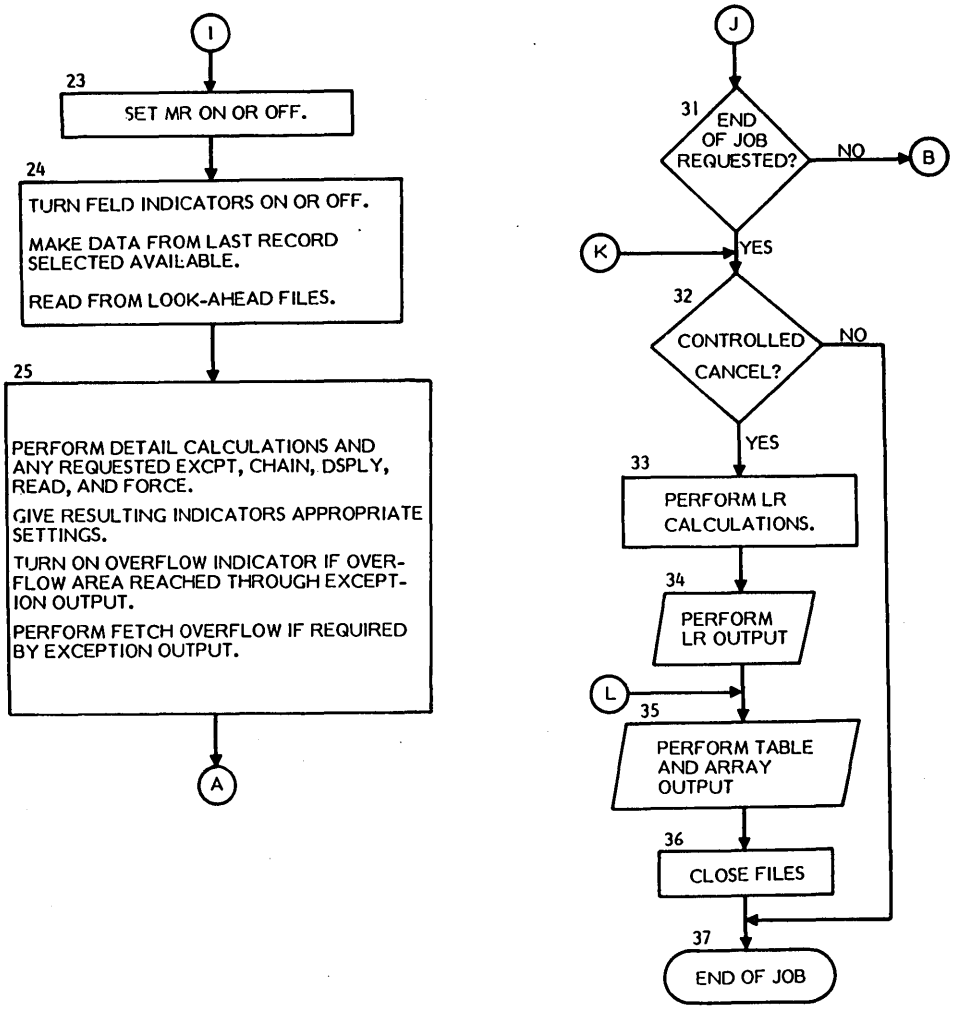


Figure F-1. Program Cycle (Sheet 4 of 4)

1. All data files referenced in the object program are opened; that is, they are made ready for use by the program. All pre-execution time tables and arrays are loaded into computer memory.

2. All output records conditioned by the first page (1P) indicator are produced. These records are written only once per object program execution and, thus, do not fall within the program cycle.

3. The program cycle begins. All heading and detail records whose conditions are satisfied are written out. This output includes records conditioned by an overflow indicator if the overflow routine has been fetched. If the overflow line was reached or if overflow was fetched during this step, the pertinent overflow indicator is turned on.

4. If the overflow line was reached during the most recent step 25, the overflow indicator is turned on. Otherwise, if the indicator did not go on during step 3, the indicator is turned off.

5. Each halt indicator is tested in turn. If they are all off or if each remaining halt indicator is off, control goes to step 6.

5A. An overflow indicator was on. The operator selects one of three options: continue, controlled cancel, or immediate cancel. These options are described in appendix B.

5B. If the operator selects continue, control returns to step 5 where the remaining halt indicators are tested. If the operator elects to cancel, the program is terminated at step 32.

6. All record identifying indicators and indicators 1P, L1 through L9, and H1 through H9 are turned off.

7. The LR indicator is tested. If it is on, a branch to step 26 is taken.

8. The program reads the next record from the file that provided the previously processed record. File

translation occurs at this point, if necessary. If this is the first execution of step 8, one record is read from each input file except forced files and demand files. If this is not the first execution of step 8 and the file has look-ahead fields, the next record of the file is identified only.

9. The program tests to see if the record just read was an end-of-file record. If the end of the file has been reached, a branch to step 11 is executed.

10. When the file was not at end-of-file, the program tests the sequence of the input record according to form I. If the record is in proper sequence, control goes to step 11. Control also goes to step 11 if this is a nonsequential input record but the program was able to identify the record.

10A. A halt occurs. When execution is resumed, control goes to step 31.

11. A test is made to see if end-of-job conditions have been met. The test is successful if all files for which an E was specified in column 17 of form F are at end-of-file, and in this case a branch to step 26 is taken.

12. If the program has multiple input files, the record just read may not be the record selected for processing in the current cycle. Record selection will occur at step 27.

13. With a single input file, the record read will indeed be the record selected for processing in this cycle; but if sequence checking through matching fields has been specified for the file, that checking takes place first at step 30.

14. A record has been selected for processing during this cycle, and the appropriate record identifying indicator is turned on. Data from the record is not made available, however, until step 24.

15. If the record contains control fields, a test for a control break is made. This is accomplished by seeing if the current control fields are not equal to the previously processed control fields. If a control break has not occurred, or if the record does not have control fields, a branch to step 17 is taken.

16. When a control break has occurred, the pertinent control level indicator and all lower level control level indicators are turned on.

17. If this is the first program cycle or first control break, total calculations and output are bypassed by a branch to step 20.

18. All total calculations (conditioned by control level indicators in columns 7 and 8) are performed. If LR is on, operations conditioned by LR are performed after all other total calculations have been performed. Any required file translation is done for exception output, chain, and read operations. Overflow is fetched if it is required by exception output. If exception output causes the overflow line to be reached, the appropriate overflow indicator is turned on.

19. All total output that is not conditioned by overflow indicators is produced. A specified overflow indicator goes on if the overflow line was reached during this total output phase. If LR is on, output records conditioned by the LR indicator are written out after all other total output records have been produced. File translation is performed on total output records, if requested. Overflow is fetched, if required.

20. If LR is on, execution of the object program is concluded at step 35.

21. When LR is not on, the program tests the status of the overflow indicators. If all overflow indicators are off, overflow output is not called for, and control goes to step 23.

22. At this point, all output operations conditioned by positive (on) overflow indicators whose other conditions are also met are performed. File translation is performed on overflow output, if requested.

23. The MR indicator is turned on if this program has multiple input files and the record about to enter processing is a matching record. Otherwise, MR is turned off.

24. Data from the record selected for processing and from specified look-ahead fields is made available for the next cycle. Field indicators are turned on or off, as appropriate.

25. All detail calculations (operations not conditioned by control level indicators in columns 7 and 8) are performed. Any required file translation is done for exception output, chain, and read operations. Overflow is fetched if it is required by exception output. If exception output causes the overflow line to be reached, the appropriate overflow indicator is turned on. Control returns to step 3, where the next cycle is commenced.

26. Control comes here when all files for which an E was specified in column 17 of form F are at end-of-file. The last record indicator (LR) and all control level indicators (L1-L9) are turned on, and processing continues at step 18.

27. If a file has been forced, the next record in that file is, by definition, the next input record to enter processing. The record is selected, and a branch to step 14 is taken.

28. If a file has not been forced, and a record with no matching fields is found in a normal and active input file, that record is selected for processing. Control goes to step 14.

29. When matching fields are specified, the normal file with the highest priority matching fields content is selected. When two or more files have equal matching fields of the highest priority, the record from the highest priority file is selected for processing. (The primary file is always the highest priority file; the order of priority of secondary files is determined by the order in which those files are listed on form F.)

30. The matching fields content of the selected record is compared to the matching fields content of the previously selected record. If the records are in proper sequence (or equal), control returns to step 14.

30A. The program halts. The operator's options are to bypass the out-of-sequence record, by reading the next record from the same file, or to cancel the job. The operator's decision is queried at step 31.

31. This step tests the operator's decision to either bypass an erroneous record or cancel the job. In the case of a bypass, control returns to step 4.

32. If the operator has chosen an immediate cancellation, the job is terminated at this point.

33. With a controlled cancel, all calculation operations conditioned by the LR indicator are performed.

34. All output operations conditioned by the LR indicator are performed.

35. All tables and arrays for which a To Filename was specified on form E are written out. File translation is applied to these tables and arrays, if requested.

36. All files used by the program are closed; for example, any output record still residing in a core buffer is written out.

37. Program execution is ended.

Control may be passed to an assembly language subroutine by an EXIT operation or to a FORTRAN language subroutine by an EXITF operation in a calculation specification. Parameters referenced in the subroutine must be specified by RLABL operations that immediately follow the EXIT operation. A parameter may be a field, table or array, file, indicator, or a literal. The format of these specification statements is given in section 9.

ENTRY

Control is passed to the user subroutine with an RTJ instruction. The entry point must therefore be an ADC 0 instruction for the return address.

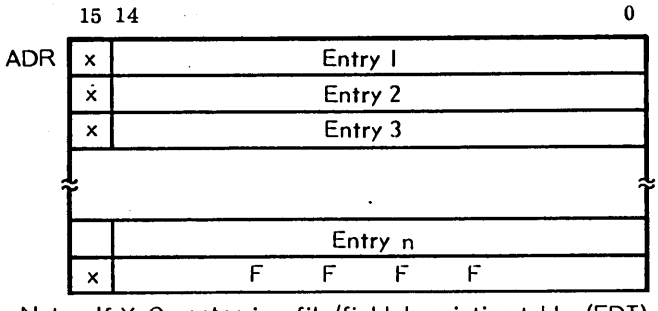
Upon entry the A-register contains a pointer to the parameter list. The parameter list contains one entry for each RLABL statement that followed the EXIT statement in the RPG II source program. The list is terminated by a word with all bits on.

The effective calling sequence is:

LDA = XADR	A register = Address of parameter list
RTJ SUB	Call to user subroutine, entry name SUB

PARAMETER LIST FORMAT

For a file, field, or table/array name in an RLABL statement, the parameter list contains the associated file/field description table entry number (FDT). For an indicator the list contains the indicator number with bit 15 also set on. The list is terminated by a word with all bits set on.



Note: If X=0, entry is a file/field description table (FDT) number
If X=0, entry is an indicator number

0317

The user subroutine may be coded to handle a variable format of parameter lists or may expect a fixed number and form of parameters. In the later case the parameter list should be checked to verify that the correct structure was coded in the RPG II source program.

SUBROUTINES TO SUPPORT USER ASSEMBLY LANGUAGE ROUTINES

The following subroutines are used to support user assembly language routines:

R9MVB Move Bytes

Entry: A-Register = Byte count
Q-Register = Source byte address
I-Register = Destination byte address
RTJ R9MVB Subroutine call
ADC R9BASE Source offset
ADC R9BASE Destination offset

R9MIB Move Bytes Immediate

Entry: A-Register = Byte count
Q-Register = Immediate constant
I-Register = Destination byte address
RTJ R9MIB Subroutine call
ADC R9BASE Destination offset

R9LBY Load Byte

Entry: Q-Register = Byte address
RTJ R9LBY Subroutine call
ADC R9BASE Source offset
Exit: A-Register = Byte (bits 7-0)

R9SBY Store Byte

Entry: A-Register = Byte to store (bits 7-0)
Q-Register = Byte address
RTJ R9SBY Subroutine call
ADC R9BASE Source offset

R9FLDL Get Field Length and Address

Entry: A-Register = File/field description table (FDT) number (bits 0-7)
RTJ R9FLDL Subroutine call

Exit: A-Register = Attribute bits

Bit 15 = Numeric field
14 = Table
13 = Array
12 = Ascending sequence if table or array
11 = Descending sequence if table or array
10 = File
9 = UMONTH, UDAY, UYEAR, or UDATE field
8 = Any form of page field
7-0 = Byte length minus 1 (alphabetic field)
7-4 = Decimal position (numeric field)
3-0 = Digit count (numeric field)

Q-Register = Length
Alphabetic field = Byte length minus 1
Numeric field = Digit count

I-Register = Address
File = Absolute address of DMPT
Field = Byte address relating to RYBASE

R9TSTN Test Indicator

Entry: A-Register: Indicator number (bits 7-0)

```
RTJ   R9TSTN
```

Exit: A-Register: 0 = Indicator on
1 = Indicator off

R9CRIN Clear Indicator

Entry: A-Register: Indicator number (bits 7-0)

```
RTJ   R9CRIN   Subroutine call
```

R9SETN Set Indicator

Entry: A-Register: Indicator number (bits 7-0)

```
RTJ   R9SETN   Subroutine call
```

DATA FORMATS

FILES

Information regarding files is contained in the data management parameter table. This table (DMPT) consists of a prefix of 4 to 14 words plus the table itself. The prefix is addressed as negative offsets from the pointer output from R9FLDL.

FIELDS/TABLES/ARRAYS

The byte address output from R9FLDL is relative to R9BASE. To get the absolute address, the byte address must be shifted right 1 and added to R9BASE. If the unit's position of the byte address is on, the first byte of the field is the right byte of the first word. The first byte is the high order position of the field. This address manipulation is performed by the R9LBY and R9SBY routines.

The table byte address is that of the last looked up entry; the array byte address is that of the first array entry.

Numeric fields are in packed decimal format with one decimal digit per byte. A sign byte is allowed to the right of the low order digit; D is minus, F is plus.

INDICATORS

Numbers 1 through 99 are used for indicators 01 through 99.

This appendix contains two sample RPG II programs. The first example is described in considerable detail as an introduction to RPG II program solutions.

EXAMPLE 1

Example 1 demonstrates the retrieval and updating of records in an indexed sequential file using the CHAIN operation code. The context is an inventory report problem in which a part number field from an input card is used to chain to the indexed sequential disk file. The card record also contains fields that specify how many of a particular part were received, issued, and returned in the transaction period. The card record is identified by an X character in column 80.

The master inventory record for the part contained in the disk file contains an alphanumeric description of the part as well as the number of parts on hand and the minimum number of the particular part that is acceptable for the inventory.

Calculations are performed to determine the new on-hand balance for the part number. That field of the disk file record is updated, and a report describing inventory activity for the month is printed.

If, during the chaining process, an unidentified part number appears at the card reader, that part number is printed in the report with the comment NOT FOUND. A master record that does not contain the record identification code I in the first character position corresponds to a deleted part number and is identified as such in the same printed report.

A new balance that is less than or equal to the minimum acceptable balance for a part number is denoted with the comment ORDER or WARNING, respectively, in the report.

The following pages contain illustrations of the input, update, and output files involved in the program (figure H-1) and the five pages of RPG II coding (figure H-2) required to solve the problem.

FILE DESCRIPTION SPECIFICATIONS

READER is the name of the primary input file assigned to the card reader. This file must be processed to end-of-file.

INVTRY is the chained update file. The file has fixed length records that are 40 characters long. The key field is six characters long and begins in record position 34. The file is processed randomly.

The output file is named PRINT and has overflow indicator OA associated with it.

INPUT SPECIFICATIONS

Records in the file READER are identified by an X in column 80. Indicator 01 is set on when this type of record is being processed.

Active records in the inventory file, INVTRY, are identified by an I in character position 1. Indicator 02 is on during processing of this type of record. Any other type of record in the same file turns on indicator 77.

CALCULATION SPECIFICATIONS

At line 010, the PARTNO field is chained to the file INVTRY. The CHAIN is conditioned by the presence of a valid input card, designated by indicator 01. If the CHAIN fails to yield the desired record, indicator 88 is set on. Otherwise, indicator 02 is on in the next detail cycle.

When indicator 02 is on, all the fields needed to perform the inventory calculations are present, and those calculations are performed in lines 020 through 050. The field WORK receives the result of the calculations.

At line 060, the test is made to determine if more parts need to be ordered. Indicator 11 is turned on if the balance is equal to the order point, and indicator 10 goes on if the part definitely needs to be ordered.

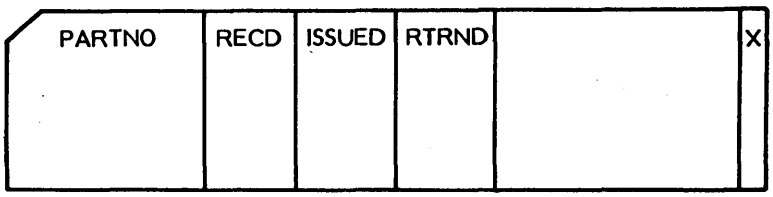
OUTPUT SPECIFICATIONS

Heading lines for the file PRINT are produced if either the first page (1P) indicator or the overflow (OA) indicator is on. Updated page numbers are printed at the top of each page.

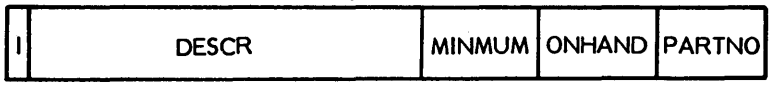
The detail record described in lines 170 through 270 is printed only if both indicators 01 and 02 are on. The individual comment fields ORDER and WARNING are conditioned additionally by the indicators 10 and 11, respectively.

Lines 280 through 310 describe the exception line, which is printed for either an unidentified part number or an inactive master record.

Lines 320 and 330 provide the updating of the master file. Update occurs only if both indicators 01 and 02 are on, showing that both a primary and secondary record were received for the part number and that the calculations were performed. Other fields of the master file record are not listed on the Output Specifications form, as they are not changed.



READER FILE



INVTRY FILE

INVENTORY 07/04/76 PAGE 1

PART NUMBER	PART DESCRIPTION	MINIMUM BALANCE	PREVIOUS BALANCE	RECEIVED	ISSUED	RETURNED	NEW BALANCE	
220260	ITEM 1	50	55		5		50	WARNING
220261	ITEM 2	1,000	1,006	100	30		1,076	
220264	ITEM 3	750	600	100	25	50	725	ORDER
267890								NOT FOUND
311235								DELETED

0334

PRINT FILE

Figure H-1. Input and Output Formats

CONTROL DATA CORPORATION

RPG CONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS

Program: EXAMPLE 1 | Programmer | Date | Punching Instruction | Graphic | Punch | Card Electro Number

Page 01 of 5 | Program Identification: EXAMP1

Control Card Specifications

Line	Form Type	Size to Compile	Object Output	Listing Options	Size to Execute	Debug	MFCM Stacking Sequence	Date Format	Date Edit	Inverted Print	360/20 2501 Buffer	Number Of Print Positions	Alternate Collating Sequence	Model 20: Address to Start	Work Tapes	Overlay Open	Overlap Printer	Binary Search	Tape Error	2152 Checking	Inquiry	Model 20: Read/Write/Compute	Keyboard Output	Sign Handling	1P Forms Position	Indicator Setting	File Translation	Punch MFCU Zeros	Nonprint Characters	Table Load Halt	Shared I/O	Field Print	Formatted Dump	RPG to RPG II Conversion	Refer to the specific System Reference Library manual for actual entries
01	H																																		

File Description Specification

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V/S/M/D	Block Length	Record Length	L/R	A/P/T/K	I/D/T or 2	Overflow Indicator	Key Field Starting Location	Extension Code	E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	Continuation Lines: K	Continuation Lines: Option	Continuation Lines: Entry	A/U	File Addition Unordered	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	File Condition U1 U8	R/U/N
02	F	READER	I	P	E		F		80								READER																
03	F	INVTRY	U	C			F		40	R	CA	I		34			DISK																
04	F	PRINT	O				F		136				OA				PRINTER																
05	F																																

CONTROL DATA CORPORATION

RPG INPUT SPECIFICATIONS

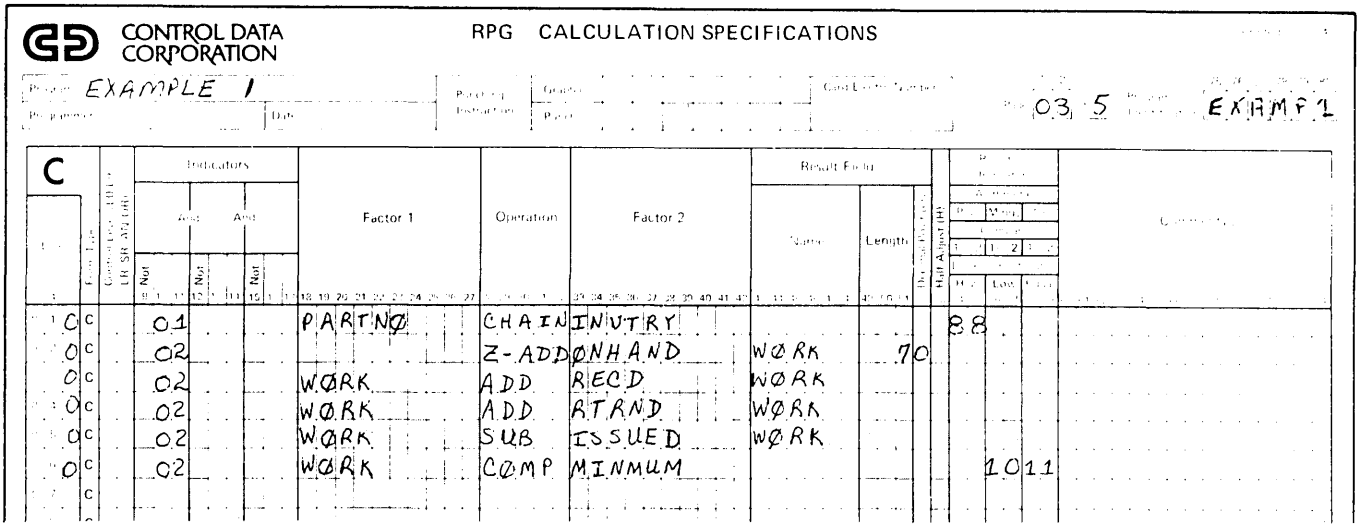
Program: EXAMPLE 1 | Programmer | Date | Punching Instruction | Graphic | Punch | Card Electro Number

Page 02 of 5 | Program Identification: EXAMP1

I

Line	Form Type	Filename	Sequence (AND/OR)	Number (1 N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank
01	I	READER	XX			C1	80		C	X																				
02	I																				1	6		ORTNO						
03	I																				7	10		RECD						
04	I																				11	14		ISSUED						
05	I																				15	18		RTRND						
06	I	INVTRY	YY			02	1		C	I																				
07	I																				2	21		DESCR						
08	I																				22	27		MINMUM						
09	I																				28	33		ONHAND						
10	I		ZZ			77		N	C	I																				
11	I																													

Figure H-2. Example 1 Coding (Sheet 1 of 3)



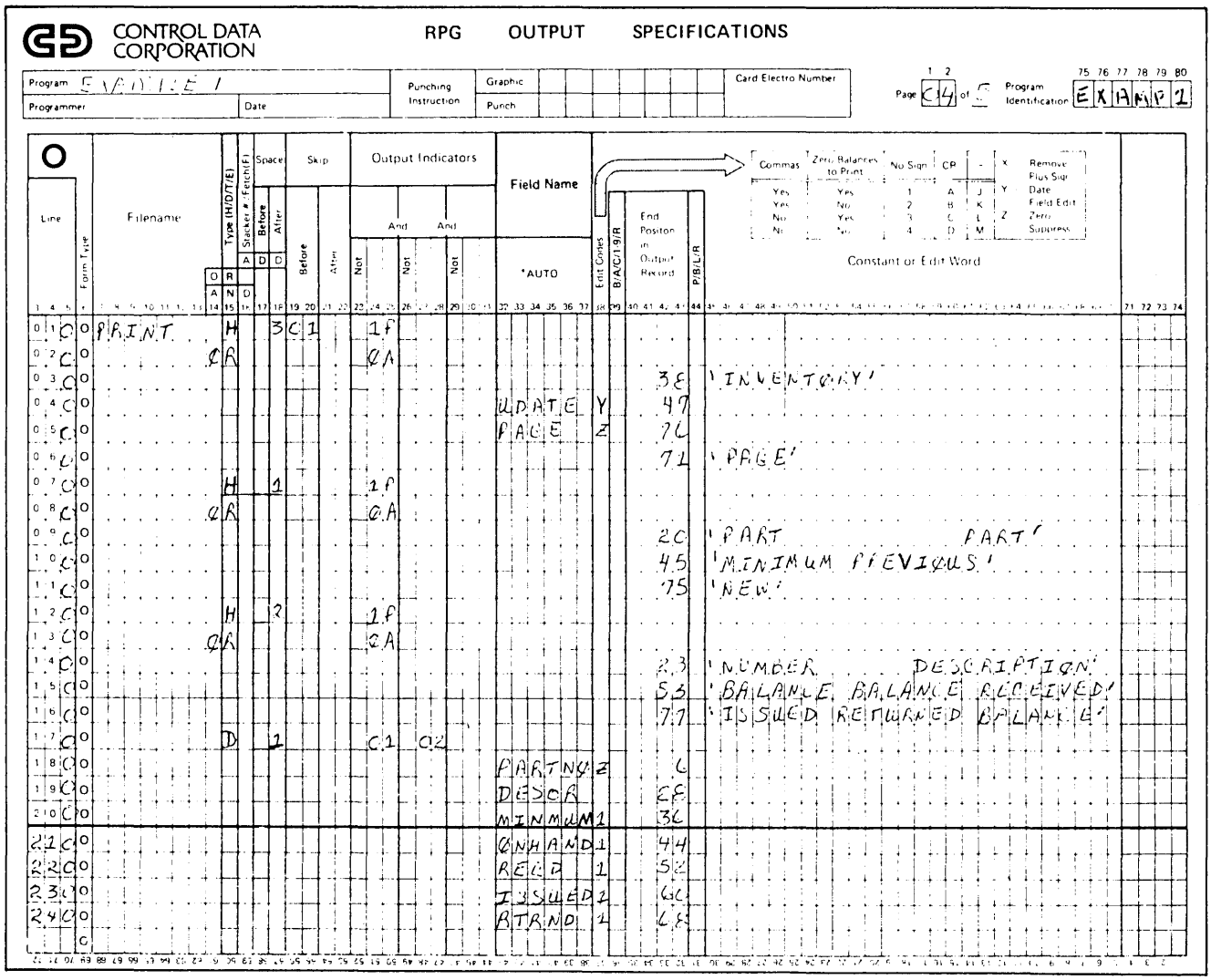
CONTROL DATA CORPORATION

RPG CALCULATION SPECIFICATIONS

Program: EXAMPLE 1 | Programmer | Date | Punching Instruction | Graphic | Punch | Card Electro Number | Page 03 5 | Program Identification EXAMP1

C

Line	Form Type	Control Level (L0-L9, LR, SR, AN, OR)	Indicators: And Not	And Not	And Not	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Plus	Minus	Zero	Comments
1	C		01			PARTNØ	CHAIN	INVTRY					88			
0	C		02				Z-ADD	ØNHAND	WØRK	70						
0	C		02			WØRK	ADD	RECD	WØRK							
0	C		02			WØRK	ADD	RTRND	WØRK							
0	C		02			WØRK	SUB	ISSUED	WØRK							
0	C		02			WØRK	COMP	MINMUM					1011			
	C															



CONTROL DATA CORPORATION

RPG OUTPUT SPECIFICATIONS

Program: EXAMPLE 1 | Programmer | Date | Punching Instruction | Graphic | Punch | Card Electro Number | Page 04 of 5 | Program Identification EXAMP1

O

Line	Form Type	Filename	Type (H/D/T/E)	Stacker # / Fetch(F)	Space: Before	Space: After	Skip: Before	Skip: After	Output Indicators: And Not	And Not	And Not	Field Name	Edit Codes	B/A/C/Z/B/R	End Position in Output Record	P/B/L/R	Constant or Edit Word
01	O	PRINT	H		3	C 1			1P								
02	O			ØR					ØA								
03	O														3E		'INVENTORY'
04	O											UDATE	Y		47		
05	O											PAGE	Z		76		
06	O														71		'PAGE'
07	O		H		1				1P								
08	O			ØR					ØA								
09	O														2C		'PART PART'
10	O														45		'MINIMUM PREVIOUS'
11	O														75		'NEW'
12	O		H		2				1P								
13	O			ØR					ØA								
14	O														23		'NUMBER DESCRIPTION'
15	O														53		'BALANCE BALANCE RECEIVED'
16	O														71		'ISSUED RETURNED BALANCE'
17	O		D		1				C1	02							
18	O											PARTNØ	Z		6		
19	O											DESCR			2F		
20	O											MINMUM	1		36		
21	O											ØNHAND	1		44		
22	O											RECD	1		52		
23	O											ISSUED	1		60		
24	O											RTRND	1		68		
	O																

Figure H-2. Example 1 Coding (Sheet 2 of 3)

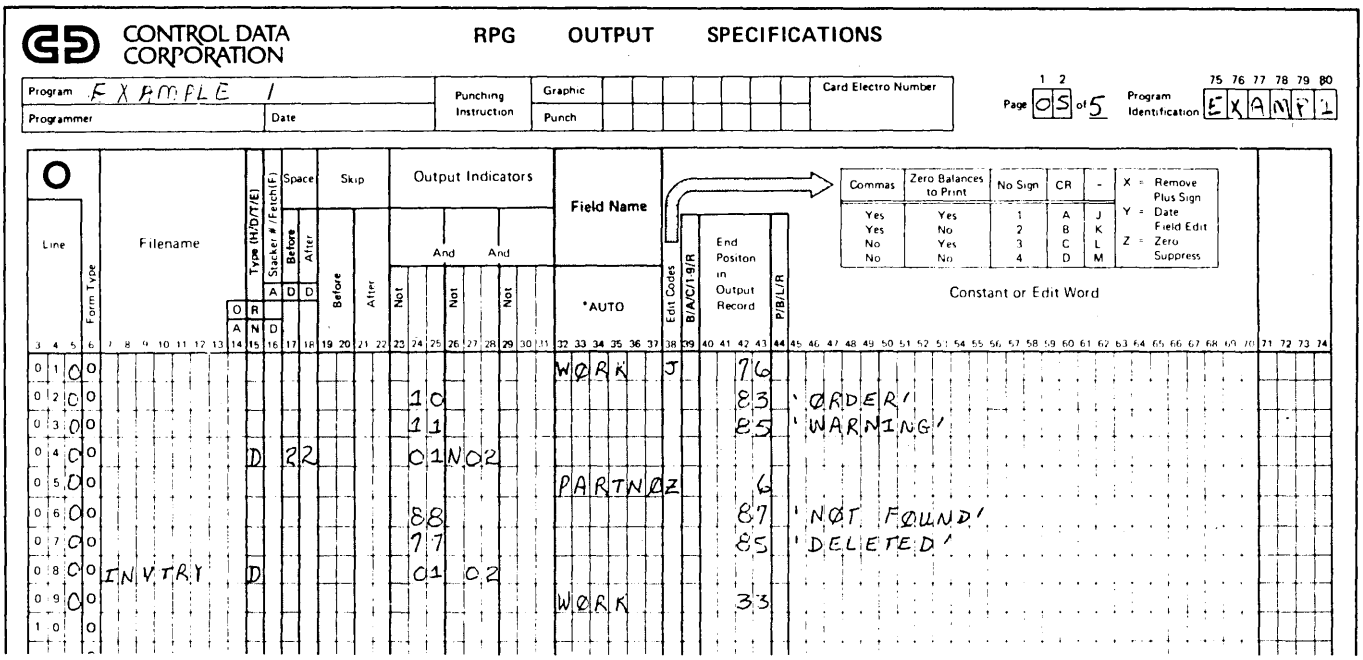


Figure H-2. Example 1 Coding (Sheet 3 of 3)

EXAMPLE 2

In example 2 the CHAIN operation code is used to create (load) a direct disk file. The disk file, OFFEXT, is a file of telephone extensions in the plant; the file is ordered by extension number, and each record contains the office number and employee number corresponding to each extension. OFFEXT is described on the File Extension Specifications form as a chained output file.

Records used to create OFFEXT are read from the card reader. The file is initialized to blanks. The key field is contained in positions 1 through 3 of each input record. Other character positions, although loaded to the file, are not described individually.

Figure H-3 shows the RPG II coding sheets required for example 2.

CONTROL DATA CORPORATION

RPG CONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS

Program EXAMPLE 2 | Punching Instruction: Graphic / Punch | Card Electro Number

Programmer | Date

Page 01 of 04 | Program Identification EXFT32

Control Card Specifications

H

Line	Form Type	Size to Compile	Object Output	Listing Options	Size to Execute	Debug	MFCM Stacking Sequence	Date Format	Date Edit	Inverted Print	360/20 2501 Buffer	Number Of Print Positions	Alternate Collating Sequence	Model 20: Address to Start	Model 20: Work Tapes	Model 20: Overlay Open	Model 20: Overlap Printer	Model 20: Binary Search	Model 20: Tape Error	Model 20: 2152 Checking	Model 20: Inquiry	Model 20: Read/Write Compute	Model 20: Keyboard Output	Sign Handling	1P Forms Position	Indicator Setting	File Translation	Punch MFCU Zeros	Nonprint Characters	Table Load Halt	Shared I/O	Field Print	Formatted Dump	RPG II Conversion	Refer to the specific System Reference Library Manual for actual entries
01	H																																	

File Description Specification

F

Line	Form Type	Filename	File Type	File Designation	End of File	Sequence	File Format	Block Length	Record Length	Mode of Processing	Length of Key Field or of Record Address Field	Record Address Type	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	Continuation Lines: K	Continuation Lines: Option	Continuation Lines: Entry	A/U	File Addition Unordered	Number of Tracks for Cylinder Overflow	Number of Extents	R/U/N	Tape Rewind	File Condition U1 U8
02	C	INPUT	I	P	E		F		80								READER																
03	C	OFFSXT	O	C			F		80								DISK																
04			F																													
05			F																													

CONTROL DATA CORPORATION

RPG CALCULATION SPECIFICATIONS

Program EXAMPLE 2 | Punching Instruction: Graphic / Punch | Card Electro Number

Programmer | Date

Page 03 of 04 | Program Identification

C

Line	Form Type	Control Level (L0-L9, LR, SR, AN, OR)	Indicators: And Not	And Not	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Decimal Positions	Half Adjust (H)	Plus	Minus	Zero	Comparison 1	Comparison 2	High	Low	Equal	Comments
01	C				PHONE	CHAIN	OFFEXT													
02	C																			

Figure H-3. Example 3 Coding (Sheet 1 of 2)

CONTROL DATA CORPORATION

RPG INPUT SPECIFICATIONS

Printed in U.S.A.

Program EXAMPLE 2 | Programmer | Date | Punching Instruction | Graphic | Punch | Card Electro Number | Page 02 of 4 | Program Identification

I

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes: 1 Position	1 Not (N)	1 C/Z/D	1 Character	2 Position	2 Not (N)	2 C/Z/D	2 Character	3 Position	3 Not (N)	3 C/Z/D	3 Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Minus	Zero or Blank
01	I	INPUT	AN			C1																								
02	I																				1	30		PHONE						
03	I																				1	80		DATA						
04	I																													

CONTROL DATA CORPORATION

RPG OUTPUT SPECIFICATIONS

Program EXAMPLE 2 | Programmer | Date | Punching Instruction | Graphic | Punch | Card Electro Number | Page 04 of 4 | Program Identification

O

Line	Form Type	Filename	Type (H/D/T/E)	Stacker #/Fetch(F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And Not	And Not	And Not	Field Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R
01	O	OUTEXT	D						C1							
02	O											DATA			F.C.	
03	O											*AUTO				

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

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

Constant or Edit Word

Figure H-3. Example 3 Coding (Sheet 2 of 2)

Table I-1 specifies the user of the utilities described in this appendix. Disk type utilities are supplied by UTIL, which is described in the ITOS reference manual.

TAPE MOUNT UTILITY

NOTE

This utility operates under ITOS only from the master terminal. It does not operate from other ITOS terminals.

The tape mount utility (MOUNT) runs under MSOS to define all tapes that are used by a subsequent RPG II program.

MOUNT builds records into an indexed MSOS file manager file that is indexed-linked, first-in, first-out. The key corresponds to the filename on an RPG F statement. Groups of records with the same keys correspond to volumes within a multivolume file. The file number of this file manager file is defined by the INIT utility and is named MOUNTFIL. Every time MOUNT is executed, the file is rewritten from the beginning.

The utility is called in by the MSOS job processor command:

```
*MOUNT
```

The utility is called in ITOS console mode in reply to REQUEST = :

```
MOUNT, parameter list
```

The format of the commands read from the MSOS standard input device by MOUNT is a string of parameters separated by commas, in any order (see table I-2). The first letter of the parameter signifies its meaning.

The last MOUNT command must be followed by /* in input columns 1 and 2.

A MOUNT command is required for each tape file in the RPG II program, including unlabeled tapes.

An RPG program has seven F statements referring to files. The following is a sample MOUNT program:

```
*MOUNT
FTAPE1
FTAPE2
FTAPE2
FTAPE3,B,P
FTAPE4,S,V123,DNEW.DATA
FTAPE5,S,V111111,DMYDATA
FTAPE5,S,V222222,DMYDATA
FTAPE6,S,OUT.DATA,V333333
FTAPE7,V345,DUPDATE,E76300
/*
```

TABLE I-1. UTILITIES USED WITH RPG

Utility	System	Comments
MOUNT	ITOS, console mode	Cannot be used except at master terminal in ITOS
CATLOG	ITOS, batch mode	Cannot be used except at master terminal in ITOS
SWITCH	ITOS, terminal mode	Note difference in way utility is called
RBDPCH	ITOS, batch mode	ITOS current does not support punch type hardware
DSORT	ITOS, procedure stream mode	
UTIL	ITOS, terminal mode	
EDITOR	ITOS, terminal mode	

TABLE I-2. MOUNT UTILITY PARAMETERS

First Letter	Subfield	Definition
A		Add records to an existing file.
B		Bypass label processing.
C	A, B, or E	Code of tape: ASCII, BCD, or EBCDIC.
D	Name; up to 17 characters long	Dataset name on HDR1 label
E	Date; in form yyddd	Expiration date of file (yy=year; ddd=day)
F	Name; up to eight characters	File name, corresponding to RPG F statement
L		Leave tape in position at close time
N		No label processing
P		Protect file. Informs operator to remove write ring.
R		Rewind tape at close time.
S		Standard label processing
U		Unload tape at close time.
V	Number; up to six characters long	Volume serial number of tape

NOTES:

1. F is compulsory.
2. Defaults are CA (ASCII code), N (no labels), and R (rewind). The rest are optional.
3. B, N, and S are mutually exclusive.
4. L, U, and R are mutually exclusive.
5. D, E, and V make sense only in conjunction with S.

Where:

TAPE1 is a non-label tape.

TAPE2 is a non-label tape on two volumes.

TAPE3 has a nonstandard label; therefore the label must be bypassed. Mount without a ring.

TAPE4 is an input file on a standard labeled tape. The serial number of the tape is 000123 and the data set name is NEW.DATA.

TAPE5 is an input file on two volumes with standard labels. The volume serial number of the first tape is 111111 and the second 222222. The data set name is MYDATA.

TAPE6 is an output file to be written on an SL tape. The data user name is OUT.DATA, VOLUME 333333.

TAPE7 is an output file to be written on an SL tape. The volume serial number is 000345. The data set name is UPDATE AND EXPIRATION DATE 76300 (day 300 in year 1976).

CATALOG UTILITY

NOTE

This utility operates under ITOS only in batch mode from the master terminal. It does not operate from other ITOS terminals.

The catalog utility (CATLOG) installs the generated RPG object program in the MSOS program library under the name given in the RPG program identification field (columns 75-80 of Common Entries).

The RPG object program will be link-edited and installed in the program library as shown in figure I-1 and described below:

1. A calling routine with the entry point the same as the RPG program name. The calling routine is stored in relocatable binary format.

2. A file containing the ROOT and initialization modules, named the same as the RPG program name with the last character of the name replaced by 0.

3. A file containing the input, calculation, output, external root, and tables. It will be named the same as the RPG program name with the last character replaced by a 1.

4. A file containing the cleanup routines, named the same as the RPG program name with the last character replaced by a 2.

Because CATLOG generates three overlays, the amount of core required to execute a program that had been cataloged is less than for direct execution from the load-and-go file.

The cataloging task takes three steps. See figure I-2 and the following description.

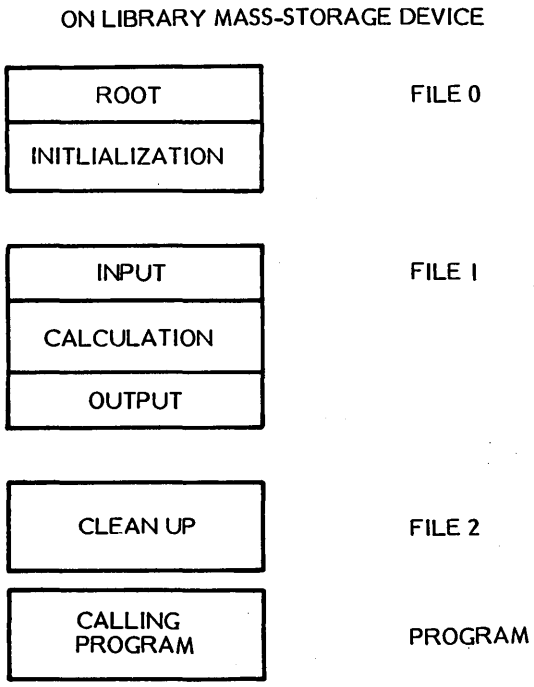


Figure I-1. RPGII Program Link-Edited and Loaded into Program Library

Step 1 – The RPG II compiler stores the object code on the scratch mass storage unit beginning at the first scratch sector. The compiler generates five object modules for each RPG program (initialization, input, calculation, output, and cleanup). Each object module begins with a NAM block and terminates with an XFR block compatible with the MSOS LOADER. The CATLOG program takes the relocatable object code generated by the RPG II compiler as input.

Step 2 – The CATLOG program inputs the ROOT and calling program from the program library, inputs the object program from scratch mass storage, and outputs the library edit load file. The load file is input to the MSOS library editor (LIBEDT) to install the RPG object program as three overlays and a calling program.

The prototype of the calling program object code is modified to have the declared RPG program name as its entry point and to use the file names as described.

Step 3 – LIBEDT is called to link-edit and load the three files and the calling program into the program library.

The catalog capability requires the use of a read/write sequential access device. Either a magnetic tape or a pseudo tape (in conjunction with the job file handler (JFH)) and the file manager in MSOS satisfy the requirement. The job control statements vary depending on which device is used.

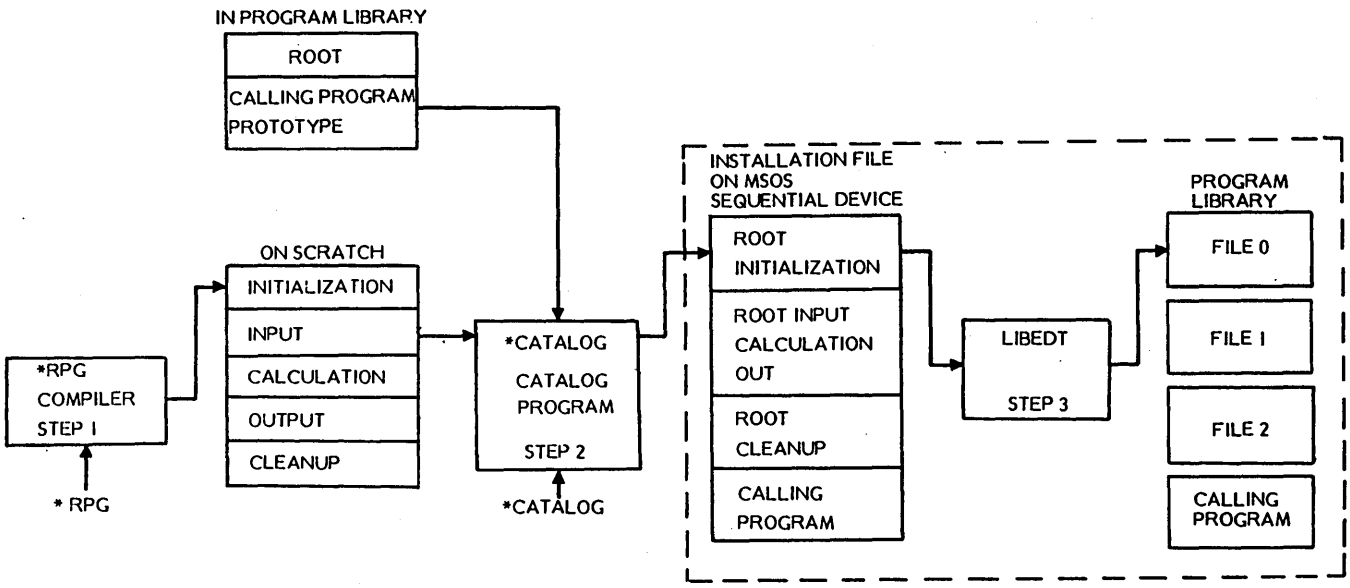


Figure I-2. Flow of Compiling/Cataloging an RPG II Program

The job control statements for magnetic tape are as follows (it is assumed that logical unit 6 is a magnetic tape):

Control Statement	Step	Comments
*JOB,RPG,CATLOG	1	Job control
*REW,6	2	Rewind the tape (unit 6).
*K,P6	3	Assign output to the tape.
*RPG	4	Execute the RPG compiler.
-RPG SOURCE- *CATLOG	6	Take the generated object code from scratch disk and create an install file on unit 6.
*V,6,B	7	Install the program on unit 6 in the program library. (Unit 6 has all the control statements to accomplish this.)
*REW,6	8	Rewind unit 6.
-END OF FILE-	9	End of job

The job control statements for pseudo tape are a follows (it is assumed that logical unit 7 is a pseudo tape):

Control Statement	Step	Comments
*JOB,RPG,CATLOG	1	Job control
*DEFINE,TAPE1,RPG	2	Define a scratch tape
*		
*OPEN,TAPE1,RPG, W,7	3	Open with read/write option as LU7.
*K,P7	4	Make the binary output unit 7.
*RPG	5	Execute the RPG compiler.
-RPG SOURCE- *CATLOG	6	Take the generated object code from scratch disk and create an install file on unit 7.
*V,7,B	7	Install the programs on unit 7 in the library.
*RELEAS,TAPE1, RPG	8	Release the scratch tape and its space.
-END OF FILE-	9	End of job

SWITCH UTILITY

The SWITCH utility is either called from a process stream or it is called from a terminal in response to REQUEST=. In the process stream case, the utility call has the form:

```
SWITCH
xxxxxxxx
```

Where: xxxxxxxx is the switch setting (0=off, 1 = on).

If called from the terminal in response to the REQUEST= option, after SWITCH is requested, the displayed message is returned to the operator:

ENTER SWITCH VALUES xxxxxxxx

The operator replies by specifying the position of each switch with a zero or a one.

BINARY PUNCH UTILITY

NOTE

This utility is not currently used for ITOS since the ITOS hardware configuration does not include punched output equipment.

The RPG II object program may be punched from MSOS load-and-go scratch mass storage by the binary punch (RBDPCH) routine.

The object output is compatible with the MSOS and the RTOS relocatable linking binary loader and optionally includes necessary RPG II runtime routines (copied from the MSOS program library).

Input is read from the standard input device and is terminated by an *T record or an end-of-file. Output is always to the standard punch unit ($FA), which may not be the same as the input unit. Output is terminated with an *T, if read, or an end-of-file.

Relocatable binary output is copied directly from the scratch unit to the punch unit. External references cause the corresponding programs to be copied from the program library.

The RBDPCH utility produces a deck of relocatable binary programs and subroutines that may then be loaded and linked into an executable program. A tape or pseudo tape or other MSOS binary output media may be used instead of punched cards. The prerequisite for using RBDPCH is that the object program(s) must be on disk scratch (LGO file) and must be terminated by an *T. The runtime routines necessary for the operation of the program(s) on LGO must be in the program library

The MSOS job control statements to punch the object code following compilation are:

*K,Pn

*RBDPCH,x

Where: n is the logical unit used to punch the resulting binary decks

x is a suppression flag. If x is any nonblank character, the referenced library programs are not punched. Trailing commas can be omitted if x is blank.

A leading message is printed on the line printer:

*** UNLINKED RBD PROGRAM AND SUBPROGRAMS ***

***

and is followed by the names of all the programs and subprograms as they are output to the punch unit.

A message is printed with the name of the missing entry point for each external reference that cannot be matched by a corresponding entry point in the program library.

Following the last subprogram, an *T is punched and a termination message appears on both the line printer and comment device:

RBD FILE FINISHED

and control returns to the job processor.

See the RTOS reference manual for information on loading the deck into the RTOS System

The following core-resident entry point names normally appear as missing externals:

- LOGIA
- FMPFLG
- RPFDIR
- SYSID
- R9SWCH
- DAYTO
- MONTO
- YERTO

Dummy programs (if any) used to equivalence these entry points must be removed from the library prior to executing RBDPCH or deleted from the output.

SORT UTILITY

A call to the sort utility allows one or more files to be sorted in order by one or more key fields into a single output file. The sort utility may be called only from a procedure stream and may not be called from within an RPG II program. The sort utility has the following capabilities:

- One or more files composed of records of equal length can be sorted. If more than one file is sorted in a single run, the records are merged to form a single sorted output file.
- The key fields upon which the files are sorted may be of any length, may overlap, and (if more than one key is specified) define a hierarchy, that is, a secondary sort within a primary, a third order sort within a secondary sort, and so forth.

- The user may select whether the sorted output file will contain (1) the full input records (tagalong sort) or (2) only the portion of the record that remains after the key fields have been removed (data sort) or (3) only the relative record numbers of the input file records (ADDROUT sort).
- Records may be selectively included in or excluded from the output file based on whether a single key field within the record satisfies logical relations specified by the user by the relational operators =, ≠, <, >, ≤, and ≥ relating the key field to another key field or to a constant.
- Data in the input records and the normal collating sequence are assumed to be ASCII. However, the user may select that EBCDIC code be used instead.

For a more complete description of the sort utility, its operation, and its use, the reader should refer to the ITOS reference manual.

UTIL

UTIL is a file-oriented utility that can be used to control equipment to be used, to list files, and to set up, change, and compress, or release files. UTIL can also be used to bring disk packs online and to load copy, or dump files.

The file manager utilities operate under the direction of an executive, UTIL, that is called through the ITOS executive. UTIL reads an individual request processor into main memory to process each UTIL command.

UTIL operates in one of two modes:

- When interactive mode, UTIL processes commands entered by the operator through the CRT terminal.
- When in procedure stream mode (entered by keying REQUEST = CARDPRO), UTIL processes commands encountered in the input stream being read on the device previously assigned by the INPUT= command.

For a more complete description of UTIL, its operation, its use, and the commands, refer to the ITOS reference manual.

TEXT EDITOR

The text editor provides line-by-line (record oriented) editing of sequential and direct files that contain only ASCII text (or blank filled records, in the case of direct files). The text editor is used to add or change records in existing file-manager files or in files previously created by the text editor. Thus the editor is well suited for creating and maintaining RPG II source files. The editor can only be used interactively; all entries are made through the CRT terminal. The following types of operations can be performed on a file:

- Change a single line
- Change a specific character string in one or more lines. The user specifies the old string of characters, the new string that is to replace it, and the range of lines in which the change is to be made.
- Add a line with a specified line number
- Add one or more lines with automatic line numbers. The text editor assigns line numbers in ascending order.
- Delete a line
- Clear all references to the current file
- List records on the CRT terminal within a range of records specified by the user
- Resequence the records in the file by assigning new sequence numbers
- Set tab stops for use with auto mode line entry

For a more complete description of the text editor, its capabilities, operation, and restrictions, the reader should refer to the ITOS reference manual

TABLE I-3. CONTROL STATEMENTS FOR SMC2

	Statement
Call	
SMC2	Sorting procedure stream begins
RUN Statement	
D,WKBKSZ,S/N,KEYCNT,FILCNT,	D = Disk sorting operation WKBKSZ = Size of working area required S/N = Sequence checks selected or ignored KEYCNT = Number of search keys FILCNT = Number of files to be sorted
KEYS Statement	
...C,A/D,KEYCOL,KEYCOLS,...	One sequence of these four parameters per key C = Character type records A/D = Ascending or descending order KEYCOL = Relative position of first character of key in record (starts with 1) KEYCOLS = Number of characters in keyword
INFILE Statement	
D,FILNAM,RECLTH,BLKSIZ,SKIPCNT,DOCNT,	One INFILE statement per input file D = Disk type records FILNAM = File name (eight characters) RECLTH = Record length (words) BLKSIZ = Block size (1000 words, nominal) SKIPCNT = Number of records skipped prior to first record to be sorted DOCNT = Number of records to process in this file
OUTFILE Statement	
D,FILNAM,LUN,BLKSIZ,	D = Disk type output file FILNAM = Output file name (eight characters) LUN = File manager logical unit BLKSIZ = Block size

S/N selects or rejects sequence checking of all merged outputs whether intermediate or final (usually sorting is done in stages and the sorted outputs from groups of records are successively merged into larger and larger groups until all of the records are in a single group). S selects sequence checking. This should be the normal selection of this parameter. Use of sequence checking guards against errors otherwise undetected during I/O transfers. A sequence error is treated as a fatal error and the sorting run is aborted. N suppresses sequence checking.

KEYCNT is the number of keys to be used for the sorting run. SMC2 will not finish processing the KEYS statement until the number of keys specified by KEYCNT has been found and processed. If this number is not found, the run is aborted and the fatal error message is delivered. If the input file is an indexed file, the key fields do not have to agree with the key fields used by the file manager.

FILCNT is the number of input files to be used for the sorting run. SMC2 will continue to process INFILE statements until it has found the total number of INFILE statements specified by the number in this parameter. If fewer or more INFILE statements are encountered, SMC2 aborts the sorting run. SMC2 processes input files in the same order that they were specified by the INFILE statements.

KEYS STATEMENT

This statement defines each key to be used. Each key requires four consecutive parameters. The order of the keys in the KEYS statement determines the sorting hierarchy: the first key presented is the primary key. Records are first sorted by this key. Then records are sorted according to the second key presented. Then records are sorted by the third key presented, etc., until records have been sorted by each of the specified keys.

The format for each of the n keys used in the sorting process is:

C,A/D,KEYCOL,KEYCOLS,

Where: C indicates that the record to be formatted is a character type record.

A/D A signifies that ascending collating order is used;

D signifies that descending collating order is used.

KEYCOL is the character position within the record for the first character of the key. The first character position of the record is numbered 1.

KEYCOLS is the length of the key in characters. The minimum length is a single character.

INFILE STATEMENT

Each file used in the sorting operation must be defined by its own INFILE statement. The order of the INFILE statements determines the order in which files are read when the sorting operation is executed. The format of the INFILE statement is:

D,FILNAM,RECLTH,BLKSIZ,SKIPCNT,DOCNT.

Where: FILNAM is an eight-character file name. The owner name is fixed by the USER ID which was supplied by the operator when he logged onto the controlling terminal. Volume name is fixed by the fact that the OUTFILE must occur on the same file where the input files are stored.

RECLTH is the number of words per record. RECLTH must be an even divisor of WKBKSZ and BLKSIZ. Note that RECLTH must be the same for every INFILE statement, and the length equals or exceeds the length of the largest key (that is, keys must be entirely contained within a single record). For a 32K word computer, maximum RECLTH is about 2,000 words.

BLKSIZ is the number of words that SMC2 reads during a read request to a file. It may differ for each INFILE statement, but each BLKSIZ must be a nonzero multiple of RECLTH. At run time, SMC2 checks the length of each block to be read to verify that the actual BLKSIZ is a nonzero multiple of the specified RECLTH and to verify that the actual BLKSIZ does not exceed the available memory storage. Error messages are generated if BLKSIZ is improperly designated. Depending on other SMC2 parameters, the maximum BLKSIZ for a 32K word machine is about 1,000 words. Other parameters, however, diminish this value.

SKIPCNT is the number of records at the beginning of the file to be skipped prior to processing the first record for that file. Its value ranges from 0 to 99,999,999.

DOCNT is the number of records to be processed in this file. Its value ranges from 1 to 99,999,999. DOCNT and SKIPCNT together allow the programmer to select a single contiguous block of records from a file, to omit records preceding the desired records, and to omit records following rhe desired records. If DOCNT equals E, all records are to be processed.

OUTFILE STATEMENT

The OUTFILE statement defines the output file. The format of the statement is:

D,FILNAM,LUN,BLKSIZ.

Where: D indicates the file is a disk file type (that is, a file manager file).

FILNAM is the eight-character output file name. Its name follows the same rules as the FILNAM parameter in the INFILE statement.

LUN is the file manager logical unit where the output file is to be defined. This unit corresponds to an assignment in SYSDAT of the file manager logical units, which are numbered (on a maximum hardware configuration) from 1 through 8. SYSVOL is always numbered 1. SMC2 converts this LUN value to a volume name as a function of the location where the volume is mounted. The operator must not use a volume designation for LUN. The LUN parameter assures

BLKSIZ is defined as in the INFILE statement.

RECLTH is inferred from the INFILE statements. RECLTH must be an even divisor of BLKSIZ.

Example:

Procedure Stream	Comments
SMC2	Procedure call
D,318,S,3,1	WKBKSZ = 6xRCDLTH, checking requested, three keys, one file
C,A,1,7,C,A,8,1,C,A,9,2,	Ascending order, keys start at positions 1, 8, and 9
D,NEWORDSØ,53,318,O,E,	RECLTH = 53, BLKSIZ = 6x53, no records skipped, sort all records
D,SNEWORDØ,1,318,	Output on SYSVOL (FM LUN1) BLKSIZ = 6xRECLTH

The RPG II Trace feature is provided to assist the analyst in locating programming errors and system problems. It is used primarily during RPG II application program development. It is not intended as a training aid. There are three types of trace:

- Trace compiler interpreter
- Trace compiler object code generation
- Trace object code execution (run time)

The trace feature requires that a special control specification be included immediately after the H-specification control card when the RPG II source program is compiled:

Column 6	X
Column 8	1 if object code execution trace required; blank if not required.
Column 10-13	Source statement number (with leading zeros); starts compiler interpreter trace, blank if not required
Columns 20-23	Source statement number (with leading zeros); starts compiler object code generation trace, blank if not required
Column 27-72	Program identification (optional; not used by trace)

The most common form is for execution time trace only: column 6 = X, column 8 = 1, and all other columns are blank. Specifying execution time trace adds about 800 words to the program main memory size.

NOTE

The program must be recompiled and recataloged before execution in normal mode.

TRACE OUTPUT FORMAT

The following examples illustrate the trace output format. Actual output on a user system may differ slightly.

COMPILER RUN-TIME TRACE

This mode is selected if column 8 = 1 on the X specification. This mode causes the trace support routine to be included when the program is cataloged. There is no special output at compile time, but a program cannot be traced at run time unless it was compiled with this option selected on the X control statement.

At run time the trace must be enabled by first executing the TRACER program and then entering one of the following options:

Option	ITOS/Terminal Mode
0	No trace, turn off E option
1	Trace at terminal
2	Full trace at terminal
3	Full trace on printer
E	Halt on any error condition
S3	3segment multiuser mode for RPG II
S5	5segment multiuser mode for RPG II

The previous option is assumed (initially zero) unless the trace utility is called, which allows the option to be entered as above. Either the terminal or the printer may be specified. The full trace includes output of a mnemonic for each of the calculation section operations actually executed.

Indicators are printed wherever a change in status of the indicator occurs (full trace only). Messages indicate the start of each portion of the RPG II execution cycle. There is no trace output during the OPEN or CLOSE RPG II segments.

When trace option 1, 2, or 3 is selected, the system will halt (enter a tight loop) in SYSMSG if an RPG runtime error message is reported. This enables the user to request a dump of main memory. To exit from the halt state (loop), the user must use the control panel to set the A-register to zero.

When trace option E is selected, the system will halt (enter a tight loop) if any error m,essage is reported from any area of ITOS (e.g., the utilities, the file manager, RPG II, etc.). To exit from the halt state (loop), the user must use the control panel to set the A-register to zero.

When trace option S3 or S5 is selected, three or five segment mode of operation (respectively) is initiated. Note that unless the RPG program to be executed is very large or the operating system has a small user area, the 3-segment mode of operation is most efficient.

The following example shows a typical run time trace output.

Trace Output	Normal Output	Comments
ON 1P L0 DETAIL OUTPUT PRINT RECORD ON		Start of trace output, indicators 1P L0 are ON

Trace Output	Normal Output	Comments
----------	----------page eject----------	
	TEST AND-LR	Header output
OFF 1P		
INPUT RECORD FROM IN		Input from primary file. Indicator 01 (record ID) ON
SELECT RECORD FROM IN		
ON 01		
GET INPUT FIELDS		
DETAIL CALCULATION		
ADD		ADD operation
COMP		COMP - no change in indicators
SETON		
ON 70		SETON 70 set ON
EXCPT		EXCPT operation led
PRINT RECORD ON OUT		
	EXCPT-01	Exception output
DETAIL OUTPUT		
PRINT RECORD ON OUT		
	DETAIL-01	Detail output. Indicator 01 (record ID) OFF
OFF 01		
INPUT RECORD FROM IN		
SELECT RECORD FROM IN		
ON 01		
TOTAL CALCULATION		
TOTAL OUTPUT		
PRINT RECORD ON OUT		
	TOTAL 01	Total output
GET INPUT FIELDS		
DETAIL CALCULATION		
ADD		
COMP		
ON 77		Indicator 77 set ON by COMP
SETON		
ON LR		Indicator LR set ON
SETON		
EXCPT		SETON - no change
PRINT RECORD ON OUT		
	EXCPT-01	Exception output
DETAIL OUTPUT		
PRINT RECORD ON OUT		
	DETAIL-01	Detail output
OFF 01		
ON L1 L2 L3 L4 L5 L6 L7 L8 L9		LR caused L1 through L9 to turn ON
TOTAL CALCULATION		
SETON		
ON 72		Indicator 72 set ON at total time
EXCPT		
TOTAL OUTPUT		
PRINT RECORD ON OUT		
	TOTAL LR	Total output

COMPILER INTERPRETER TRACE

This mode occurs when a source statement line number is entered in columns 10 through 13 of the X specification. This mode generates a large amount of printout even though a line is printed only when one of the following changes:

PC – Program counter – if changed by more than one

RL – Roll position – contents of roll

PF – Program flag – true or false

BP – Source statement pointer – column number (decimal)

OP – Old pointer – program counter

Values are in hexadecimal (except BP).

An example of a compiler interpreter trace is shown in figure J1.

COMPILER OBJECT CODE GENERATION TRACE

This mode occurs when a source statement line number is entered in columns 20 through 23 of the X specification. A line of output is printed for each compiler object output operation.

RBD Value

For output of an RBD value, the first field printed is the object program counter. The second field printed is the value output.

A denotes an absolute value

R denotes a program relocatable value

B denotes output of the lower byte (bit 0-7)

B* denotes output of the high byte (bits 15-8)

* denotes a new origin

PC= Program counter in the specified compiler overlay

ENT

For output of an ENT, the first field is the entry point name followed by the second field, which is the defined value (address).

EXT Reference

For output of an EXT reference (with an ADC), the first field is the object program counter followed by the second field, which is the external name.

NAM

For output of an NAM, the following is printed:

******* SEGMENT NO ***

An example of object code generation is shown in figure J-2.

Output						Comments
RPG II COMPILER						Normal heading and list output
0001		H				Source line 1
0002		FREAD	IPE F	80	80	READER Source line 2
0003		FPRINT	O F		132	PRINTER Trace begins at line 0003
PC=0194	F	RL=3-0008	PF=T		BP=0006	OP=8BC0 Interpreter is at 0194 in F-overlay
PC=0196	F	RL=4-0000	PF=F			Roll position 4 loaded with 0000
PC=0198	F		PF=T			Program flag changed to TRUE
PC=019B	F				BP=0007	Source pointer moved to column 7
PC=019C	F		PF=F			
PC=01A3	F		PF=T			
PC=01A5	F	RL=5-0000	PF=F			Roll-5 loaded with 0000
PC=01A7	F	RL=6-0008				Roll-6 loaded with 0008
PC=01A8	F	RL=5-0000				Roll-6 was unloaded
PC=01AB	F				BP=0008	Source pointer moved to column 8
PC=01A8	F					OP=01AC
PC=01AB	F				BP=0009	Interpreter jumper to 01AB from 01AC
PC=01A8	F					OP=01AC
PC=01AB	F				BP=0010	
PC=01A8	F					OP=01AC
PC=01AB	F				BP=0011	
PC=01A8	F					OP=01AC
PC=01AB	F				BP=0012	
PC=01A8	F					OP=01AC
PC=01AB	F				BP=0013	
PC=01A8	F					OP=01AC
PC=01AB	F				BP=0014	
PC=01A8	F					OP=01AC
PC=01AB	F				BP=0015	
PC=01AC	F		PF=T			End of interpreter loop for columns
PC=01AE	F				BP=0007	8 through 15
PC=01AF	F		PF=F			

Figure J-1. Compiler Interpreter Trace

Output					Comments
RPG II COMPILER					Normal list output
0001	H			S	
0002	FINPUT		IPE F	80	
****** SEGMENT 0 ***			PC=018F	H	Start of object output
R9BASE--	0000	R	PC=0190	H	Entry point R9BASE and R9RPGX are at location 0000
R9RPGX--	0000	R	PC=0191	H	
0000	1400	A	PC=0194	H	Absolute values of 1400 and 0000 were output for locations
0001	0000	A	PC=0197	H	0000 and 0001
0002	=R9RPRT		PC=0199	H	Location 0002 will contain address of R9RPRT which is an
R9ONES--	0003	R	PC=01A3	H	external reference. At this point program counter is at
0003	00F1	A	PC=01A4	H	location 01A3-01A4 in compiler H overlay.
R9COMA--	0004	R	PC=01A3	H	
0004	006B	A	PC=01A4	H	
R9PERD--	0005	R	PC=01A3	H	Entry point (R9PERD) is at location 0005
0005	004B	A	PC=01A4	H	Location 0005 contains value 004B
R9DOLR--	0006	R	PC=01A3	H	
0006	005B	A	PC=01A4	H	
R9ZERO--	0007	R	PC=01A3	H	
0007	00F0	A	PC=01A4	H	
R9STAR--	0008	R	PC=01A3	H	
0008	005C	A	PC=01A4	H	

Figure J-2. Object Code Generation

INDEX

COMMENT SHEET

MANUAL TITLE CDC® RPG II Version 2 Reference Manual

PUBLICATION NO. 96768710 REVISION C

FROM NAME:

BUSINESS
ADDRESS:

COMMENTS: This form is not intended to be used as an order blank. Your evaluation of this manual will be welcomed by Control Data Corporation. Any errors, suggested additions or deletions, or general comments may be made below. Please include page number.

CUT ALONG LINE

STAPLE

STAPLE

FOLD

FIRST CLASS
PERMIT NO. 333

LA JOLLA CA.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
PUBLICATIONS AND GRAPHICS DIVISION
4455 EASTGATE MALL
LA JOLLA, CALIFORNIA 92037

CUT ALONG LINE

FOLD

STAPLE

STAPLE

CORPORATE HEADQUARTERS, P.O. BOX O, MINNEAPOLIS, MINNESOTA 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

LITHO IN U.S.A.

CONTROL DATA CORPORATION