Burroughs

B 1800/B 1700 Systems

Report Program Generator (RPG)

REFERENCE MANUAL

PRICED ITEM

Printed in U.S.A. August 1979 1057189

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

Section Page No.

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

Section Page No.

TABLE OF CONTENTS (Cont)

Section Page No.

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

Section Page No.

TABLE OF CONTENTS (Cont)

Section Page No.

TABLE OF CONTENTS (Cont)

Section Page No.

Appendix

TABLE OF CONTENTS (Cont)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Cont)

LIST OF ILLUSTRATIONS (Cont)

LIST OF TABLES

LIST OF TABLES (Cont)

LIST OF APPLICABLE B 1800/B 1700 PUBLICATIONS

The following is a list of publications relative to the B 1800/B 1700 RPG System that are referenced in this manual:

Publication Title	Title Page Date	Form No.
B 1800/B 1700 System Software Operational Guide	8-78	1068731
B 1800/B 1700 Systems COBOL Reference Manual	8-78	1057197
B 1800/B 1700 Systems Network Definition Language (NDL) Reference Manual	9-78	1073715
B 1800/B 1700 Systems Data Management System II (DMS II) Reference Manual	1-76	1089794

INTRODUCTION

Burroughs RPG (Report Program Generator) is a machine-independent programming language suitable for implementation in a wide variety of data processing applications. B 1800/B 1700 RPG embraces RPG as implemented on IBM 360/20 Systems and RPG II as implemented on IBM S/3. In addition, Burroughs has taken advantage of the sophisticated operating system of the B 1800/B 1700 to allow optional extensions to the above-mentioned implementations.

Burroughs B 1800/B 1700 RPG defaults to that of IBM S/3. A simple control option signals the compiler to generate code as in 360/20 RPG. Throughout this manual, reference to 360/20 RPG will be denoted by "RPG 1".

Burroughs B 1800/B 1700 RPG offers the following advantages to the user:

a. Simple, generative syntax for ease of program implementation.

b. Accelerated programmer training and simplified retraining requirements.

c. Ease of conversion through standard implementation.

d. Ease of program modification.

e. Standardized documentation.

This manual describes Burroughs B 1800/B 1700 RPG, and is a reference to the RPG Specification Forms together with each of their appropriate fields.

Columns not mentioned in the specifications sections are not used and must be left blank.

When left and right broken brackets (< >) are used in syntax descriptions they denote that a metalinguistic variable of the language is to be placed at that point in the syntax.

SECTION 1

RPG OPERATION

A program written in Burroughs RPG, called a source program, is accepted as input by the RPG Compiler. The compiler first verifies that the source program is syntactically error-free, then converts this source into RPG S-Language, which is then ready for execution on the system. The S-Language generated by the compiler can be executed by using an Interpreter. The Interpreter causes the system hardware to perform the operations specified by the S-Language Program and, thus, by the programmer who wrote the source program.

For a more detailed description of the function of the S-Language as it relates to the Interpreter and the hardware, refer to B 1800/B 1700 System Software Operational Guide, Form No. 1068731.

RPG SOURCE PROGRAM

An RPG Source Program is divided into eight parts which must appear in the following order:

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

A description of the above specifications is contained in the paragraphs that follow.

CONTROL CARD SPECIFICATIONS

These specifications provide certain information about the program to the B 1800/B 1700 RPG Compiler.

FILE DESCRIPTION SPECIFICATIONS

These specifications provide information about the equipment being used, and associate files with the hardware devices that will be used. The file types (i.e., input, output, combined) and blocking factors are also given.

EXTENSION SPECIFICATIONS

These specifications are used to describe tables, arrays, and record address files that will be used with the program.

LINE COUNTER SPECIFICATIONS

These specifications provide information about the number of lines to be printed on each page of the output forms that are used. Also, line-channel equation can be given to associate a print line with a channel punch in a carriage control tape.

TELECOMMUNICATIONS CARD SPECIFICATION

The Telecommunications card is used to further define a file specified on the File Description Specifications as a REMOTE, DATACOM, or BSCA file. There is no special form for coding Telecommunications Specifications, but the format described in Section 7 must be used.

INPUT SPECIFICATIONS

These specifications are used to describe the record layouts of all input files used by the program.

CALCULATION SPECIFICATIONS

These specifications define the steps necessary to accomplish the desired task when operating on data described in the program.

OUTPUT-FORMAT SPECIFICATIONS

These specifications are used to specify the type and arrangement of data that will be written as output from the program.

DOLLAR CARD SPECIFICATIONS

The Dollar Card ($ Card) Specifications sheet is also defined for Burroughs RPG. These specifications are used to accommodate machine- and system-dependent features, and may appear anywhere in a RPG Source Program, unless otherwise specified (see Chapter 11).

The above specifications are coded using the following Report Program Generator forms:

a. Control Card Specifications/File Description Specifications (Form No. 1055837)

b. Extension Specifications/Line Counter Specifications (Form No. 1055852)

c. Input Specifications (Form No. 1107562)

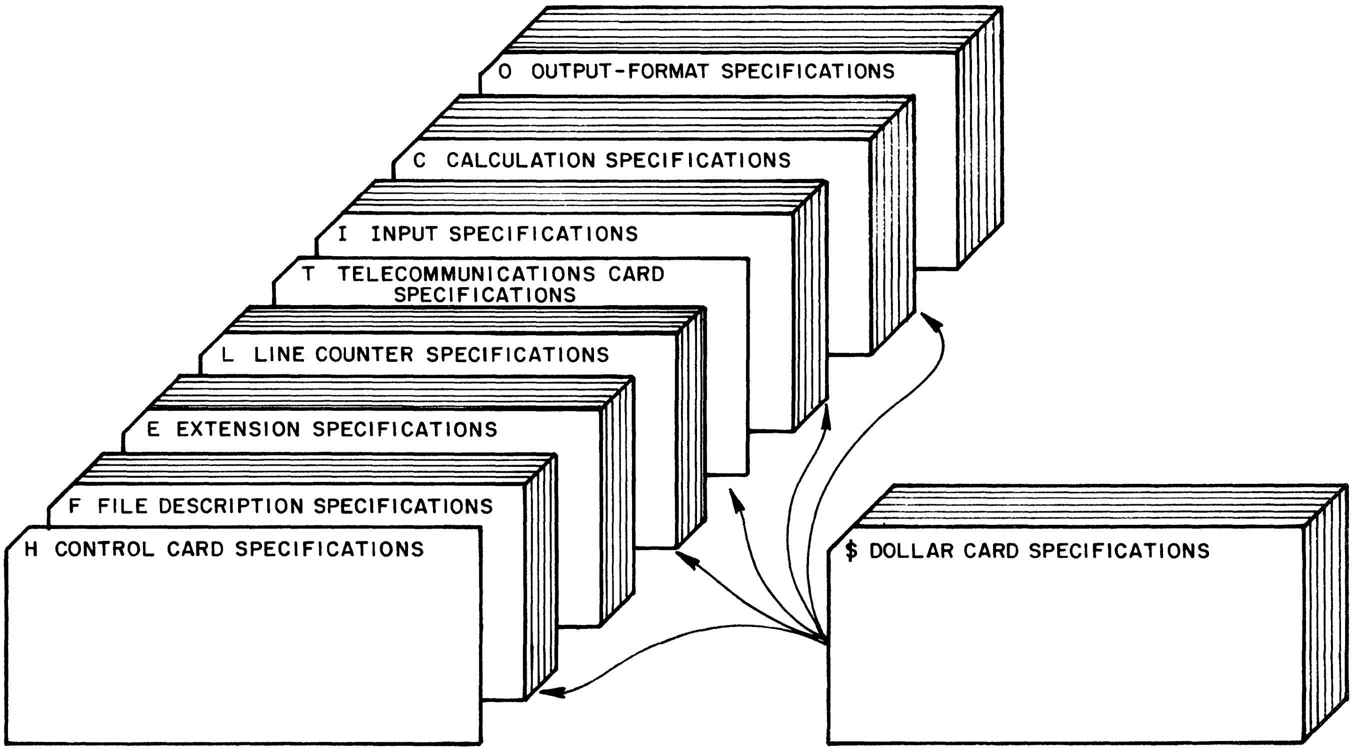
d. Calculation Specifications (Form No. 1055878)

e. Output-Format Specifications (Form No. 1055886)

f. Dollar Card Specifications (Form No. 1055845)

RPG SOURCE PROGRAM DECK

The coded information contained on the specification forms is recorded in punched cards which constitute the source program. The arrangement of these cards in the source deck is illustrated in figure 1-1.


O OUTPUT-FORMAT SPECIFICATIONS
C CALCULATION SPECIFICATIONS
I INPUT SPECIFICATIONS
T TELECOMMUNICATIONS CARD SPECIFICATIONS
L LINE COUNTER SPECIFICATIONS
E EXTENSION SPECIFICATIONS
F FILE DESCRIPTION SPECIFICATIONS
H CONTROL CARD SPECIFICATIONS
$ DOLLAR CARD SPECIFICATIONS

G14003

Figure 1-1. RPG Source Program Deck

Should source corrections become necessary, appropriate changes can be made and the program recompiled. Thus, the source program deck always reflects the S-Language being operationally executed. See Section 12 for operating instructions.

RPG FORMAT

As mentioned in the introduction to this manual, Burroughs RPG accepts source statement input in the format of either RPG 1 or RPG II.

The following is a list of exceptions found in B 1700 RPG:

a. The Control Card requires simplified recoding before compiling programs from systems other than B 1700.

b. B 1700 RPG allows 511 characters for alpha fields and 31 for numeric.

c. IBM's RPG uses standard right signed numeric fields, whereas Burroughs B 1700 RPG uses standard left sign. IBM's System 3 standard plus sign is 'F' right sign, the 360/20 standard plus sign is 'C' right sign, whereas the Burroughs standard plus sign is 'C' left sign. B 1700 RPG allows the right sign option for system compatibility. (Refer to $RSIGN option in Chapter 11.)

d. Numeric fields must be edited if the C zone punch is not wanted on output. Otherwise, a positive unedited numeric field will contain an alpha character in the left-most position of the field. For example, C1F0F1F3 would appear as A013 on the printed output. However, if this field is edited, it will assign a F to the left-most character, changing the output to appear as 1013.

e. 96-column devices will not allow packed decimal format.

f. The following features and language constructs are not provided for by Burroughs RPG:

   1. Variable length records

   2. Sterling data format.

   3. Alternate collating sequence.

   4. Inquiry programs.

   5. Printer keyboard output files.

   6. External assembler subroutines.

   7. Blank after literals.

   8. Redefined field lengths.

   9. Factor 1 and Factor 2 both literals.

   10. Card print feature of IBM's 360/20.

   11. File translation.

SECTION 2

RPG LANGUAGE ELEMENTS

Burroughs RPG is a programming language based upon a fixed series of events, called the RPG "program cycle", which takes place during program execution. Due to the strict limitations of the generated program, the source language must conform to rigid rules of syntax. The following paragraphs and sections define the rules for writing programs using the RPG language.

CHARACTER SET

The minimum RPG character set consists of the following characters:

0-9	
A-Z	
	blank or space
&	ampersand
.	period or decimal point
-	minus sign
$	dollar sign
*	asterisk
,	comma
'	apostrophe

The following RPG characters are optional and may be added to the above list:

+	plus sign
<	less than sign
>	greater than sign
(	left parenthesis
)	right parenthesis
[	left bracket
]	right bracket
|	logical OR
¬	logical NOT
!	exclamation point
;	semicolon
/	slash (virgule)
%	percent sign
_	underscore
?	question mark
:	colon
#	pound sign
@	at sign
=	equal sign
"	quotation mark

CHARACTERS USED FOR NAMES

The character set used to form names consists of the 36 characters: 0 through 9 and A through Z.

CHARACTERS USED FOR EDITING

The character set used for special purposes within edit words in the Output-Format Specifications consists of the following nine characters:

	blank or space
0	zero
$	dollar sign
.	decimal point
,	comma
CR	credit symbol (two characters)
*	asterisk (check protect)
&	amperand
-	minus sign

DEFINITION OF NAMES

A name must be left-justified in the field, must begin with an alphabetic character, and is ended by a space or the end of the field, whichever comes first. All characters in a name except the first may be any combination of alphabetic and numeric characters (special characters are not allowed). Blanks may not appear between the characters in a name.

RPG defines the following four types of names:

Filenames
Vector names (table or array names)
Field names (variable names)
Labels

FILENAMES

A filename is a collective name or word that designates a set of data items. The contents of a file are divided into logical records which are made up of any consecutive set of data items. Filenames cannot exceed eight characters and the first seven characters must be unique among filenames (e.g., DISKFILE and DISKFIL are considered by the compiler to be equal). The compiler truncates the eighth character.

VECTOR NAMES

A vector name is used to identify a data item which is actually a table or an array. The table will be loaded with a number of elementary data items which are accessed through use of the vector name that identifies the entire table. Vector names cannot exceed six characters and must be unique among vector and field names.

FIELD NAMES

A field name is used to identify an individual element of data. Field names cannot exceed six characters and must be unique among vector and field names. A separate memory area is reserved for each unique field name which is completely unrelated to any memory area reserved for any other field name.

LABELS

A label is used only to identify a point in the Calculation Specifications to which a GOTO operation will branch, or to identify the beginning of a subroutine. Labels cannot exceed six characters and must be unique among labels.

DEFINITION OF LITERALS

A literal is an item of data which contains a value identical to the characters being described. There are two classes of literals: numeric and alphanumeric.

NUMERIC LITERAL

A numeric literal is defined as an item composed of characters chosen from the digits 0 through 9, an optional plus sign (all numeric literals not signed minus (-) are assumed plus) or minus sign (-), and the decimal point or decimal comma (see item e below). The rules for the formation of a numeric literal are:

a. There must be at least one digit in a numeric literal.

b. The sign of a numeric literal must appear as the left-most character. If no sign is present, the literal is defined as a positive value.

c. Only one sign character and/or one decimal point may be contained in a numeric literal.

d. The maximum total length of a numeric literal is 10 characters, including sign and decimal point.

e. Decimal commas must be used in place of decimal points if the Inverted Print Options I or J are used (Control Card, Column 21).

f. Embedded blanks are not allowed.

The following are examples of numeric literals:

```
13427
.005
+1.808 = 1.808
-.0968
7894.54
```

ALPHANUMERIC LITERAL

An alphanumeric literal may be composed of any allowable character. The beginning and end of an alphanumeric literal is denoted by an apostrophe. Any character enclosed within apostrophes is part of the alphanumeric literal. Consequently, all spaces enclosed within the apostrophes are considered part of the literal. Two consecutive apostrophes within an alphanumeric literal cause a single apostrophe to be inserted into the literal string. An alphanumeric literal which consists only of four consecutive apostrophes results in a single apostrophe. The rules for the formation of an alphanumeric literal are:

a. There must be at least one character in an alphanumeric literal.

b. The maximum length of an alphanumeric literal used in the Calculation Specifications is eight characters, and in the Output-Format Specification is 24 characters.

c. Alphanumeric literals may not be used for arithmetic operations.

The following are examples of alphanumeric literals:

Literal on Source Program Level	Literal Stored by Compiler
'ACTUAL'	ACTUAL
'-1234.56'	-1234.56
'WEEK''S'	WEEK'S
'TODAY''S DATE'	TODAY'S DATE
''''	'
'A''B'	A'B
'A''''B'	A''B

DEFINITION OF RESERVED WORDS

Reserved words have a specific function in the RPG syntax and are of two types: special words and operation codes.

SPECIAL WORDS

Reserved words are used in the Input, Calculation, and Output-Format Specifications, and specify such things as page numbering, date fields, and card interpreting.

The following special words are reserved for use as variables:

JDATE
PAGE
PAGEn (where n=1-8)
TIME
UDATE
UMONTH
UDAY
UYEAR
*PRINT
*PLACE

Their use is discussed fully in the sections where they are used.

OPERATION CODES

These reserved words are used in the Calculation Specifications, and specify operations to be performed upon data items. A complete description of the operation codes is presented in Section 9.

COMMON FIELD DEFINITIONS

The RPG specification forms have certain common fields which have consistent entries within a RPG Program. These fields and their respective entries are described below, so that they need not be repeated in subsequent sections.

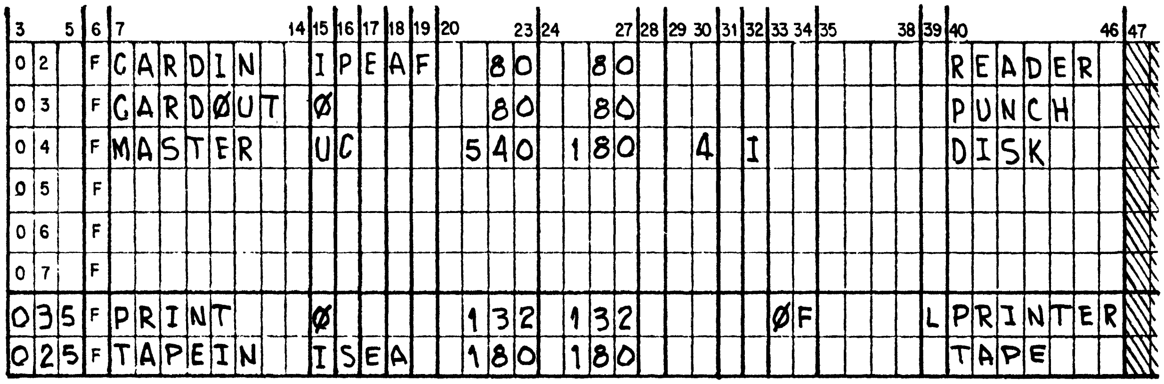
1-2 PAGE

The PAGE entry in the upper left-hand corner of each specification form is used to number the coding forms sequentially for each source program. Normally, only numeric characters in the range 01-99 would be used; however, any EBCDIC characters (including blanks) are valid entries.

3-5 LINE

The LINE entry is used to number the individual lines on each specification form sequentially. Normally, only numeric characters in the range 000-999 would be used; however, any EBCDIC characters (including blanks) are valid entries.

Columns 3-4 are preprinted so that in most cases the entry is already made. For example, the File Description Specifications form contains line numbers for lines 02 through 07. If more than six lines are needed, additional entries may be made below line 07 and numbered 08, 09, etc. The units position (column 5) may be used to insert a line between two previously written lines (see figure 2-1).



3 5	6	7 14	15	16	17	18	19	20 23	24 27	28	29 30	31	32	33 34	35 38	39	40 46	47
02	F	CARDIN	I	P	E	A	F	80	80								READER	
03	F	CARDØUT	Ø					80	80								PUNCH	
04	F	MASTER	U	C				540	180		4		I				DISK	
05	F																	
06	F																	
07	F																	
035	F	PRINT	Ø					132	132					ØF		L	PRINTER	
025	F	TAPEIN	I	S	E	A		180	180								TAPE	

G14004

Figure 2-1. Insertion of Coding Lines

The compiler sequence-checks all input cards on columns 1-5 so that all lines should be numbered in ascending numerical order. If a sequence number (page and line number) is encountered that is equal to or less than the preceding sequence number, a sequence error warning message (S) will be emitted by the compiler.

6 FORM TYPE

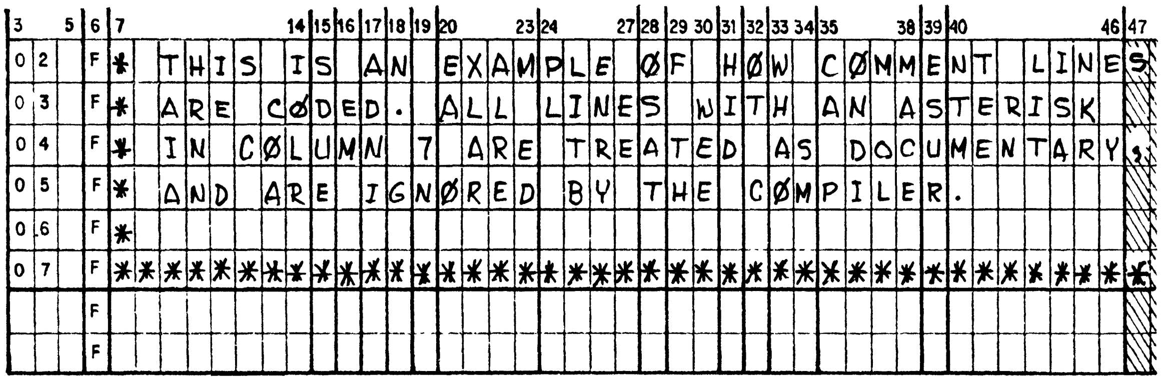
This entry identifies the type of specification for each line of code. This entry is preprinted for all but the Telecommunications Card Specifications.

Valid entries for this field are:

Entry	Definition
H	Header Card (Control Card Specification).
F	File Description Specification.
E	Extension Specification.
L	Line Counter Specification.
T	Telecommunications Card.
I	Input Specification.
C	Calculation Specification.
O	Output-Format Specification.
Blank	Not allowed except for Comment Card or Dollar Card (depending upon entry in column 7).

7 COMMENTS/DOLLAR CARD

Since it is often necessary to write explanatory statements within the source program, the Comment Card allows the entire line to the right of column 7 (which contains an asterisk, "*") to be produced on the source program listing for documentation clarity. Comments are not instructions to the RPG Program or Compiler, but serve only as a means of including program documentation. Any valid EBCDIC characters may be used in a comment line. An asterisk in column 7 overrides column 6. An example of comment line coding is illustrated in figure 2-2.



```
3 5 6 7 14 15 16 17 18 19 20 23 24 27 28 29 30 31 32 33 34 35 38 39 40 46 47
  02 F* THIS IS AN EXAMPLE ØF HØW CØMMENT LINES
  03 F* ARE CØDED. ALL LINES WITH AN ASTERISK
  04 F* IN CØLUMN 7 ARE TREATED AS DOCUMENTARY,
  05 F* AND ARE IGNØRED BY THE CØMPILER.
  06 F*
  07 F*****************************************
     F
     F
```

G14005

Figure 2-2. Comment Line Coding

The $ specification in column 7 designates the line to be a Dollar Card Specification which allows the RPG Compiler to accommodate machine- and system-dependent features. A dollar sign in column 7 overrides column 6. Dollar Cards may appear at any point in the RPG Source Program Deck. Section 11 describes the use of Dollar Card Specifications.

75-80 PROGRAM IDENTIFICATION

The PROGRAM IDENTIFICATION entry in the upper right-hand corner of each specification form is ignored by the compiler, but will appear on the source program listing. Thus, it can be used for documentation to identify different portions of the program, if desired.

SECTION 3

CONTROL CARD SPECIFICATIONS

RPG can use a special card, called a Control Card, to provide certain information about the program to the RPG Compiler. The Control Card can transmit to the compiler such information as type of source input, whether debugging is to take place, sign positions, and so forth. If columns 7 through 74 are blank, the Control Card is not required. One line is provided on the Control Card Specifications and File Description Specifications form for coding of the Control Card information (see figure 3-1).

FIELD DEFINITIONS

The fields for the Control Card Specification form are defined in the paragraphs that follow.

1-2 PAGE

Refer to Section 2 for a complete description.

3-5 LINE

Refer to Section 2 for a complete description.

6 FORM TYPE

This field must be coded with the letter H.

15 DEBUG

Column 15 specifies whether or not the DEBUG operation is to be significant during program compilation. To perform the DEBUG operation:

a. Column 15 of the Control Card must be coded with a 1.

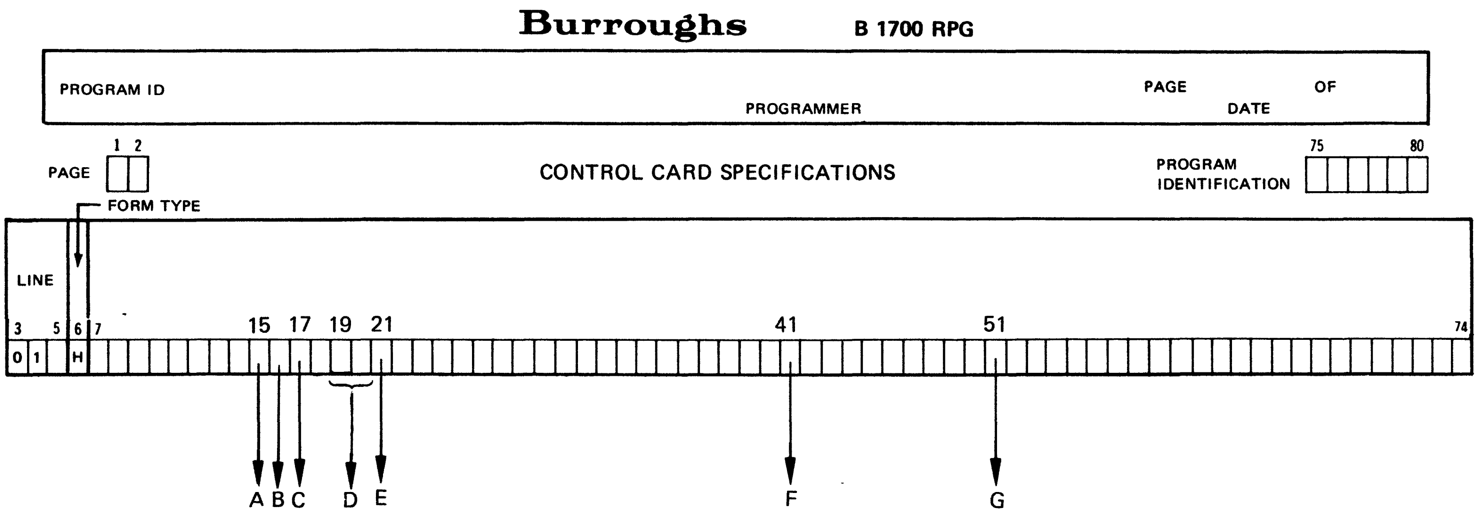
b. The operation code for DEBUG must appear in the Calculation Specifications.

If this field is left blank, any DEBUG operations encountered in the Calculation Specifications will not be executed at run-time, but they will still be checked syntactically by the compiler.

16 B-INDEXED FILES

Column 16 can be used to specify whether B-Indexed Files are to be created or used by the program. Valid entries are as follows:

Entry	Definition
Blank	Indexed file.
B	B-Indexed file.



A. 15 Specifies whether the DEBUG operation is to be used during compilation. Entries: Blank or 1.

B. 16 Specifies whether B-Indexed files are to be created or used by the program. Entries: Blank or B.

C. 17 Specifies the location of the sign in all numeric data items. Entries: Blank or L, or R.

D. 19-20 Specifies the amount of memory in K-bytes to be allocated for sorting indexed files. The system defaults to 8000 bytes if these columns are blank. Entry: A two digit number.

E. 21 Specifies the punctuation (INVERTED PRINT) to be used for numeric literals. Entries: Blank, I, J, or D.

F. 41 Causes output lines to be conditioned by the 1P indicator. Entries: Blank or 1.

G. 51 Specifies which RPG dialect to use, RPG 1 or RPG II. Entries: Blank or 1.

G14006

Figure 3-1. Control Card Specifications Summary Sheet

17 SIGN POSITION

This field specifies the location of the sign in all numeric data items and may have the following entries:

Entry	Definition
Blank or L	Sign in left-most (high-order) character position.
R	Sign in right-most (low-order) character position.

The sign position specified by this field may be overridden by the RSIGN Dollar Card Specification (refer to Section 11) if different sign positions are required within the program.

19-20 SORT MEMORY SIZE

This field is used to specify the amount of memory to be used for sorting. The number specified is used to allocate memory in thousands of bytes. An entry of 24 results in the allocation of 24K. bytes of memory for sorting. The default value of 8K. bytes is used when this field is blank or no header card is included in the source card deck.

21 INVERTED PRINT

This field is used to specify the type of punctuation to be used for numeric literals in the Calculation Specifications, the order of the system date field, and the edit codes used on output. The valid codes are:

Entry	Definition
Blank	Domestic format.
I	International format.
J	International format (leading zeros not suppressed for zero balances).
D	United Kingdom format.

Table 3-1 shows inverted print specifications and their resulting formats.

Table 3-1. Inverted Print Specifications

Column 21 Entry	Numeric Literal With a Comma or a Period as a Decimal Point	Edit Codes With a Comma or a Period as a Decimal Point	Zero Suppress to the Left or Right of the Decimal Point	UDATE With a Slash or a Period
Blank	5678.90	7,654.32	.60	Mon/Day/Year
D	5678.90	7,654.32	.60	Day/Mon/Year
I	5678,90	7.654,32	,60	Day.Mon.Year
J	5678,90	7.654,32	0,60	Day.Mon.Year

41 FORMS POSITIONING

Since it sometimes becomes necessary to align special forms in the line printer before beginning to print a report, a 1 in column 41 will cause all output lines conditioned by the 1P indicator to be printed more than once when the program is executed. Each time the lines are printed, the program is temporarily suspended to allow the operator to reposition the forms. Printing of the lines may be requested by the operator as many times as necessary to align the forms properly. If this field is left blank, 1P lines are printed only once.

51 SOURCE INPUT DIALECT

Burroughs RPG supports two dialects. RPG 1 dialect is compatible to IBM 360/20 RPG. RPG II dialect is compatible to IBM System/3 RPG II. Both dialects support Burroughs optional extensions. The RPG II dialect is assumed by default. If it is required to use RPG 1 features, a 1 must be entered in column 51 of the control card, otherwise this column should be blank. The source program is not checked against the syntax of the chosen dialect; this option is only used to resolve conflicts between the dialects. Providing there is no conflict, RPG II features can be used when RPG 1 dialect is specified and vice versa.

75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a complete description.

SECTION 4

FILE DESCRIPTION SPECIFICATIONS

Every file to be used by an RPG program, except compile-time vector files, must be described to the RPG compiler through the File Description Specifications. These specifications are used to provide information about file types, file names, how the files are used, record and block lengths, and hardware devices. The information is used to associate files with hardware devices as well as for other purposes.

The B 1800/B 1700 system provides the facility for the object program to use the record and block lengths from the header record of existing disk files. This function can be inhibited by use of the dollar option REFORM.

The maximum number of files allowed in any RPG program is 31. An index file and its corresponding tag file count as one file.

The lower portion of the Control Card Specifications and File Description Specifications is used for coding the File Description information. Each file to be described requires one line on the form.

FIELD DEFINITIONS

Figure 4-1 can be used in conjunction with the following field definitions for the File Description Specifications.

1-2 PAGE

Refer to Section 2 for a complete description.

3-5 LINE

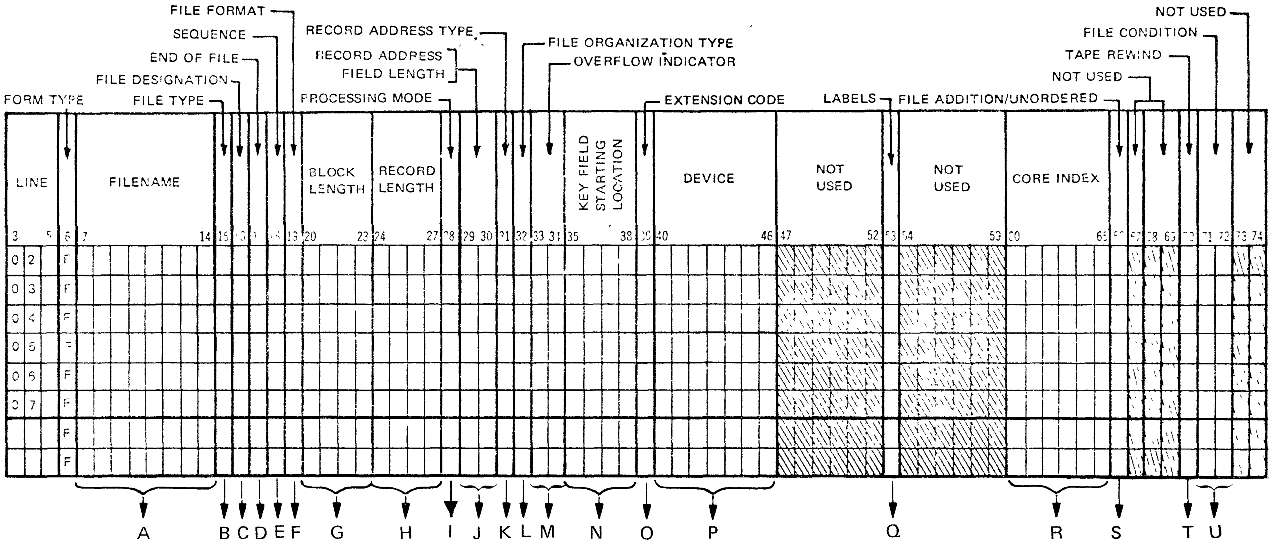
Refer to Section 2 for a complete description.

6 FORM TYPE

This field must contain a F.

7-14 FILENAME

This field is used to assign a unique name to every file used by the program. The filename must begin in column 7. Filenames must be assigned in accordance with the rules outlined in Section 2 of this manual.



A. Contains name unique in the first 7 characters for every file to be used.

B. 15 Specifies how each file is to be used. Entries: 1, O, U, C, or D.

C. 16 Describes the use of input, update, and combined files. Entries: Blank, P, S, C, T, or D.

D. 17 Specifies which files are to be checked for End-of-File during multifile processing. Entries: Blank or E.

E. 18 Specifies the order matching fields are to be checked. Entries: Blank or A, or D.

F. 19 Specifies whether the file contains fixed- or variable-length records. Entries: Blank or F.

G. 20-23 Specifies the block size. Entry is device dependent.

H. 24-27 Specifies the record size. Entry is device dependent.

I. 28 Specifies Record Address or Sequential Processing disk files only. Entries: Blank, L, or R.

J. 29-30 Specifies the length of the key field. Entries: 1-99, right-justified.

K. 31 Describes the format of the keys within the records. Entries: Blank, K or A, P, or N.

G14007

L. 32 Identifies how a file is to be accessed or whether multiple buffers are required. Entries: I, 1-9, or Blank.

M. 33-34 Specifies the overflow indicator used to condition records. Entries: OA-OG, OV or Blank.

N. 35-38 Specifies the key field starting position in each record for indexed files. Entries: 1-4095, right-justified.

O. 39 Indicates whether an output table or array file will be further described on Extension or Line Counter Specifications. Entries: E, L, or Blank.

P. 40-46 Identifies the input/output device to which each file is assigned. Entries: READER, MFCU1, MFCU2, PUNCH, PRINTER, PRINTR2, TAPE, DISK, CONSOLE, BSCA, DATACOM, REMOTE, QUEUE.

Q. 53 Entries only meaningful to the B 1700. Entries: U (unlabelled file), F (special forms), or B (printer backup).

R. 60-65 Specifies number of bytes of memory to be set aside for an index. Entries: 1-9999 right-justified.

S. 66 For sequential or indexed disk files it specifies addition to existing file or unordered records to be loaded. Entries: A, U, or Blank.

T. 70 Specifies action to be taken during close of the file (tape and disk files). Entries: P, U, N, R, or Blank.

U. 71-72 Indicates whether file is to be conditioned by an external indicator. Entries: U1-U8 or Blank.

Figure 4-1. File Description Specifications Summary Sheet

15 FILE TYPE

This field is used to identify the manner in which the program uses the file. Valid entries are:

Entry	Definition
I	Input file.
O	Output file.
U	Update file.
C	Combined file.
D	Display file.

I Input Files

Input files contain records that the program uses as a source of data. If a file is described as input, it indicates that records will be read from that file. Input files must be further described on the Input Specifications form with the exception of table files, which must be described on the Extension Specifications form.

O Output Files

Output files contain records that are written, printed, or punched as output from the program. All output files should (not required) be further described on the Output-Format Specifications form, except for output table files. FILE DESIGNATION (column 16) must be blank for all output files except chained direct files.

U Update Files

Update files are disk files from which a program reads a record, changes fields in the record, and writes it back into the same location from which it was read. All update files must be further described on the Input Specifications and should (not required) be described on the Output-Format Specifications form. A chained file or a demand file may be updated at detail time, at total time, or at exception time. All other disk files can be updated only at detail time.

C Combined Files

Combined files are card files or remote files that can be used for both input or output. These files consist of:

a. Cards that are read by the program and are subsequently punched and/or printed as output. The punching is into the same cards that have been read, and output may occur only once per cycle.

b. Remote files that will be written to (SEND) and read from (RECV) during calculations.

Combined files must be further described on the Input Specifications and should (not required) be described on the Output-Format Specifications form.

D Display Files

Display files are used to print a field or record directly on the console printer. The DSPLY operation code must be used in the Calculation Specifications in order to perform the print operation. Display files are described only on the File Description Specifications form.

16 FILE DESIGNATION

This field is used to further describe the use of input, update, and combined files. It must be left blank for all output files (including display files), except for chained output files. Acceptable entries for this field are:

Entry	Definition
Blank	Output file.
P	Primary sequential file.
S	Secondary sequential file.
C	Chained (random) file.
T	Input table file.
D	Demand file.
R	Record address file.

P Primary File

A primary file is the principal file from which the program reads input records, and may be designated as input, update, or combined. Every RPG program must have one and only one primary file; all primary files described after the first one named are considered to be terminal errors. The primary file must be declared before any secondary files or a warning message is emitted.

If there is no P entry in the File Description Specifications, a warning message is emitted and the first S (secondary) file defined is assumed to be the primary file. When no primary or secondary files are present, a syntax error is emitted.

S Secondary Files

Secondary files are all files other than the primary file involved in record selection during multifile processing. Note that this excludes table, chained, and demand files. These files are processed in the same order in which they are written in the File Description Specifications. A secondary file must be an input, update, or combined file.

C Chained File

A chained file may be an input, output, or update file assigned to a disk that uses the CHAIN operation code to read or write records randomly. When the chained file is an input file, records are read from the file. When the chained file is an update file, records are read from the file and can be modified and rewritten to the same location in the file.

Chained indexed files may be input or update files only.

T Table File

A table file is a sequential input file that contains vector entries which can be read into the program during pre-execution-time. Only pre-execution-time vector files are described on the File Description Specifications form. Pre-execution-time vectors must be described on the Extension Specifications form.

Table files are only a means for supplying entries for tables used by the program and are not involved in record selection during processing at execution time. All records in table files read during program execution are read before any other data records.

Both compile-time and pre-execution-time vectors may be changed at execution time; however, vector entries read during compilation can be permanently altered only by recompiling the program.

A vector output file, written or punched at End-of-Job, is defined as a normal output file and does not require an entry in column 16.

D Demand File

A demand file may be an update, input, or combined file from which records are read through use of the READ or RECV operation codes in the Calculation Specifications. Demand files and indexed disk files which are read sequentially by key can only be read sequentially.

Remote devices are accessed on a demand basis through the data communications facilities of RPG. The RECV and SEND operation codes are used to cause records to be read from and exception records to be written to REMOTE files respectively.

Demand files are processed as follows:

a. During input mode, the RPG program reads a record in calculations and does not depend on the RPG program cycle to make a record available.

b. During output mode, the RPG program allows exception records to be written during calculations. The desired output operation is performed, and program execution continues with the next calculation statement. The demand file does not have to wait for the RPG cycle to write the exception record.

R Record Address File

A record address file must be an input file and must be described on the Extension Specifications. No entries are required on the Input Specifications for a record address file.

A record address file can be either:

a. A limits file that contains records consisting of lower and upper bound record keys that are used to process indexed files sequentially within limits, or

b. An addrout file that consists of 4-byte, relative-record numbers used to randomly process a disk file.

17 END OF FILE

This column specifies which files are to be checked for End-of-File during multifile processing in order to turn the last record indicator (LR) ON. It applies only to input, update, and combined files declared as primary or secondary files and also to record address file, and is used to indicate whether or not the program may end before all of the records from the file are processed. Valid entries for this field are:

Entry	Definition
Blank	If this entry is blank for all files, all records must be read from all files before the program may end. If this entry is blank for only some files the program may end whether or not all records from the file have been read.
E	All records in the file must be read and processed before the program may end.

If all records from all input files must be read and processed before the LR indicator can be turned ON, this column must be blank (or contain E) for all files.

Specifying an E in column 17 of the File Description Sheet indicates that the job is to end after all records are processed from the file for which the E was specified. In most cases, the job will end at the time all records from that file are processed. However, under certain conditions, additional records may be processed after all records from the file with the E designation are processed. The exceptional situation is in matching records when an E is designated for the primary file and all records from that file have been processed. The job will end only after all secondary records that match the last primary record have been processed or the first secondary record without a match field has been encountered.

Technically, a data file never reaches end-of-file when processed by a record address file. The program can be forced to go to end-of-job when the record address file reaches end-of-file by making an entry in column 17 of the File Description Card for the record address file.

18 SEQUENCE

This field is used only by primary and secondary files to indicate whether or not the program is to check the sequence of the input records. The Input Specifications form (columns 61-62) must be used to specify the match fields within the input file records. This entry must be left blank for all files other than primary and secondary files. Acceptable entries for this field are:

Entry	Definition
Blank or A	Records with matching fields are to be sequence-checked in ascending order.
D	Records with matching fields are to be sequence-checked in descending order.

Sequence checking is performed when matching fields have been specified for the records in a file. If a record from a matching input file is found to be out of sequence, the program halts. If the program halts, the operator may make one of the following entries:

Console Message	Definition
<program number> AXGO	Ignore the record out of sequence and read the next record from the same file.
<program number> AXSTOP	Ignore the record out of sequence, turn on the LR indicator, and perform all final detail and total calculation procedures.
<program number> DS	Discontinue the program.

All sequence checking is performed according to the EBCDIC collating sequence (see Appendix B). If any matching file specifies descending (D) sequence, all files must specify descending sequence.

19 FILE FORMAT

This field specifies whether the file contains fixed- or variable-length records. Variable-length records may not be specified.

Entry	Definition
Blank or F	Fixed record length.

20-27 BLOCK AND RECORD LENGTH

These fields are used to specify the block and record sizes for the file. The minimum and maximum record and block length allowed depend upon the device to which the file is assigned. This information is detailed for the B 1800/B 1700 in table 4-1.

Table 4-1. The Maximum and Minimum Values Allowed For Block and Record Sizes

Device	Block Length		Record Length	
	Minimum	Maximum	Minimum	Maximum
CARD (80 col)	Same as record length	Same as record length	1	80
CARD (96 col)	Same as record length	Same as record length	1	96
PRINTER	Same as record length	Same as record length	1	132
DISK				
Data files	1	9999	1	4095
Addrout files	4	180	4	4
TAPE	8	9999	1	4095
CONSOLE	Same as record length	Same as record length	1	50

Block and record length entries may in certain cases be left blank. The treatment of the various cases is tabulated in table 4-2.

Table 4-2. Resulting Action of Block Length and Record Length

Block Length	Record Length	Result
Entry	Entry	The values entered are used.
Blank	Blank	Block and Record Length are defaulted to the same value.
Entry	Blank	Invalid.
Blank	Entry	Block length is defaulted to the value specified for Record Length.

Entries in the BLOCK LENGTH (columns 20-23) and RECORD LENGTH (columns 24-27) fields must be right-justified, and leading zeros may be omitted. If the BLOCK LENGTH is left blank, it is assumed to be the same as the RECORD LENGTH. If both the RECORD LENGTH and BLOCK LENGTH entries are left blank, the default sizes given in table 4-3 are assumed.

Table 4-3. Default Block and Record Lengths

Device	Block and Record Length
All Printers	132
DISK	
Data file	180
Addrout file	4
DATA 96, MFCU1, MFCU2	96
All other devices	80

The BLOCK LENGTH must be an integral multiple of the RECORD LENGTH. For unblocked files (note that card and printer files are unblocked), the BLOCK LENGTH will be equal to the RECORD LENGTH. For card and printer files, the RECORD LENGTH specified may be less than the maximum record length for the device, e.g., printer files may be specified as 96, 120, or 132 characters in length.

28 PROCESSING MODE

This field applies to disk files only. The acceptable entries are:

Entry	Processing Mode
Blank	Sequential by key. Consecutive.
L	Sequential within limits.
R	Addrout.

29-30 RECORD ADDRESS FIELD LENGTH

This field applies only to indexed files and record address files.

For a record address file, the entry in this field specifies either the length in bytes of each record in an addrout file or the length in bytes of each key in a limits file.

For an indexed file, the entry in this field specifies the length of the key field within the records of the file. All key fields of an indexed file must be the same length, and the maximum key length is as follows:

a. Indexed files - 29 characters.

b. B-Indexed files - 99 characters.

The maximum key length is the same for a limits file or an indexed file.

The entry in this field must be numeric and must be right-justified. Leading zeroes are not required. The numeric entry specifies the length in bytes of the key field.

31 RECORD ADDRESS TYPE

This field is used to provide additional information about the format of the keys and/or the processing mode of indexed disk files and record address files. The acceptable entries for this field are:

Entry	Definition
K, A	Indexed or limits file with alphanumeric keys.
P	Indexed or limits file with packed keys.
N	Indexed file with numeric keys in alphanumeric format.
I	Addrout file or processed by an addrout file (refer to note below).
Blank	Not an indexed file, a limits file, or an addrout file. Also, the file is not processed by an addrout file.

NOTE

If column 31 contains the letter I and column 16 contains the letter R, the file is an addrout file. If column 16 does not contain the letter R, the file is processed by an addrout file.

A numeric key specified for a limits file may be of a different data format than the numeric key specified for the associated indexed file.

32 FILE ORGANIZATION TYPE

Column 32 identifies file organization or indicates whether multiple buffers are required. Valid entries are:

Entry	Definition
I	Indexed file.
T	Addrout file.
1-9	Sequential or direct file with 1-9 I/O areas.
Blank	Sequential or direct file with a single I/O area.

NOTE

B 1700 RPG organizational techniques are such that file accessing and file organization are independent of each other. This is not so for other RPG implementations.

Indexed files are assigned to disk. Data keys are required when using indexed files. For additional information about indexed files, refer to the description of columns 29-30, RECORD ADDRESS FIELD LENGTH and columns 35-38, KEY FIELD STARTING LOCATION in this section.

An indexed file can be loaded in ascending key sequence or it may be loaded in unordered sequence. Refer to the description of column 66, FILE ADDITION/UNORDERED in this section for additional information.

Sequential and direct files can have multiple I/O areas assigned to them. Multiple I/O areas may increase the efficiency of execution of the RPG program, but their use also increases the size of the program. A good balance between the number of I/O areas and program size must be found in order to achieve the greatest throughput from the system when programs are multiprogrammed.

Multiple I/O areas cannot be used with table, combined, display, demand, indexed disk, or stacker selected files.

If column 32 contains the letter T specifying an addrout file, column 16 must contain the letter R and the file must be assigned to disk or magnetic tape.

A limits file is defined by entering a blank or a numeric (1-9) in column 32 and the letter R in column 16.

File Organization

The following types of file organization can be used with B 1700 RPG:

a. Sequential.

b. Indexed.

c. Direct.

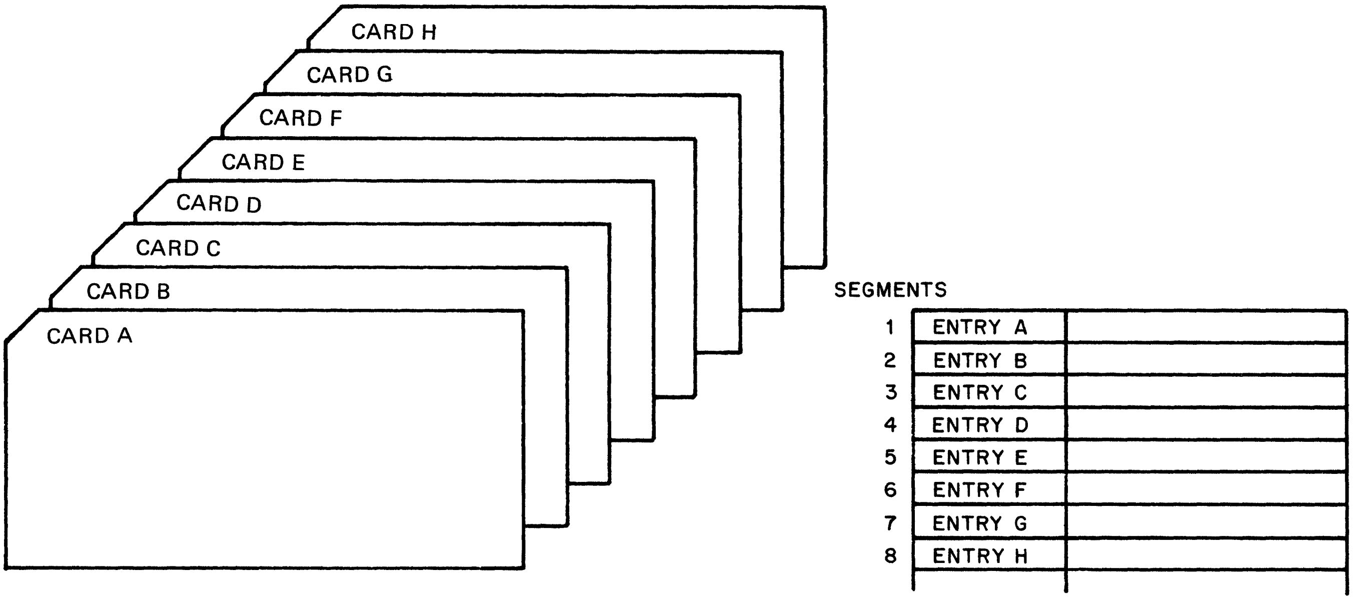
A detailed description of each type of file organization is presented in the following pages.

Sequential Files

The order in which records appear in a sequential file is determined by the order in which the records are placed in the file when the file is created.

Card files and magnetic tape files are sequential files. Disk files can be sequential, indexed, or direct.

Figure 4-2 shows a card file and the order in which the information from the cards is stored in a sequential disk file. The contents of cards A, B, C, D, E, F, G, and H are stored in disk segments 1 through 8 as entries A, B, C, D, E, F, G, and H respectively.



G14008

Figure 4-2. Sequentially Organized Card and Disk Files

Indexed Files

The following type of indexed files can be used with B 1700 RPG:

a. Indexed.

b. B-Indexed.

Only one of the two indexed file types can be used in a given program, and a file must always use the same indexing method.

A detailed description of both types of indexed files follows:

Indexed. Indexed files are data files on disk that have associated tag files. The records of the indexed file are in random sequence and are accessed by use of the tag file. The records of the indexed file contain record keys and data. The record keys are specified by the user. The file description for the indexed file contains information that describes the length, type, and starting position in the record of the record key. Each record contains one key whose size can range from one byte to a maximum of 29 bytes.

Indexed files can be created with the record keys in either ascending or random sequence. When an indexed file is loaded in a random (unordered) sequence, the tag file is sorted in order to properly sequence the record keys before the user's program goes to end-of-job. When an ordered load is specified, the records are checked for proper sequencing.

A tag file is a sequential file that contains the record keys of the associated indexed file and indices that reference the data records of the indexed file.

A tag file record contains a key and a six-digit decimal address. The key is the same as the key in the record in the indexed file that the address points to. The tag file records are small, and are blocked under control of the RPG compiler so that as many records as possible are placed in 180 byte blocks, thereby contributing to efficient searching and sorting of the tag file. For example, when the key field requires seven bytes, the tag file record size is ten bytes, and the tag file contains 18 records per block.

B-Indexed. When the user enters a B in column 16 of the header card, a B-Indexed file type is specified. B-indexed files do not have associated tag files, so their records are normally ordered in ascending sequence according to the record key specified by the user. The file description entry for the B-indexed file contains information describing the length, type, and starting position in the record of the record key.

A B-indexed file can be created with the record keys in random sequence, in which case the file is sorted according to the specified key before the user's program goes to end-of-job. When an ordered load is specified, the records are checked for proper sequencing.

Direct Files

A direct file is a disk file whose records always occupy a specific position (disk address) in the file, regardless of the order in which the records are placed in the file. Therefore, direct files are classified as randomly organized files.

When a direct file is created or accessed, relative record numbers are used to identify the record locations within the file.

Figure 4-3 shows a card file and a direct file on disk that was created from the card file. The relative records number of each card is the relative record number within the disk file that the record occupies.

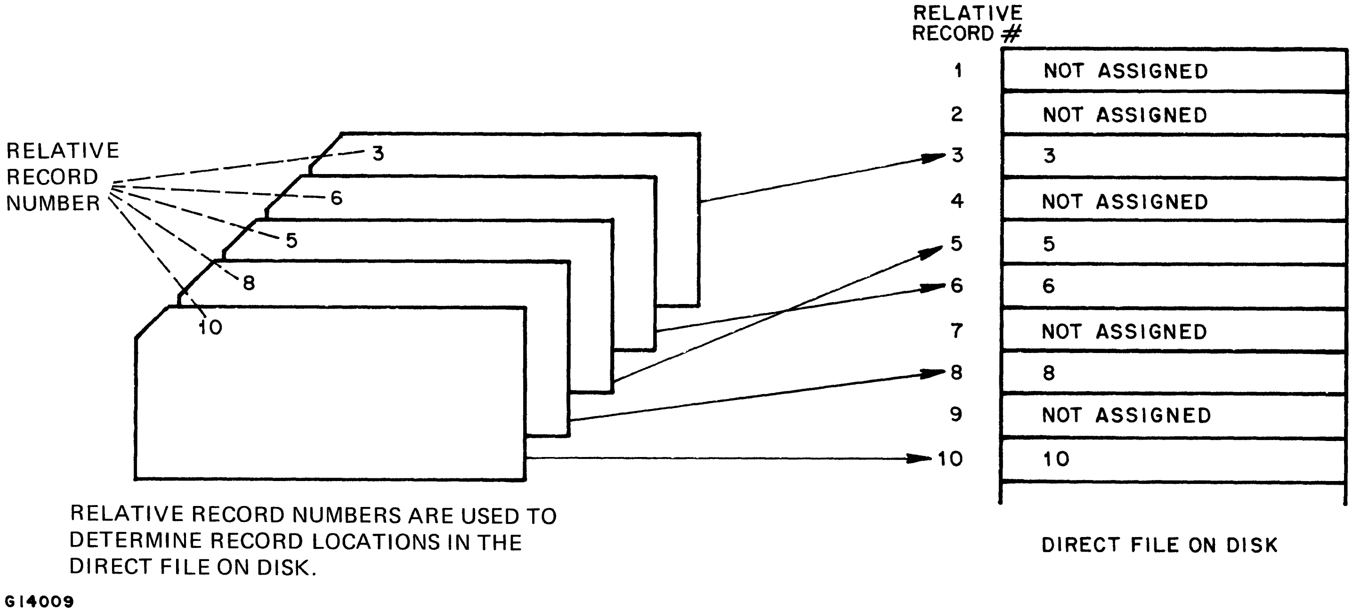


Figure 4-3. Direct File Organization

Note that the card file records do not have to be in sequence by relative record number when the direct disk file is created. Record areas in the direct disk file for which no records are assigned remain the same, but can be overlayed with new data at a later time.

File Processing

The manner in which files are processed is largely independent on the type of file organization used to create the file.

Files that were not created with the use of keys or relative record numbers cannot be read randomly by accessing them through an addrout file. The four major types of file processing that are described in the following pages are:

a. Consecutive.

b. Sequential by key.

c. Sequential within limits.

d. Random.

Consecutive

Consecutive file processing refers to the method of reading the records of a file in the order of their occurrence in the file. Sequential files are normally processed by consecutive file processing. Direct files and files designated as demand files can also be processed by use of consecutive file processing.

When consecutive file processing is used to read a direct file, all of the records are read from the file, so that embedded records that do not logically belong to the file must be handled by the user's program.

Sequential By Key

Sequential by key file processing is used for indexed disk files that are designated as primary, secondary, or demand files. Records are read from the file in ascending key sequence until all of the records in the file are processed or the program goes to end of job.

Sequential Within Limits

Sequential within limits file processing is used for processing indexed disk files by using either:

a. A limits file.

b. A SETLL operation code.

The use of a limits file and SETLL operation code for sequential within limits file processing are explained in the following pages.

Use Of A Limits File. Any indexed disk file that is designated a primary, secondary, or demand file can be processed sequentially within limits by utilizing a limits file. The purpose of the limits file is to specify the lower and upper limits (bounds) of the indexed file, so that only the records within those limits are processed. Each record in the limits file contains two record keys that describe the lower and upper bounds of the indexed file. The limits file can contain more than one record, but each record describes only one pair of lower and upper bounds.

For example, an indexed file that is alphabetically ordered by employee name can be accessed by using a limits file that has one or more records specifying the alphabetic range to be accessed. There are no restrictions on either the number of records or the bounds they specify. Therefore, the records in the limits file used to access such an indexed file could specify the following bounds:

A and J

D and J

A and I

F and Z

A and Z

The limits file can be on cards, disk, magnetic tape, or can be entered from the console printer.

Limits File Requirements And Restrictions. The restrictions and format requirements for records of a limits file follow:

a. Each limits file record can contain only one set of limits (lower and upper bounds).

b. The same set of limits (lower and upper bounds) can appear in more than one record of a limits file.

c. The lower bound key must begin in position one of the record. The upper bound key must immediately follow the lower bound key.

d. The lower bound key and the upper bound key in a limits file record can be the same. The result is that only the one data record described by the keys is read.

e. The length of limits file records must be twice the length of the record key specified in the index file. Therefore, the addition of leading zeroes or blanks may be necessary. If the key length specified in the indexed file is 10 bytes, the length of the limits file records is 20 bytes.

f. When the limits file is on disk, either A or P (whichever is applicable) must be specified for the record address type in column 31 of the file description of the limits file. A further description of the use of A and P follows in item g. When the limits file is not on disk, column 31 must remain blank, and alphanumeric keys are assumed.

g. The keys of the limits file can have a different format than the keys of the indexed file that is to be processed within limits. For example, the indexed file can have packed keys and the associated limits file can have alphanumeric keys.

When the keys of the indexed file and limits file have different formats, the following limitations apply:

1. The length of alphanumeric key fields must be twice the length specified for packed key fields, minus 1 or 2.

2. At execution time of the RPG program, the format of the key in the limits file is changed to the same format as specified for the key in the associated indexed file.

3. Only the digit portion of alphanumeric keys is significant.

h. The normal conventions for specifying the position of signs (left or right) within packed key fields applies to both indexed files and limits files. The files can have their signs in different positions within their key fields. The appropriate Dollar Card entries must be included in the file description specifications when necessary.

i. When an indexed file is to be processed within limits with the use of a limits file and an external indicator is specified in columns 71 and 72 of the file description, the external indicator must be coded for the indexed file.

j. When an indexed file is to be processed within limits with the use of a limits file and an end-of-file specification is desired, an E should be specified in column 17 of the file description specifications for the limits file.

<u>File Description Specifications - Indexed File</u>. The following fields must be coded in the File Description Specifications when the indexed file is to be processed sequentially within limits, except for the optional entries for columns 17 and 71-72.

Table 4-3a. File Description Specifications - Indexed File

Column Number	Field Name	Entry	Function or Description
15	File type	I	Input.
		U	Update.
16	File designation	P	Primary.
		S	Secondary.
		D	Demand.
17	End of file	E	End of file. This entry is optional.
28	Processing mode	L	Processing within limits.
31	Record address type	A	Alphanumeric keys.
		P	Packed keys.
40	Device	DISK	The indexed file is assigned to disk.
66	File addition/un-ordered	Blank	Additions to the indexed file are not allowed.
71-72	File condition	U1-U8	External indicator. This entry is optional.

File Description Specifications - Limits File. The following fields must be coded in the File Description Specifications for a limits file that is used in conjunction with an indexed file.

Table 4-3b. File Description Specifications - Limits File

Column Number	Field Name	Entry	Function or Description
15	File type	I	Input.
16	File designation	R	Record address file.
17	End of file	E	End of file. This entry is optional
19	File format	F	Fixed length records.
		Blank	Fixed length records.
20-23	Block length	Blank	Unblocked records.
		nnnn	A multiple of the record length.
24-27	Record length	nnnn	Record length must be at least twice key length.
29-30	Record address field	nn	Key length in bytes.
31	Record address type	A P	Alphanumeric keys. Packed keys.
32	File organization	Blank	One buffer.
		1-9	Number of buffers.
39	Extension code.	E	A further description of the limits file is coded on the extension specifications.
40-46	Device	CARD DISK TAPE CONSOLE	The limits file is assigned to the named device.

Extension Specifications - Limits File. The following fields must be coded in the Extension Specifications when a limits file is used in conjunction with an indexed file.

Table 4-3c. Extension Specifications - Limits File

Column Number	Field Name	Entry
11-18	From filename	The name of the limits file must be stated as it appears in the File Description Specifications.
19-26	To filename	The name of the indexed file associated with this limits file must be stated as it appears in the File Description Specifications.

Figure 4-4 shows an example program which illustrates the use of limits file processing. The indexed file name DATA is processed by means of the limits file named LIMITS. The program reads a record in the limits file to obtain the first limits pair. A limits pair contains the range in which the file named DATA is to be processed. There are two values which make up a limits pair. The first value is the low key and the second value is the high key. Processing of the data file begins with the low key and ends with the high key. Once the high key is processed, the next record in the limits file is read to obtain the next limits pair. Processing of the data file continues until the high key of the last record in the limits file is processed.

The program in figure 4-4 updates records in the file named DATA, by moving the literal UPDATE to the field named VALUE.

Table 4-3d lists the contents of the file named LIMITS.

Table 4-3d. Contents of the File Named LIMITS

Relative Record Number	Key Values in Hex	
	Low	High
1	0A	0B
2	2A	2D
3	0H	0I
4	0D	0E
5	1C	1D

NOTE

For reader convenience, the low and high keys are separated. The two keys are actually concatenated. For example, "0A" and "0B" actually appear as "0A0B".

Since the keys are alphanumeric, the zone portion of the right-most character contains the sign. Refer to Appendix B for the hexadecimal equivalents of B 1800/ B 1700 characters.

Table 4-3e lists the contents of the indexed file named DATA before and after the program in figure 4-4 is executed.

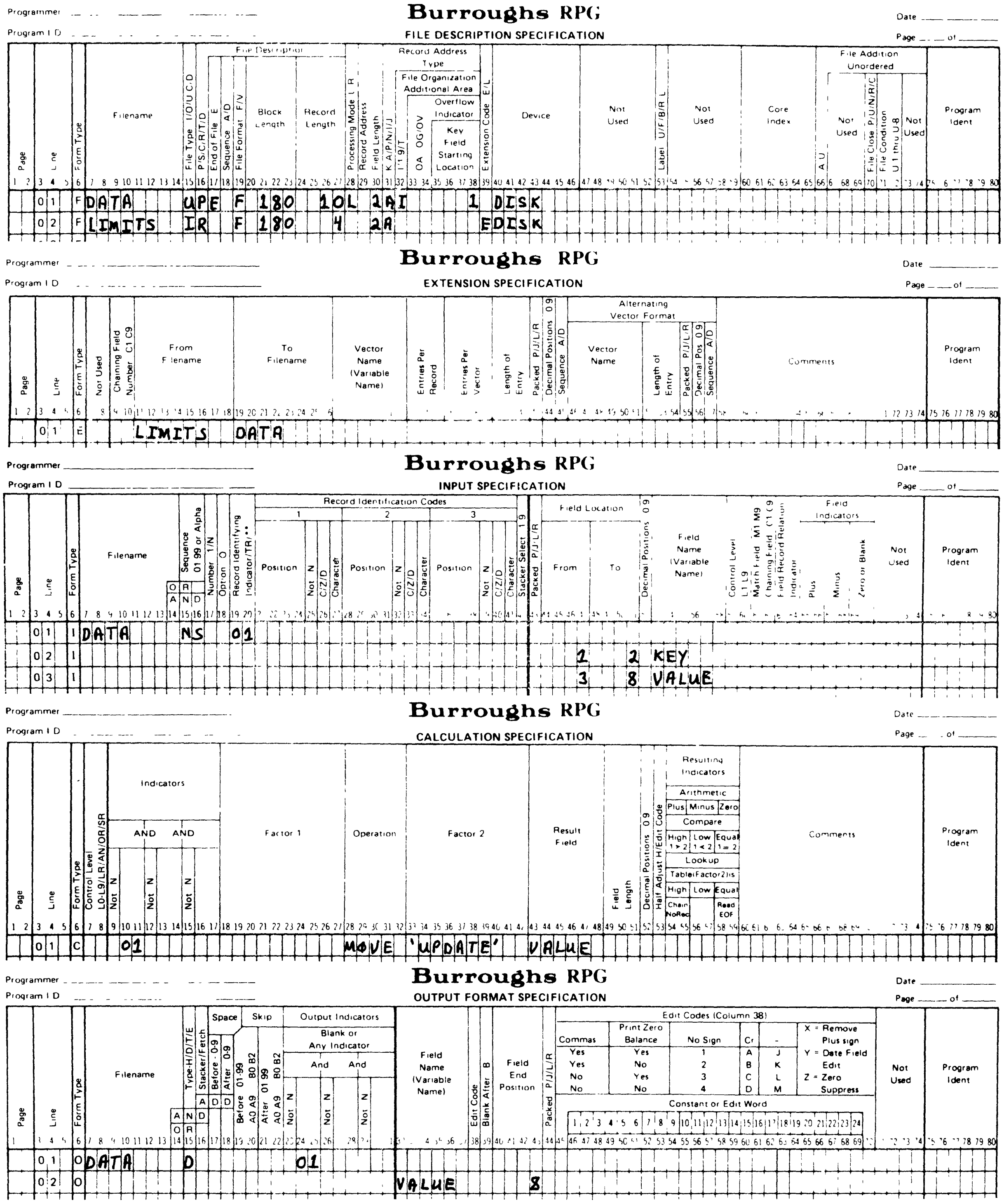
Table 4-3e. Contents of the File Named DATA

Relative Record Number	Contents of Each Record in Hex	
	Before	After
01	0A	0AUPDATE
02	0B	0BUPDATE
03	0C	0C
04	0D	0DUPDATE
05	0E	0EUPDATE
06	0F	0F
07	0G	0G
08	0H	0HUPDATE
09	0I	0IUPDATE
10	1? *	1? *
11	1A	1A
12	1B	1B
13	1C	1CUPDATE
14	1D	1DUPDATE
15	1E	1E
16	1F	1F
17	1G	1G
18	1H	1H
19	1I	1I
20	2? *	2? *
21	2A	2AUPDATE
22	2B	2BUPDATE
23	2C	2CUPDATE
24	2D	2DUPDATE
25	2E	2E

* The question mark character (?) represents positive zero (+0).

NOTE

Since the keys are alphanumeric, the zone portion of the right-most character contains the sign. Refer to Appendix B for the hexadecimal equivalents of B 1800/ B 1700 characters.



Programmer
Program I D

Burroughs RPG
FILE DESCRIPTION SPECIFICATION

Date
Page of

```
  01 FDATA    UPE F 180  10L 2AI     1 DISK
  02 FLIMITS  IR  F 180   4  2A       EDISK
```

Programmer
Program I D

Burroughs RPG
EXTENSION SPECIFICATION

Date
Page of

```
  01 E    LIMITS  DATA
```

Programmer
Program I D

Burroughs RPG
INPUT SPECIFICATION

Date
Page of

```
  01 IDATA    NS  01
  02 I                                        1   2 KEY
  03 I                                        3   8 VALUE
```

Programmer
Program I D

Burroughs RPG
CALCULATION SPECIFICATION

Date
Page of

```
  01 C   01                MOVE 'UPDATE'  VALUE
```

Programmer
Program I D

Burroughs RPG
OUTPUT FORMAT SPECIFICATION

Date
Page of

```
  01 ODATA    D        01
  02 O                         VALUE      8
```

G12067

Figure 4-4. Example Program Illustrating the Use of Limits

Use of SETLL Operation Code. Indexed files that are designated as demand files can be processed sequentially within limits by specifying SETLL in the Calculations Specifications. The indexed file can be designated as an input or update file, record additions are not allowed, and the record keys must be either alphanumeric or packed.

NOTE

> Indexed files designated as demand files cannot be processed sequentially within limits by both a limits file and the SETLL operation code in the same program.

Calculation Specifications - SETLL. The following fields of the Calculation Specifications must be coded when the SETLL instruction is to be used.

Table 4-3f. Calculation Specifications - SETLL

Column Number	Field Name	Entry	Function or Description
18-27	Factor 1		Field name or literal.*
28-32	Operation	SETLL	Operation code.
33-42	Factor 2		File name of indexed file.*

* This entry must be left-justified.

The record keys can be either alphanumeric or packed. When the keys are alphanumeric, the field or literal length must be equal to the length of the key specified in the file description specifications for the indexed file. When the keys are packed, the field or literal must be numeric and the length should be less than or equal to the length of the key specified for the indexed file, or truncation occurs.

When SETLL is specified, the lower limit of the record key for the indexed file to be processed sequentially within limits is established by the absolute value contained in a field or literal. The field or literal is specified in columns 18-27 of the Calculation Specifications. The upper limit of the record key defaults to the highest record key that exists in the file.

At execution time of the RPG program, records of the indexed file are processed consecutively, starting with either the lowest record key in the file or with the record key specified by the SETLL operation.

File Description Specifications - SETLL. When an indexed disk file is to be processed sequentially within limits using the SETLL operation, the following entries are required in the File Description Specifications. All other fields are coded as they normally are for sequentially processed indexed files.

Table 4-3g. File Description Specifications - SETLL

Column Number	Field Name	Entry	Function or Description
15	File type	I U	Input file. Update file.
16	File designation	D	Demand file.
17	End of file	Blank	
28	Processing mode	L	Limits processing.
31	Record address type	A P	Alphanumeric keys. Packed keys.
40	Device	DISK	The indexed file is assigned to disk.
66	File addition/ Unordered	Blank	Additions to the indexed file are not allowed.
71-72	File condition	U1-U8	External indicator. This entry is optional.

Record Processing Using SETLL. There is one exception to the normal convention of reading a demand file, as follows:

> When a read is performed on a demand file that is to be processed sequentially within limits using SETLL, the entire file can be read as many times as desired even though end of file has been reached.

Read operations performed on a demand file prior to execution of a SETLL instruction are performed consecutively starting with the record having the lowest record key. When end of file is reached, the file can be read again starting with the record having the lowest record key.

Read operations performed on a demand file after a SETLL instruction has been executed are performed consecutively starting with the first record in the file whose key is equal to or greater than the low record key established by the SETLL instruction. When end of file is reached, the file can be processed consecutively starting with either:

a. The record with the lowest record key if a new SETLL instruction is not executed.

b. The first record in the file whose key is equal to or greater than the low record key established by a new SETLL instruction.

When a read operation is specified in the Calculation Specifications, an indicator should be specified in columns 58-59. The indicator is turned on when end of file is reached on the indexed file, and remains on until the program turns it off.

When a read operation is specified in the Calculation Specifications and no indicator is specified, the following occurs during program execution:

> When an end-of-file condition occurs during reading of the indexed file, the following message is displayed on the console printer:
>
> REOF
>
> Following the display of REOF, the operator must enter either the GO or ST system control instruction on the console printer. Entry of GO causes the program to continue, and reading can be performed and the records processed consecutively starting with either:
>
> a. The record with the lowest record key if a new SETLL instruction is not executed.
>
> b. The first record in the file whose key is equal to or greater than the low record key established by a new SETLL instruction.

Random

Random file processing can be used for direct, indexed, and sequential files. Records are written to or read from the files only when chain statements are executed that identify the records. During program execution, records are read during the time that calculations are being performed, which allows the programmer to use data from the records for detail or total calculations. When a chained update file is processed randomly, records can be:

a. Read and modified during the time in which detail or total calculations are being performed, and rewritten in the file during the detail or total output phase, or

b. Rewritten during the time in which detail or total calculations are being performed by use of the EXCPT operation code.

Random Processing - Sequential And Direct Files. When sequential or direct files are processed randomly, records are selected by the use of relative record numbers. These relative record numbers identify the positions of the records in the file relative to the beginning of the file. For example, the relative record numbers for the first, third, seventh, and eighth records are 1, 3, 7, and 8 respectively.

Random Processing - Indexed Files. When indexed files are processed randomly, records are selected on the basis of record key values. An indexed file that is an input file can be processed randomly by the use of relative record numbers.

Random Processing - Addrout Files. Addrout files are created by the tagsort option of SORT, and can be used to randomly process a data file. The addrout file is comprised of 8-digit, 4-bit, decimal-encoded relative record numbers that indicate the relative position of the associated records in the data file.

Any sequential, indexed, or direct organized disk file that is designated as a primary or secondary file can be randomly processed by using an addrout file. During program execution, a record containing a relative record number is read from the addrout file, and the relative record number is then used to locate and read a record from the data file that is being randomly processed. Only the records of the data file whose relative record numbers are in the addrout file are processed. File processing continues until an end-of-file condition occurs for the addrout file.

Advantages of Using Addrout Files. The advantages of using addrout files for random processing of data files are as follows:

a. A data file can be sorted in many different sequences by using various control fields from the records in the data file.

b. Addrout files can be used in the processing of files that are organized sequential, indexed, or direct.

c. Addrout files require less disk space than comparable tagfiles composed of both record key fields and relative record numbers.

d. More than one addrout file can be used with a program.

Creating Addrout Files. An addrout file can be created by sorting a file with the tagsort option of SORT. Specify the same blocking factor in the tagsort parameters as specified in the File Description Specifications for the addrout file. A blocking factor of 45 is preferred. If the file is not blocked, each 4-byte relative record number occupies one disk segment. For additional information about SORT and tagsort, refer to the B 1800/B 1700 Systems Software Operation Guide, Form No. 1068731.

File Description Specifications - Data Files. The following entries are required in the File Description Specifications for a disk file that is to be processed randomly by using an addrout file. All other fields are coded as they normally are for consecutively processed disk files.

Table 4-3h. File Description Specifications - Data Files

Column Number	Field Name	Entry	Function or Description
15	File type	I	Input file.
		U	Update file.
16	File designation	P	Primary file.
		S	Secondary file.
17	End of file	E	If an end-of-file test is desired, it should be coded for the addrout file. This entry is optional.
19	File format	F	Fixed length records.
		Blank	Fixed length records.
20-23	Block length	Blank	File is to be unblocked.
		nnnn	Must be a multiple of the record length.
28	Processing mode	R	Addrout processing.
31	Record address type	I	Addrout processing.
40	Device	DISK	The data file is assigned to disk.
66	File addition/ Unordered	Blank	Additions to the data file are not allowed.
71-72	File condition	U1-U8	External indicator. This entry is optional.

File Description Specifications - Addrout File. The following entries are required in the File Description Specifications for an addrout file:

Table 4-3i. File Description Specifications - Addrout File.

Column Number	Field Name	Entry	Function or Description
15	File type	I	Input file.
16	File designation	R	Record address file.
17	End of file	E	If an end-of-file test is desired, it should be coded for the addrout file. This entry is optional.
19	File format	F Blank	Fixed length records. Fixed length records.
20-23	Block length	Blank	Unblocked.
		nnnn	Block length in bytes. Must be a multiple of record length.
24-27	Record length	4	Record length in bytes.
29-30	Record address field	4	Record key length in bytes.
31	Record address type	I	Addrout file.
32	File organization	T	Tag records.
39	Extension code	E	Extension specifications.
40	Device	DISK	The addrout file is assigned to disk.
71-72	File condition	Blank	Not applicable.

Extension Specifications - Addrout File. The following fields must be coded in the Extension Specifications when a data file is processed with the use of an addrout file.

Table 4-3j. Extension Specifications - Addrout File

Column Number	Field Name	Entry
11-18	From filename	The name of the addrout file must be stated as it appears in the File Description Specifications.
19-26	To filename	The name of the data file associated with this addrout file must be stated as it appears in the File Description Specifications.

Table 4-3k. shows which combinations of file processing methods and file organization types can be used with primary, secondary, demand, and chained files. Allowable combinations are indicated by an X.

Table 4-3k. Valid Combinations of File Types and File Processing

File Processing Method	Primary, Secondary, and Demand Files				Chained Files			
Consecutive	X	X	X	X				
Sequential By key			X	X				
Sequential Within Limits			X	X				
Addrout	X	X	X	X				
Random by Relative Record Number					X	X	X	X
Random By key							X	X
	SEQUENTIAL	DIRECT	B INDEXED	INDEXED	SEQUENTIAL	DIRECT	B INDEXED	INDEXED
	File Organization Type							

33-34 OVERFLOW INDICATOR

This field applies only to files assigned to the printer, and is used to specify the overflow indicator used to condition records being printed in the file. Each printer file must have a unique overflow indicator assigned to it, if overflow printing (i.e., the printing of special lines when the overflow file is fetched) is desired for that file. Acceptable entries for this field are:

Entry	Definition
OA-OG, OV	Specified indicator is used to condition records in the file.
Blank	Specifies automatic skipping. No default overflow indicator is assigned.

If this field is left blank, no overflow indicator is assigned to the printer file. Overflow is then handled automatically by the RPG program.

If this field contains an overflow indicator but no output is conditioned on that indicator, a continuous printer listing is produced.

For additional information concerning overflow, refer to the subsection of this manual titled Printer File Handling in Section 10.

35-38 KEY FIELD STARTING LOCATION

This field applies only to indexed files and specifies the character position within each record where the key field begins. All key fields in the file must occupy the same position in each record. The entry must be numeric, and right-justified. Leading zeroes may be omitted.

39 EXTENSION CODE

This field applies to printer output files, table or array files that are to be read during program execution, and record address files. The contents of this field indicates whether the file is further described on the Extension Specifications (table and array files, or record address files) or the Line Counter Specifications (printer files) form. Valid entries for this field are:

Entry	Definition
E	An Extension Specification must further describe the file. This entry is only allowed for input table files and record address files
L	A Line Counter Specification must further describe the file. This entry is only allowed for printer files.
Blank	No Line Counter or Extension Specifications are required for this file; however, they may still occur.

40-46 DEVICE

This field is used to identify the input/output device to which the file is assigned, or the data communication entry that indicates that the file is a Telecommunications file. All entries must be left-justified. Valid entries for this field are:

Entry	Definition
READER	80-column card reader.
MFCU1	96-column MFCU - primary hopper.
MFCU2	96-column MFCU - secondary hopper
PUNCH	80-column card punch.
PRINTER or PRINTR2	Line printer. (A maximum of 5 printer files may be specified).
TAPE	Magnetic tape.
DISK	Disk file.
CONSOLE*	Console printer,
BSCA	Telecommunications file.
REMOTE	Telecommunications file.
DATACOM	Telecommunications file,

*Only one CONSOLE file may be specified and must be specified as a display file (see DSPLY in Section 9), Any card device name (i.e., READER, PUNCH, MFCU1, etc.) is considered only to denote a card file. The actual type (input, output, combined) is determined by the entry in column 15 of the File Specifications. Table 4-4 lists the devices available for each of the file types.

Limits files can be assigned to any input device. Addrout files can only be assigned to an input device that can also be used by the systems SORT utility for output. Only disk files can be processed sequentially within limits or by addrout files.

Table 4-4. Device Assignment for Files

File	Type	Devices
Primary or Secondary Input Files	Card	MFCU1 or MFCU2 READER
	Disk	DISK
	Tape	TAPE
Demand Files	Card	MFCU1 or MFCU2 READER
	Data Communications	REMOTE
	Disk	DISK
	Tape	TAPE
Table Files	Card	MFCU1 or MFCU2 READER
	Disk	DISK
	Tape	TAPE
Chained Input Files	Disk	DISK
Update Files	Disk	DISK
Combined Files	Card	MFCU1 or MFCU2
	Data Communications	REMOTE
Output Files	Card	MFCU1 or MFCU2 PUNCH
	Disk	DISK
	Listing	PRINTER, PRINTR2
	Data Communications	REMOTE
Display File	Console Printer	CONSOLE

The B 1800/B 1700 will also accept the following additional device names in column 40-46:

READ01
MFCM1
MFCM2
READ20
READ40
READ42
CRP
CRP20
DATA96

PUNCH20
PUNCH42
PRINTUF
PRINTLF
DISK11
DISK11F
DISK45
SPO

If communications with a remote device is desired, the entry REMOTE, DATACOM, or BSCA is used. A Telecommunication Card is required for each data communications file declared in an RPG program. Refer to Section 7, Telecommunications Card Specifications, for a detailed description of the entries required on the Telecommunication Card.

Data communications files can be input, output, or combined files, and are used to reference Network Definition Language (NDL) files associated with the network controller. The filename in columns 7-14 of the File Description Card entry must be the same as the filename of the NDL file being referenced, or the names should be file-equated with a File statement. For information concerning the Network Definition Language, refer to the B 1800/B 1700 System Network Definition Language Reference Manual, Form Number 1073715.

REMOTE

When REMOTE is specified in columns 40-45, the remote devices are accessed on a demand basis. The fields of the file description specifications shown in Table 4-4a must be coded with the applicable entry when REMOTE is specified. The remaining fields must be coded as they would be for any demand file.

Table 4-4a. File Description Specifications - REMOTE

Column Number	Field Name	Entry	Function or Description
15	File type	I	Input.
		O	Output.
		C	Combined.
16	File designation	D	Demand file.
32	File organization	Blank	One buffer.
		1-9	Number of buffers.
40	Device	REMOTE	Remote devices are to be accessed on a demand basis.

The SEND and RECV operation codes are used in calculations when REMOTE is specified, and allow the user to specify indicators to report on:

a. Exception conditions.

b. Incomplete I/O operations.

c. End-of-file conditions.

Refer to the subsection titled Programmed Control Of Input And Output in section 9 for detailed information about SEND and RECV.

DATACOM Or BSCA

When DATACOM or BSCA is specified starting in column 40, the fields of the file description specification shown in Table 4-4b must be coded as shown.

Table 4-4b. File Description Specifications - DATACOM or BSCA

Column Number	Field Name	Entry	Function or Description
40	Device	DATACOM	Telecommunications file.
		BSCA	Telecommunications file.

53 LABELS ("FILE OPEN" field)

Any entry is allowed here. Only entries meaningful to the B 1700 will be used. Meaningful entries are:

Entry	Definition
U	File is unlabelled.
F	Special forms.
B	Forces backup.
Blank or S	→ Indicates the B 1700 Standard Label.

L → I or U files - cates file not available to other pgms. while the disk file is opened by this pgm.

ENTRY	DEFINITION
R	I or U disk files - indicates this file is available to othe pgms. for I only.

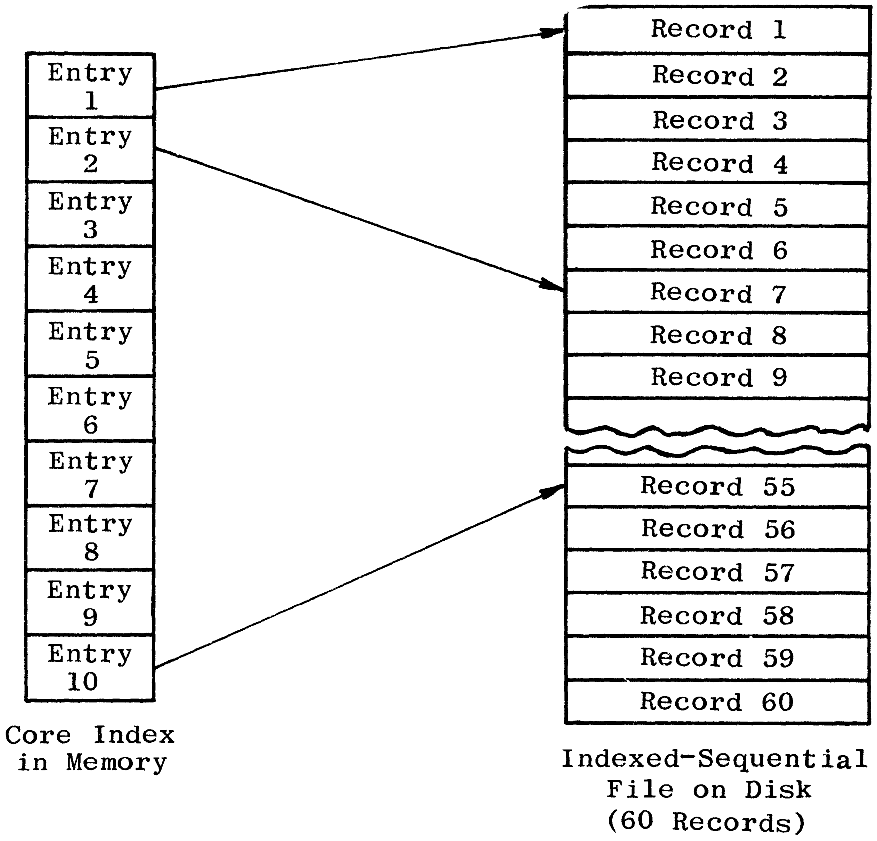
60-65 CORE INDEX

The core index is a rough table in memory which contains entries representing the key fields of the data file on disk. The use of the core index may significantly reduce the time needed to process an indexed file because it allows a more direct access to the specific record required. The program must search only a portion of the file instead of all entries in the file preceding the required record. The core index field must be blank when files other than chained indexed files are specified.

This field is used only by files accessed using the CHAIN operator (i.e., indexed-sequential files) to specify the number of bytes of memory to be set aside for indexes. The entry must be right-justified, and leading zeros may be omitted. The maximum size that may be entered is 9999.

At Beginning-of-Job, the core index is built in memory to speed the access time to process the file. The number of keys contained in memory is determined by taking the key length as specified in columns 29-30 and dividing it into the core index as specified in positions 60-65. The program then looks at the End-of-File pointer contained in the File Information Block to determine the total size of the file. The program divides the file into even partitions for the allowable number of keys in memory, and reads every nth record filling the core table in memory with just the data key fields. If no entry is made, a minimum (default) number of 10 of these keys (20 keys if the keys are in packed or numeric form) is automatically specified.

Once the core index is built, the program will begin processing. As an access occurs, the code emitted will do a binary search of the core index to determine in which partition the record should be found. It calculates the lowest actual address and one greater than the highest to be used as the area to be searched for the record. At this point the program reverts to a binary search of the disk.



G14014

Figure 4-5. Core Index Selection of Data Keys

Several considerations should be made in space/time trade-offs when processing indexed-sequential files. The core index should reflect the size of the data file to be accessed and the number of accesses to be performed. When a significant number of accesses are to be performed, the core index should be as large as practical; however, the core index should be moderate-to-small if only a few accesses are desired.

66 FILE ADDITION/UNORDERED

This field applies only to sequential disk and indexed disk files and indicates:

a. New records are to be added to an existing file; or

b. Unordered records are to be loaded into an output file.

When an entry is made in column 66 for an indexed file, it implies that the tag or data file is to be ordered in ascending sequence according to the key specified in columns 29-31 and 35-38 of the File Description Specifications.

For indexed files, the tag file is maintained in ascending sequence.

For B-Indexed files, the data file is maintained in ascending sequence.

Column 66 must be left blank when indexed files that are processed by record address files are specified.

Valid entries for the file addition/unordered field are:

Entry	Definition
A	Records are to be added to an existing sequential output file or to any indexed disk file.
U	Records are to be loaded for an indexed file in unordered sequence (creating a new indexed file).
Blank	All files other than sequential output files and indexed disk files that are processed by record address files. If the file is an indexed output file, the records must be loaded in ascending key sequence.

The letter A must be entered in column 66 if there is an ADD entry in columns 16-18 of the Output-Format Specifications record description of the file.

If the letter A is entered in this field for any file other than an update file, all output records must have ADD specified on the Output-Format Specifications (columns 16-18). However, the Output-Format Specifications are optional.

Records can only be added to an existing disk file. When records are added to a sequential output file or indexed disk file the records are added at the end of the file. After all records have been added to an indexed disk file, the tag file or data file is sorted into ascending key sequence depending on the type of indexed file specified, and the program then goes to end-of-job.

The coding options for columns 15 and 66 are shown in Table 4-5.

Table 4-5. Columns 15 and 66 Coding Options

Column 15	Column 66	Function
I	Blank	Read only file. Records cannot be added or updated.
I	A	Read only file. Existing records cannot be updated but new records can be added.
0	Blank	The indexed file is created in the order in which records are loaded to the file. It is the user's responsibility to ensure that records are in the desired order. A run-time error occurs if records are improperly ordered.
0	A	Data records are to be added to an existing file.
0	U	An indexed file is created from data records which may not be in ascending key sequence. For B-Indexed files, the data file is created in ascending key sequence. For indexed files, the corresponding tag file is created in ascending key sequence.
U	Blank	Read and update existing records.
U	A	Read and update existing records and add new records.

70 TAPE REWIND

This field specifies the action to be taken during closing of the file, and includes the provision for rewind and/or lock for tape reels, where desired. Valid entries are:

Entry	Definition
P	Close with purge.
U	Close with unload (lock).
N	Close with no rewind.
Blank	Close with release.
R	Close with remove.

To show the effects of the various options, each type of file is discussed separately in the paragraphs that follow.

Card Input

All options are ignored. The input areas are released and the unit is returned to the MCP.

Card Output

All options are ignored. The output areas are released, the trailer label is punched, and the unit is returned to the MCP.

Tape Input

The effects of the various options for tape input are:

Entry	Effect
U	Releases the input areas, rewinds the tape, and the MCP marks the unit "not ready".
N	The input areas are not released, the tape is not rewound, and the device remains assigned to the program.
Blank	Releases the input areas, rewinds and returns the unit to the MCP.

Tape Output

The effects of the various options for tape output are:

Entry	Effect
P	Releases the output areas, writes the trailer label, rewinds the tape, and overwrites the label, thus making the tape a scratch tape.
U	Releases the output areas, writes the trailer label, rewinds the tape, and the MCP marks the unit "not ready".
N	Writes the trailer label. The tape is not rewound, and the device remains assigned to the program.
Blank	Releases the output areas, writes the trailer label, rewinds the tape and returns the unit to the MCP.

Printer Output

All options are ignored. A page is ejected, a trailer label is written, and the printer is returned to the MCP.

Disk Files

The effects of the various options assigned to disk are described in terms of "old files" and "new files". An old file is one that already exists on disk and appears in the MCP Disk Directory. A new file is one created by the program, and does not appear in the Directory. A new file may only be referenced by the program which creates it.

Entry	Effect
P	An old file is removed from the disk and deleted from the Directory, or a new file is removed from disk.
U	For an old file, the file remains in the Directory and is made available. A new file is entered in the Directory (thereby making it an old file) and made available.
N	Use of this option is not permitted with disk files.
Blank	Same as for U.
R	An old file is removed from the disk and deleted from the Directory. The new file is entered in the Directory and made available.

71-72 FILE CONDITION

This field applies to input (excluding table input files), update, output, and combined files, and indicates whether or not the file is conditioned by an external indicator. The entry indicates (at execution time) whether or not the file is to be used by the program. If a file is conditioned by an external indicator, the file is used only when that indicator is ON. When the indicator is OFF, the file is treated as though End-of-File had been reached, and no records may be read or written into the file. Valid entries for this field are:

Entry	Definition
U1-U8	The specified external indicator is used to condition the file.
Blank	The file is not conditioned by an external indicator.

External indicators (U1-U8) may be used to condition files. For example, a single program may have two uses. On some occasions, two files may be required to be read (or written); on others, only one file may be required to be read (or written). The optional file can be conditioned by an external indicator.

For example, in a program that has two applications, external indicators determine which application has been selected and what files are to be read (or written).

If a file is conditioned in this way, records written to the output file may also be conditioned by the same indicator on Output Specifications. If an output file is conditioned by an external indicator, every output record described on the Output-Format Specifications for that file should be conditioned by the same external indicator, otherwise the object program will still build the output record and perform blank-after operations but suppress the write operation. Any calculation operation which should not be done when the file is not in use (especially CHAIN and READ operations) may also be conditioned by the same indicator, otherwise the object program will automatically condition the CHAIN or READ on the same external indicator.

If an external indicator is entered in columns 71-72 for a record address file, the same indicator must be assigned to the associated data file.

It is the user's responsibility to set the external indicators (U1-U8). The external setting and interrogation of the external indicators is done by the use of eight one-bit program switches numbered 1 through 8. Each switch can contain a value of 0 or 1, and can be set at run time by including the switch attribute as part of the EXECUTE statement. The syntax for assigning values to the program switches is as follows:

```
EXECUTE <program name> SW <switch number> <switch value>
```

The following EXECUTE statement would turn on external indicator U1:

```
EXECUTE RPG/TEST SW 1=1
```

The program switches can also be permanently initialized by using the MODIFY control instruction as follows:

```
MODIFY <program name> SW  <switch number>  <switch value>
```

When any (but not necessarily all) of the program switches are initialized with the SWITCH attribute used with either the EXECUTE or MODIFY control instruction, console printer input is not requested at program execution time. If the switches are not initialized, operator action is required to set the external indicators by use of the AX input message in response to an ACCEPT message. The following examples show the use of the AX input message:

```
<program number> AX1    (This message results in setting U1.)
<program number> AX01   (This message resets U1 and sets U2.)
```

75-80 PROGRAM IDENTIFICATION

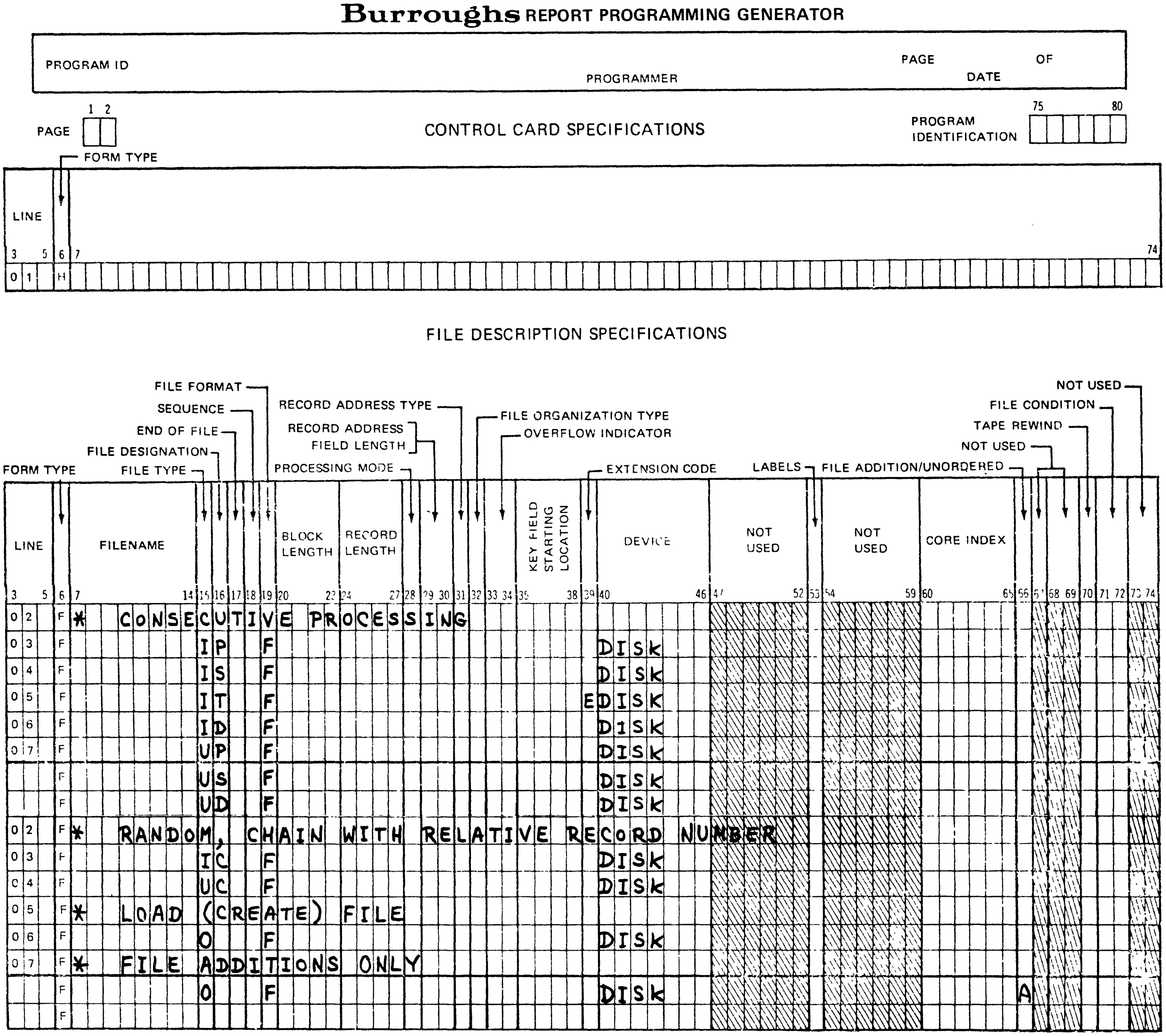
Refer to Section 2 for a complete description.

Figures 4-6 through 4-13 show coding examples for a variety of file types.

FILE DESCRIPTION SPECIFICATIONS

LINE 3-5	FORM TYPE 6	FILENAME 7-14	FILE TYPE 15	FILE DESIGNATION 16	END OF FILE 17	SEQUENCE 18	FILE FORMAT 19	BLOCK LENGTH 20-23	RECORD LENGTH 24-27	PROCESSING MODE 28	RECORD ADDRESS FIELD LENGTH 29-30	RECORD ADDRESS TYPE 31	FILE ORGANIZATION TYPE 32	OVERFLOW INDICATOR 33-34	KEY FIELD STARTING LOCATION 35-38	EXTENSION CODE 39	DEVICE 40-46	NOT USED 47-52	LABELS 53	NOT USED 54-59	CORE INDEX 60-65	FILE ADDITION/UNORDERED 66	NOT USED 67-69	TAPE REWIND 70	FILE CONDITION 71-72	NOT USED 73-74
02	F																									
03	F	* SEQUENTIAL, BY KEY, NO ADDITIONS																								
04	F		I	P			F				nn	A	I		1		DISK									
05	F		I	S			F				nn	A	I		1		DISK									
06	F		I	D			F				nn	A	I		1		DISK									
07	F		U	P			F				nn	A	I		1		DISK									
	F		U	S			F				nn	A	I		1		DISK									
	F		U	D			F				nn	A	I		1		DISK									
02	F																									
03	F	* SEQUENTIAL, BY KEY, WITH ADDITIONS																								
04	F		I	P			F					A	I				DISK					A				
05	F		I	S			F					A	I				DISK					A				
06	F		I	D			F					A	I				DISK					A				
07	F		U	P			F					A	I				DISK					A				
	F		U	S			F					A	I				DISK					A				
	F		U	D			F					A	I				DISK					A				
03	F	* RANDOM, BY KEY, NO ADDITIONS																								
04	F		I	C			F					A	I				DISK									
05	F		U	C			F					A	I				DISK									
06	F																									
07	F	* RANDOM, BY KEY, WITH ADDITIONS																								
	F		I	C			F					A	I				DISK					A				
	F		U	C			F					A	I				DISK					A				
02	F	* UNORDERED LOAD																								
03	F		O				F					A	I				DISK					U				
04	F	* ORDERED LOAD																								
05	F		O				F					A	I				DISK									
06	F	* FILE ADDITIONS																								
07	F		O				F					A	I				DISK					A				
	F																									
	F																									

Figure 4-6. Example of File Description Entries for Indexed Disk Files



Burroughs REPORT PROGRAMMING GENERATOR

PROGRAM ID

PROGRAMMER

PAGE OF

DATE

PAGE 1 2

CONTROL CARD SPECIFICATIONS

PROGRAM IDENTIFICATION 75 80

FORM TYPE

LINE 3 5	FORM TYPE 6	7 … 74
0 1	H	

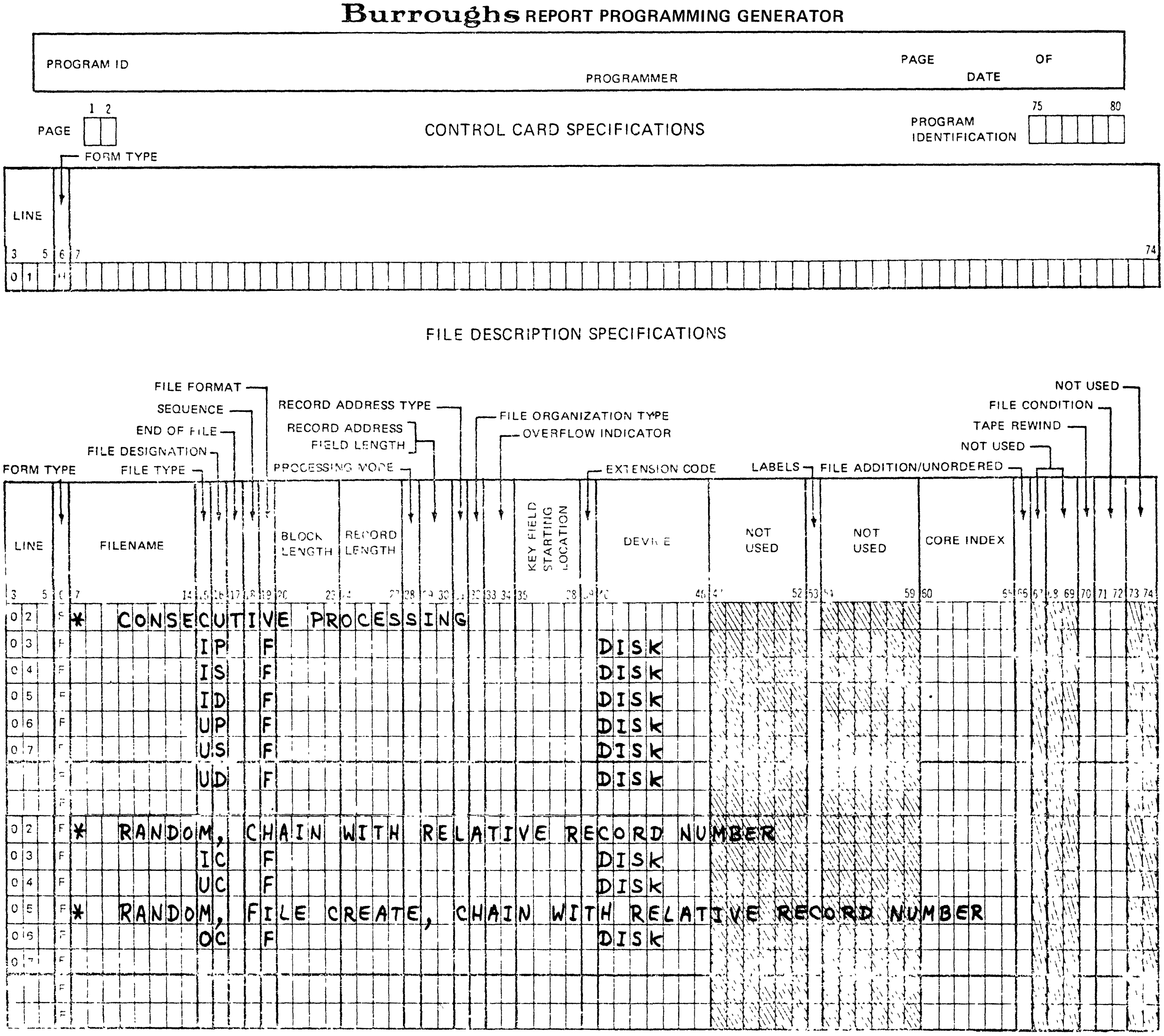
FILE DESCRIPTION SPECIFICATIONS

LINE 3 5	FORM TYPE 6	FILENAME 7 14	FILE TYPE 15	FILE DESIGNATION 16	END OF FILE 17	SEQUENCE 18	FILE FORMAT 19	BLOCK LENGTH 20 23	RECORD LENGTH 24 27	EXTENSION CODE 39	DEVICE 40 46	FILE ADDITION/UNORDERED 66
0 2	F	* CONSECUTIVE PROCESSING										
0 3	F		I	P			F				DISK	
0 4	F		I	S			F				DISK	
0 5	F		I	T			F			E	DISK	
0 6	F		I	D			F				DISK	
0 7	F		U	P			F				DISK	
	F		U	S			F				DISK	
	F		U	D			F				DISK	
0 2	F	* RANDOM, CHAIN WITH RELATIVE RECORD NUMBER										
0 3	F		I	C			F				DISK	
C 4	F		U	C			F				DISK	
0 5	F	* LOAD (CREATE) FILE										
0 6	F		O				F				DISK	
0 7	F	* FILE ADDITIONS ONLY										
	F		O				F				DISK	A
	F											

Other column headings on the form: RECORD ADDRESS FIELD LENGTH (28), PROCESSING MODE (29), RECORD ADDRESS TYPE (31), FILE ORGANIZATION TYPE (32), OVERFLOW INDICATOR (33 34), KEY FIELD STARTING LOCATION (35 38), NOT USED (47 52), LABELS (53), NOT USED (54 59), CORE INDEX (60 65), NOT USED (67), TAPE REWIND (70), FILE CONDITION (71 72), NOT USED (73 74).

G12031

Figure 4-7. Example of File Description Entries for Sequential Files



Burroughs REPORT PROGRAMMING GENERATOR

PROGRAM ID PROGRAMMER PAGE OF DATE

PAGE □□

CONTROL CARD SPECIFICATIONS

PROGRAM IDENTIFICATION □□□□□□

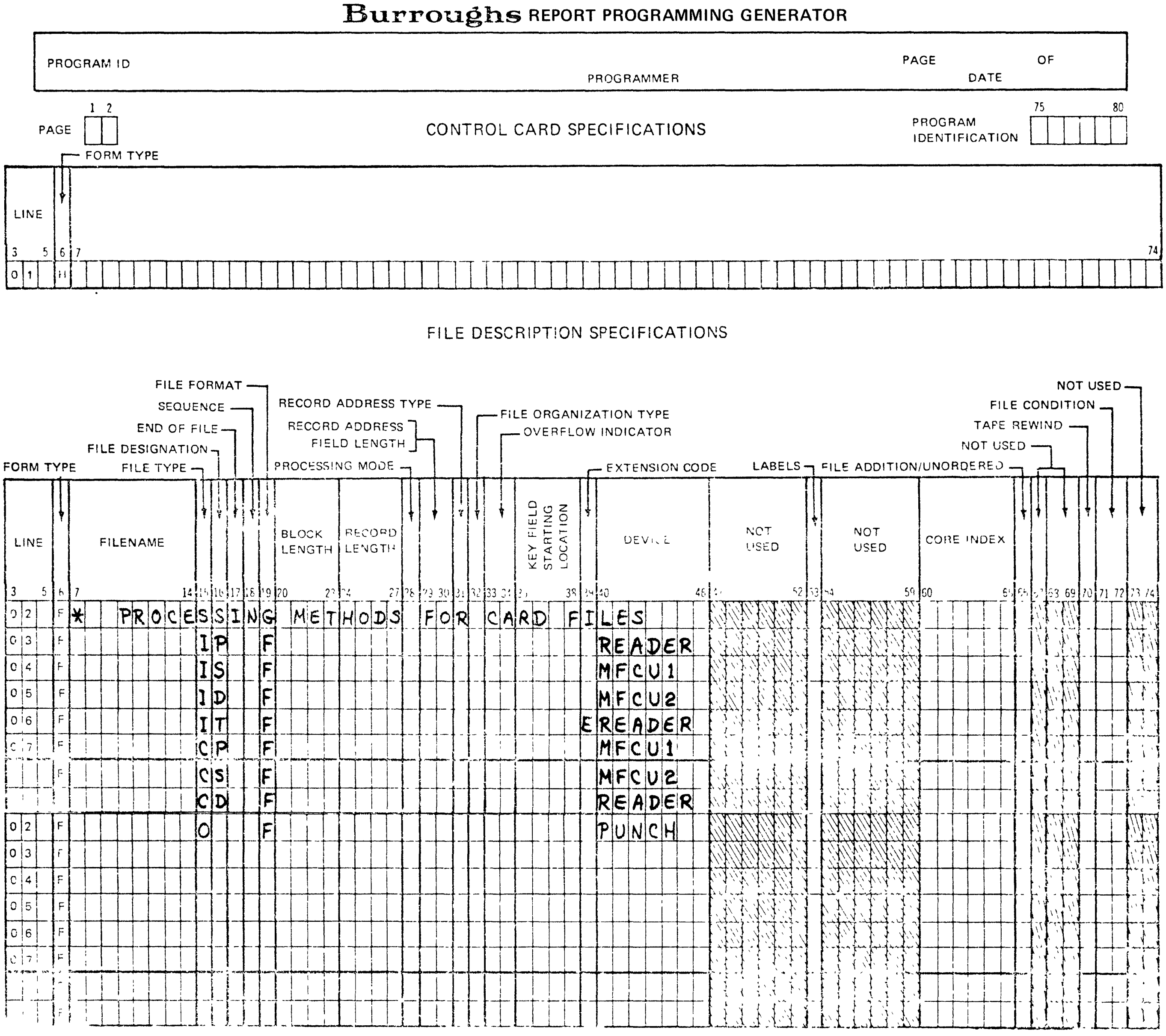
LINE	FORM TYPE	7–74
01	H	

FILE DESCRIPTION SPECIFICATIONS

LINE	FORM TYPE	FILENAME	FILE TYPE	FILE DESIGNATION	END OF FILE	SEQUENCE	FILE FORMAT	BLOCK LENGTH	RECORD LENGTH	PROCESSING MODE	RECORD ADDRESS FIELD LENGTH	RECORD ADDRESS TYPE	FILE ORGANIZATION TYPE	OVERFLOW INDICATOR	KEY FIELD STARTING LOCATION	EXTENSION CODE	DEVICE	NOT USED	LABELS	NOT USED	CORE INDEX	FILE ADDITION/UNORDERED	NOT USED	TAPE REWIND	FILE CONDITION	NOT USED
02	F	* CONSECUTIVE PROCESSING																								
03	F		I	P			F										DISK									
04	F		I	S			F										DISK									
05	F		I	D			F										DISK									
06	F		U	P			F										DISK									
07	F		U	S			F										DISK									
	F		U	D			F										DISK									
	F																									
02	F	* RANDOM, CHAIN WITH RELATIVE RECORD NUMBER																								
03	F		I	C			F										DISK									
04	F		U	C			F										DISK									
05	F	* RANDOM, FILE CREATE, CHAIN WITH RELATIVE RECORD NUMBER																								
06	F		O	C			F										DISK									
07	F																									
	F																									
	F																									

G12032

Figure 4-8. Example of File Description Entries for Direct Files



Burroughs REPORT PROGRAMMING GENERATOR

PROGRAM ID — PROGRAMMER — PAGE — OF — DATE

CONTROL CARD SPECIFICATIONS

PAGE (1 2) — PROGRAM IDENTIFICATION (75–80)

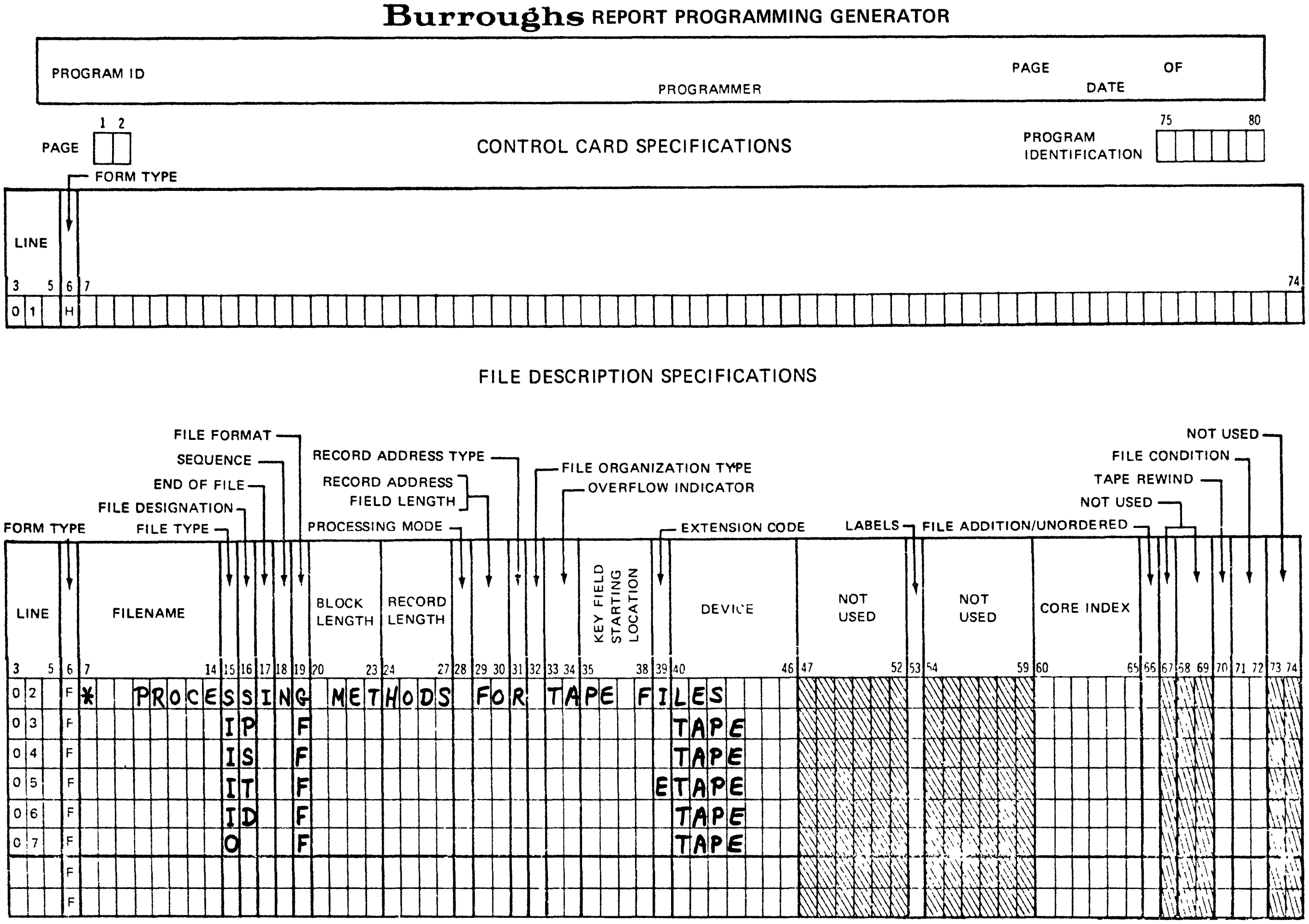
LINE (3–5)	FORM TYPE (6)	7–74
01	H	

FILE DESCRIPTION SPECIFICATIONS

LINE (3–5)	FORM TYPE (6)	FILENAME (7–14)	FILE TYPE (15)	FILE DESIGNATION (16)	END OF FILE (17)	SEQUENCE (18)	FILE FORMAT (19)	BLOCK LENGTH (20–23)	RECORD LENGTH (24–27)	PROCESSING MODE (28)	RECORD ADDRESS FIELD LENGTH (29–30)	RECORD ADDRESS TYPE (31)	FILE ORGANIZATION TYPE (32)	OVERFLOW INDICATOR (33–34)	KEY FIELD STARTING LOCATION (35–38)	EXTENSION CODE (39)	DEVICE (40–46)	NOT USED (47–52)	LABELS (53)	NOT USED (54–59)	CORE INDEX (60–65)	FILE ADDITION/UNORDERED (66)	NOT USED (67–68)	TAPE REWIND (69–70)	FILE CONDITION (71–72)	NOT USED (73–74)
02	F	* PROCESSING METHODS FOR CARD FILES																								
03	F		I	P			F										READER									
04	F		I	S			F										MFCU1									
05	F		I	D			F										MFCU2									
06	F		I	T			F									E	READER									
07	F		C	P			F										MFCU1									
	F		C	S			F										MFCU2									
			C	D			F										READER									
02	F		O				F										PUNCH									
03	F																									
04	F																									
05	F																									
06	F																									
07	F																									
	F																									
	F																									

G12033

Figure 4-9. Example of File Description Entries for Card Files



Burroughs REPORT PROGRAMMING GENERATOR

PROGRAM ID
PROGRAMMER
PAGE OF
DATE

PAGE (1 2)

CONTROL CARD SPECIFICATIONS

PROGRAM IDENTIFICATION (75–80)

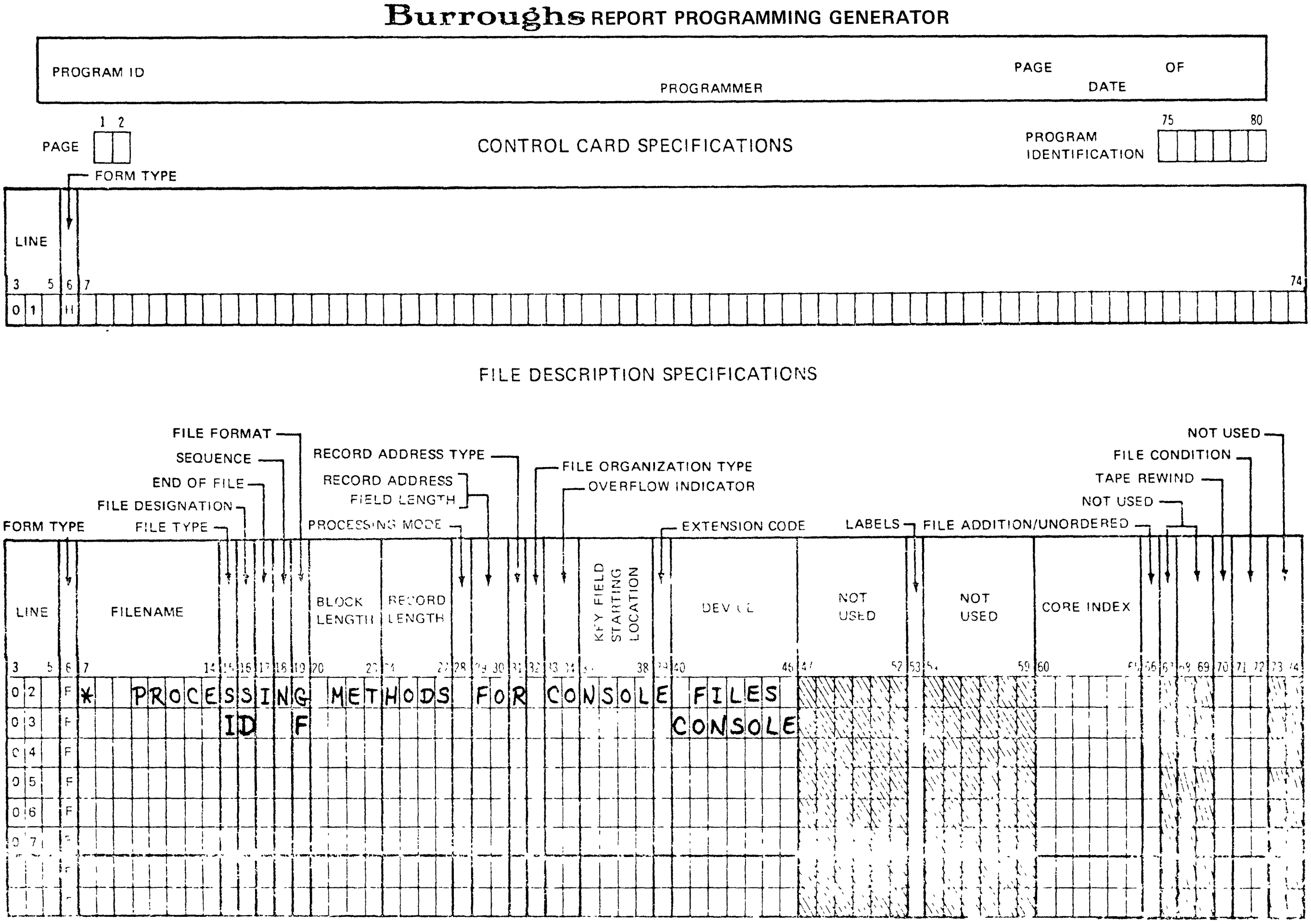
LINE (3–5)	FORM TYPE (6)	7–74
0 1	H	

FILE DESCRIPTION SPECIFICATIONS

LINE (3–5)	FORM TYPE (6)	FILENAME (7–14)	FILE TYPE (15)	FILE DESIGNATION (16)	END OF FILE (17)	SEQUENCE (18)	FILE FORMAT (19)	BLOCK LENGTH (20–23)	RECORD LENGTH (24–27)	PROCESSING MODE (28)	RECORD ADDRESS FIELD LENGTH (29–30)	RECORD ADDRESS TYPE (31)	FILE ORGANIZATION TYPE (32)	OVERFLOW INDICATOR (33–34)	KEY FIELD STARTING LOCATION (35–38)	EXTENSION CODE (39)	DEVICE (40–46)	NOT USED (47–52)	LABELS (53)	NOT USED (54–59)	CORE INDEX (60–65)	FILE ADDITION/UNORDERED (66)	NOT USED (67–69)	TAPE REWIND (70)	FILE CONDITION (71–72)	NOT USED (73–74)
0 2	F	* PROCESSING METHODS FOR TAPE FILES																								
0 3	F		I	P			F										TAPE									
0 4	F		I	S			F										TAPE									
0 5	F		I	T			F									E	TAPE									
0 6	F		I	D			F										TAPE									
0 7	F		O				F										TAPE									
	F																									
	F																									

G12034

Figure 4-10. Example of File Description Entries for Tape Files



Burroughs REPORT PROGRAMMING GENERATOR

PROGRAM ID PROGRAMMER PAGE OF DATE

PAGE (1 2) CONTROL CARD SPECIFICATIONS PROGRAM IDENTIFICATION (75 80)

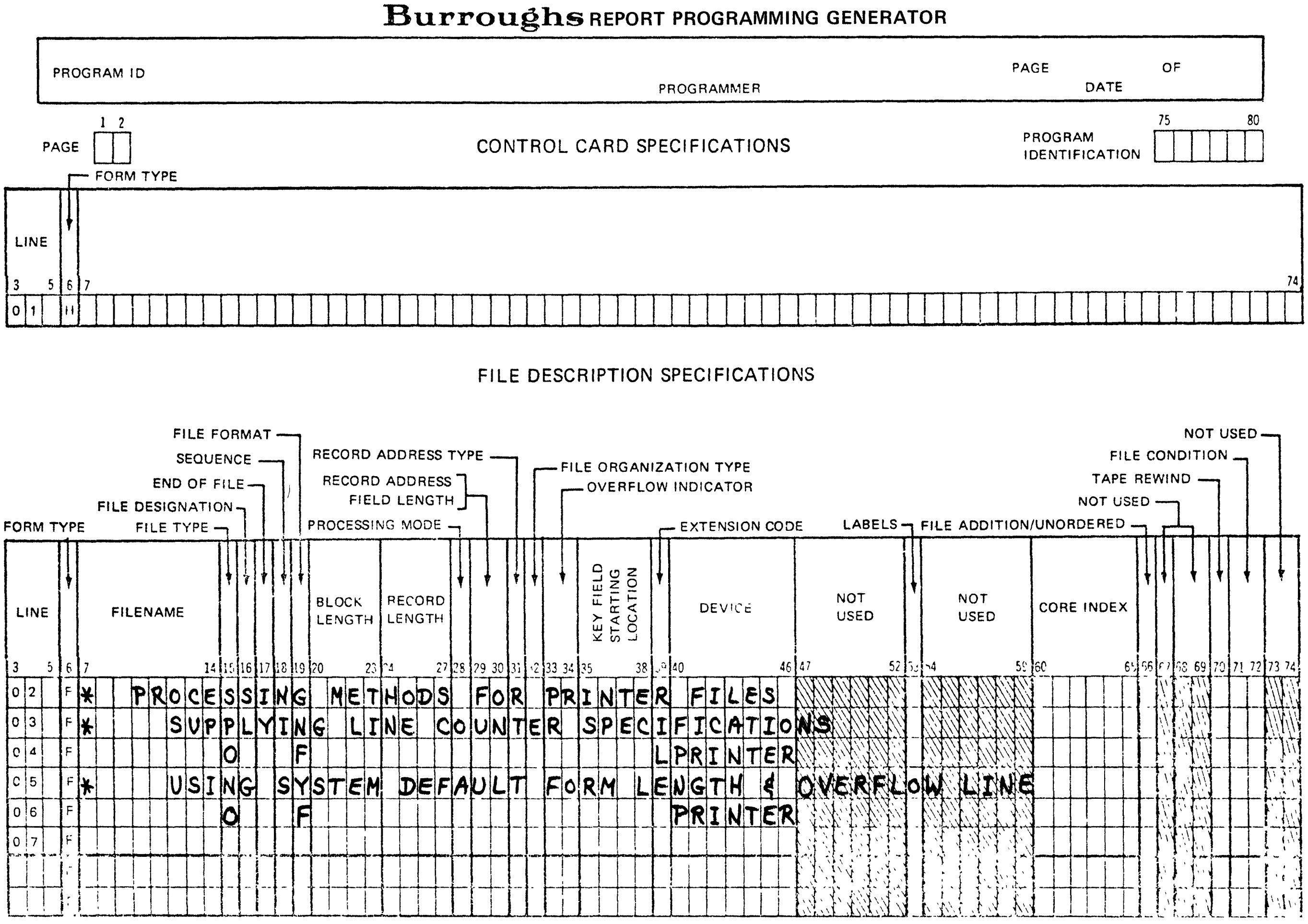
LINE (3 5)	FORM TYPE (6)	7 … 74
0 1	H	

FILE DESCRIPTION SPECIFICATIONS

LINE (3 5)	FORM TYPE (6)	FILENAME (7 14)	FILE TYPE (15)	FILE DESIGNATION (16)	END OF FILE (17)	SEQUENCE (18)	FILE FORMAT (19)	BLOCK LENGTH (20 23)	RECORD LENGTH (24 27)	PROCESSING MODE (28)	RECORD ADDRESS FIELD LENGTH (29 30)	RECORD ADDRESS TYPE (31)	FILE ORGANIZATION TYPE (32)	OVERFLOW INDICATOR (33 34)	KEY FIELD STARTING LOCATION (35 38)	EXTENSION CODE (39)	DEVICE (40 46)	NOT USED (47 52)	LABELS (53)	NOT USED (54 59)	CORE INDEX (60 65)	FILE ADDITION/UNORDERED (66)	NOT USED (67 69)	TAPE REWIND (70)	FILE CONDITION (71 72)	NOT USED (73 74)
0 2	F	* PROCESSING METHODS FOR CONSOLE FILES																								
0 3	F		I	D			F										CONSOLE									
0 4	F																									
0 5	F																									
0 6	F																									
0 7	F																									
	F																									
	F																									

G12035

Figure 4-11. Example of File Description Entries for Console Files



Burroughs REPORT PROGRAMMING GENERATOR

PROGRAM ID PROGRAMMER PAGE OF DATE

CONTROL CARD SPECIFICATIONS

PAGE (1-2) PROGRAM IDENTIFICATION (75-80)

```
LINE  FORM TYPE
3  5  6  7                                                                  74
01    H
```

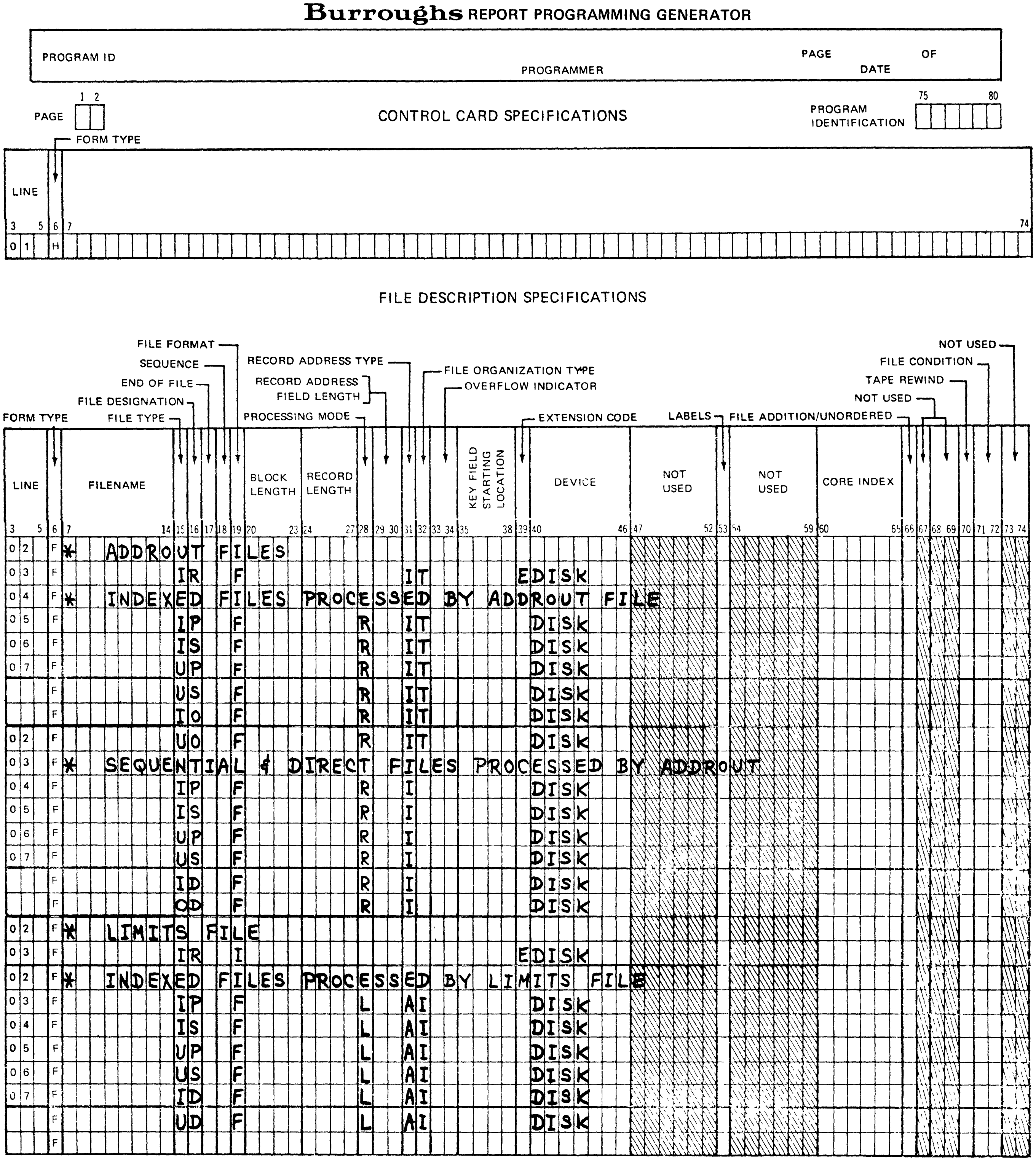
FILE DESCRIPTION SPECIFICATIONS

FORM TYPE
FILE TYPE
FILE DESIGNATION
END OF FILE
SEQUENCE
FILE FORMAT
PROCESSING MODE
RECORD ADDRESS FIELD LENGTH
RECORD ADDRESS TYPE
FILE ORGANIZATION TYPE
OVERFLOW INDICATOR
EXTENSION CODE
LABELS
FILE ADDITION/UNORDERED
NOT USED
TAPE REWIND
FILE CONDITION
NOT USED
LINE
FILENAME
BLOCK LENGTH
RECORD LENGTH
KEY FIELD STARTING LOCATION
DEVICE
NOT USED
NOT USED
CORE INDEX

```
3  5 6 7      14  19      27        38  46       53      59
02  F*  PROCESSING METHODS FOR PRINTER FILES
03  F*    SUPPLYING LINE COUNTER SPECIFICATIONS
04  F        O   F                   LPRINTER
05  F*    USING SYSTEM DEFAULT FORM LENGTH & OVERFLOW LINE
06  F        O   F                    PRINTER
07  F
```

G12036

Figure 4-12. Example of File Description Entries for Printer Files



Burroughs REPORT PROGRAMMING GENERATOR

PROGRAM ID PROGRAMMER PAGE OF DATE

PAGE (1 2) CONTROL CARD SPECIFICATIONS PROGRAM IDENTIFICATION (75 80)

LINE (3 5)	FORM TYPE (6)	(7 74)
01	H	

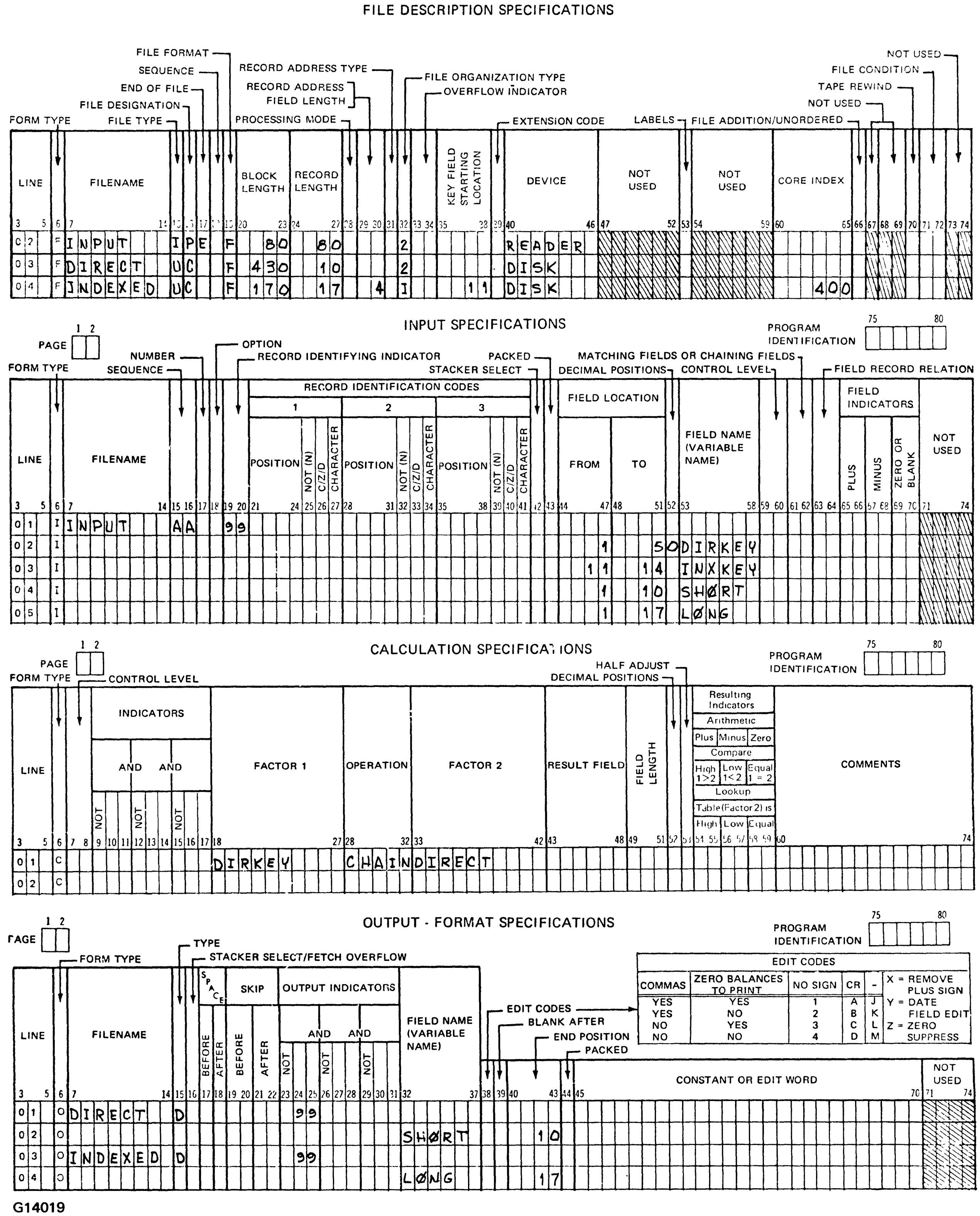
FILE DESCRIPTION SPECIFICATIONS

LINE (3 5)	FORM TYPE (6)	FILENAME (7 14)	FILE TYPE (15)	FILE DESIGNATION (16)	END OF FILE (17)	SEQUENCE (18)	FILE FORMAT (19)	BLOCK LENGTH (20 23)	RECORD LENGTH (24 27)	PROCESSING MODE (28)	RECORD ADDRESS FIELD LENGTH (29 30)	RECORD ADDRESS TYPE (31)	FILE ORGANIZATION TYPE (32)	OVERFLOW INDICATOR (33 34)	KEY FIELD STARTING LOCATION (35 38)	EXTENSION CODE (39)	DEVICE (40 46)	NOT USED (47 52)	LABELS (53)	NOT USED (54 59)	CORE INDEX (60 65)	FILE ADDITION/UNORDERED (66)	NOT USED (67 69)	TAPE REWIND (70)	FILE CONDITION (71 72)	NOT USED (73 74)
02	F	* ADDROUT FILES																								
03	F		I	R			F					I	T			E	DISK									
04	F	* INDEXED FILES PROCESSED BY ADDROUT FILE																								
05	F		I	P			F			R		I	T				DISK									
06	F		I	S			F			R		I	T				DISK									
07	F		U	P			F			R		I	T				DISK									
	F		U	S			F			R		I	T				DISK									
	F		I	O			F			R		I	T				DISK									
02	F		U	O			F			R		I	T				DISK									
03	F	* SEQUENTIAL & DIRECT FILES PROCESSED BY ADDROUT																								
04	F		I	P			F			R		I					DISK									
05	F		I	S			F			R		I					DISK									
06	F		U	P			F			R		I					DISK									
07	F		U	S			F			R		I					DISK									
	F		I	D			F			R		I					DISK									
	F		O	D			F			R		I					DISK									
02	F	* LIMITS FILE																								
03	F		I	R			I									E	DISK									
02	F	* INDEXED FILES PROCESSED BY LIMITS FILE																								
03	F		I	P			F			L		A	I				DISK									
04	F		I	S			F			L		A	I				DISK									
05	F		U	P			F			L		A	I				DISK									
06	F		U	S			F			L		A	I				DISK									
07	F		I	D			F			L		A	I				DISK									
	F		U	D			F			L		A	I				DISK									
	F																									

G12037

Figure 4-13. Example of File Description Entries for Record Address Processing

The following figures illustrate RPG coding examples for indexed and direct files. Figure 4-14 illustrates the updating of an existing file; figure 4-15 illustrates the creation of a new file.



FILE DESCRIPTION SPECIFICATIONS

```
  02 FINPUT   IPE F  80  80    2       READER
  03 FDIRECT  UC  F 430  10    2       DISK
  04 FINDEXED UC  F 170  17  4 I    11 DISK                400
```

INPUT SPECIFICATIONS

```
  01 IINPUT   AA  99
  02 I                                        1  50 DIRKEY
  03 I                                       11  14 INXKEY
  04 I                                        1  10 SHØRT
  05 I                                        1  17 LØNG
```

CALCULATION SPECIFICATIONS

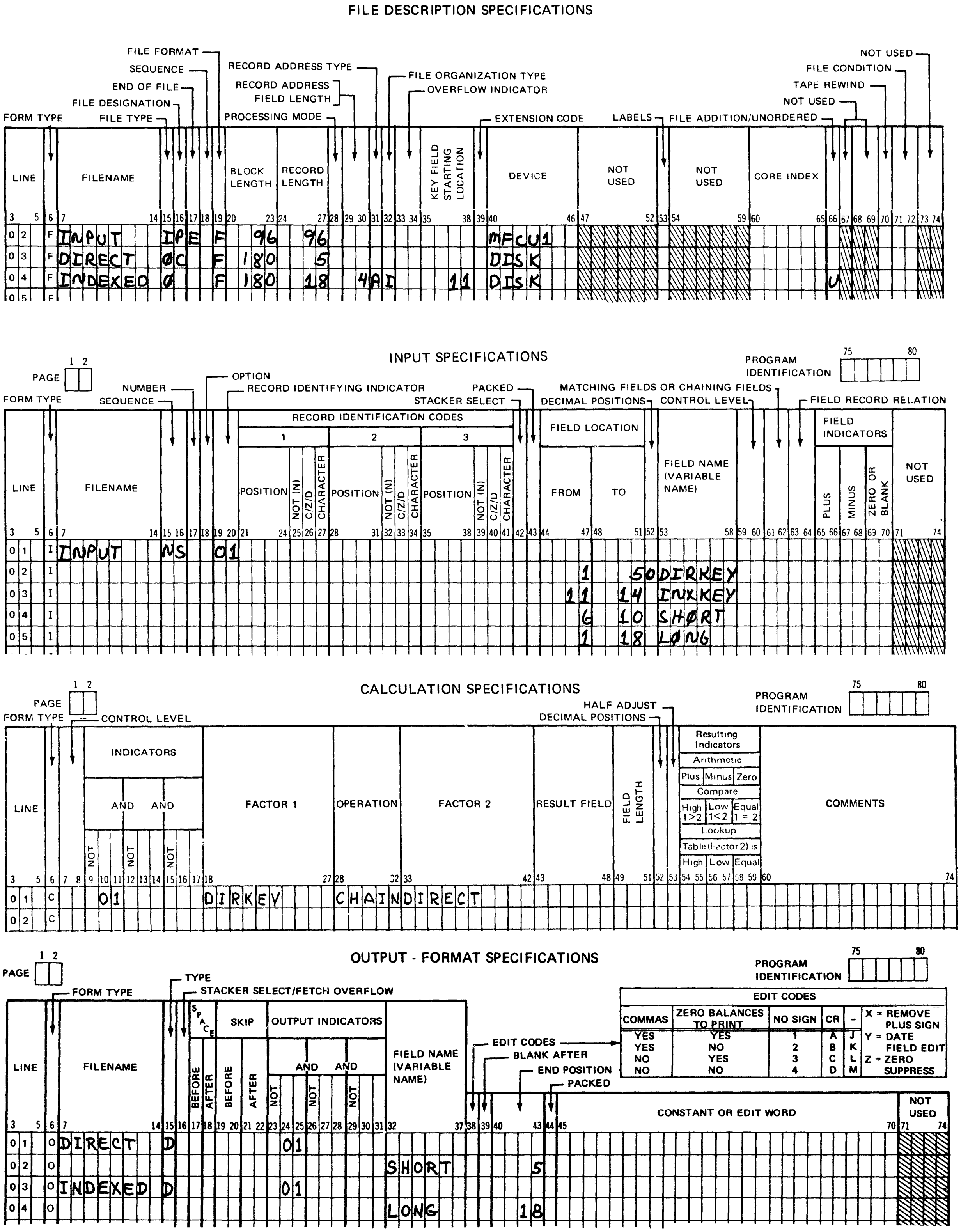
```
  01 C           DIRKEY    CHAINDIRECT
  02 C
```

OUTPUT - FORMAT SPECIFICATIONS

```
  01 ODIRECT  D        99
  02 O                         SHØRT    10
  03 OINDEXED D        99
  04 O                         LØNG     17
```

G14019

Figure 4-14. Indexed and Direct Files - File Update



FILE DESCRIPTION SPECIFICATIONS

```
  02 FINPUT   IPE F  96  96            MFCU1
  03 FDIRECT  ØC  F 180   5            DISK
  04 FINDEXED Ø   F 180  18  4AI    11 DISK                      U
  05 F
```

INPUT SPECIFICATIONS

```
  01 IINPUT   NS  01
  02 I                                        1    50DIRKEY
  03 I                                       11  14 INXKEY
  04 I                                        6  10 SHØRT
  05 I                                        1  18 LØNG
```

CALCULATION SPECIFICATIONS

```
  01 C   01      DIRKEV    CHAINDIRECT
  02 C
```

OUTPUT - FORMAT SPECIFICATIONS

```
  01 ODIRECT  D        01
  02 O                         SHORT      5
  03 OINDEXED D        01
  04 O                         LONG      18
```

G14020

Figure 4-15. Indexed and Direct Files - File Creation

SECTION 5

VECTORS AND EXTENSION SPECIFICATIONS

Extension specifications are used to describe all tables, arrays, and record address files that are specified in an RPG program.

The Extension Specifications Summary Sheet, form number 1057924, is used for coding the field information that is described in the following pages.

FIELD DEFINITIONS

Figure 5-1 can be used in conjunction with the following field definitions for the Extension Specifications.

1-2 PAGE

Refer to Section 2 for a complete description.

3-5 LINE

Refer to Section 2 for a complete description.

6 FORM TYPE

An E must appear in this field.

11-18 FROM FILENAME

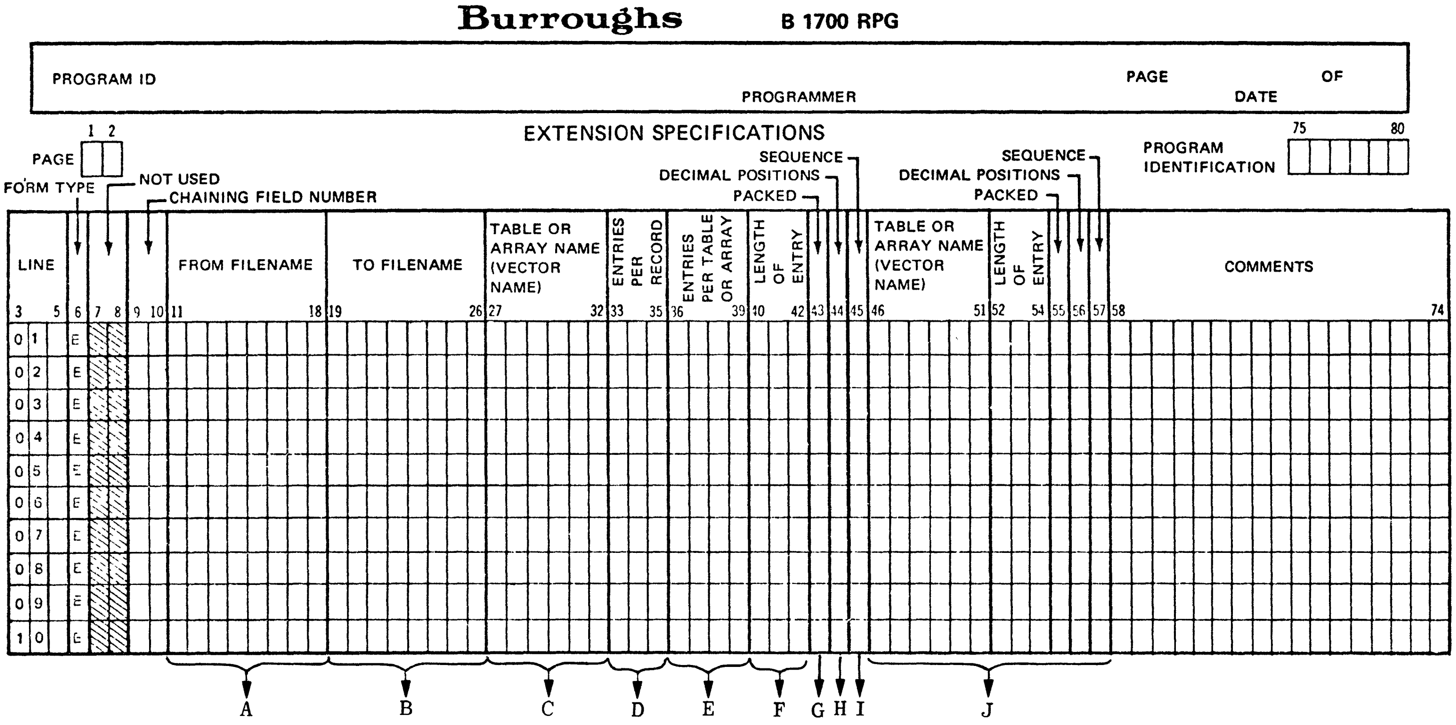
This field is used to name a record address file or a pre-execution-time vector file, and must contain the filename of every record address file and every pre-execution-time vector file to be used by the program. The file must be specified on the File Description Specifications as an input record address file or as an input table file. The same record address file must not be named in more than one Extension Specification entry.

If the vector is to be loaded at compile time or by Input or Calculation Specifications during program execution, this field must be left blank.

Filenames must always be entered in this field left-justified. When a vector is loaded at compile time, it becomes a permanent part of the program so that a vector file is not needed when the program is executed. Only those vectors that do not change often should be compiled with the program. When vectors are being compiled with the program, the vector data must follow the source program deck.

19-26 TO FILENAME

The TO FILENAME field defines the relationship between a file named in this field and a file named in the FROM FILENAME field, columns 11-18.



A. 11-18 Contains the file identification of a record address file or the name of a pre-execution time vector file.

B. 19-26 Contains the file identification of the file to be processed by a record address file or the name of the file to which vector will be outputted. If blank, vector not to be output at EOJ.

C. 27-32 Contains the name of the input table or array. Entries: TABXXX (X= any alphanumeric character) or 1-6 alphanumeric characters. Also used to name the first of two alternating vectors.

D. 33-35 Specifies the exact number of entries contained in each vector input record.

E. 36-39 Specifies the maximum number of items contained in the first vector named. Entry maximum: 4095, right-justified.

F. 40-42 Specifies the length (in bytes) of each element in the vector. Maximum numeric entry is 31, maximum alphanumeric entry is 511, right-justified.

G. 43 Specifies if the external vector elements are in binary, packed, or unpacked decimal format. Entries: Blank, B, or P.

H. 44 Specifies the number of decimal positions contained in each element. Entries: Blank or 0-9.

I. 45 Specifies the sequence in which elements will be loaded for the first vector. Entries: Blank, A or D.

J. 46-57 These columns are used to describe the second of two alternating vectors. Entries are of the same type as specified for the first vector. The second vector is loaded in alternating format with the first.

G14022

Figure 5-1. Extension Specifications Summary Sheet

Filenames must be left-justified beginning in column 19. Valid entries for this field are:

Entry	Description
Name of an input or update file	The file processed via the record address file named in the FROM FILENAME field.
Name of an output file	The output file on which a table or array is to be written at ECJ.

If a record address file is named in the FROM FILENAME field, columns 11-18, then the name of the primary or secondary file that contains the data records to be processed must be entered in the TO FILENAME field, columns 19-26. This primary or secondary file can be an input or update file and must not be specified in any other Extension Specifications entry.

If a file is specified on the File Description Specifications as being processed by an addrout file, then its filename must be entered in the TO FILENAME field, columns 19-26, and an appropriate filename must also be entered in the FROM FILENAME field, columns 11-18.

If a file is specified on the File Description Specifications as being processed as a limits file, then its filename must be entered in the TO FILENAME field, columns 19-26, and an appropriate filename must be entered in the FROM FILENAME field, columns 11-18.

If a table or array is to be written or punched, then enter the designated output file in the TO FILENAME field, columns 19-26. This output file must have been previously defined in the File Description Specifications.

Execution time arrays cannot be written at end of job. In order tc produce an array or table, the FROM FILENAME, TO FILENAME, and ENTRIES PER RECORD fields must have entries specified.

If a table or array is to be written or punched, it is automatically written or punched at the end of job after all other records have been written or punched.

An array or table can be written to only one output device. More than one array or table may be written to the same output file, but it is the user's responsibility to ensure that this is a meaningful thing to do.

27-32 VECTOR NAME

This field is used to name tables and arrays to be used by the program. Each vector name must be unique, and must follow the rules for the formation of vector names as described in Section 2. Table names must begin with the letters TAB (note that this includes the name TAB itself); any name appearing in this field which does not begin with TAB is considered an array name.

Vector files are processed in the same order in which they appear on the Extension Specifications form. Thus, if more than one vector file is specified for the program, the files must be loaded in the same order in which they appear on the form.

If two related vectors are in alternating form within one vector file, the first vector must be named in columns 27-32, and the second vector must be named in columns 46-51. Any combination of vector types (table or array) is allowed in alternating format. (For more information, see columns 46-57 in this section.)

33-35 ENTRIES PER RECORD

This field is used to specify the number of entries in each vector input record. Every record except the last one must contain the number of entries specified in this field. The last record may contain fewer entries than specified in this field. The possible entries that can be made in this field follow:

Entry	Definition
1-999	The number of vector entries contained in each vector input record.
Blank	This vector is a dynamic/ execution-time vector.

Entries in this field must be right-justified, and leading zeroes are not required. Corresponding items from related (alternating) vectors must be on the same record and in alternating format. Each pair of items is considered one entry. The number of entries per record must not exceed the number of entries per vector specified in columns 36-39.

The FROM FILENAME and ENTRIES PER RECORD entries are used to determine the type of vector and therefore how it will be loaded. Table 5-1 is a guide for determining vector type.

Table 5-1. Guide for Determining Vector Type

Vector Type	FROM FILENAME	ENTRIES PER RECORD
Compile-Time	Blank	Filled
Pre-Execution Time	Filled	Filled
Dynamic/Execution Time	Blank	Blank

36-39 ENTRIES PER VECTOR

This field is used to specify the maximum number of elements that can be contained in the vector named in the first VECTOR NAME field (columns 27-32). A maximum of 4095 elements per vector is allowed. For vectors to be loaded in alternating format, this number also applies to the one named in the second VECTOR NAME field (columns 46-51). Entries in this field must be right-justified; leading zeros may be omitted.

40-42 LENGTH OF ENTRY

This field is used to specify the length (in bytes) of each element in the vector named in the first VECTOR NAME field (columns 27-32). For numeric vectors in packed decimal format, enter the number of digits.

Entries in this field must be right-justified; leading zeros may be omitted.

Numeric items in the vector input records must have leading zeros added if their length is less than that specified; alphanumeric entries must have either leading or trailing blanks.

The maximum length of a numeric vector element is dependent on the format of the data, and is as follows:

Format	Maximum Length
Unpacked numeric	31 characters
Packed numeric	31 digits
Binary	4 or 9 digits

The maximum for an alphanumeric vector element is 511 characters. However, an element must be completely contained on one record; therefore, input record sizes will also limit the maximum element sizes.

43 PACKED

This field specifies the external format of the vector data. Acceptable entries are:

Entry	Definition
Blank	Vector elements are in either unpacked decimal or alphanumeric format.
P	Vector elements must be in packed decimal format.
B	Vector elements must be in binary format.

Any vector (including compile-time vectors) may be packed and may be either right or left signed (as specified in the Control Card or by the dollar option RSIGN).

Pre-execution-time vectors can be in packed decimal format unless the filename specified in columns 19-26 is that of a printer file.

For vectors that are loaded or modified as a result of Input Specification entries, the sign position and format (packed or unpacked) described on the Input Specifications dictate the external data format.

Binary compile-time vectors are not allowed.

44 DECIMAL POSITIONS

This field:

a. Is used to specify the number of decimal positions contained in each element of the vector named in the first VECTOR NAME field (columns 27-32). If the elements have no decimal positions, a zero must be entered in column 44. This field must not be blank for a numeric vector or if column 43 contains a P.

b. Defines whether a data item is in numeric or alphanumeric format. Column 40 (LENGTH OF ENTRY) defines the length of the item. For alphanumeric data items, the entry specifies the number of bytes. For numeric data items, including those in packed or binary format, the entry specifies the number of digits. For binary data items, either a 4 or a 9 must be specified.

Note that it is possible to define the same numeric data items differently on the Extension Specifications and on the Input Specifications.

Example:

A pre-execution-time array that is specified on the Extension Specifications can indicate that the data items are in packed numeric format, and the Input Specifications can specify that the data item is in unpacked numeric format.

The acceptable entries for this field are:

Entry	Definition
0-9	Number of positions to the right of the implied decimal point for numeric vector elements.
Blank	Alphanumeric vector.

45 SEQUENCE

This field is used to specify the sequence in which elements will be loaded for the vector named in the first VECTOR NAME field (columns 27-32). A vector loaded at compile or pre-execution time is checked for the specified sequence. A sequence error at compile time generates warnings in the source listing. A sequence error at pre-execution time causes the program to be discontinued. The sequence check does allow two consecutive elements to be equal. This column must contain an entry if high or low LOKUP is to be used. Alternate vectors need not have the same sequence.

Valid entries for this field are:

Entry	Definition
Blank	Unordered elements.
A	Elements arranged in ascending order.
D	Elements arranged in descending order.

46-57 VECTOR NAME, LENGTH OF ENTRY, PACKED, DECIMAL POSITIONS, SEQUENCE

These fields are used only when describing a second vector which is loaded in alternating format with the vector named in the first VECTOR NAME field (columns 27-32). All of these fields require the same type of entries as the corresponding fields in columns 27-45, but entries in columns 46-57 apply only to the second vector. For a single vector description, these fields must be left blank. Compile-time vectors, pre-execution-time vectors, and dynamic/execution-time arrays can be specified as alternating vectors.

58-74 COMMENTS

This field is available for inclusion of comments and documentary remarks, and may contain any valid EBCDIC characters.

75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a complete description.

VECTORS

GENERAL

Tables and arrays are logical configurations of data elements that have similar characteristics. Within the scope of this manual, very little distinction is made between tables and arrays, and therefore they are usually referred to as vectors. Where differences exist in their characteristics, they are referred to individually as tables and arrays.

Each element of a vector must be of the same length and data type (numeric or alphanumeric). All numeric elements must have the same number of decimal positions.

TABLE AND ARRAY DIFFERENCES

Every vector to be used by the program must be given a name. All table names must begin with the letters TAB. Array names can begin with any alphabetic character.

Indices can be used to access specific elements within tables and arrays.

If a table name appears without an index, it refers to the last item referenced in the table.

If an array name appears without an index, it refers to the entire array. Such a reference specifies that the designated operation be performed repetitively for each element of the array.

TYPES OF VECTORS

The three types of vectors and the time at which they are loaded with data follows:

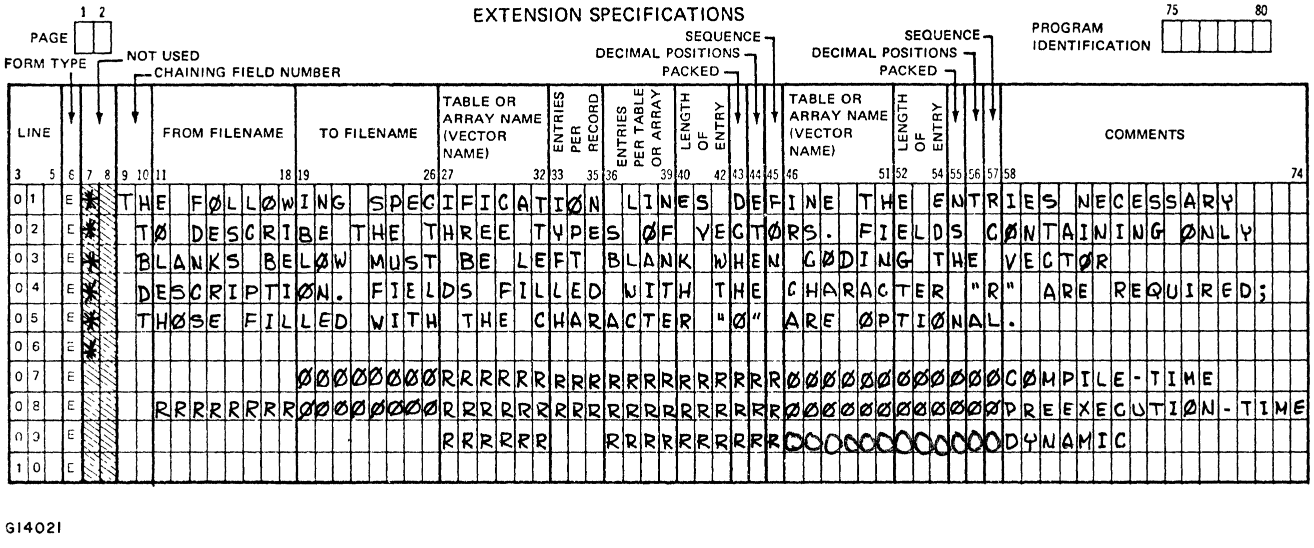
Vector Type	Time Loaded
Compile time	During RPG program generation.
Pre-execution time	At the beginning of RPG program execution.
Dynamic/execution time	During RPG program execution.

Data elements for all types of vectors can be altered at any time during program execution.

REQUIRED ENTRIES FOR EXTENSION SPECIFICATIONS

All vectors for a program must be described on the Extension Specifications form. Several of the fields of the Extension Specifications require entries for each type of vector, regardless of the time at which the vector is loaded.

Figure 5-2 illustrates which fields require entries for each of the three types of vectors


EXTENSION SPECIFICATIONS
PAGE 1 2
FORM TYPE
NOT USED
CHAINING FIELD NUMBER
SEQUENCE
DECIMAL POSITIONS
PACKED
PROGRAM IDENTIFICATION 75 80
LINE
FROM FILENAME
TO FILENAME
TABLE OR ARRAY NAME (VECTOR NAME)
ENTRIES PER RECORD
ENTRIES PER TABLE OR ARRAY
LENGTH OF ENTRY
COMMENTS
3 5 6 7 8 9 10 11 18 19 26 27 32 33 35 36 39 40 42 43 44 45 46 51 52 54 55 56 57 58 74
G14021

```
01 E*THE FØLLØWING SPECIFICATIØN LINES DEFINE THE ENTRIES NECESSARY
02 E*  TØ DESCRIBE THE THREE TYPES ØF VECTØRS. FIELDS CØNTAINING ØNLY
03 E*  BLANKS BELØW MUST BE LEFT BLANK WHEN CØDING THE VECTØR
04 E*  DESCRIPTIØN. FIELDS FILLED WITH THE CHARACTER "R" ARE REQUIRED;
05 E*  THØSE FILLED WITH THE CHARACTER "Ø" ARE ØPTIØNAL.
06 E*
07 E                ØØØØØØØØRRRRRRRRRRRRRRRRRRRØØØØØØØØØØØØCØMPILE-TIME
08 E        RRRRRRRRØØØØØØØØRRRRRRRRRRRRRRRRRRRØØØØØØØØØØØØPREEXECUTIØN-TIME
09 E                        RRRRRR   RRRRRRRRRROOOOOOOOOOOODYNAMIC
10 E
```

Figure 5-2. Entries Necessary to Describe a Vector

Columns 27-45 are used as needed to specify the name assigned to identify the vector (VECTOR NAME), the number of vector elements occurring in each input record (ENTRIES PER RECORD), the size of the vector (ENTRIES PER TABLE OR ARRAY and LENGTH OF ENTRY), whether the input data is in binary, packed or unpacked decimal format (PACKED), the number of decimal positions in each entry (DECIMAL POSITIONS), and the order in which the elements are sequenced (SEQUENCE).

Additional entries are required if:

a. The vector is to be loaded at pre-execution time from the file named in the FROM FILENAME field.

b. The vector is to be written to the output file named in the TO FILENAME field at End-of-Job.

Any file named for vector loading or vector output must also be described on the File Description Specifications.

For dynamic/execution-time vectors, the FROM FILENAME, TO FILENAME, and the ~~ENTRIES PER RECORD fields m~~ust be left blank because dynamic vectors cannot be written at End-of-Job.

COMPILE TIME VECTORS

For a compiler-time vector load, the data to be loaded is read into the program storage area reserved by the entries in the Extension Specifications at the same time that the source program is compiled. Physically, the data is placed in a card file called "RPG/VECTOR", which immediately follows the source deck (see figure 5-3).

For a compile-time vector load, short vectors (those which are only partially full) are not allowed. Each vector file must occur in the order in which it was specified, and must contain exactly the number of records necessary to fill it with data. No separators are used to delimit the end of one vector file and the beginning of the next; the compiler reads input records until one vector is full, and then proceeds to fill the next one in order.

Compile-time vectors can be changed by either Input or Calculation Specifications.

COMPILE-TIME VECTOR LOAD

To load a compile-time vector, all that is necessary is to include the data cards for the vector in the proper order in the file "RPG/VECTOR" (figure 5-3). The compiler will automatically read this file after the source program has been read, and will load the data into the vectors declared.

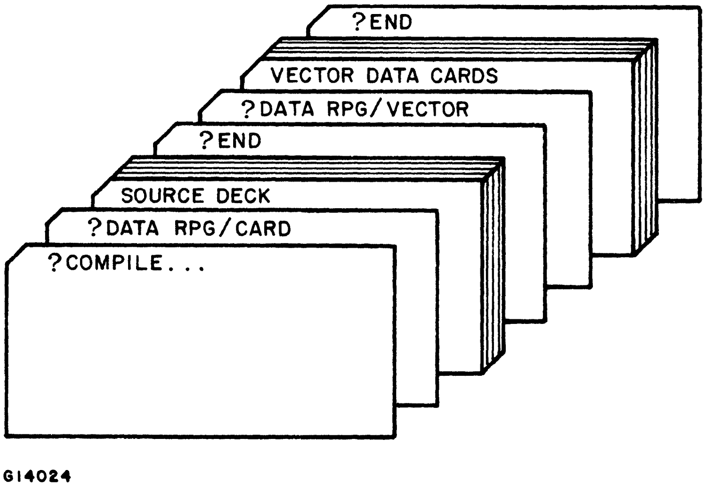
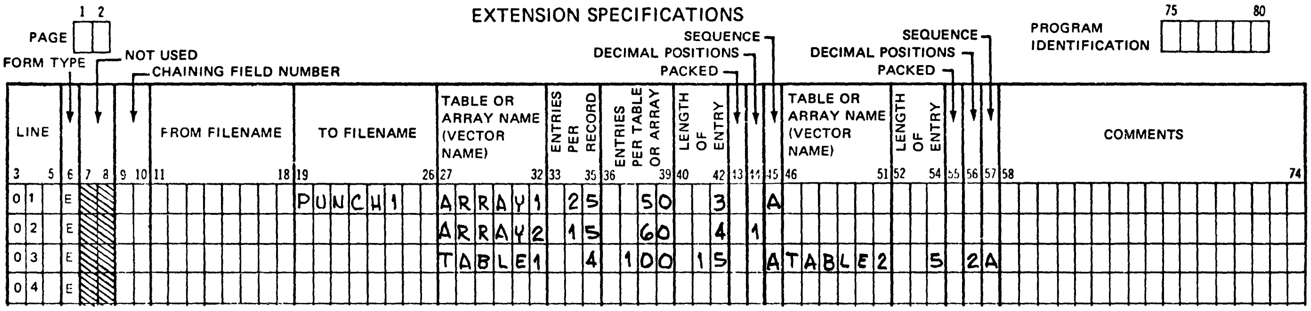


Figure 5-3. Setting Up a Data Deck For a Compile-Time Vector

If more than one compile-time vector has been declared, care must be exercised in setting up the data deck. The compiler expects data for the vectors to be entered in the same order in which they were declared in the Extension Specifications form. The compiler reads data cards and stores the elements in the first vector until it is full, then proceeds to do the same for the second vector, and so forth. If more or fewer records are present than are necessary to exactly fill a vector, data will be placed into the wrong vector. See figure 5-4 for an example of a compile-time vector load declaration.



EXTENSION SPECIFICATIONS

PAGE □□ (1 2) PROGRAM IDENTIFICATION □□□□□□ (75 80)

LINE (3-5)	FORM TYPE (6)	NOT USED (7-8)	CHAINING FIELD NUMBER (9-10)	FROM FILENAME (11-18)	TO FILENAME (19-26)	TABLE OR ARRAY NAME (VECTOR NAME) (27-32)	ENTRIES PER RECORD (33-35)	ENTRIES PER TABLE OR ARRAY (36-39)	LENGTH OF ENTRY (40-42)	PACKED (43)	DECIMAL POSITIONS (44)	SEQUENCE (45)	TABLE OR ARRAY NAME (VECTOR NAME) (46-51)	LENGTH OF ENTRY (52-54)	PACKED (55)	DECIMAL POSITIONS (56)	SEQUENCE (57)	COMMENTS (58-74)
01	E	▨			PUNCH1	ARRAY1	25	50	3			A						
02	E	▨				ARRAY2	15	60	4		1							
03	E	▨				TABLE1	4	100	15			A	TABLE2	5		2	A	
04	E	▨																

G14025

Figure 5-4. Compile-Time Vector Load

PRE-EXECUTION TIME VECTORS

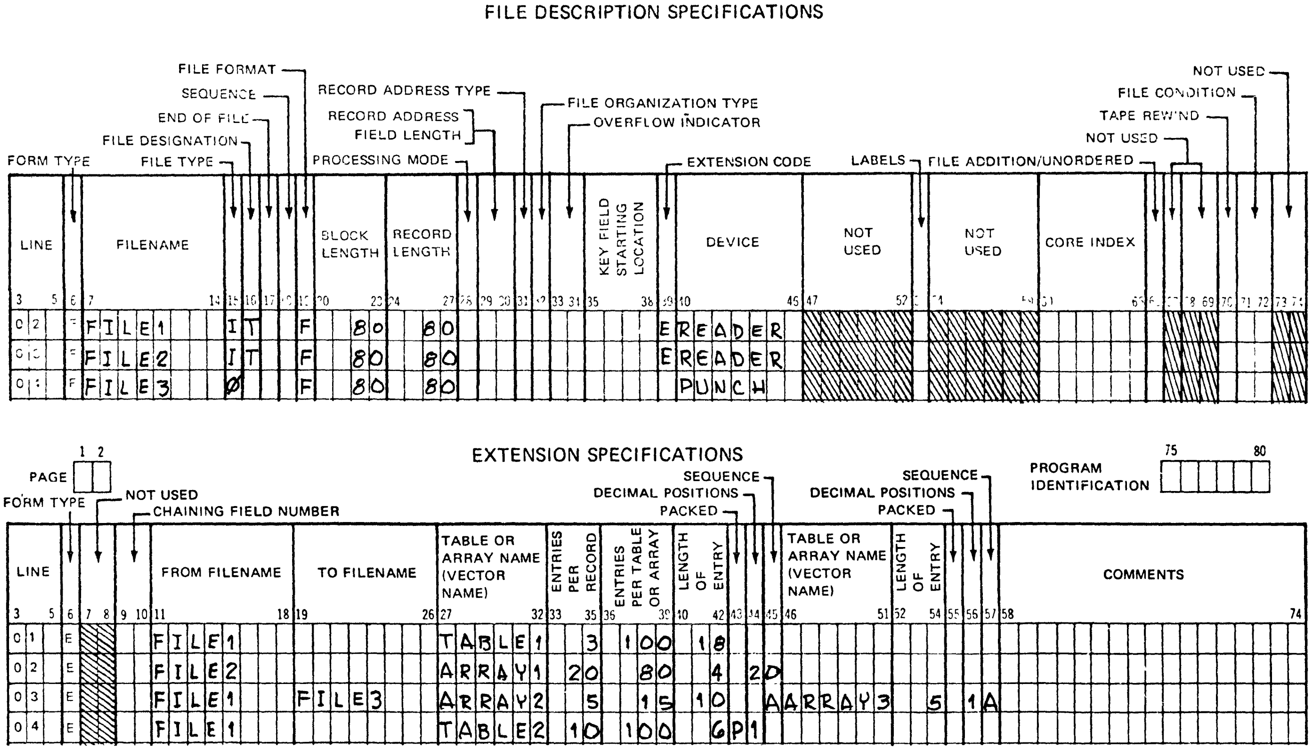
For a pre-execution time vector load, the data to be loaded is read into the program storage area reserved by the entries in the Extension Specifications at the beginning of object program execution, before the normal operations in the program cycle begin. The data for each vector is placed in a file, identified by the names assigned in the File Description Specifications.

More than one vector may be loaded from the same file. Short vectors are not allowed with pre-execution time vector loads.

Pre-execution time vectors can be changed by either Input or Calculation Specifications.

PRE-EXECUTION TIME VECTOR LOAD

To load a pre-execution time vector, the data to be loaded must be placed in the file described in the File Description Specifications to which the vector is assigned (FROM FILENAME). Data is read in at the beginning of program execution and placed in the vector until the vector is full. If more than one vector (not in alternating format) is assigned to a single file, some special considerations must be taken into account (see figure 5-5). Vectors are still loaded in the same order that they are specified in the Extension Specifications, so that files will be opened, read, and closed as necessary to load the designated vectors. If two vectors are assigned to the same table file, and no other pre-execution time vector declaration comes between them, the data for both must be in the same card file. No separators are allowed between the data decks, so that restrictions are imposed the same as those for compile-time vector loads.



FILE DESCRIPTION SPECIFICATIONS

LINE (3-5)	FORM TYPE (6)	FILENAME (7-14)	FILE TYPE (15)	FILE DESIGNATION (16)	END OF FILE (17)	SEQUENCE (18)	FILE FORMAT (19)	BLOCK LENGTH (20-23)	RECORD LENGTH (24-27)	PROCESSING MODE (28)	RECORD ADDRESS FIELD LENGTH (29-30)	RECORD ADDRESS TYPE (31)	FILE ORGANIZATION TYPE (32)	OVERFLOW INDICATOR (33-34)	KEY FIELD STARTING LOCATION (35-38)	EXTENSION CODE (39)	DEVICE (40-46)	NOT USED (47-52)	LABELS (53)	NOT USED (54-59)	CORE INDEX (61-65)	FILE ADDITION/UNORDERED (66)	NOT USED (67-69)	TAPE REWIND (70)	FILE CONDITION (71-72)	NOT USED (73-74)
02	F	FILE1	I	T			F	80	80							E	READER									
03	F	FILE2	I	T			F	80	80							E	READER									
04	F	FILE3	O				F	80	80								PUNCH									

EXTENSION SPECIFICATIONS

PAGE □□ (1-2) PROGRAM IDENTIFICATION □□□□□□ (75-80)

LINE (3-5)	FORM TYPE (6)	NOT USED (7-8)	CHAINING FIELD NUMBER (9-10)	FROM FILENAME (11-18)	TO FILENAME (19-26)	TABLE OR ARRAY NAME (VECTOR NAME) (27-32)	ENTRIES PER RECORD (33-35)	ENTRIES PER TABLE OR ARRAY (36-39)	LENGTH OF ENTRY (40-42)	PACKED (43)	DECIMAL POSITIONS (44)	SEQUENCE (45)	TABLE OR ARRAY NAME (VECTOR NAME) (46-51)	LENGTH OF ENTRY (52-54)	PACKED (55)	DECIMAL POSITIONS (56)	SEQUENCE (57)	COMMENTS (58-74)
01	E			FILE1		TABLE1	3	100	18									
02	E			FILE2		ARRAY1	20	80	4		2	D						
03	E			FILE1	FILE3	ARRAY2	5	15	10			A	ARRAY3	5		1	A	
04	E			FILE1		TABLE2	10	100	6	P	1							

G14026

Figure 5-5. Pre-Execution Time Vector Load

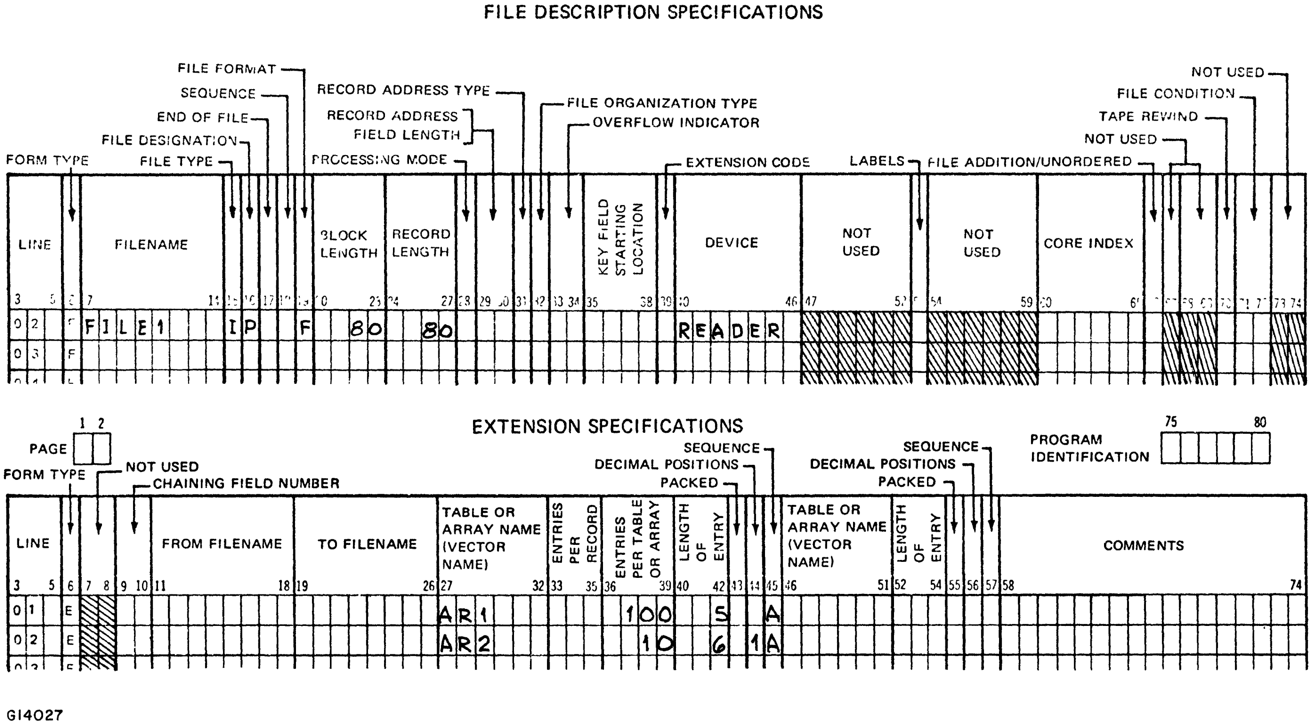
DYNAMIC/EXECUTION TIME VECTORS

Dynamic vectors are loaded during program execution through entries in the Input Specifications or Calculation Specifications. Certain fields of input records or the results of calculation operations may be used to load the elements of a dynamic vector. Such loading, unlike the automatic loading of compile-time and pre-execution ~~time vect~~ors, is completely under programmatic control.

All vectors can be altered during program execution, regardless of when they were loaded initially. Because of this, all vectors may be considered to have "dynamic" characteristics.

DYNAMIC/EXECUTION TIME VECTOR LOAD

To load a dynamic vector, the data elements may be obtained from fields within input records or from the result of operations in the Calculation Specifications. Figure 5-6 provides coding examples on the File Description Specifications and the Extension Specifications for dynamic vector loading.



FILE DESCRIPTION SPECIFICATIONS

LINE	FORM TYPE	FILENAME	FILE TYPE	FILE DESIGNATION	FILE FORMAT	BLOCK LENGTH	RECORD LENGTH	DEVICE
02	F	FILE1	I	P	F	80	80	READER
03	F							

EXTENSION SPECIFICATIONS

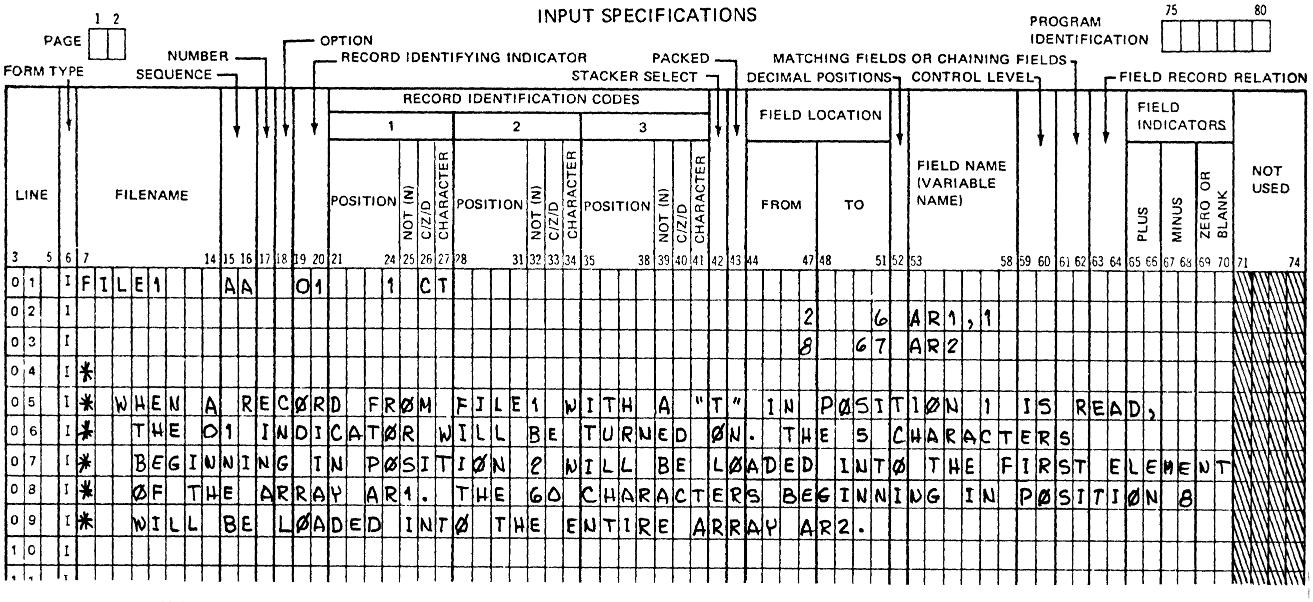
LINE	FORM TYPE	FROM FILENAME	TO FILENAME	TABLE OR ARRAY NAME (VECTOR NAME)	ENTRIES PER RECORD	ENTRIES PER TABLE OR ARRAY	LENGTH OF ENTRY	PACKED	DECIMAL POSITIONS	SEQUENCE
01	E			AR1		100	5			A
02	E			AR2		10	6		1	A

GI4027

Figure 5-6. Dynamic Vector Loading

Input Specifications Load

Fields within input records may contain data for vector loading. This is done by assigning a vector name with an index or an array name without an index as a field name within an input record description (see figure 5-7). If a FIELD NAME (VARIABLE NAME) designates a single element of the vector, the input field will be placed into the vector element when the record is selected. If FIELD NAME designates an entire array (no index assigned), the input field length must be an integral multiple of the element size (LENGTH OF ENTRY) and equal to or less than the total size of the array. If the input field is less than the size of the array, the elements not referenced will not be affected.

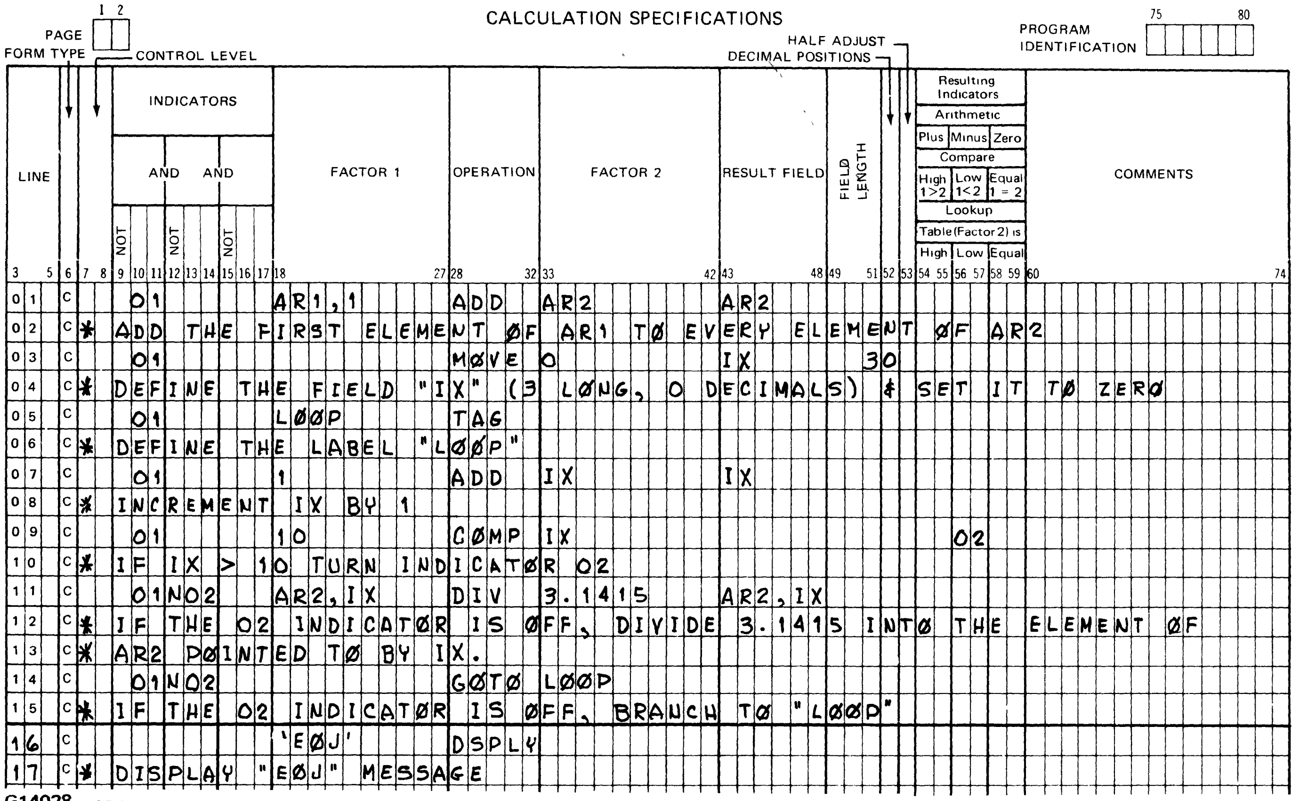


```
01 IFILE1   AA  01   1 CT
02 I                                        2   6 AR1,1
03 I                                        8  67 AR2
04 I*
05 I* WHEN A RECØRD FRØM FILE1 WITH A "T" IN PØSITIØN 1 IS READ,
06 I* THE 01 INDICATØR WILL BE TURNED ØN. THE 5 CHARACTERS
07 I* BEGINNING IN PØSITIØN 2 WILL BE LØADED INTØ THE FIRST ELEMENT
08 I* ØF THE ARRAY AR1. THE 60 CHARACTERS BEGINNING IN PØSITIØN 8
09 I* WILL BE LØADED INTØ THE ENTIRE ARRAY AR2.
10 I
11 I
```

Figure 5-7. Dynamic Vector Load - Input Specifications

Calculation Specifications Load

Any operation which specifies a vector (with or without an index) as the RESULT FIELD will cause the designated vector element (or entire array, if no index is specified) to be loaded with the result of the operation. See figure 5-8 for an example of a load via the Calculation Specifications. Refer to Section 9 for the operations which may be specified for vectors.



```
01 C   01      AR1,1     ADD  AR2       AR2
02 C* ADD THE FIRST ELEMENT ØF AR1 TØ EVERY ELEMENT ØF AR2
03 C   01                MØVE 0         IX      30
04 C* DEFINE THE FIELD "IX" (3 LØNG, 0 DECIMALS) & SET IT TØ ZERØ
05 C   01      LØØP      TAG
06 C* DEFINE THE LABEL "LØØP"
07 C   01      1         ADD  IX        IX
08 C* INCREMENT IX BY 1
09 C   01      10        CØMP IX                     02
10 C* IF IX > 10 TURN INDICATØR 02
11 C   01N02   AR2,IX    DIV  3.1415    AR2,IX
12 C* IF THE 02 INDICATØR IS ØFF, DIVIDE 3.1415 INTØ THE ELEMENT ØF
13 C* AR2 PØINTED TØ BY IX.
14 C   01N02             GØTØ LØØP
15 C* IF THE 02 INDICATØR IS ØFF, BRANCH TØ "LØØP"
16 C           'EØJ'     DSPLY
17 C* DISPLAY "EØJ" MESSAGE
```

G14028 Figure 5-8. Dynamic Vector Load - Calculation Specifications

RULES FOR LOADING A VECTOR

The following rules must be observed in regard to loading vectors:

a. For a vector to be loaded at pre-execution time, entries are necessary in the FROM FILENAME and ENTRIES PER RECORD fields.

b. For a vector to be loaded at compile time, the FROM FILENAME field must be blank and an entry must be made in the ENTRIES PER RECORD field.

c. Vector loading is not implied if the FROM FILENAME, TO FILENAME, and ENTRIES PER RECORD fields are blank. Vectors whose specifications contain blanks in those fields can be loaded as a result of Input or Calculation Specifications, which are execution-time loads.

VECTOR OUTPUT

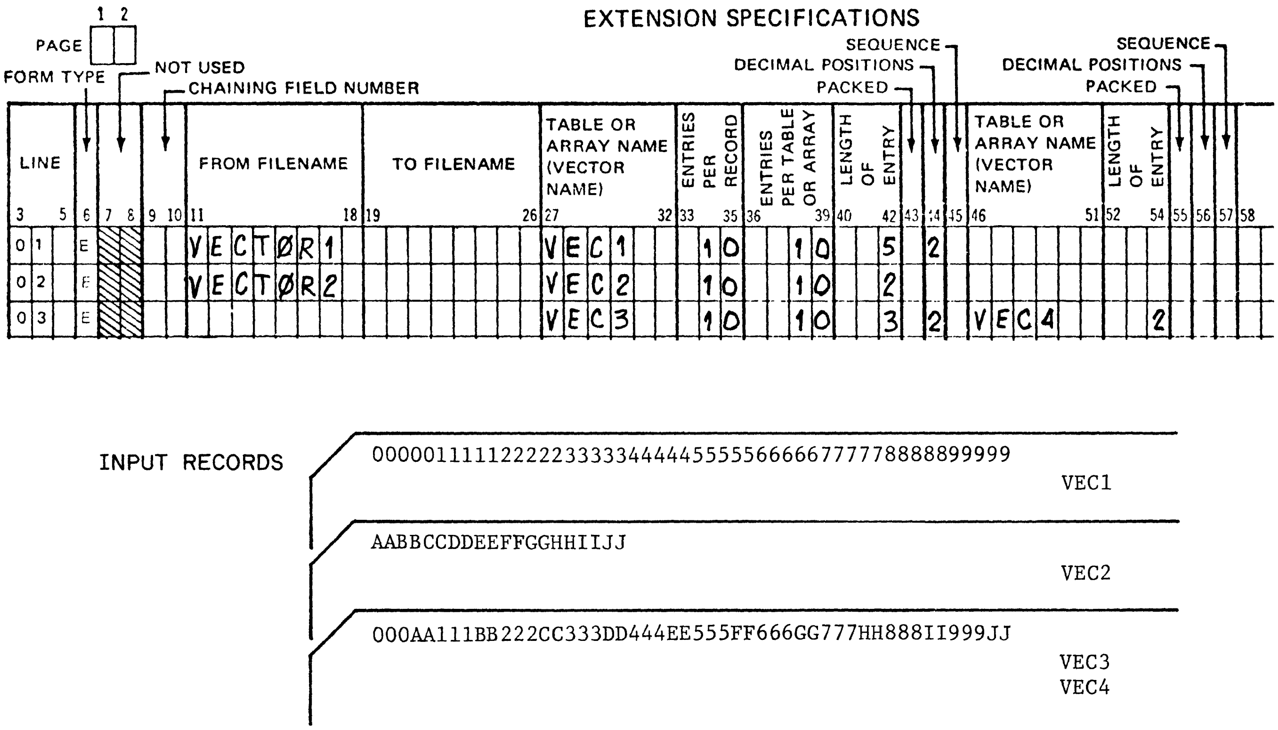
Compile-time and pre-execution time vectors may be written to an output device at End-of-Job by specifying an output file in the TO FILENAME field of the Extension Specifications. This vector output is performed automatically after all processing has been completed. Vector records will be in the format specified by the Extension Specifications.

Dynamic vectors cannot specify a TO FILENAME, and thus cannot be automatically written out at End-of-Job.

Also, an entire array may be written during output time by specifying the array name without an index as field names (VARIABLE NAME) in the Output-Format Specifications. If an entire array is to be output in this way, the end position specified must allow sufficient space for all elements of the array, allowing for any editing. For editing of a whole array see Section 10.

VECTORS IN ALTERNATING FORMAT

Vectors specified as occurring in alternating format have related elements contained in alternating format on each input record (see figure 5-9). The two vectors need not be of the same size (LENGTH OF ENTRY), type (numeric or alphanumeric), or sequence order, but they must have the same number of elements contained on each input record, and each vector must contain the same number of elements (NUMBER OF ENTRIES PER TABLE OR ARRAY). Each pair of elements on the input record is considered one entry. The first element of each pair belongs to the vector described in columns 27-45; the second element belongs to the vector described in columns 46-57.



EXTENSION SPECIFICATIONS

PAGE 1 2

LINE 3-5	FORM TYPE 6	NOT USED 7-8	CHAINING FIELD NUMBER 9-10	FROM FILENAME 11-18	TO FILENAME 19-26	TABLE OR ARRAY NAME (VECTOR NAME) 27-32	ENTRIES PER RECORD 33-35	ENTRIES PER TABLE OR ARRAY 36-39	LENGTH OF ENTRY 40-42	PACKED 43	DECIMAL POSITIONS 44	SEQUENCE 45	TABLE OR ARRAY NAME (VECTOR NAME) 46-51	LENGTH OF ENTRY 52-54	PACKED 55	DECIMAL POSITIONS 56	SEQUENCE 57	58
01	E	▨		VECTØR1		VEC1	10	10	5		2							
02	E	▨		VECTØR2		VEC2	10	10	2									
03	E	▨				VEC3	10	10	3		2		VEC4	2				

INPUT RECORDS

000001111122222333334444455555666667777788888899999
VEC1

AABBCCDDEEFFGGHHIIJJ
VEC2

000AA111BB222CC333DD444EE555FF666GG777HH888II999JJ
VEC3
VEC4

G14023

Figure 5-9. Vectors in Alternating Format

SECTION 6

LINE COUNTER SPECIFICATIONS

Line Counter Specifications are used only for line printer files to:

a. Specify form length. The default is 66 lines.

b. Specify the overflow line.

c. Define the line - channel equations that are associated with the carriage control format tape that must be installed on the printer when the program is executed.

Line Counter Specifications are required when using the RPG I dialect but are optional when using the RPG II dialect.

LINE - CHANNEL EQUATIONS

The LINE NUMBER, FL OR CHANNEL NUMBER, and OL OR CHANNEL NUMBER fields are used to specify line number - channel relationships referred to as line - channel equations

CHANNEL

The word "channel" is used in conjunction with the format or carriage control tape that is installed on the line printer. The carriage control tape is used to assist in the movement and positioning of the paper as it passes through the print mechanism.

LINE NUMBER - CHANNEL NUMBER

The LINE NUMBER and CHANNEL NUMBER fields are used to relate a line that is to be printed with a channel punch in the line printer carriage control tape.

FIELD DEFINITIONS

Figure 6-1 can be used in conjunction with the following field definitions for Line Counter Specifications.

1-2 PAGE

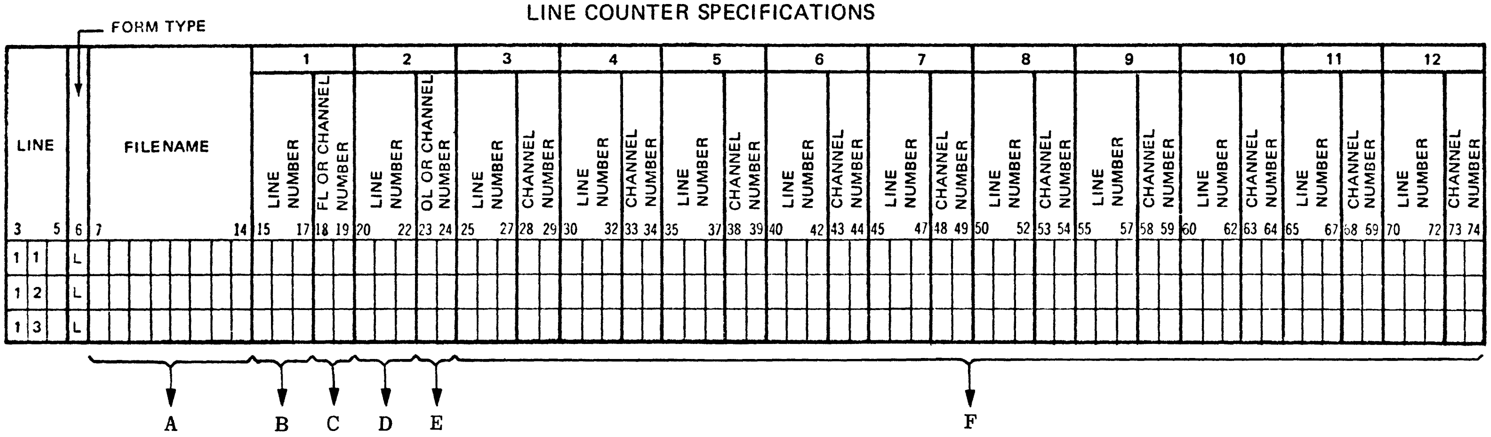
Refer to Section 2 for a complete description.

3-5 LINE

Refer to Section 2 for a complete description.

6 FORM TYPE

This field must contain the letter L.



A. 7-14 Specifies the name of a printer file.

B. 15-17 If 18-19 contains FL, this entry specifies the length of the page or form length. If 18-19 are numeric, this entry specifies the number of lines from the top of the page to associate with that channel number. Entries: 1-112, right-justified.

C. 18-19 Designates the use of the numeric entry in columns 15-17. Entries: FL or 1-12, right-justified.

D. 20-22 If 23-24 contains OL, this entry specifies the line number of the overflow line. If 23-24 are numeric, this entry specifies the number of lines from the top of the page to associate with that channel number. Entries: 1-112, right-justified.

E. 23-24 CHANNEL NUMBER associated with the overflow line designated in columns 20-22. Entries: OL or 1-12, right-justified.

F. 25-74 CHANNEL NUMBER related to preceding LINE NUMBER entry. Entries: 1-12, right-justified LINE NUMBER designates a particular line on each page. Entries: 1-112, right-justified.

G14029

Figure 6-1. Line Counter Specifications Summary Sheet

7-14 FILENAME

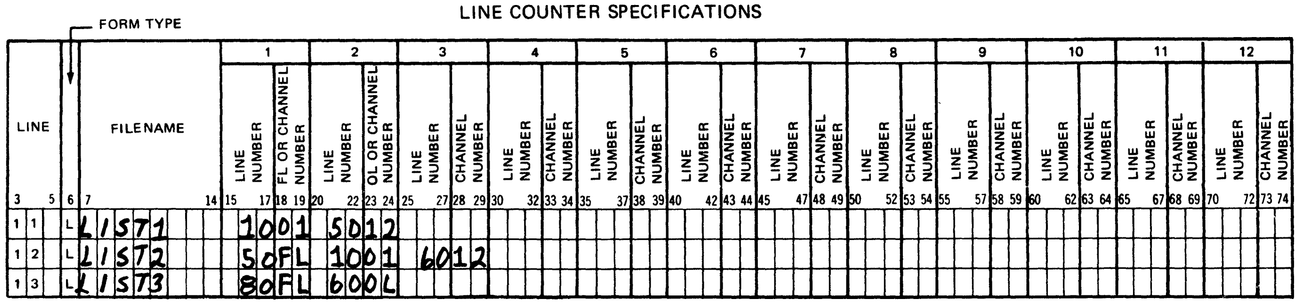
This field is used to specify the name of the printer file to which these Line Counter Specifications apply. The filename must also be described on the File Description Specifications and must be assigned to a printer. The entry in this field is required and must be left-justified.

15-19 LINE NUMBER, FL OR CHANNEL NUMBER

These fields have two possible meanings, depending upon the entry in columns 18-19.

If columns 18-19 contain the entry FL (Forms Length), the entry in columns 15-17 specifies the length (in print lines) of each page. The LINE NUMBER field entry must be between one and 112, inclusive, and be right-justified in the field, and leading zeroes are optional (see figure 6-2).

If columns 18-19 contain a numeric entry between one and 12, inclusive, the entry in columns 15-17 specifies the number of lines from the top of the form to be associated with the CHANNEL NUMBER entry designated in columns 18-19. The LINE NUMBER entry must be between one and 112, inclusive. Both entries must be right-justified in their respective fields, and leading zeros are optional (see figure 6-2).



LINE COUNTER SPECIFICATIONS

			1		2		3		4		5		6		7		8		9		10		11		12	
LINE	FORM TYPE	FILENAME	LINE NUMBER	FL OR CHANNEL NUMBER	LINE NUMBER	OL OR CHANNEL NUMBER	LINE NUMBER	CHANNEL NUMBER	LINE NUMBER	CHANNEL NUMBER	LINE NUMBER	CHANNEL NUMBER	LINE NUMBER	CHANNEL NUMBER	LINE NUMBER	CHANNEL NUMBER	LINE NUMBER	CHANNEL NUMBER	LINE NUMBER	CHANNEL NUMBER	LINE NUMBER	CHANNEL NUMBER	LINE NUMBER	CHANNEL NUMBER	LINE NUMBER	CHANNEL NUMBER
3 5	6	7 14	15 17	18 19	20 22	23 24	25 27	28 29	30 32	33 34	35 37	38 39	40 42	43 44	45 47	48 49	50 52	53 54	55 57	58 59	60 62	63 64	65 67	68 69	70 72	73 74
1 1	L	LIST1	10	01	50	12																				
1 2	L	LIST2	50	FL	10	01	60	12																		
1 3	L	LIST3	80	FL	60	0L																				

G14030

LIST1 specified channel 1 as line 10 and channel 12 as line 50.

LIST2 specifies the form length (50 lines) and channels 1 and 12.

LIST3 specifies the form length (80 lines) and the overflow line (line 60).

Figure 6-2. Line Counter Specifications Code Example

20-24 LINE NUMBER, OL OR CHANNEL NUMBER

This field has two possible functions, depending upon the entry in columns 23-24.

RPG II Dialect Only

When OL is entered in columns 23-24, the entry in columns 20-22 specifies the line number that will be considered the overflow line. The overflow line must be less than or equal to the form length. When the destination of a space or skip operation is a line beyond the overflow line but not beyond the form length the overflow indicator specified for the file is turned ON to indicate that the end of the page is near. If the destination is beyond the form length the overflow indicator does not turn ON.

When the output is conditioned on an overflow indicator or fetch overflow is specified and the overflow indicator is ON, the following actions will take place before the forms are advanced to the next page:

a. Detail lines still to be printed as part of the current program cycle will be completed.

b. Total lines will be printed.

c. Total line conditioned by the overflow indicator for this file will be printed

d. Heading and detail lines conditioned by the overflow indicator for this file will be printed.

Since all these actions will take place after the overflow line is reached, the programmer should be certain that enough space is left between the overflow line and the bottom of the page to allow all the lines to be printed. Refer to the subsection titled Printer File Handling in the Output-Format Specifications section for detailed information about overflow handling.

RPG I Or RPG II Dialect

If columns 23-24 contain a numeric entry between one and 12, inclusive, the entry in columns 20-22 must specify the number of lines from the top of the form to be associated with the CHANNEL NUMBER entry (columns 23-24). The LINE NUMBER field entry must be between one and 112, inclusive. Both entries must be right-justified in their respective fields, and leading zeros are optional (see figure 6-2).

25-74 LINE NUMBER AND CHANNEL NUMBER

The rest of the form is divided into ten 5-character fields, each consisting of a 3-character LINE NUMBER field and a 2-character CHANNEL NUMBER field. All fields are optional and must be left blank if they are not to be used (see figure 6-2).

The CHANNEL NUMBER fields may contain a numeric entry between one and 12, inclusive. This CHANNEL NUMBER entry is associated with the corresponding LINE NUMBER field entry (between one and 112, inclusive) and is used to relate a channel number to a particular line on each page of the output forms. The same channel numbers must not be specified more than once on the same Line Counter Specification.

All entries must be right-justified in their respective fields. Leading zeros are optional.

The LINE NUMBER field entry must not be greater than the form length specified in columns 18-19 or the default value of 66.

75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a complete description.

PRINTER CHANNEL SKIPPING

The user can control printer channel skipping by making appropriate entries in the RPG source program specifications. The entries in the specifications and the results obtained are different for the two RPG dialects, as defined in the following paragraphs.

CHANNEL SKIPPING - RPG II

When forms skipping is specified in columns 19-22 of the Output-Format Specifications, the entry refers to the line number on the form or page to be printed.

When a LINE NUMBER and CHANNEL NUMBER are entered in columns 15-19 of the Line Control Specifications form, a reference to a line in output for which a corresponding line - channel equation exists will result in the generation of a channel skip.

Example:

If the Line Control Specifications describe a line - channel equation which specifies that line 28 is equal to channel 3, then any time an output specification references a skip to line 28, a skip to channel 3 results.

CHANNEL SKIPPING - RPG I

When forms skipping is specified in columns 19-22 of the Output-Format Specifications, the entry refers to channel numbers, not lines. If line - channel equations are specified on the line counter specifications form, then a reference to a channel in output for which a corresponding line - channel equation exists results in the generation of a channel skip. The equations are necessary so the program can determine where the paper is positioned in relation to the overflow line.

CHANNEL SKIPPING - RPG I AND RPG II

When line - channel equations are specified but are not referenced on the Output-Format Specifications, no syntax error or warning message results.

It is the responsibility of the user to insure that the proper carriage control tape is installed on the line printer.

PRINTER BACKUP

If a line printer is not available during RPG program execution, the output data is sent to a printer backup file and is subsequently printed exactly as intended providing that the proper carriage control tape is installed on the line printer.

Line Counter Specifications are required when generating programs with RPG I. A summary of the required and optional entries follows:

a. Columns 15-19 can optionally contain the form length and columns 18-19 can optionally contain "FL". If these entries are omitted, the default value of 66 is assumed for the form length.

b. If the form length is specified in columns 15-19, then columns 20-74 can be used for line - channel equations that can specify a maximum of 11 channels.

c. If a form length is not specified in columns 15-19, the default value of 66 is assumed and columns 15-74 can be used to specify up to 12 channels.

d. The user must specify channel 1 and channel 12. Channel 12 defines the overflow line.

e. The user must provide line - channel specifications for any channel used in the Output-Format Specifications.

f. "OL" is not specified.

g. A skip to channel 12 is permitted.

SECTION 7

TELECOMMUNICATIONS CARD SPECIFICATIONS

The function of the Telecommunications Card is to further define a data communications file specified on the File Description Specifications as a REMOTE, DATACOM, or BSCA file. A Telecommunications Card is required for each data communications file specified on the File Description Specifications.

There is no special form used for coding Telecommunications specifications. The Control Card Specification form is used for illustrating Telecommunications coding positions and entries in this manual.

Standard coding procedures are used for the Input and Output Specifications when data communications files are declared. On output, exception records are written only to the REMOTE file whose filename is specified in the FACTOR 2 field with the SEND operation code that is being executed.

DATA COMMUNICATIONS FILES

Data communications files are required when the RPG program is to communicate with a remote device, and they are used to transmit and/or receive information contained in corresponding NDL files of the network controller. The filename of the NDL file must be entered in columns 7-14 of the Telecommunications Card. The data communications files used with B 1800/B 1700 RPG are of the following two types:

a. REMOTE

b. DATACOM and BSCA

The following two subsections titled REMOTE and DATACOM AND BSCA contain the detailed information required to define data communications files. The Telecommunications Card entries and the coding positions in which the entries must be made are explained for both types of data communications files.

REMOTE FILES

One Telecommunications Card is required for each REMOTE file declared in a program. Multiple stations can be assigned to a REMOTE file, and the remote devices are accessed on a demand basis.

A description of the required fields to be coded for a Telecommunications Card follows. Following the field definitions is figure 7-1, which illustrates the use of the Control Card Specifications form for coding Telecommunications entries for REMOTE files.

Field Definitions - REMOTE Files

1-2 PAGE

Refer to Section 2 for a complete description of this field.

3-5 LINE

Refer to Section 2 for a complete description of this field.

6 FORM TYPE

This field must contain a T.

7-14 FILENAME

This field must contain the name of the REMOTE file specified on the File Description Specifications.

15 -

Column 15 must be blank.

16-18 NUMBER OF STATIONS

A numeric entry must be made in this field to specify the maximum number of terminals that can be assigned to the data communications file. Valid entries are 001-999.

19-21 MAXIMUM MESSAGES

A numeric entry must be made in this field to specify the maximum number of input messages per station that can be queued.

Example:

If MAXIMUM MESSAGES is set to 2 and a terminal attempts to transmit three messages before the program reads a message, the following events occur:

a. The terminal remains in transmit mode following the attempted transmission of the third message since the input queue is full.

b. When the program performs a read operation, a message is read from the input queue, thereby making space available in the input queue for the third message.

c. The third message is then transmitted from the terminal buffer to the input queue, and the terminal returns to local mode.

22-27 STATION NUMBER

Prior to the execution of a SEND operation code, a valid relative station number must be entered in this 3-byte field. After a RECV operation code is executed, the MCP indicates to the program which station was read, and the station number is automatically placed in this field.

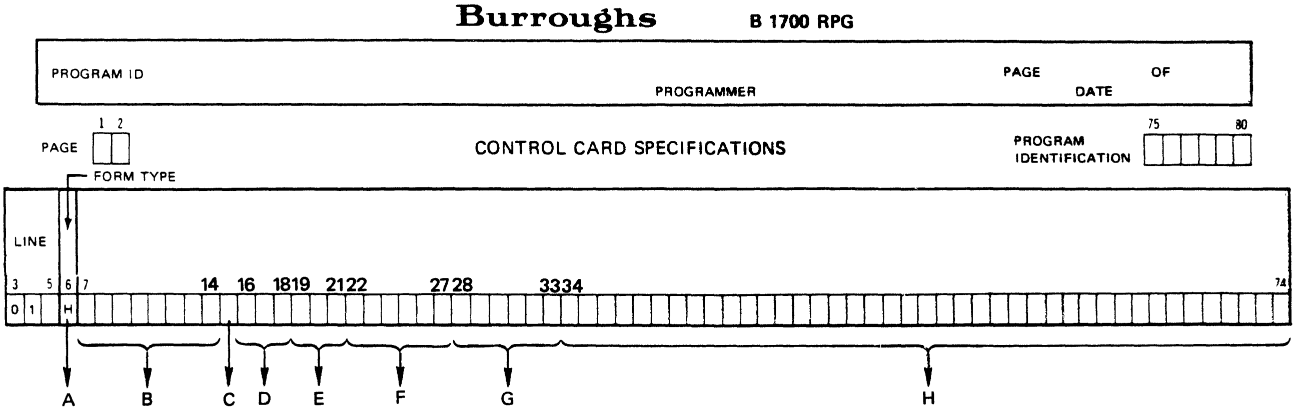
28-33 MESSAGE LENGTH

Prior to the execution of a SEND operation code, a message length equal to or less than the declared record size must be entered in the 4-byte message length field. After a RECV operation code is executed, the MCP indicates to the program the length of the message read, which is then placed in this field.

34-80 -

This field must remain blank.

Figure 7-1 illustrates the use of the Control Card Specifications form for coding Telecommunications entries for REMOTE files.



A. 6 Must contain the letter T.

B. 7-14 Must contain the name of a REMOTE file specified on the File Description Specifications.

C. 15 Must remain blank.

D. 16-18 A numeric entry is required to specify the maximum number of terminals that can be assigned to the REMOTE file.

E. 19-21 A numeric entry is required to specify the maximum number of input messages per station that can be queued.

F. 22-27 A field name which will contain the relative station number of the station to which a message is being sent must be entered in this field.

G. 28-33 A field name which will contain a message length equal to or less than the declared record size must be entered in the message length field.

H. 34-80 Must remain blank.

Figure 7-1. Telecommunications Entries - REMOTE Files

DATACOM OR BSCA FILES

One Telecommunications Card is required for each DATACOM or BSCA file declared in a program. Only one station (terminal) can be assigned to a DATACOM or BSCA file.

A description of the fields of a Telecommunications Card for DATACOM or BSCA files follows. Entries must be made in columns 6, and 7-14.

Figure 7-2 illustrates the use of the Control Card Specifications form for coding Telecommunications entries for DATACOM or BSCA files.

FIELD DEFINITIONS - DATACOM OR BSCA FILES

1-2 PAGE

Refer to Section 2 for a complete description.

3-5 LINE

Refer to Section 2 for a complete description.

6 FORM TYPE

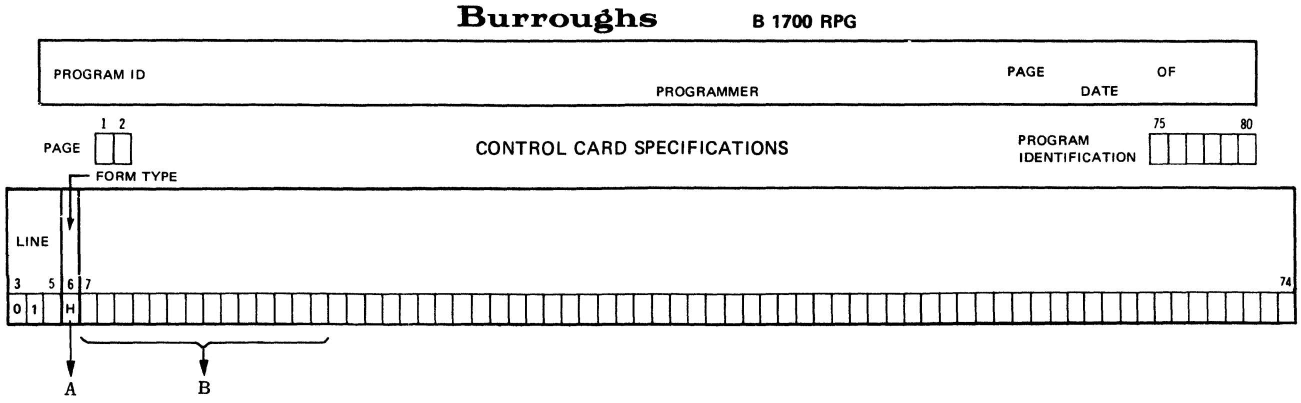
This field must contain a T.

7-14 FILENAME

This field must contain the name of the DATACOM or BSCA file specified on the File Description Specifications.

15 CONFIGURATION

This field is unused. Any entry other than blank, P, M, or S will be given a syntax error.



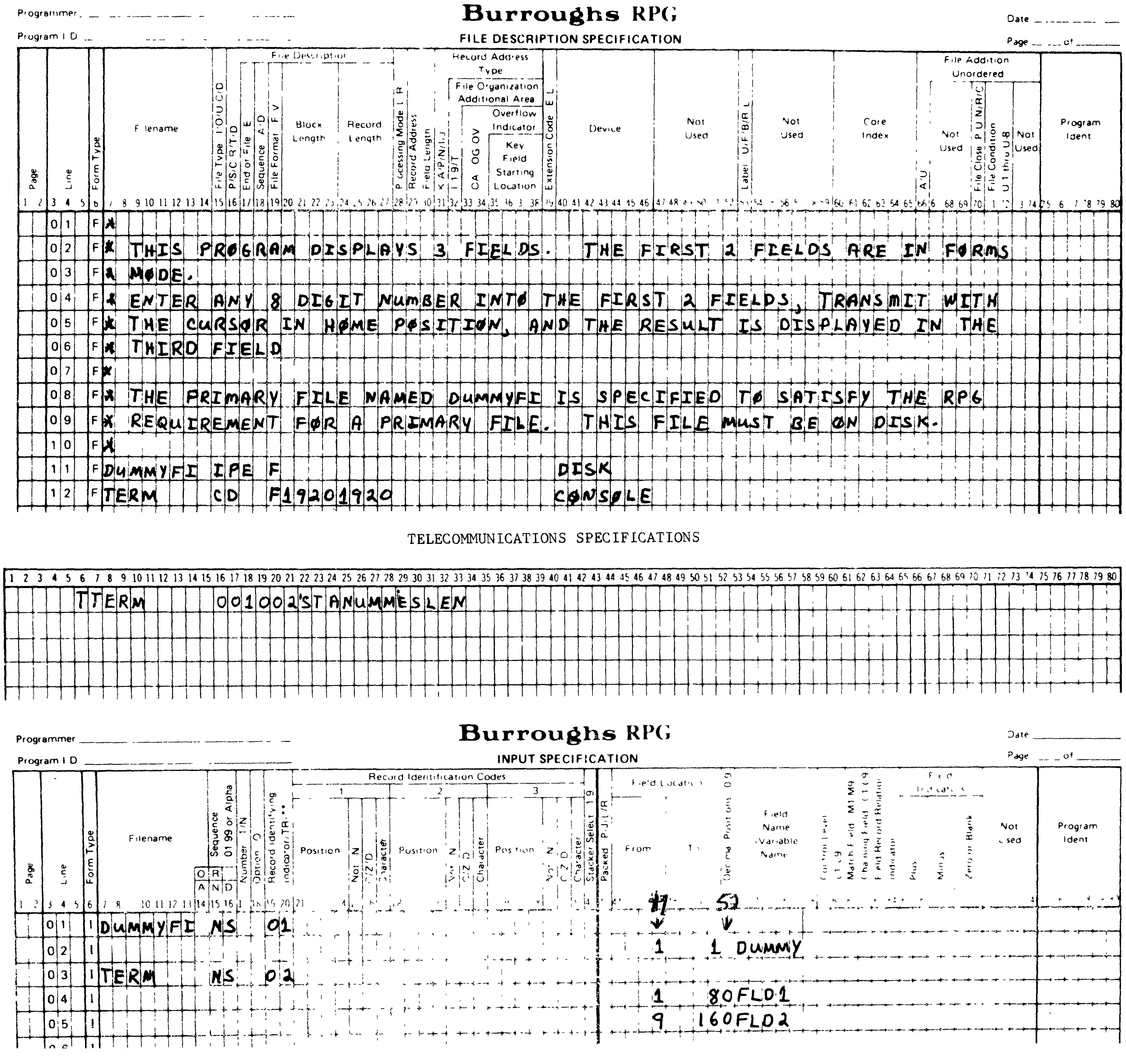
A. 6 Must contain the letter T.

B. 7-14 Contains the name of a DATACOM or BSCA file entered on the File Description Specifications.

G14033

Figure 7-2. Telecommunications Entries - DATACOM or BSCA

Figure 7-3 is an example of an RPG data communications program. The program uses the BITON and BITOF operation codes to build the special characters required in data communications programming. Refer to Section 9 for a complete description of the BITON and BITOF operation codes.



Programmer
Program I D

Burroughs RPG

FILE DESCRIPTION SPECIFICATION

Date
Page of

```
  01 F*
  02 F* THIS PRØGRAM DISPLAYS 3 FIELDS.  THE FIRST 2 FIELDS ARE IN FØRMS
  03 F* MØDE.
  04 F* ENTER ANY 8 DIGIT NUMBER INTØ THE FIRST 2 FIELDS, TRANSMIT WITH
  05 F* THE CURSØR IN HØME PØSITIØN, AND THE RESULT IS DISPLAYED IN THE
  06 F* THIRD FIELD
  07 F*
  08 F* THE PRIMARY FILE NAMED DUMMYFI IS SPECIFIED TØ SATISFY THE RPG
  09 F* REQUIREMENT FØR A PRIMARY FILE.  THIS FILE MUST BE ØN DISK.
  10 F*
  11 FDUMMYFI IPE F                    DISK
  12 FTERM    CD  F19201920            CØNSØLE
```

TELECOMMUNICATIONS SPECIFICATIONS

```
     TTERM     001002'STANUMMESLEN
```

Programmer
Program I D

Burroughs RPG

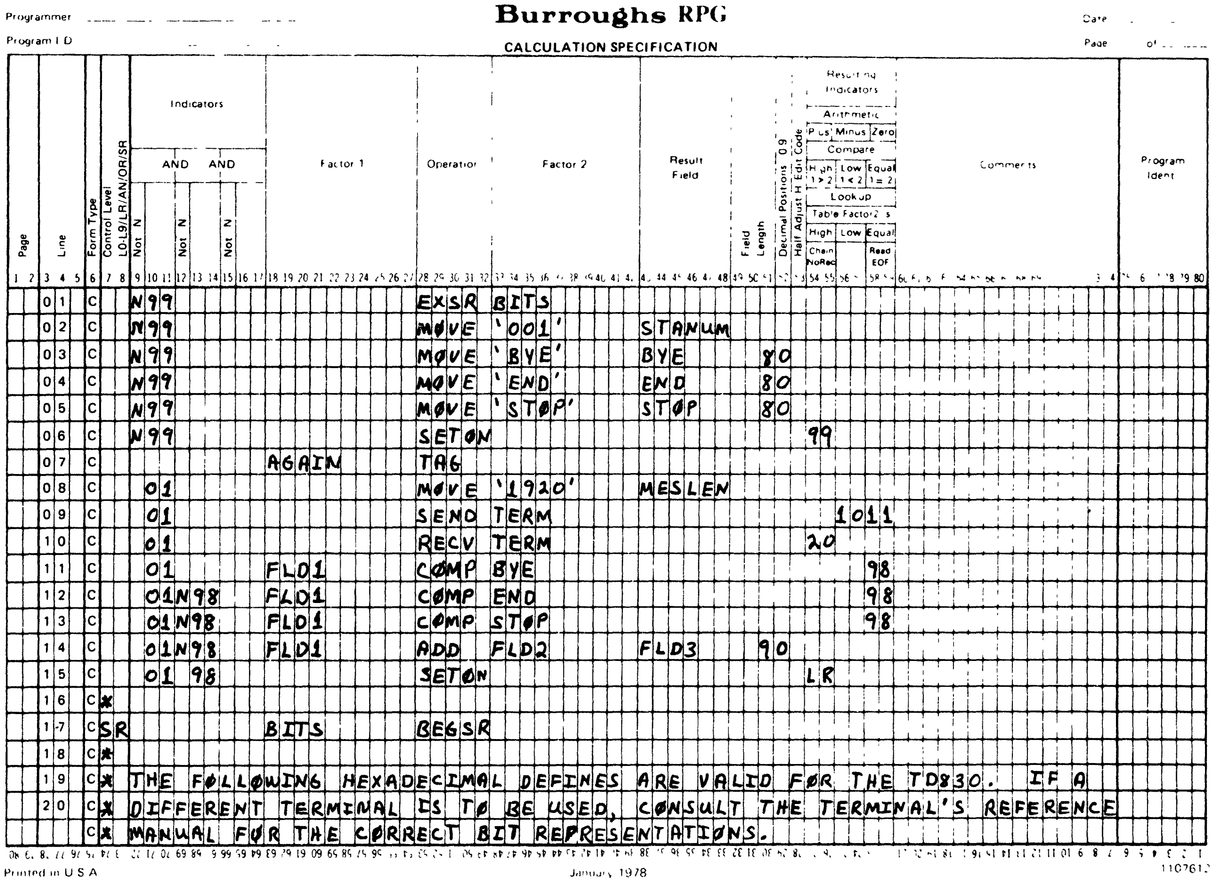
INPUT SPECIFICATION

Date
Page of

```
                                             47   52
  01 IDUMMYFI NS  01
  02 I                                        1   1 DUMMY
  03 ITERM    NS  02
  04 I                                        1   80FLD1
  05 I                                        9  160FLD2
```

G12068/SHEET 1 OF 5

Figure 7-3. Example of an RPG Data Communications Program (Sheet 1 of 5)


Page
Line
Form Type
Control Level
L0-L9/LR/AN/OR/SR
Indicators
AND
AND
Not N
Not N
Not N
Factor 1
Operation
Factor 2
Result Field
Field Length
Decimal Positions 0-9
Half Adjust H Edit Code
Resulting Indicators
Arithmetic
Plus
Minus
Zero
Compare
High 1>2
Low 1<2
Equal 1=2
Lookup
Table Factor 2 is
High
Low
Equal
Chain NoRec
Read EOF
Comments
Program Ident

Programmer Burroughs RPG Date

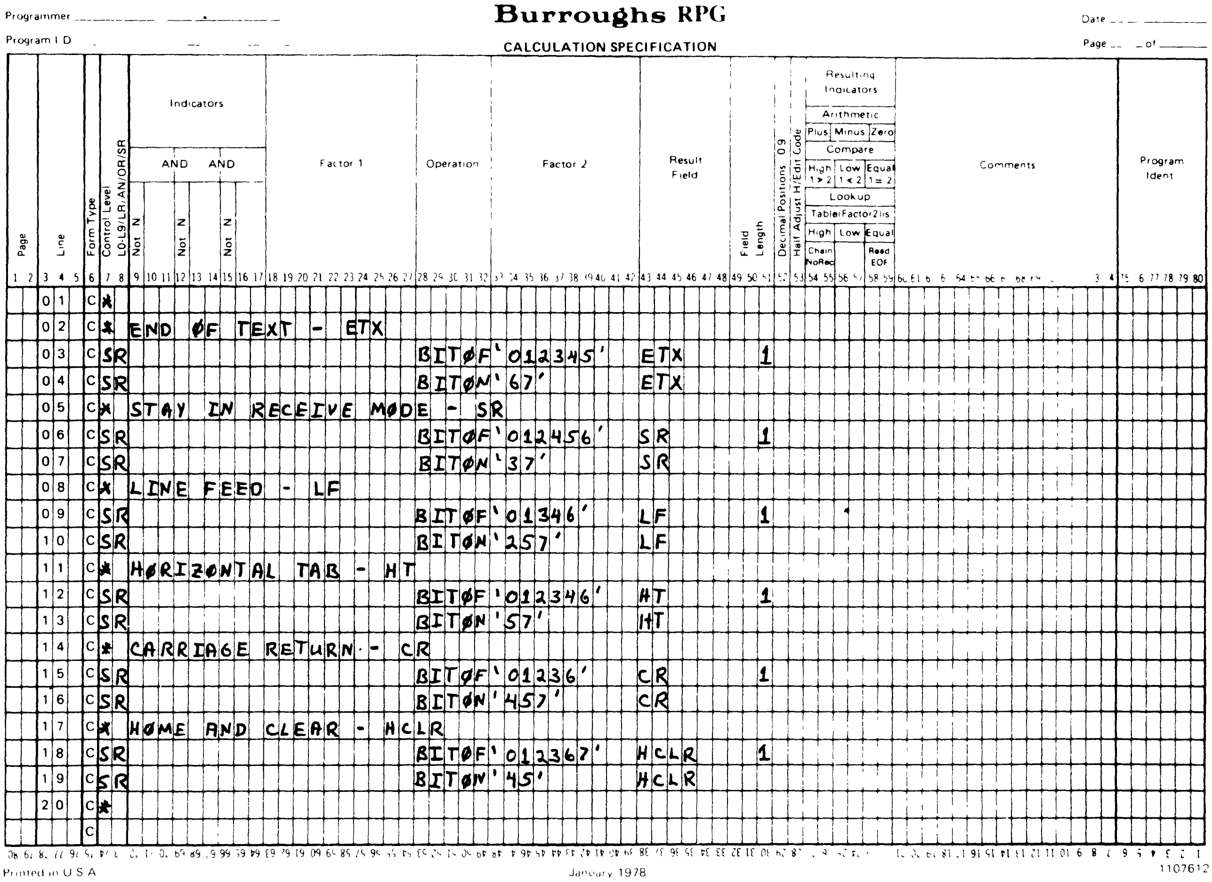
Program I D CALCULATION SPECIFICATION Page of

```
  01 C  N99                EXSR BITS
  02 C  N99                MØVE '001'     STANUM
  03 C  N99                MØVE 'BYE'     BYE     80
  04 C  N99                MØVE 'END'     END     80
  05 C  N99                MØVE 'STØP'    STØP    80
  06 C  N99                SETØN                     99
  07 C           AGAIN     TAG
  08 C   01                MØVE '1920'    MESLEN
  09 C   01                SEND TERM                   1011
  10 C   01                RECV TERM                 20
  11 C   01      FLD1      CØMP BYE                      98
  12 C   01N98   FLD1      CØMP END                      98
  13 C   01N98   FLD1      CØMP STØP                     98
  14 C   01N98   FLD1      ADD  FLD2      FLD3    90
  15 C   01 98             SETØN                     LR
  16 C*
  17 CSR         BITS      BEGSR
  18 C*
  19 C* THE FØLLØWING HEXADECIMAL DEFINES ARE VALID FØR THE TD830.  IF A
  20 C* DIFFERENT TERMINAL IS TØ BE USED, CØNSULT THE TERMINAL'S REFERENCE
     C* MANUAL FØR THE CØRRECT BIT REPRESENTATIØNS.
```

Printed in U S A January 1978 1107612

G12068/SHEET 2 OF 5

Figure 7-3. Example of an RPG Data Communications Program (Sheet 2 of 5)



Programmer

Program I D

Burroughs RPG

CALCULATION SPECIFICATION

Date

Page ___ of ___

Page	Line	Form Type	Control Level (L0-L9/LR/AN/OR/SR)	Indicators (Not / AND Not / AND Not)	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust H/Edit Code	Resulting Indicators	Comments	Program Ident
	01	C	*											
	02	C	*	END ØF TEXT - ETX										
	03	C	SR			BITØF	'012345'	ETX	1					
	04	C	SR			BITØN	'67'	ETX						
	05	C	*	STAY IN RECEIVE MØDE - SR										
	06	C	SR			BITØF	'012456'	SR	1					
	07	C	SR			BITØN	'37'	SR						
	08	C	*	LINE FEED - LF										
	09	C	SR			BITØF	'01346'	LF	1					
	10	C	SR			BITØN	'257'	LF						
	11	C	*	HØRIZØNTAL TAB - HT										
	12	C	SR			BITØF	'012346'	HT	1					
	13	C	SR			BITØN	'57'	HT						
	14	C	*	CARRIAGE RETURN - CR										
	15	C	SR			BITØF	'01236'	CR	1					
	16	C	SR			BITØN	'457'	CR						
	17	C	*	HØME AND CLEAR - HCLR										
	18	C	SR			BITØF	'012367'	HCLR	1					
	19	C	SR			BITØN	'45'	HCLR						
	20	C	*											
		C												

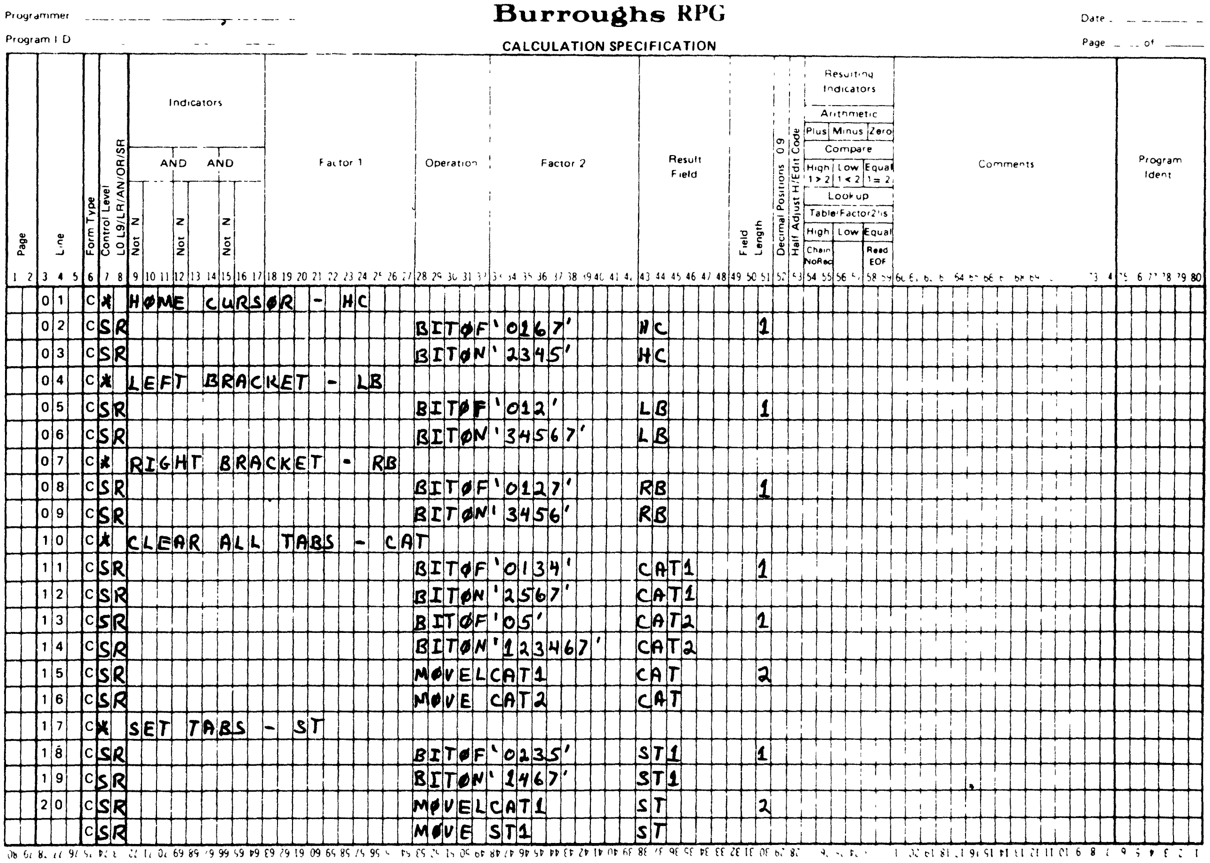
Printed in U S A

January 1978

1107612

G12068/SHEET 3 OF 5

Figure 7-3. Example of an RPG Data Communications Program (Sheet 3 of 5)



Programmer
Program I D
Date
Page of

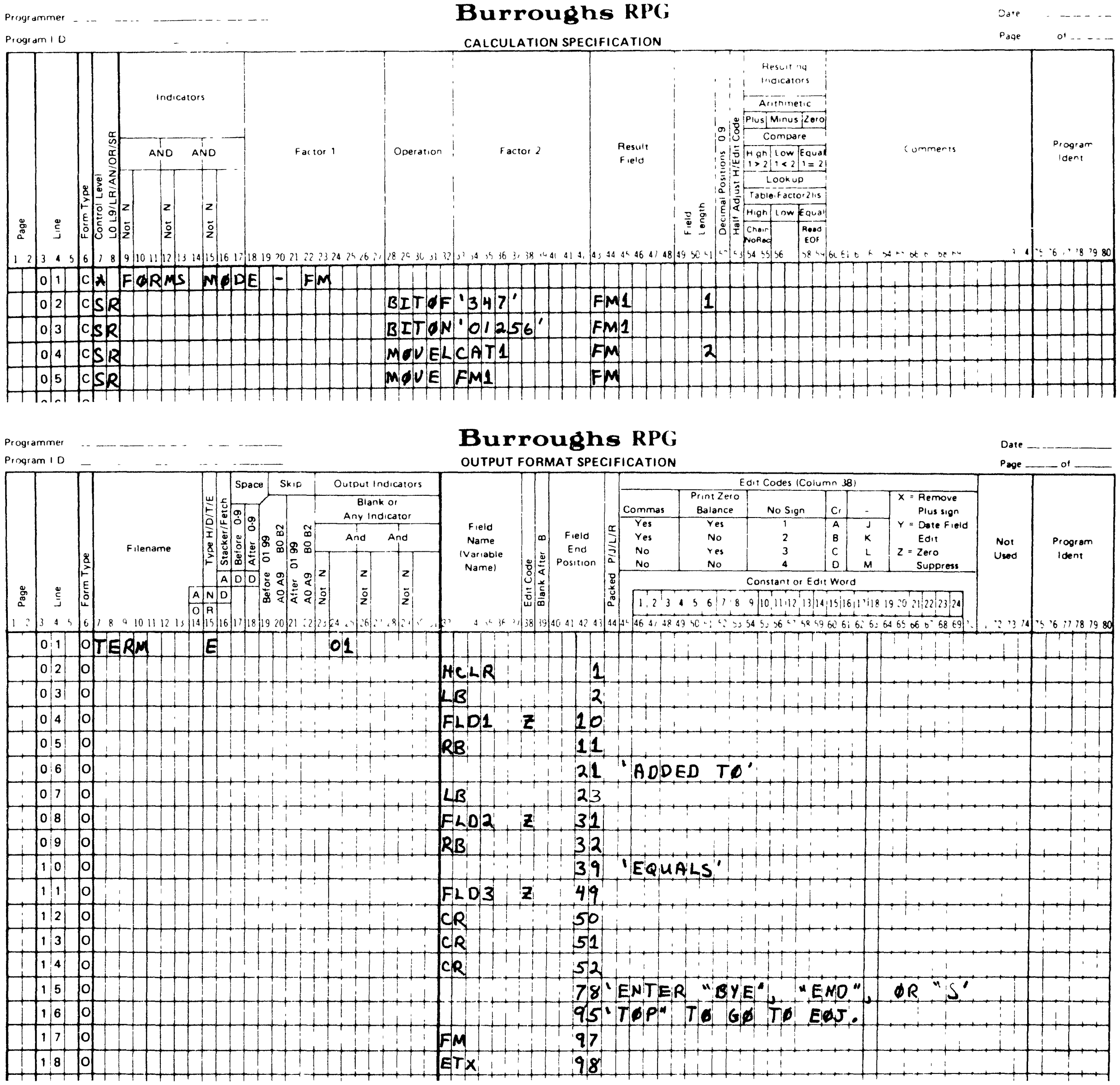
Burroughs RPG

CALCULATION SPECIFICATION

Line	Form Type	Control Level	Indicators / Factor 1	Operation	Factor 2	Result Field	Field Length
01	C	*	HØME CURSØR - HC				
02	C	SR		BITØF	'0167'	HC	1
03	C	SR		BITØN	'2345'	HC	
04	C	*	LEFT BRACKET - LB				
05	C	SR		BITØF	'012'	LB	1
06	C	SR		BITØN	'34567'	LB	
07	C	*	RIGHT BRACKET - RB				
08	C	SR		BITØF	'0127'	RB	1
09	C	SR		BITØN	'3456'	RB	
10	C	*	CLEAR ALL TABS - CAT				
11	C	SR		BITØF	'0134'	CAT1	1
12	C	SR		BITØN	'2567'	CAT1	
13	C	SR		BITØF	'05'	CAT2	1
14	C	SR		BITØN	'123467'	CAT2	
15	C	SR		MØVEL	CAT1	CAT	2
16	C	SR		MØVE	CAT2	CAT	
17	C	*	SET TABS - ST				
18	C	SR		BITØF	'0235'	ST1	1
19	C	SR		BITØN	'1467'	ST1	
20	C	SR		MØVEL	CAT1	ST	2
	C	SR		MØVE	ST1	ST	

G12068/SHEET 4 OF 5

Figure 7-3. Example of an RPG Data Communications Program (Sheet 4 of 5)



Burroughs RPG

CALCULATION SPECIFICATION

Programmer
Program I D
Date
Page of

Line	Form Type	Control Level	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length
01	C	* FØRMS MØDE - FM						
02	C	SR			BITØF	'347'	FM1	1
03	C	SR			BITØN	'01256'	FM1	
04	C	SR			MØVEL	CAT1	FM	2
05	C	SR			MØVE	FM1	FM	

Burroughs RPG

OUTPUT FORMAT SPECIFICATION

Programmer
Program I D
Date
Page of

Line	Form Type	Filename	Type H/D/T/E	Output Indicators	Field Name (Variable Name)	Edit Code	End Position	Constant or Edit Word
01	O	TERM	E	01				
02	O				HCLR		1	
03	O				LB		2	
04	O				FLD1	Z	10	
05	O				RB		11	
06	O						21	'ADDED TØ'
07	O				LB		23	
08	O				FLD2	Z	31	
09	O				RB		32	
10	O						39	'EQUALS'
11	O				FLD3	Z	49	
12	O				CR		50	
13	O				CR		51	
14	O				CR		52	
15	O						78	'ENTER "BYE", "END", ØR "S'
16	O						95	'TØP" TØ GØ TØ EØJ.
17	O				FM		97	
18	O				ETX		98	

G12068/SHEET 5 OF 5

Figure 7-3. Example of an RPG Data Communications Program (Sheet 5 of 5)

SECTION 8

INPUT SPECIFICATIONS

Input Specifications describe the records within each file and fields within each record to be used as input data for the program. The two types of input specifications are:

a. Record type descriptions (columns 7-42) which define the various input records and their relationship to other records in the file. Columns 43-70 must be blank.

b. Field descriptions (columns 43-70) which define each field within the records. Columns 7-72 must be blank.
7-42

Field description entries must start one line below the associated record type descriptions, or an error will occur. A warning is emitted if a record type description is not followed by a field description. Field and record descriptions must not be specified on the same line.

FIELD DEFINITIONS

Figure 8-1 can be used in conjunction with the following field definitions for the Input Specifications.

1-2 PAGE

Refer to Chapter 2 for a complete description.

3-5 LINE

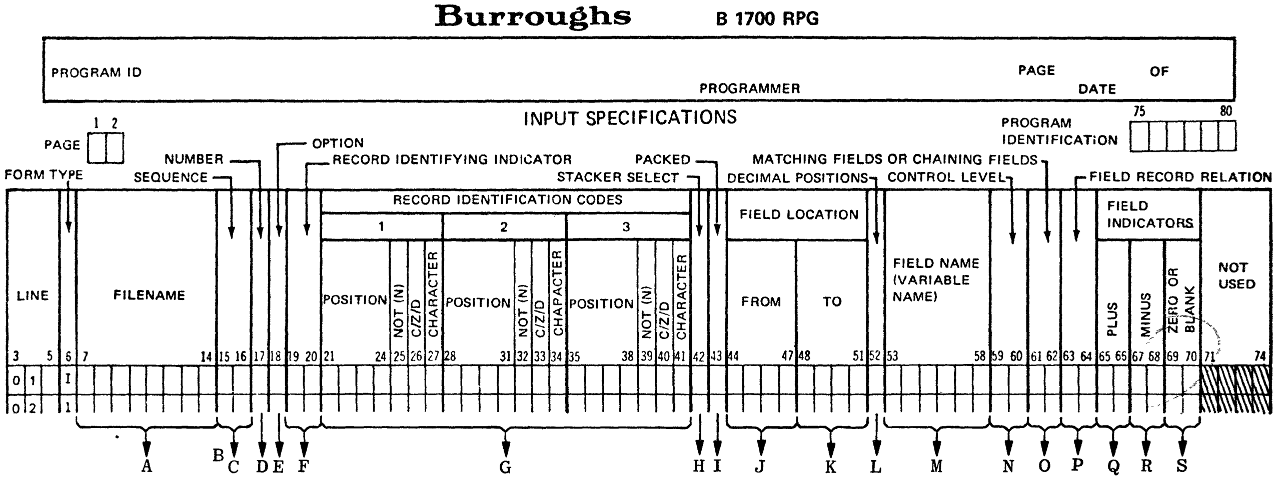
Refer to Chapter 2 for a complete description.

6 FORM TYPE

This field must contain the letter I.

7-14 FILENAME

This field is used to identify the file to which the subsequent record type and field descriptions belong. The file specified must have been previously described on the File Description Specifications form as an input, update, or combined file. Every input, update, or combined file (except input table files and record address files), described in the File Description Specifications, must be described on the Input Specifications form. The FILENAME entry must be the same as the one used in the File Description Specifications. It must appear on the first line containing information about the records in the file; if the entry is left blank, the last filename entered is assumed to be the file being described. The first record type description must not have a blank FILENAME entry:



A. 7-14 Contains a filename specified in the File Description Specifications.

B. 14-16 Puts the record identification codes in an AND or OR relationship. Entries: AND or OR.

C. 15-16 Specifies if records are to be processed in a predetermined sequence. Entries: 01-99 or any alphabetic character.

D. 17 If sequence is specified, the entry indicates the number of records of each type in a sequence group. Entries: 1 or N.

E. 18 Specifies if records sequenced must be present. Entries: Blank or O.

F. 19-20 Contains either a record identifying indicator (01-99), a control level indicator (L1-L9), a halt indicator (H1-H9), the spread card designator (TR), or specifies look ahead records (**).

G. 21-41 Record Identification Codes:

21-24, 28-31, 35-38 (specifies the position within the record that contains a record identification code. Entries: Blank or 1-N where N = record length).

25, 32, 39 (indicates if the character in columns 27, 34, or 41 is or is not present. Entries: Blank or N.

26, 33, 40 (Specifies what part of the code character in columns 27, 34, or 41 is to be read. Entries: C - the entire character, Z - the zone portion, D - the digit portion).

27, 34, 41 (Contains the code character. Entries: Any EBCDIC character).

H. 42 Specifies which stacker will be used. Entries: Blank or 1-6.

I. 43 Specifies if numeric input is in packed or unpacked decimal format, or binary format. Entries: Blank, P, or B.

J. 44-47 Contains the left-most position of the input field. Entries: numeric, right-justified.

K. 48-51 Contains the right-most position of the input field. Entries: numeric, right-justified.

L. 52 Contains the number of decimal positions for numeric fields. Entries: Blank or 0-9.

M. 53-58 Contains field names (1-6 alphanumeric characters, left-justified) or one of the special field names, PAGE, PAGE1, PAGE2, UDATE, UMONTH, UDAY, UYEAR, TABXXX (X=any alphanumeric character).

N. 59-60 Contains the control level indicator. Entries: Blank or L1-L9.

O. 61-62 Specifies sequence checking for a single input or combined file or sequence checking with matching records for two or more input and/or combined files. Entries: Blank or M1-M9.

P. 63-64 Contains one of the following field record relations indicators or blank: 01-99 (record identifying indicator), L1-L9 (control level indicator previously defined), MR (matching record indicator), U1-U8 (external indicator), or H0-H9 (halt indicator).

Q. 65-66 Used to indicate if the specified field is greater than blank or positive. Entries: Blank, 01-99, H0-H9.

R. 67-68 Used to indicate if the specified field is less than blank or negative. Entries: Blank, 01-99, H0-H9.

S. 69-70 Used to indicate if the specified field is blank or zero. Entries: Blank, 1-99, H0-H9.

Figure 8-1. Input Specifications Summary Sheet

Primary and secondary files are processed in the same order as they are described in the File Description Specifications. If primary or secondary files are not described on Input Specifications in the same order as they are described on the File Description Specifications, a warning is emitted. The warning is emitted because the object program from some RPG compilers may process primary and secondary files according to the order in which they appear on the Input Specifications

All record type and field descriptions for a particular file must be grouped together on the Input Specifications. Descriptions of records from different files must not be interspersed.

14-16 AND/OR LINES

There is no limit on the number of AND or OR lines that may be specified; however, it is recommended that the user not exceed 20 if compatibility with other Burroughs systems is desired. AND/OR lines must be preceded by a line containing at least one record identification code entry.

AND Line

If it is necessary to specify more than three record identifying codes to identify a record type, an AND line may be used. The word AND shquld be entered in columns 14-16 and the additional record identifying codes should be entered in columns 21-41. There must be at least one record identification code entry on each AND line.

OR Line

In some cases, a particular record type may be identified by two or more different codes. For this condition, the word OR entered in columns 14-15 indicates that only one of the codes specified need be present to identify the record type (see figure 8-2). Record identification codes are not required on OR lines, although this is not necessarily a meaningful thing to do. Other uses of the OR relationship are discussed later in this chapter (refer to figures 8-10 and 8-18).

15-16 SEQUENCE

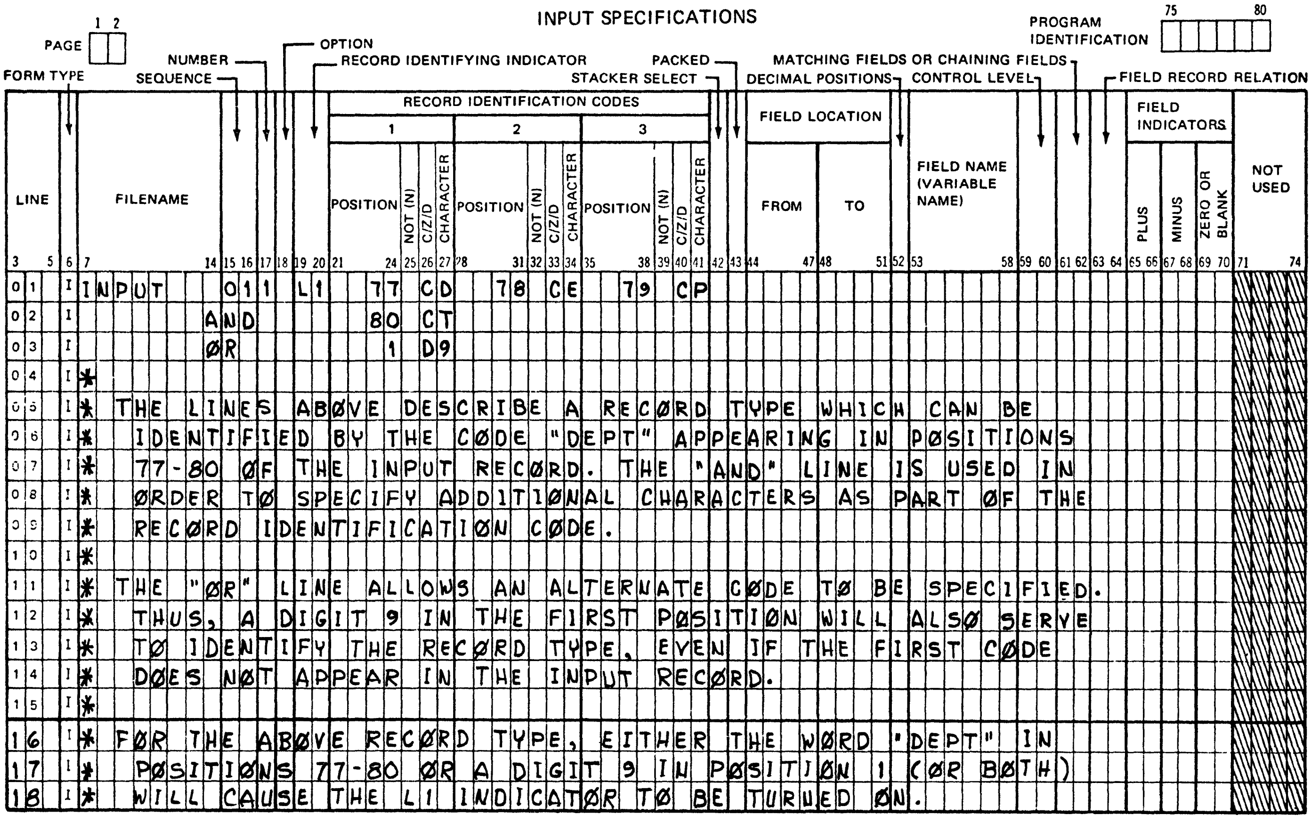
This field is used to specify a special sequence to different record types in a file. If this field contains an alphabetic entry (note that this includes a blank entry, although a warning will be emitted), it specifies that the record types need not be in any special order.

Within each file, all record types having alphabetic entries in the SEQUENCE field must be specified before those with numeric entries. Refer to figure 8-4 for an example of how to code the SEQUENCE field when both alphabetic and numeric entries are desired. All chained and demand files must have an alphabetic entry in this field.

[redacted]

A sequence group is data file records which are defined by a numeric entry specified in columns 15-16 of the Input Specification line.

If this field contains a numeric entry, it indicates that sequence checking is to be done. The order of precedence is the sequence in which the records are declared on the Input Specifications. This allows the programmer to specify that one record type must appear before


INPUT SPECIFICATIONS
PAGE
FORM TYPE
NUMBER
SEQUENCE
OPTION
RECORD IDENTIFYING INDICATOR
PACKED
STACKER SELECT
DECIMAL POSITIONS
MATCHING FIELDS OR CHAINING FIELDS
CONTROL LEVEL
FIELD RECORD RELATION
PROGRAM IDENTIFICATION
RECORD IDENTIFICATION CODES
POSITION
NOT (N)
C/Z/D
CHARACTER
FIELD LOCATION
FROM
TO
FIELD NAME (VARIABLE NAME)
FIELD INDICATORS
PLUS
MINUS
ZERO OR BLANK
NOT USED
LINE
FILENAME
G14035

```
01 IINPUT   011 L1  77 CD  78 CE  79 CP
02 I       AND      80 CT
03 I       ØR        1 D9
04 I*
05 I* THE LINES ABØVE DESCRIBE A RECØRD TYPE WHICH CAN BE
06 I*   IDENTIFIED BY THE CØDE "DEPT" APPEARING IN PØSITIØNS
07 I*   77-80 ØF THE INPUT RECØRD. THE "AND" LINE IS USED IN
08 I*   ØRDER TØ SPECIFY ADDITIØNAL CHARACTERS AS PART ØF THE
09 I*   RECØRD IDENTIFICATIØN CØDE.
10 I*
11 I* THE "ØR" LINE ALLOWS AN ALTERNATE CØDE TØ BE SPECIFIED.
12 I*   THUS, A DIGIT 9 IN THE FIRST PØSITIØN WILL ALSØ SERVE
13 I*   TØ IDENTIFY THE RECØRD TYPE, EVEN IF THE FIRST CØDE
14 I*   DØES NØT APPEAR IN THE INPUT RECØRD.
15 I*
16 I* FØR THE ABØVE RECØRD TYPE, EITHER THE WØRD "DEPT" IN
17 I*   PØSITIØNS 77-80 ØR A DIGIT 9 IN PØSITIØN 1 (ØR BØTH)
18 I*   WILL CAUSE THE L1 INDICATØR TØ BE TURNED ØN.
```

Figure 8-2. AND/OR Relationships - Record Identification Codes

another record type within a sequenced group. The program will automatically check the designated order as the records are read.

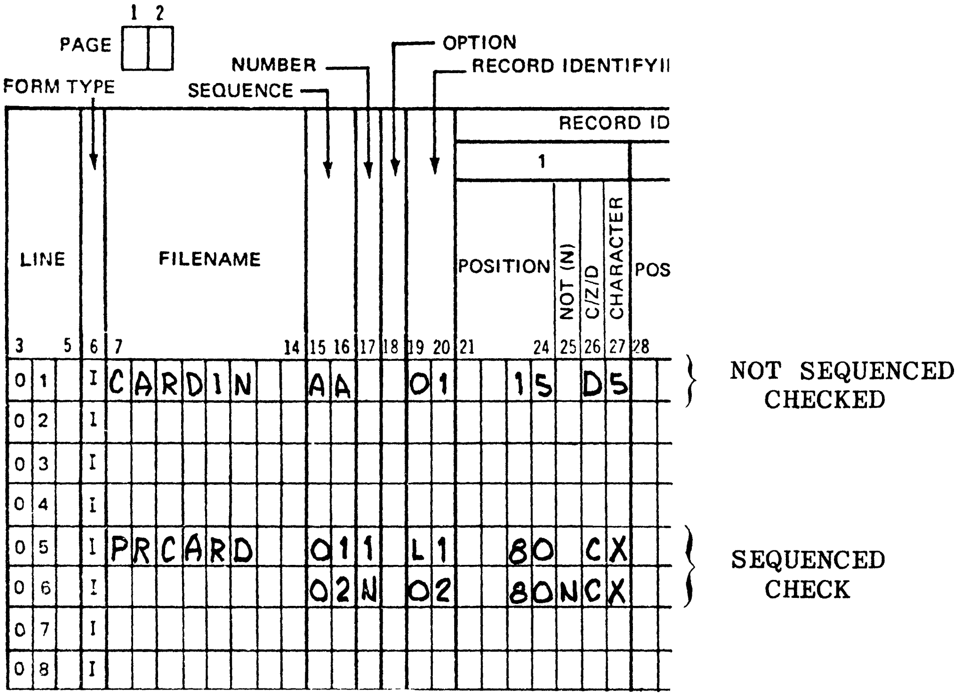
The first sequenced record type specified must have the lowest sequence number (01), the next record type should be given a higher number, etc. Gaps in sequence numbers are allowed, but the numbers used must be used in ascending order.

If a record is encountered that is out of sequence, the program will halt. The system operator can order the program to resume, at which time it will ignore the record that is out of sequence and read the next record from the file.

Records in an AND or OR line cannot have a sequence field entry; the entry from the previous line also applies to the line with the AND or OR entry.

In the example shown in figure 8-3, the input file PRCARD contains two record types which are to be sequence checked. Each group of input records of the input file PRCARD must contain exactly one of the first record type which may be followed by any number of records of the second type. A record identifying indicator of L1 is assigned to the first record type, so that a control break will occur each time the first record of a new group is read.

Refer to figure 8-4 for an example of how to code the SEQUENCE field when both alphabetic and numeric entries are desired.


1 2
PAGE
FORM TYPE
NUMBER
SEQUENCE
OPTION
RECORD IDENTIFYII
RECORD ID
NOT SEQUENCED CHECKED
SEQUENCED CHECK

LINE (3-5)	FORM TYPE (6)	FILENAME (7-14)	SEQUENCE (15-16)	NUMBER (17)	OPTION (18)	RECORD IDENTIFYII (19-20)	POSITION (21-24)	NOT (N) (25)	C/Z/D (26)	CHARACTER (27)	POS (28)
01	I	CARDIN	AA			01	15		D	5	
02	I										
03	I										
04	I										
05	I	PRCARD	01	1		L1	80		C	X	
06	I		02	N		02	80	N	C	X	
07	I										
08	I										

G14036

Figure 8-3. Input Sequence Checking

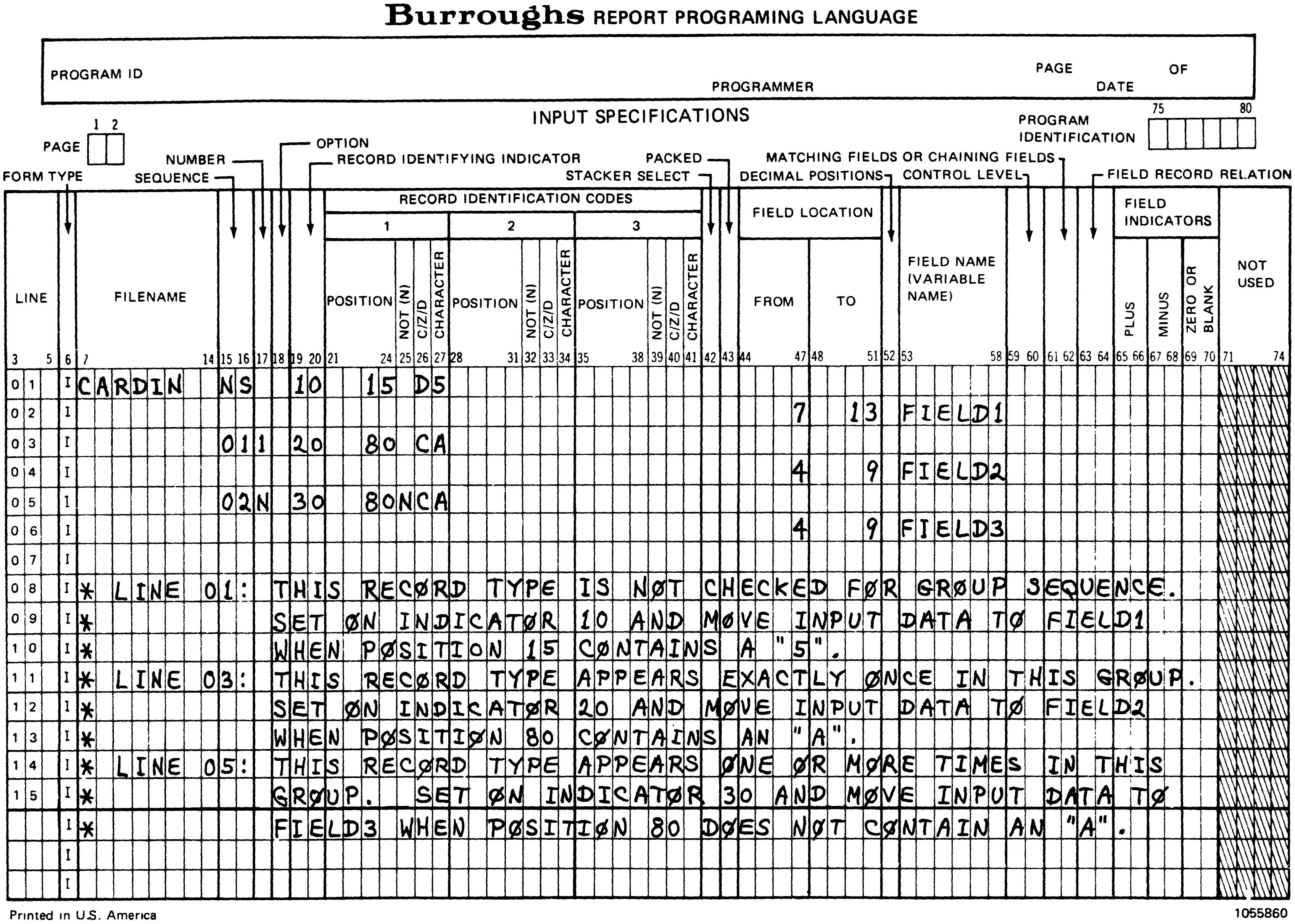
17 NUMBER

This field is used only if sequence checking is to be done (i.e., the SEQUENCE field contains a numeric entry). An entry in this field indicates whether more than one record of the designated type may appear in each group of a sequenced input file (see figure 8-4).

Records in an AND or OR line cannot have a NUMBER field entry; the entry from the previous line also applies to the line with the AND or OR entry.

Valid entries for the NUMBER field are:

Entry	Definition
Blank	Record types are not being sequence checked (SEQUENCE field contains alphabetic entry).
1	Not more than one record of this type will be present in each sequence group.
N	One or more records of this type will be present in each sequence group.



Burroughs REPORT PROGRAMING LANGUAGE

PROGRAM ID PROGRAMMER PAGE OF DATE

INPUT SPECIFICATIONS

PAGE 1 2 □□ PROGRAM IDENTIFICATION 75 80 □□□□□□

FORM TYPE · SEQUENCE · NUMBER · OPTION · RECORD IDENTIFYING INDICATOR · PACKED · STACKER SELECT · DECIMAL POSITIONS · MATCHING FIELDS OR CHAINING FIELDS · CONTROL LEVEL · FIELD RECORD RELATION

LINE (3–5) · FILENAME (7–14) · RECORD IDENTIFICATION CODES 1: POSITION (21–24), NOT (N) (25), C/Z/D (26), CHARACTER (27); 2: POSITION (28–31), NOT (N) (32), C/Z/D (33), CHARACTER (34); 3: POSITION (35–38), NOT (N) (39), C/Z/D (40), CHARACTER (41) · FIELD LOCATION: FROM (44–47), TO (48–51) · FIELD NAME (VARIABLE NAME) (53–58) · FIELD INDICATORS: PLUS (65–66), MINUS (67–68), ZERO OR BLANK (69–70) · NOT USED (71–74)

```
01 ICARDIN  NS  10  15 D5
02 I                                        7  13 FIELD1
03 I        011 20  80 CA
04 I                                        4   9 FIELD2
05 I        02N 30  80NCA
06 I                                        4   9 FIELD3
07 I
08 I* LINE  01: THIS RECØRD TYPE IS NØT CHECKED FØR GRØUP SEQUENCE.
09 I*           SET ØN INDICATØR 10 AND MØVE INPUT DATA TØ FIELD1
10 I*           WHEN PØSITION 15 CØNTAINS A "5".
11 I* LINE  03: THIS RECØRD TYPE APPEARS EXACTLY ØNCE IN THIS GRØUP.
12 I*           SET ØN INDICATØR 20 AND MØVE INPUT DATA TØ FIELD2
13 I*           WHEN PØSITIØN 80 CØNTAINS AN "A".
14 I* LINE  05: THIS RECØRD TYPE APPEARS ØNE ØR MØRE TIMES IN THIS
15 I*           GRØUP.  SET ØN INDICATØR 30 AND MØVE INPUT DATA TØ
   I*           FIELD3 WHEN PØSITIØN 80 DØES NØT CØNTAIN AN "A".
   I
   I
```

Printed in U.S. America

1055860

G12039

Figure 8-4. Example of Alphabetic and Numeric Entries in SEQUENCE Field

18 OPTION

This field is used only if sequence checking is to be done (i.e., the SEQUENCE field contains a numeric entry). An entry in this field indicates whether certain record types are optional. An alphanumeric "0" entry specifies that a record of this type may or may not be present in each group of a sequenced file. If this field is left blank, each group may contain one or any number of records of this type, depending on the entry in column 17 (see figure 8-4).

Records in an AND or OR line cannot have an OPTION field entry; the entry from the previous line also applies to the line with the AND or OR entry. Valid entries are:

Entry	Definition
Blank	Record type must be present in each group.
0	Record type is optional, and may not be present in each group.

If all record types in a file are designated as optional, no sequence errors will be detected.

Example Coding of Sequence, Number, and Option Fields

Figures 8-4A, B, and C illustrate coding of the sequence, number, and option fields on the Input Specifications. Following each figure is a description of the specified record sequence. In conjunction, tables 8-1, 8-2, and 8-3 list possible sequences which records from SEQ1, SEQ2, and SEQ3 can appear in for figures 8-4A, B, and C, respectively. The records from the SEQ1, SEQ2, and SEQ3 files are read sequentially.

The sequence the records from SEQ1 (as specified in figure 8-4A) can appear in is as follows:

a. A record with a 1 in column one must be read first. Subsequent records with a 1 in column one must follow records with a 2 or 3 in column one.

b. A record with a 2 in column one must follow a record with a 1 in column one.

c. A record with a 3 in column one must follow a record with either a 1 or 2 in column one.

Records with a 2 or 3 in column one are optional, but at least one of these records must appear after each record with a 1 in column one.

Figure 8-4A is used in conjunction with table 8-1.

Burroughs REPORT PROGRAMING LANGUAGE

PROGRAM ID PROGRAMMER PAGE OF DATE

INPUT SPECIFICATIONS

PAGE 1 2

PROGRAM IDENTIFICATION 75 80

							RECORD IDENTIFICATION CODES														FIELD LOCATION							FIELD INDICATORS			
							1				2				3																
LINE	FORM TYPE	FILENAME	SEQUENCE	NUMBER	OPTION	RECORD IDENTIFYING INDICATOR	POSITION	NOT (N)	C/Z/D	CHARACTER	POSITION	NOT (N)	C/Z/D	CHARACTER	POSITION	NOT (N)	C/Z/D	CHARACTER	STACKER SELECT	PACKED	FROM	TO	DECIMAL POSITIONS	FIELD NAME (VARIABLE NAME)	CONTROL LEVEL	MATCHING FIELDS OR CHAINING FIELDS	FIELD RECORD RELATION	PLUS	MINUS	ZERO OR BLANK	NOT USED
3 5	6	7 14	15 16	17	18	19 20	21 24	25	26	27	28 31	32	33	34	35 38	39	40	41	42	43	44 47	48 51	52	53 58	59 60	61 62	63 64	65 66	67 68	69 70	71 74
0 1	I	SEQ1	01	1		01	1		C	1																					
0 2	I																				1	10	0	FIELD1							
0 3	I		02	1	Ø	02	1		C	2																					
0 4	I																				1	10	0	FIELD2							
0 5	I		03	1	Ø	03	1		C	3																					
0 6	I																				1	10	0	FIELD3							
0 7	I																														
0 8	I																														
0 9	I																														
1 0	I																														
1 1	I																														
1 2	I																														
1 3	I																														
1 4	I																														
1 5	I																														
	I																														
	I																														
	I																														

Printed in U.S America 1055860

G12064

Figure 8-4A. Input Specifications for DISKIN (SEQ1)

Table 8-1 lists one possible sequence in which records may appear from SEQ1. Asterisks denote out of sequence records. This table is used with figure 8-4A.

Table 8-1. SEQ1 Data File

Relative Record Number	Data in Column One
1	1
2	2
3	3
4	1
5	1*
6	2
7	1
8	3
9	2**
10	1
11	2
12	2***
13	3

* Expects a record with a 2 or 3 in column one.

** Expects a record with a 1 in column one.

*** Expects a record with a 1 or 3 in column one.

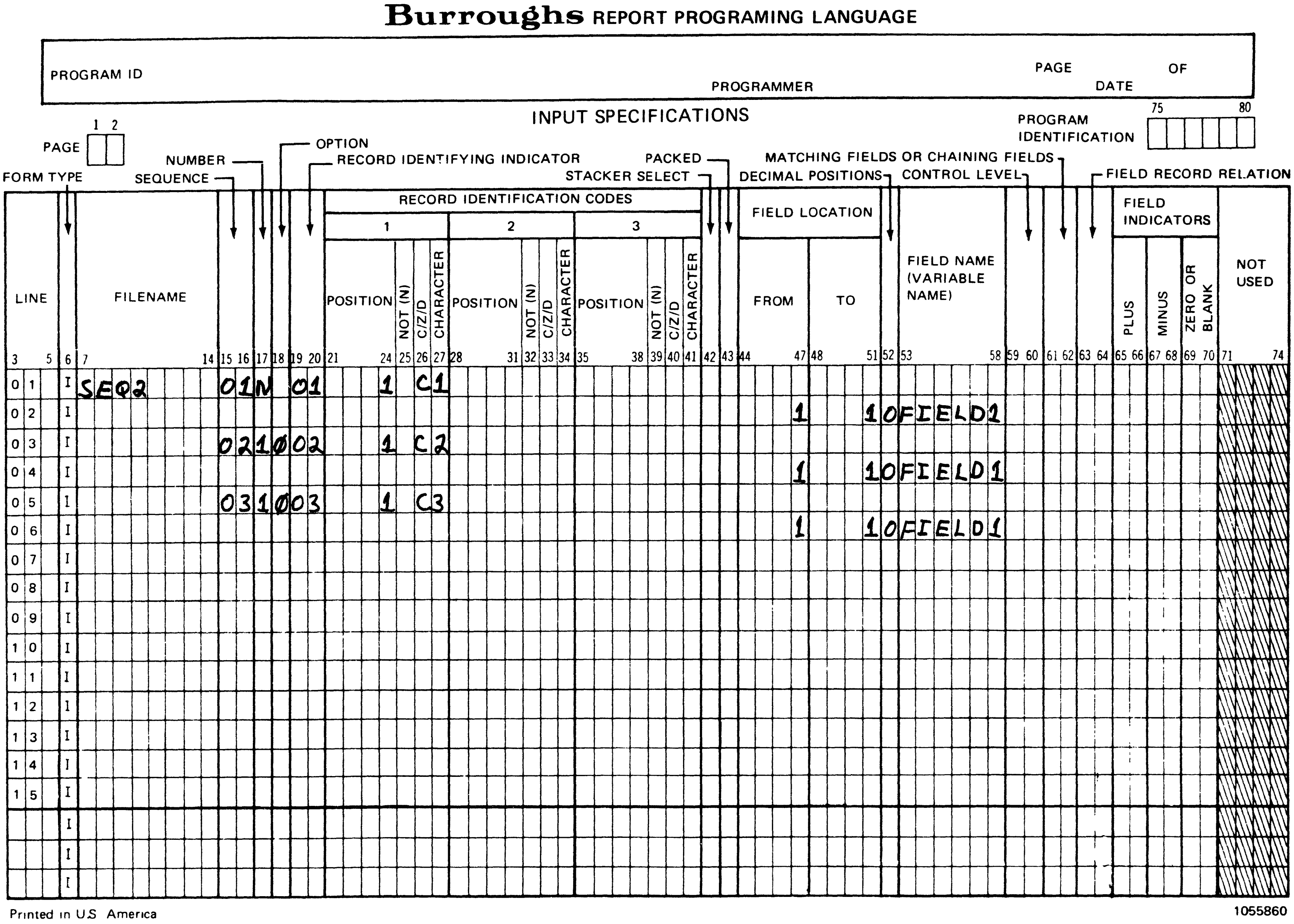
Figure 8-4B is used in conjunction with table 8-2.

The sequence the records from SEQ2 (as specified in figure 8-4B) can appear in is as follows:

a. A record with a 1 in column one must be read first. Subsequent records with a 1 in column one must follow records with either a 1, 2, or 3 in column one.

b. A record with a 2 in column one must follow a record with a 1 in column one.

c. A record with a 3 in column one must follow a record with a 1 or 2 in column one.



Burroughs REPORT PROGRAMING LANGUAGE

PROGRAM ID PROGRAMMER PAGE OF DATE

INPUT SPECIFICATIONS

PAGE 1 2

PROGRAM IDENTIFICATION 75 80

LINE 3-5	FORM TYPE 6	FILENAME 7-14	SEQUENCE 15-16	NUMBER 17	OPTION 18	RECORD IDENTIFYING INDICATOR 19-20	RECORD IDENTIFICATION CODES 1: POSITION 21-24	NOT (N) 25	C/Z/D 26	CHARACTER 27	2: POSITION 28-31	NOT (N) 32	C/Z/D 33	CHARACTER 34	3: POSITION 35-38	NOT (N) 39	C/Z/D 40	CHARACTER 41	STACKER SELECT 42	PACKED 43	FIELD LOCATION FROM 44-47	TO 48-51	DECIMAL POSITIONS 52	FIELD NAME (VARIABLE NAME) 53-58	CONTROL LEVEL 59-60	MATCHING FIELDS OR CHAINING FIELDS 61-62	FIELD RECORD RELATION 63-64	FIELD INDICATORS PLUS 65-66	MINUS 67-68	ZERO OR BLANK 69-70	NOT USED 71-74
01	I	SEQ2	01	N		01	1		C	1																					
02	I																				1	1	0	FIELD1							
03	I		02	1	Ø	02	1		C	2																					
04	I																				1	1	0	FIELD1							
05	I		03	1	Ø	03	1		C	3																					
06	I																				1	1	0	FIELD1							
07	I																														
08	I																														
09	I																														
10	I																														
11	I																														
12	I																														
13	I																														
14	I																														
15	I																														
	I																														
	I																														
	I																														

Printed in U.S. America

1055860

G12065

Figure 8-4B. Input Specifications for DISKIN (SEQ2)

Table 8-2 lists one possible sequence in which records may appear from SEQ2. Asterisks denote out of sequence records. This table is used with figure 8-4B.

Table 8-2. SEQ2 Data File

Relative Record Number	Data in Column One
1	1
2	2
3	2*
4	3
5	1
6	3
7	2**
8	1
9	1
10	1

* Expects a record with a 1 or 3 in column one.

** Expects a record with a 1 in column one.

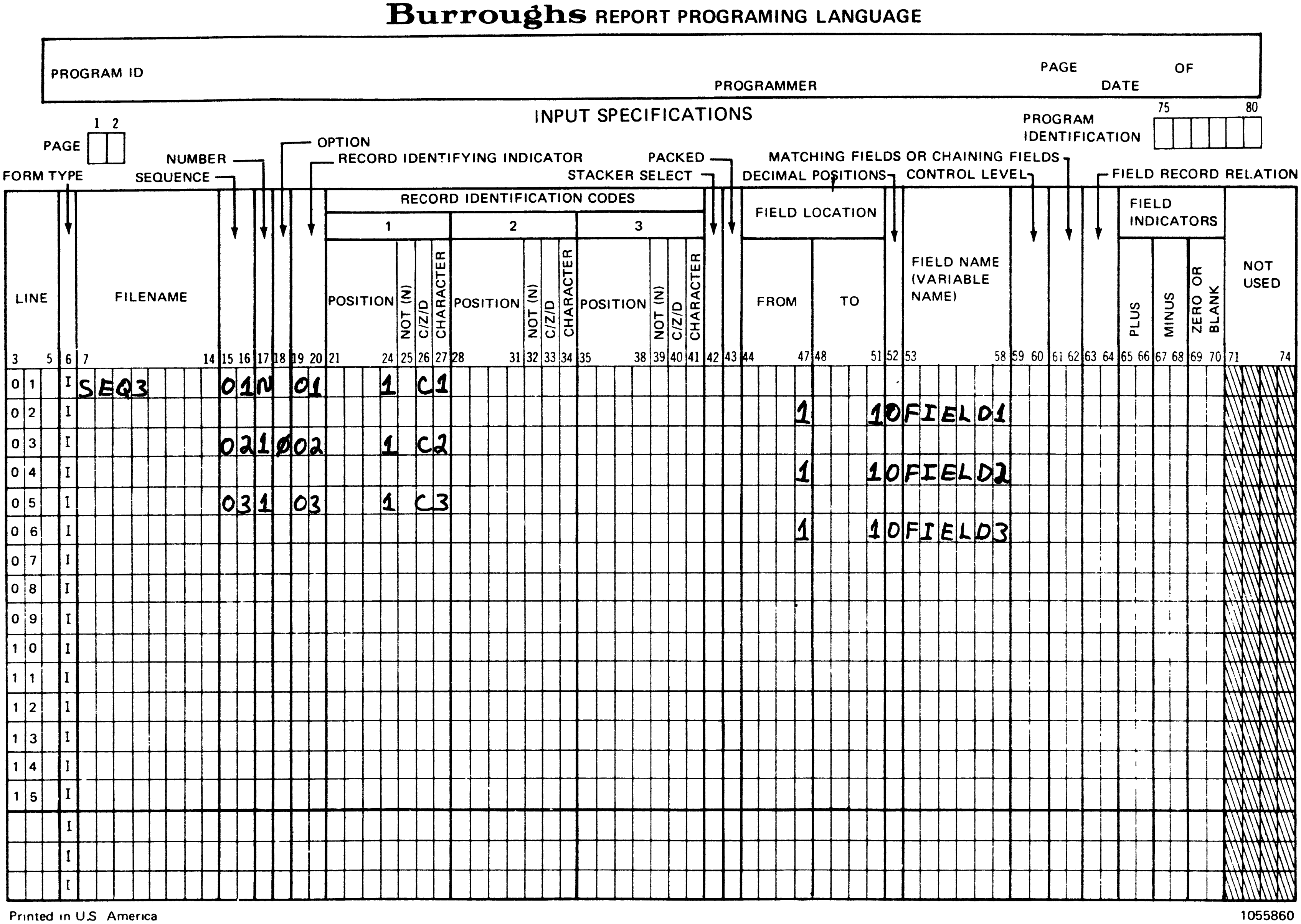
Figure 8-4C is used in conjunction with table 8-3.

The sequence the records from SEQ3 (as specified in figure 8-4C) can appear in is as follows:

a. A record with a 1 in column one must be read first. Subsequent records with a 1 in column one must follow records with a 1, 2, or 3 in column one.

b. A record with a 2 in column one must follow a record with a 1 in column one.

c. A record with a 3 in column one must follow a record with a 1 or 2 in column one.



Burroughs REPORT PROGRAMING LANGUAGE

PROGRAM ID PROGRAMMER PAGE OF DATE

INPUT SPECIFICATIONS

PAGE 1 2 □□

PROGRAM IDENTIFICATION 75 80 □□□□□□

LINE	FORM TYPE	FILENAME	SEQUENCE	NUMBER	OPTION	RECORD IDENTIFYING INDICATOR	RECORD IDENTIFICATION CODES 1				RECORD IDENTIFICATION CODES 2				RECORD IDENTIFICATION CODES 3				STACKER SELECT	PACKED	FIELD LOCATION		DECIMAL POSITIONS	FIELD NAME (VARIABLE NAME)	CONTROL LEVEL	MATCHING FIELDS OR CHAINING FIELDS	FIELD RECORD RELATION	FIELD INDICATORS			NOT USED
							POSITION	NOT (N)	C/Z/D	CHARACTER	POSITION	NOT (N)	C/Z/D	CHARACTER	POSITION	NOT (N)	C/Z/D	CHARACTER			FROM	TO						PLUS	MINUS	ZERO OR BLANK	
3 5	6	7 14	15 16	17	18	19 20	21 24	25	26	27	28 31	32	33	34	35 38	39	40	41	42	43	44 47	48 51	52	53 58	59 60	61 62	63 64	65 66	67 68	69 70	71 74
0 1	I	SEQ3	01	N		01	1		C	1																					
0 2	I																				1	1	0	FIELD1							
0 3	I		02	1	Ø	02	1		C	2																					
0 4	I																				1	1	0	FIELD2							
0 5	I		03	1		03	1		C	3																					
0 6	I																				1	1	0	FIELD3							
0 7	I																														
0 8	I																														
0 9	I																														
1 0	I																														
1 1	I																														
1 2	I																														
1 3	I																														
1 4	I																														
1 5	I																														
	I																														
	I																														
	I																														

Printed in U.S. America

1055860

G12066

Figure 8-4C. Input Specifications for DISKIN (SEQ3)

Table 8-3 lists one possible sequence in which records may appear from SEQ3. Asterisks denote out of sequence records. This table is used with figure 8-4C.

Table 8-3. SEQ3 Data File

Relative Record Number	Data in Column One
1	1
2	1
3	2
4	2*
5	3
6	1
7	3
8	3**
9	1
10	3
11	1
12	2
13	1***

* Expects a record with a 3 in column one.

** Expects a record with a 1 in column one.

*** Expects a record with a 3 in column one.

19-20 RECORD IDENTIFYING INDICATOR

This field may be used for the following purposes:

a. To assign an indicator to each record type.

b. To indicate look-ahead fields.

c. To specify spread cards.

Entry	Definition
01-99	Record identifying indicator.
L1-L9	Control level indicator.
LR	Last record indicator.
HO-H9	Halt indicator.
TR	Spread cards.
**	Look-ahead field.

If this entry is blank a warning will be emitted. The various indicators are defined in the following paragraphs.

Record Identifying Indicator 01-99

Each input file may contain different types of records requiring different operations. Record identifying indicators are used to signal to the rest of the program cycle the type of record just read. When a specific record type is selected for processing, its corresponding identifying indicator is turned ON. This indicator remains ON for the rest of the current program cycle and may be used to condition various calculation and output operations, as desired. All record identifying indicators are turned off at the same point in the program cycle. Each record identifying indicator should be unique and only one record identifying indicator may be ON for any one file at any one time. However, there may be more than one record identifying indicator ON at any one time, each one associated with a different file (i.e., through CHAIN or READ operations).

Record identifying indicators do not have to be assigned in any order. If the same operations are to be performed on different record types, the same indicator may be assigned to more than one type.

Record identifying indicators are not allowed in an AND line, but indicators may be specified for every record type that requires special processing in an OR relationship.

Control Level Indicator L1-L9

A control level indicator is used instead of a record identifying indicator when a record type, rather than a control field, signals the start of a new control group. This use of the control level indicator does not cause the lower control levels to turn ON. Refer to the CONTROL LEVEL field for a complete description of control level indicators.

Last Record Indicator LR

The last record indicator is used instead of a record identifying indicator when a record type, rather than automatic End-of-File, signals the end of processing. Final total operations are conditioned by this indicator.

Halt Indicator H0-H9

A halt indicator is used instead of a record identifying indicator when the occurrence of a specific record type denotes a desired condition requiring a program halt.

Look-Ahead Field **

Look-ahead fields are specified by placing asterisks in columns 19 and 20. All fields named in columns 53 through 58 on the specifications lines following the look-ahead specifications are look-ahead fields.

Look-ahead fields can be used to:

a. Determine when the last record in a particular control group is being processed.

b. Extend the use of the matching record function.

Rules for Look-Ahead Fields.

The following rules must be observed regarding look-ahead fields:

a. Look-ahead fields can be specified for input, update, or combined files that are primary or secondary files, regardless of whether or not they are processed by record address files.

b. Look-ahead fields cannot be specified for combined files, demand files, or for files that specify spread card records.

c. One set of look-ahead fields can be specified per file, and the field descriptions apply to all records in that file.

d. Look-ahead fields can not be used as result fields in calculation operations.

e. The name given to a look-ahead field must not occur on any other Input or Extension Specification.

f. If the look-ahead field occurs on an Output Specification, blank after must not be specified.

g. When a program needs to access information before and after the record is selected for processing, the field must be described twice with different names (once as a look-ahead field and once in the normal way). Refer to figure 8-5.

h. The ** line cannot follow a record type description that has a numeric sequence entry.

i. Columns 17-18 and 21-74 must be left blank.

j. Any combination of alphabetic characters or blanks can be entered in columns 15-16 except ND and Rɓ (ɓ = blank).

k. The fields themselves are described on the lines following the ** line. When fields are described, columns 7-42 and 59-74 must be blank.

l. When the last record of a file is being processed, any look-ahead fields for that file contain all "9's" (signed numeric or alphanumeric according to field type).

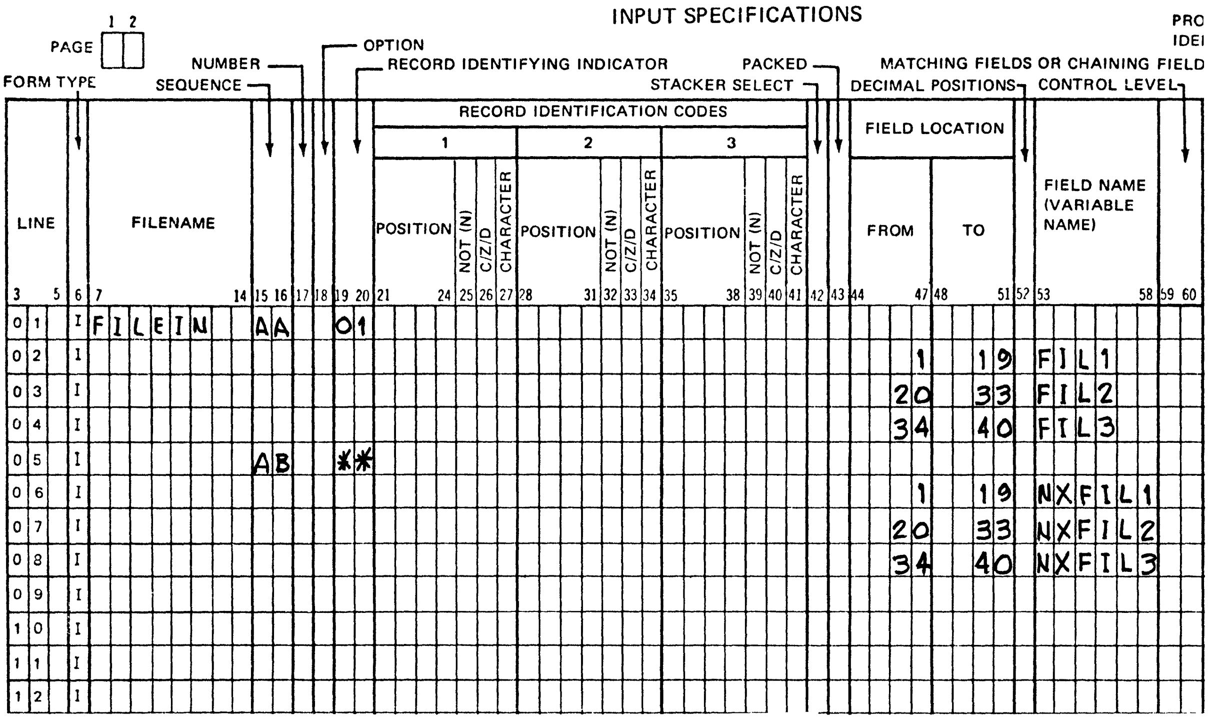
Figure 8-5 provides an example of how to code look-ahead fields.

Use of the Look-Ahead Feature with Input Files.

When the look-ahead feature is used with an input file, the look-ahead field allows the program to access information in a field of the next record that is available for processing. Thus, the program can use information from the look-ahead field to condition certain operations prior to the time the record is normally available for processing.

Use of the Look-Ahead Feature with Update and Combined Files.

When the look-ahead feature is used with an update or combined file, the look-ahead field usually references the current record being processed. The look-ahead file references only the next record in the



INPUT SPECIFICATIONS

LINE (3-5)	FORM TYPE (6)	FILENAME (7-14)	SEQUENCE (15-16)	NUMBER (17)	OPTION (18)	RECORD IDENTIFYING INDICATOR (19-20)	RECORD IDENTIFICATION CODES (21-41)	PACKED (43)	FIELD LOCATION FROM (44-47)	FIELD LOCATION TO (48-51)	DECIMAL POSITIONS (52)	FIELD NAME (VARIABLE NAME) (53-58)	CONTROL LEVEL (59-60)
01	I	FILEIN	AA			01							
02	I								1	19		FIL1	
03	I								20	33		FIL2	
04	I								34	40		FIL3	
05	I		AB			**							
06	I								1	19		NXFIL1	
07	I								20	33		NXFIL2	
08	I								34	40		NXFIL3	
09	I												
10	I												
11	I												
12	I												

G14039

Figure 8-5. Coding Look-Ahead Fields

file when the current record was not read from the file. Therefore, when an update or combined file is the only file being read, the look-ahead field always references the current record.

Processing Two Input Files Using the Look-Ahead Feature.

Figure 8-6 shows processing of records from two input files, one of which is a primary file and the other a secondary file. All primary records are processed before any secondary records are available. Therefore, to use data from the secondary file while processing the primary file record it is necessary to use the look-ahead feature.

Record Processed	Look-Ahead Records Available
P1	P2 S1
S1	P2 S2
S2	P2 S3
S3	P2 S4
P2	P3 S4
S4	P3 S5
S5	P3 S6
S6	P3 S7
P3	P4 S7
P4	P5 S7
S7	P5 S8
P5	P6 S8

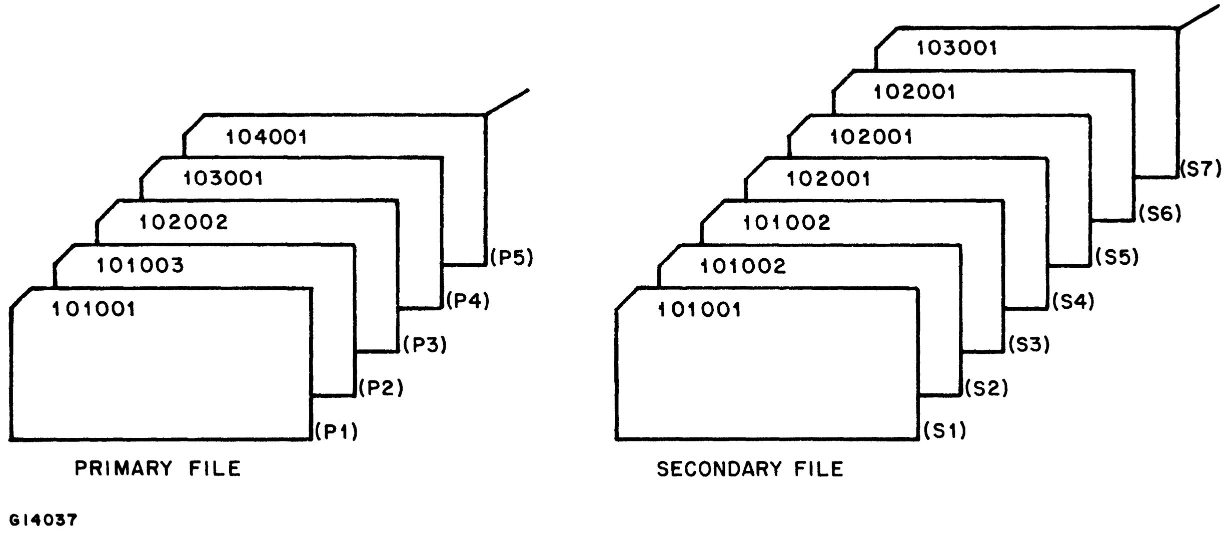


Figure 8-6. Records Available for Look-Ahead: Two Input Files

Processing an Update File and a Secondary File Using Look-Ahead.

Figure 8-7 shows processing records from an update file and a secondary input file.

Record Processed	Look-Ahead Records Available
U1	U1 S1
S1	U2 S2
S2	U2 S3
S3	U2 S4
U2	U2 S4
S4	U3 S5
S5	U3 S6
S6	U3 S7
U3	U3 S7
U4	U4 S7
S7	U5 S8

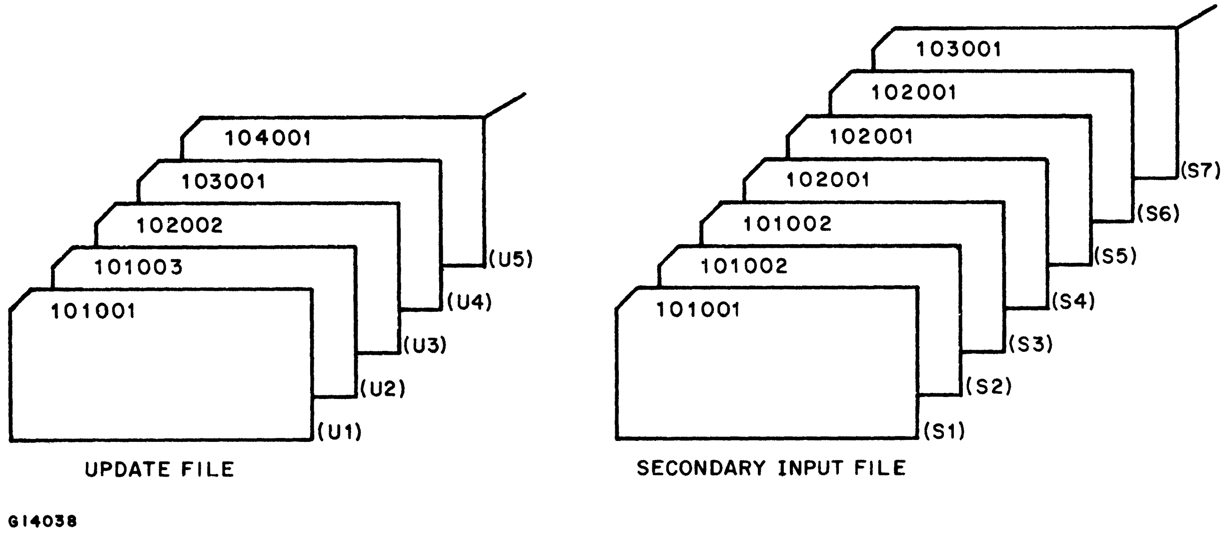


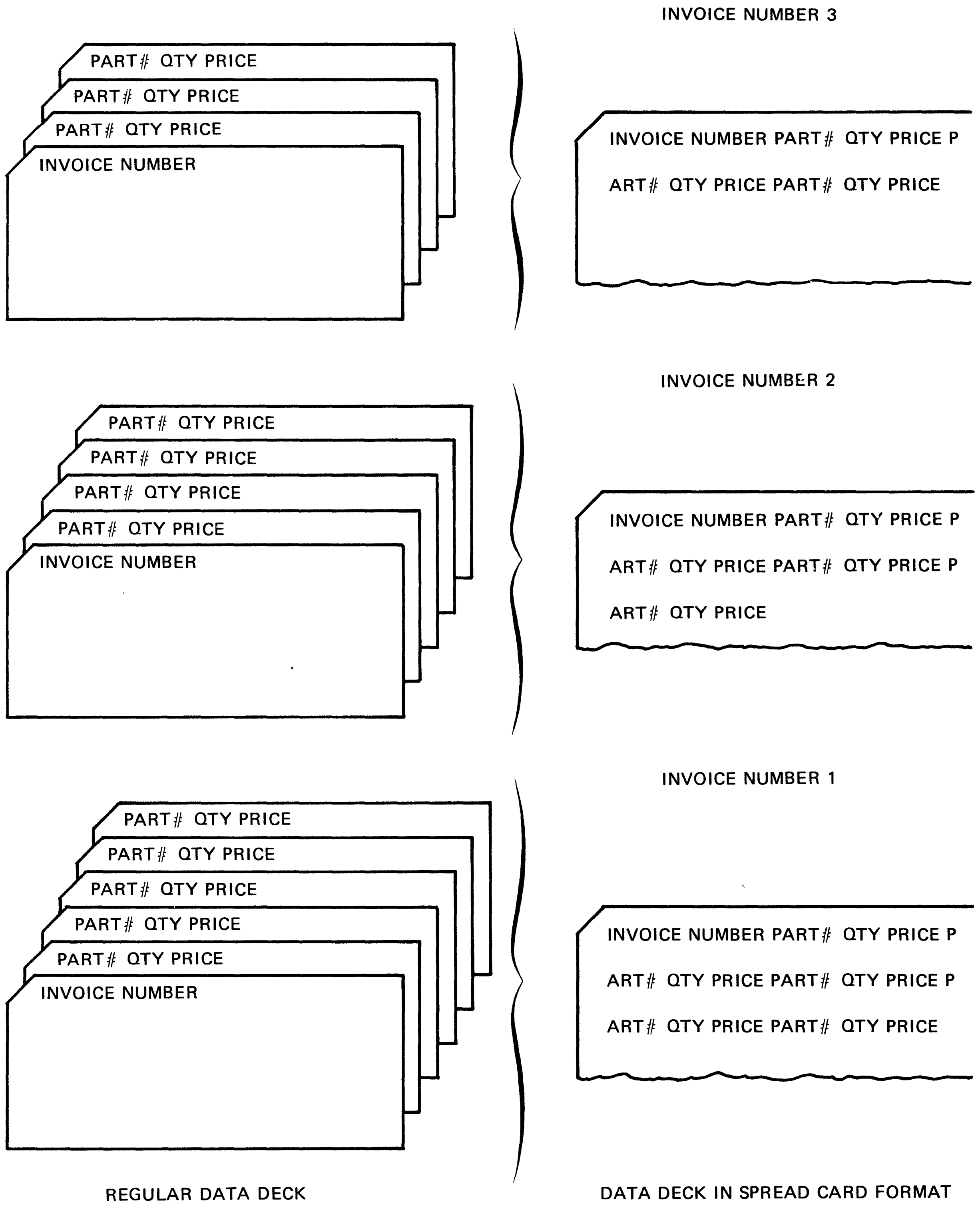
Figure 8-7. Records Available for Look-Ahead: One Update File, One Input File

Spread Card Indicator TR

Spread card records can only be used with primary or secondary input card files that do not have look-ahead fields. A spread card record consists of a header field and one or more associated trailer fields, thereby permitting the storage of more data on each card. For example, a file for an order filling program can contain an invoice number, part number, quantity, and price.

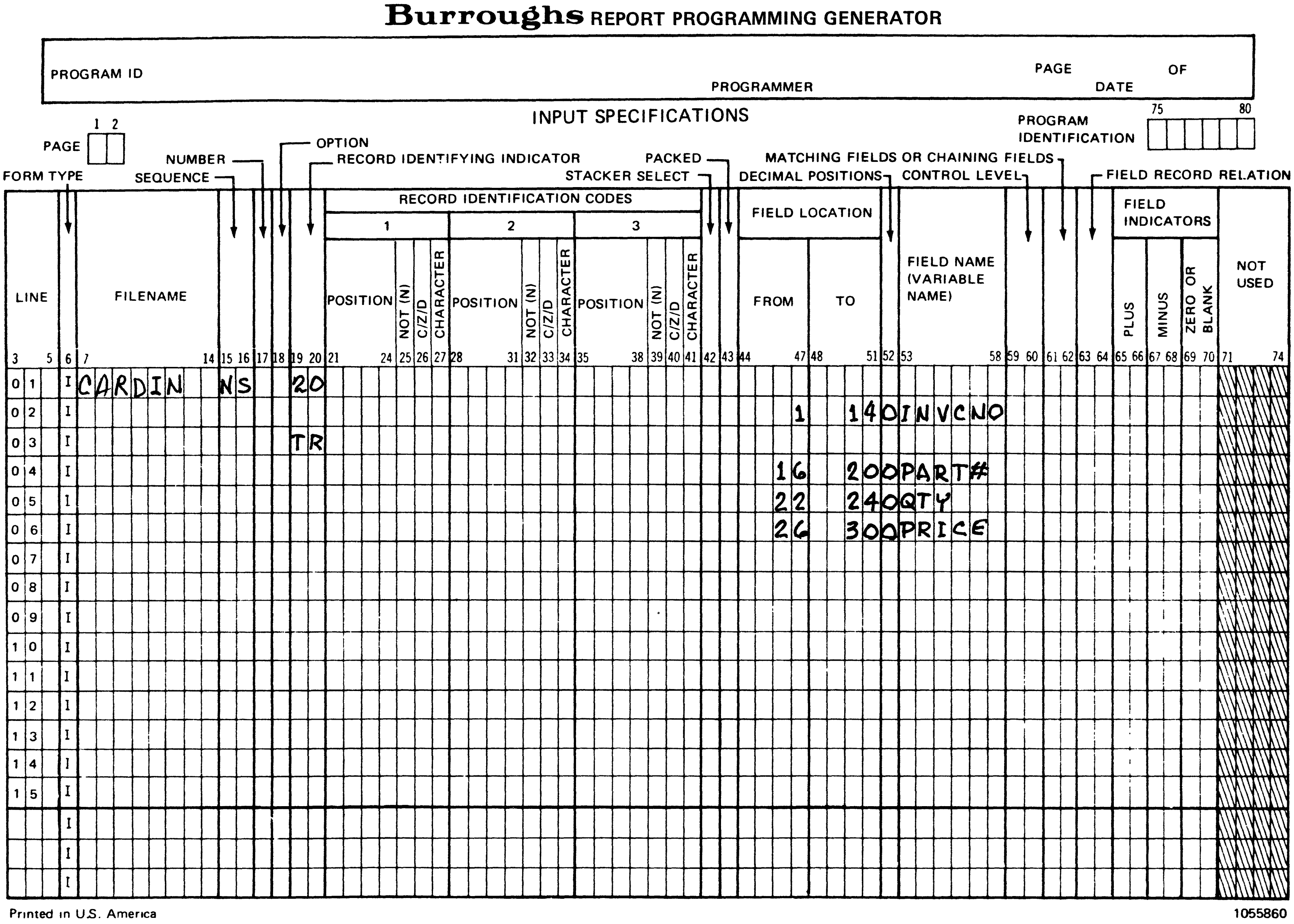
Figure 8-8 illustrates the difference between a regular data deck and a data deck in spread card format. Note that six cards are required for invoice number 1 in regular format, but only one spread card is required for the identical information. For each spread card, the header field contains INVOICE NUMBER and the trailer fields contain the set PART# QTY PRICE.

Figure 8-9 shows the Input Specifications for the spread card data deck shown in figure 8-8.



G12040

Figure 8-8. Comparison of Regular and Spread Card Data Decks



Burroughs REPORT PROGRAMMING GENERATOR

PROGRAM ID | PROGRAMMER | PAGE OF | DATE

INPUT SPECIFICATIONS

PAGE 1 2

PROGRAM IDENTIFICATION 75 80

LINE	FORM TYPE	FILENAME	SEQUENCE	NUMBER	OPTION	RECORD IDENTIFYING INDICATOR	RECORD IDENTIFICATION CODES 1 POSITION	NOT (N)	C/Z/D	CHARACTER	2 POSITION	NOT (N)	C/Z/D	CHARACTER	3 POSITION	NOT (N)	C/Z/D	CHARACTER	STACKER SELECT	PACKED	FIELD LOCATION FROM	TO	DECIMAL POSITIONS	FIELD NAME (VARIABLE NAME)	CONTROL LEVEL	MATCHING FIELDS OR CHAINING FIELDS	FIELD RECORD RELATION	FIELD INDICATORS PLUS	MINUS	ZERO OR BLANK	NOT USED
3 5	6	7 14	15 16	17	18	19 20	21 24	25	26	27	28 31	32	33	34	35 38	39	40	41	42	43	44 47	48 51	52	53 58	59 60	61 62	63 64	65 66	67 68	69 70	71 74
01	I	CARDIN	NS			20																									
02	I																				1	14	0	INVCNO							
03	I					TR																									
04	I																				16	20	0	PART#							
05	I																				22	24	0	QTY							
06	I																				26	30	0	PRICE							
07	I																														
08	I																														
09	I																														
10	I																														
11	I																														
12	I																														
13	I																														
14	I																														
15	I																														
	I																														
	I																														
	I																														

Printed in U.S. America

1055860

Figure 8-9. Coding for Spread Cards

Rules for Spread Cards.

The following rules must be observed regarding spread cards.

a. Spread cards can be specified for primary or secondary input card files only.

b. Look-ahead fields are not allowed.

c. If sequencing is specified in columns 15-16, then an "N" must be entered in column 17.

d. The header field specification is optional, but when specified is coded as follows:

   1. Describe the header fields on separate specifications lines immediately following the file and record type entries.

   2. Describe only the header fields that are used within the program.

e. Enter "TR" in columns 19-20 to specify that the trailer fields are described on the specification lines that follow. All other entries on the "TR" line must be left blank.

f. It is necessary to describe only the trailer fields that are to be used in the program. However, the fields that specify the beginning and ending position of each trailer portion must be described.

g. All trailer fields must be of the same length and contain the same fields. Therefore, it is necessary to describe only the first trailer portion.

h. Trailer field entries are made in columns 44-58, and all other columns are left blank.

Processing Spread Cards.

Spread cards are processed as follows:

a. The header portion of the spread card record and one trailer portion are processed each program cycle and are treated as one logical record.

b. The process described in step a continues until:

   1. All trailer portions of a record have been processed, or

   2. A trailer portion of a record is encountered whose fields are all blank.

When either of the described conditions occurs, the next spread card is read and processing continues.

21-41 RECORD IDENTIFICATION CODES

When more than one record type is used within a file, only one record type will be selected for processing during each program cycle. The record identifying indicator for that record type will be turned ON when it is selected and will remain ON for the rest of the current program cycle.

In order to identify the various record types to the program for the purpose of record selection, each record type must have a unique code assigned to it. This code consists of a certain character or combination of characters occurring in certain positions of the record. The record identification codes are checked according to the sequence in which they were specified, and that checking is terminated when a condition is encountered that causes a record identifying indicator to turn ON. For that reason, a record within a file that does not contain record identification codes should be specified last.

The RECORD IDENTIFICATION CODES field is used to describe the code for each record type. If all records are to be processed alike regardless of their type, or if all records are of the same type, this field should be left blank.

This field is subdivided into three subfields of seven columns each, allowing up to three code characters to be described on one line. The three subfields are taken to be in an AND relationship, and any that are not needed to specify code characters should be left blank. Each of the three subfields is divided into four entries, and coding is the same for all three subfields. The subfields are discussed in the following paragraphs.

21-24,28-31,35-38 POSITION

These fields are used to give the locations in the record of each character in the record identification code. Entries must be numeric, between one and the record length specified, inclusive, and right-justified (leading zeros are optional).

25,32,39 NOT

These fields are used to indicate whether the specified character must be present in the record at the designated position. Valid codes are:

Entry	Definition
Blank	Character must be present in the location specified by the POSITION entry.
N	Character must not be present in the location specified by the POSITION entry.

26,33,40 C/Z/D

These fields are used to indicate which portion of the character specified in columns 27, 34, and 41 should be used for comparison: the zone, the digit, or the entire character. Valid entries are:

Entry	Definition
C	Entire character.
Z	Zone portion.
D	Numeric (digit) portion.

Every alphabetic character, numeric character, or special character is represented by a different combination of punches in the 80-column or 96-column cards. Each character punched on the card is composed of two parts, a zone portion and a digit portion. Even after a character has been read into the machine, it is still composed of these two parts (see appendixes A and B).

A character is represented in the computer by eight bits. The first four bits comprise the zone portion and the last four bits comprise the digit portion. The configuration of these bits is set in the binary equivalent of their hexadecimal value. In appendix B, the hexadecimal value of the character A is given as C1. Therefore, the corresponding bit configuration for A would be:

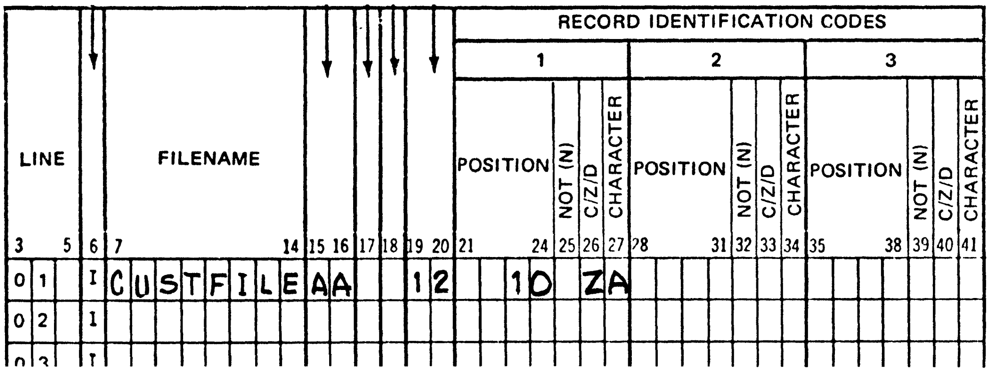
1 1 0 0 0 0 0 1 .

Since the character is represented by 12 punch positions on an 80-column card and six punch positions on a 96-column card, translation must take place so that it can be represented by eight bits in storage. This is an automatic function. As a result of it, however, the way characters are represented in the machine and the way they appear on the punched card are not always identical. Not all characters that have the same zone punched in the card have identical zone structures in the machine. For example, character $ has the same zone punch in the card as character K. However, they do not have the same zone representation in the machine.

Whenever just the zone or just the digit portions of characters are used in specific functions, such as sequencing, testing, or identifying records, the exact structure of the characters in the machine must be known. For example, when identifying a record type on the basis of the zone portion of the character D, notice that several characters have the same zone structure as the letter D. If a card with the record identifying code of E is read, it is still considered to be a D type record because the zone of character E is the same as the zone of character D.

The zone of the plus (+) character is treated like the zone of the characters A through I, and the zone of the minus (-) character is treated like the zone of the characters J through R, irrespective of the internal codes actually used for plus and minus.

In figure 8-10, only the records of customers whose last names begin with the letters A through I will be processed since the zone portion of A is the same as for the characters B through I. The first letter of each last name begins in column 10.

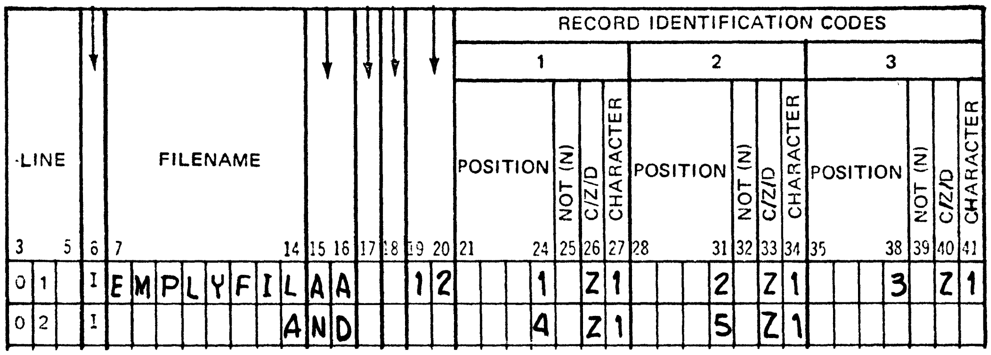


LINE		FILENAME					RECORD IDENTIFICATION CODES 1				2				3			
							POSITION	NOT (N)	C/Z/D	CHARACTER	POSITION	NOT (N)	C/Z/D	CHARACTER	POSITION	NOT (N)	C/Z/D	CHARACTER
3 5	6	7 14	15 16	17	18	19 20	21 24	25	26	27	28 31	32	33	34	35 38	39	40	41
01	I	CUSTFILE	AA			12	10		Z	A								
02	I																	
03	I																	

G14040

Figure 8-10. C/Z/D Coding Example 1

In figure 8-11, 5-digit employee numbers are checked to see that all 5 digits are numeric. The zone for all numeric characters is the same.

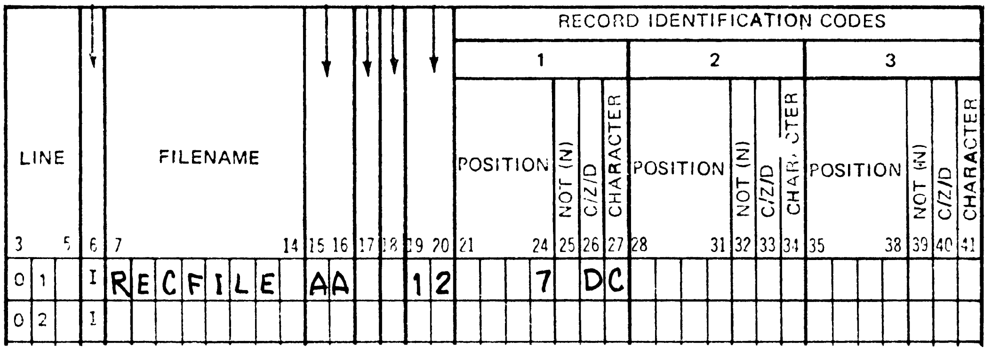


LINE		FILENAME					RECORD IDENTIFICATION CODES 1				2				3			
							POSITION	NOT (N)	C/Z/D	CHARACTER	POSITION	NOT (N)	C/Z/D	CHARACTER	POSITION	NOT (N)	C/Z/D	CHARACTER
3 5	6	7 14	15 16	17	18	19 20	21 24	25	26	27	28 31	32	33	34	35 38	39	40	41
01	I	EMPLYFIL	AA			12	1		Z	1	2		Z	1	3		Z	1
02	I	A	ND				4		Z	1	5		Z	1				

G14041

Figure 8-11. C/Z/D Coding Example 2

In figure 8-12, only persons whose names start with C, L, or T will be processed, since the digit portion of the characters L and T is the same as C.



							RECORD IDENTIFICATION CODES											
							1				2				3			
LINE		FILENAME					POSITION	NOT (N)	C/Z/D	CHARACTER	POSITION	NOT (N)	C/Z/D	CHARACTER	POSITION	NOT (N)	C/Z/D	CHARACTER
3 5	6	7 14	15 16	17	18	19 20	21 24	25	26	27	28 31	32	33	34	35 38	39	40	41
01	I	RECFILE	AA			12	7		D	C								
02	I																	

G14042

Figure 8-12. C/Z/D Coding Example 3

NOTE

If packed decimal format is specified, the zone portion and the digit portion of each byte contains a numeric character. Therefore, the user must know where the numeric character he is referencing is located, i.e., either the zone portion or the digit portion.

27,34,41

CHARACTER

Any valid EBCDIC character may be used to identify the input record type. These fields are used to specify the character to be used for comparison as part of the identification code.

If none of the RECORD IDENTIFICATION codes are found on a record, the program will halt. The system operator may request resumption of the program, at which time it will ignore the record in error and read the next record from the same file (see appendix E).

Additional Record Identification Codes

If necessary, more than three record identification codes can be specified by entering "AND" in columns 14-16 of the next line. Columns 21-41 should then be coded as previously described.

It is also possible to specify an "or" relationship between records by entering "OR" in columns 14-15 of the next line. Columns 21-40 should then be coded as previously described. The capability of specifying more than three record identification codes can be used when:

a. Multiple record types have the same fields but the fields are in different positions.

b. Multiple record types have the same field descriptions.

c. A field occurs in one type of record but not in another.

d. Various combinations of a, b, and c.

42 STACKER SELECT

This field is used to indicate the stacker into which the input card is to be placed after being read. Only card input or combined files may be stacker selected. Input files may be stacker selected only in the Input Specifications; combined files may be stacker selected either in the Input or Output-Format Specifications. If a combined file is stacker selected in both the Input and Output-Format Specifications, the Output-Format Stacker Specification overrides the stacker specified in the Input Specifications. Valid entries for this field are:

Entry	Definition
Blank	Cards automatically go to default stacker.
Numeric Entry (1-6)	Cards go into the stacker specified.

Card types identified by OR lines may be stacker selected for a special stacker by an entry in this field; however, if the STACKER SELECT field entry is left blank, the card type selected by the OR line will go to the default stacker. AND lines may not have an entry in STACKER SELECT.

At execution time, any record types specifying a stacker number higher than that available on the device being used will go to the default stacker.

This entry should be left blank for input files with multiple I/O areas, otherwise a warning is emitted that the results may not be those the user intended.

43 PACKED OR BINARY FIELD

This field is used to specify that a numeric field is in packed decimal format or binary format. Valid entries are:

Entry	Description
Blank	Field is in unpacked decimal format or is alphanumeric.
P	Field is in packed decimal format.
B	Field is in binary format.

When the input field named in columns 53-58 is in packed decimal format, column 43 must contain a "P". When the input field is in binary format, column 43 must contain a "B".

Whether in packed decimal, or unpacked decimal format, the data may be signed at the most significant or least significant position as specified by means of column 17 of the Control Card or by use of the dollar option RSIGN. The object program automatically converts all numeric data internally during execution with the sign at the most significant position.

UNPACKED DECIMAL FORMAT

Unpacked decimal format means that each byte of storage contains one character. Each byte is divided into a 4-bit zone portion and a 4-bit digit portion. The format for unpacked decimal (left signed) is shown in figure 8-13.

	1		9		7		6
Positive Sign	0001		1001		0111		0110
zone	digit	zone	digit	zone	digit	zone	digit

G12042

Figure 8-13. Unpacked Format for Decimal Number 1976 (Left Signed)

The format for unpacked decimal (right signed) is shown in figure 8-14:

	1		9		7		6
	0001		1001		0111	Positive Sign	0110
zone	digit	zone	digit	zone	digit	zone	digit

G12043

Figure 8-14. Unpacked Format for Decimal Number 1976 (Right Signed)

When processing numeric data in the unpacked format, the zone portion is included for each digit in the number, but only the zone in the rightmost (right signed) or leftmost (left signed) digit serves as the sign (determines if the number is positive or negative).

PACKED DECIMAL FORMAT

Packed decimal format means that each byte of storage can contain two decimal numbers. The sign is included for the number, but the zone portion is omitted for each digit in the number. Each byte consists of two 4-bit digit positions. Within each number, either the leftmost or rightmost byte contains a digit and the sign, depending on whether left or right signs are desired.

When describing external data in RPG, the smallest element is a byte or character. Packed data is always character-aligned on external storage media such as magnetic tape or disk.

Unpacked decimal format means that each byte of storage contains one character. Each byte is divided into a 4-bit zone portion and a 4-bit digit portion:

zone	digit	zone	digit	zone	digit	zone	digit

Each digit portion holds one digit of the number. On conversion to packed decimal format, the zone portions are dropped, except for the sign position.

Packed data input always causes the field that is to contain the data to be of an odd size, since even-numbered digits are padded with a zero. All digits except the sign digit are considered data.

The following example illustrates the format for packed decimal data:

S	O	d_n ...	d_2	d_1	left signed output
O	d_n	...d_2	d_1	S	right signed output

S = sign

O = digit zero - (even number digits are zero padded)

$d_n \ldots d_1$ = digits of the field.

Table 8-4 shows the corresponding packed field length in bytes for unpacked fields of one byte through 15 bytes in length.

BINARY FORMAT

Binary format means that two bytes of storage can contain up to four decimal numbers, and that nine decimal numbers can be contained in four bytes of storage.

Rules for Binary Format

The following rules must be observed when binary format is used:

a. Each binary field must be either two bytes or four bytes in length.

Table 8-4. Packed Equivalents for Unpacked Fields

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

b. Binary fields cannot be used as control fields or matching fields.

c. Each two-byte field consists of a 1-bit sign followed by a 15-bit numeric value. The numeric value must be within the range -9,999 and +9,999 inclusive.

d. Each four-byte field consists of a 1-bit sign and a 31-bit numeric value. The numeric value must be within the range -999,999,999 and +999,999,999 inclusive.

e. When a two-byte or four-byte binary field is assigned a value greater than it can store, the result is truncation of the leftmost (high order) digits of the decimal number.

f. For binary fields, the leftmost bit is the sign bit and is used to indicate whether the number is positive or negative. If the sign bit is a 0 (OFF) the number is positive. If the sign bit is a 1 (ON) the number is negative. Figure 8-15 shows the format of a two-byte binary field.

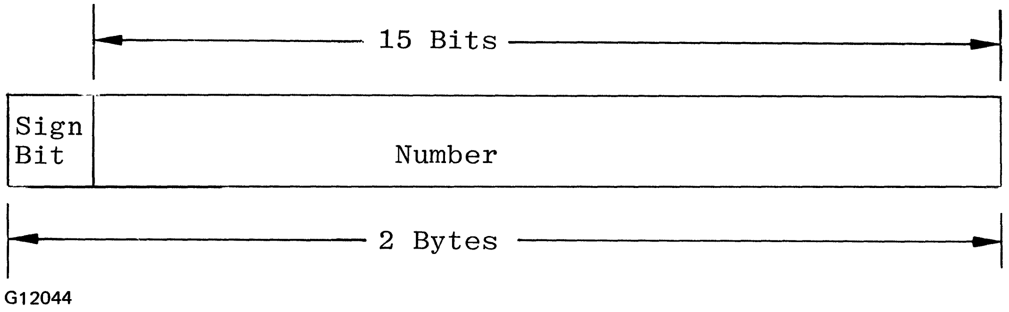


Figure 8-15. Format of Two-Byte Binary Field

g. The decimal value of a binary field can be determined by adding the decimal equivalents of the binary bits that are ON. The sign bit is not included in the addition. Figure 8-16 shows how the decimal equivalents of binary bits are obtained.

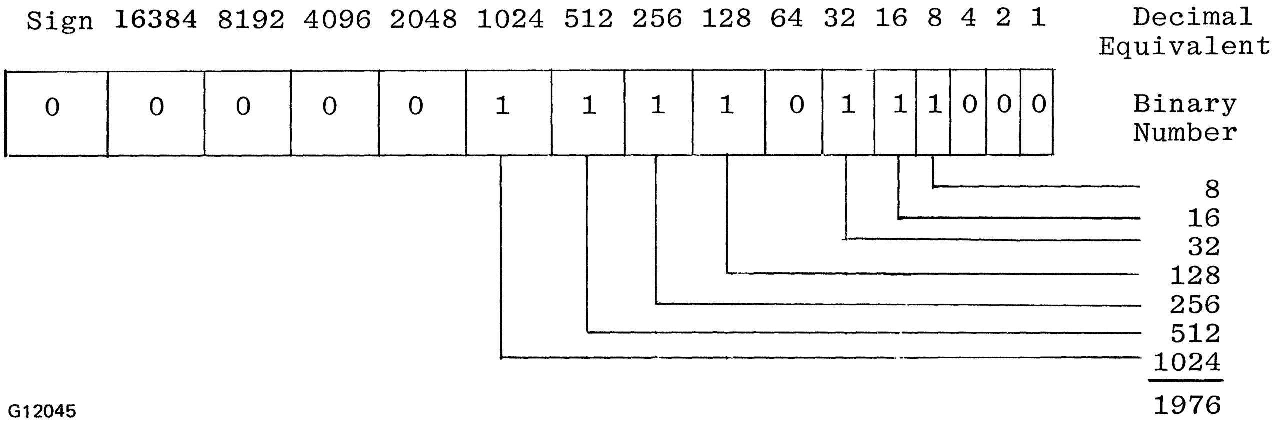


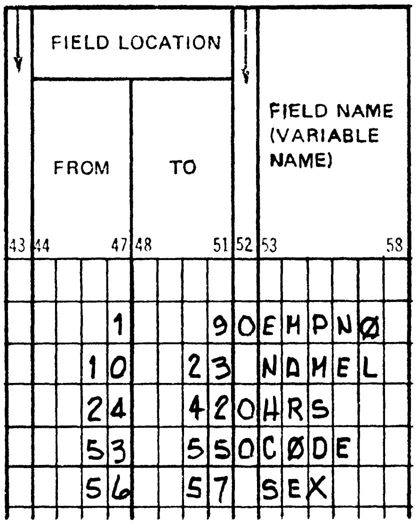
Figure 8-16. Binary Representation of the Decimal Number 1976

44-51 FIELD LOCATION

This field is used to describe the location of data fields within a record, and is divided into two subfields that specify the beginning (FROM) and ending (TO) positions of the data field. A field of only one character will have the same position number entered in both subfields. Both entries must lie between 1 and the RECORD LENGTH. The TO entry must be greater than or equal to the FROM entry. Entries in the FROM and TO subfields must be right-justified; leading zeros are optional (figure 8-17).

The length of a packed decimal field in digits (P in column 43) is 2n-1, where n is the number of bytes occupied by the data as specified by FROM and TO.

If the FIELD NAME entry (columns 53-58) specifies an array name without an index, it is not necessary that the FROM and TO entries provide



FIELD LOCATION FROM (44–47)	FIELD LOCATION TO (48–51)	52	FIELD NAME (VARIABLE NAME) (53–58)
1	9	0	EHPNØ
10	23		NAMEL
24	42	0	HRS
53	55	0	CØDE
56	57		SEX

G14043

Figure 8-17. FIELD LOCATION Coding Example

sufficient space for the whole array, as long as it is big enough for an integral number of array elements. The array will be read in from element 1 up to as many elements as will fit into the locations specified. The decimal positions must contain the same entry as specified on the Extension Specifications for that array.

52 DECIMAL POSITIONS

This field is used to specify the number of positions to the right of the implied decimal point in a numeric field. This entry may not be blank for a numeric field; if the data field contains only integral values, a 0 should be entered to indicate no decimal positions. Valid entries are:

Entry	Definition
Blank	Alphanumeric field.
0-9	Number of decimal positions in a numeric field.

Any field to be used for arithmetic operations, or to be edited, must be numeric. The number of decimal positions specified cannot exceed the length of the field (as specified in the FIELD LOCATION field).

If the FIELD NAME entry (columns 53-58) on the Input Specifications specifies an array name, then the decimal positions field must contain the same entry as specified in the decimal positions field of the Extension Specification for that array.

53-58 FIELD NAME (VARIABLE NAME)

This field is used to assign an identifier (name) to an input data field. All fields that will be referenced by the program must be named. Names must be assigned in accordance with the rules for forming field names as described in Chapter 2. A previously defined vector name may be used, which allows loading of the vector during input. Refer to Section 5 for a complete discussion of this method of vector loading. A separate line must be used for each field description.

All fields within one record type should have unique names; if two or more fields within the same record have identical names, only the last one defined is used. Fields from different record types may have the same name, but all names not uniquely defined must have the same length and data type (decimal position entry). These fields do not have to occur in the same location in each record.

OR Relationship

To eliminate duplicate coding of identical fields within different record types, the OR relationship may be used. The OR relationship, illustrated in figure 8-18, shows two record types which have identical fields in the same record positions. Refer to columns 14-16 in this section.

Special Words

The following special words are reserved for use as variable names in columns 53-58 of the Input Specifications:

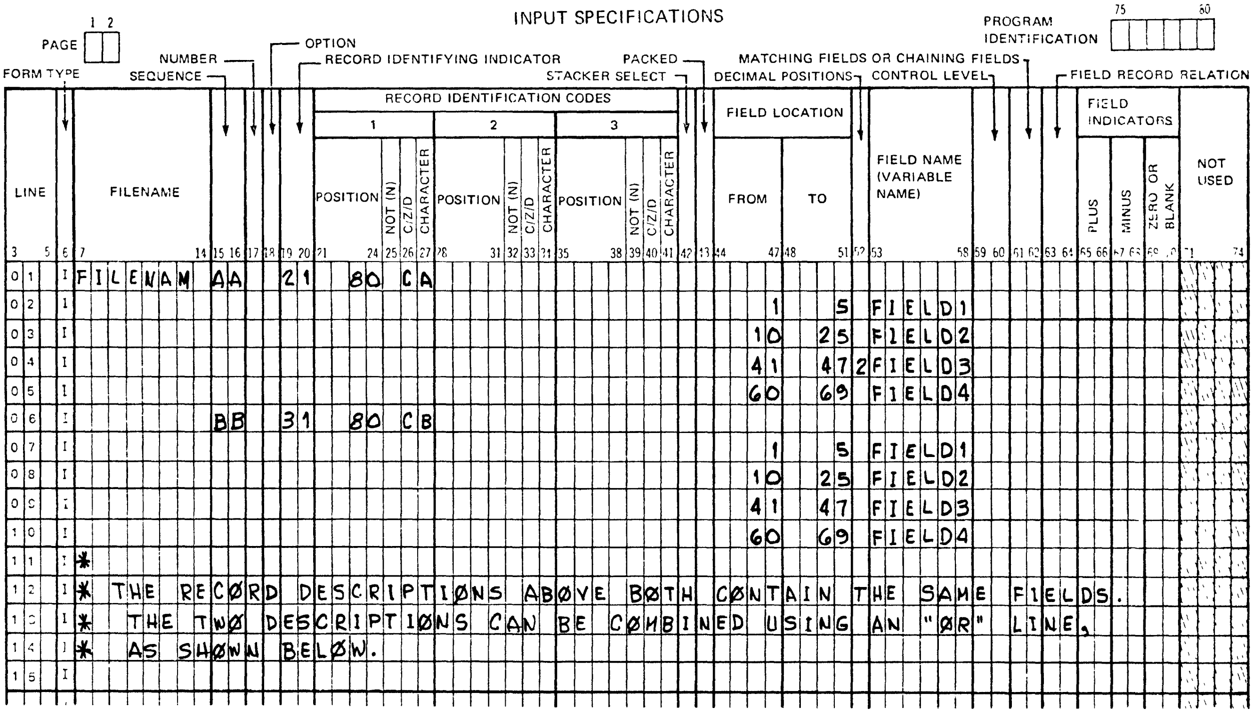
PAGE
PAGEn (where n=1-8)

If page numbering is desired on output, the special word "PAGE" or "PAGEn" for two more printer files is used to indicate that page numbering is to be done. Page field coding is illustrated in figure 8-19.

This feature allows a page number to be entered through a field in an input record, the field called "PAGE". The page number printed will be one greater than the page number contained in the "PAGE" field of the input record. A page field is incremented by 1 each time before it is printed. The field may be defined as any length, but it must contain zero decimal positions. Unless otherwise specified, it is assumed to be four digits in length with zero decimal positions (see figure 8-19). The "PAGE" field may be used in calculations like any other field.

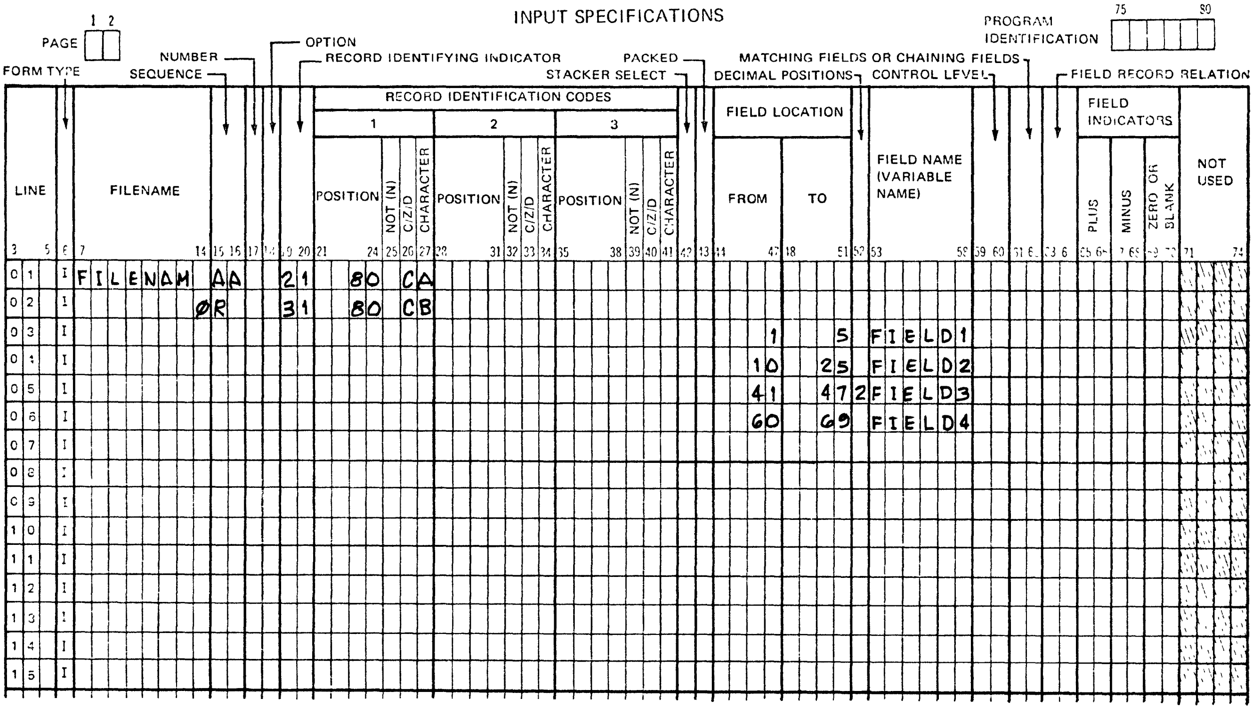
The same "PAGE" entry may be used for two different output files, but this is not recommended.

Figure 8-19 is an example of PAGE field coding on Input Specifications.



INPUT SPECIFICATIONS

```
  01 IFILENAM AA  21  80 CA
  02 I                                        1   5 FIELD1
  03 I                                       10  25 FIELD2
  04 I                                       41  472FIELD3
  05 I                                       60  69 FIELD4
  06 I        BB  31  80 CB
  07 I                                        1   5 FIELD1
  08 I                                       10  25 FIELD2
  09 I                                       41  47 FIELD3
  10 I                                       60  69 FIELD4
  11 I*
  12 I* THE RECØRD DESCRIPTIØNS ABØVE BØTH CØNTAIN THE SAME FIELDS.
  13 I*  THE TWØ DESCRIPTIØNS CAN BE CØMBINED USING AN "ØR" LINE,
  14 I*  AS SHØWN BELØW.
  15 I
```

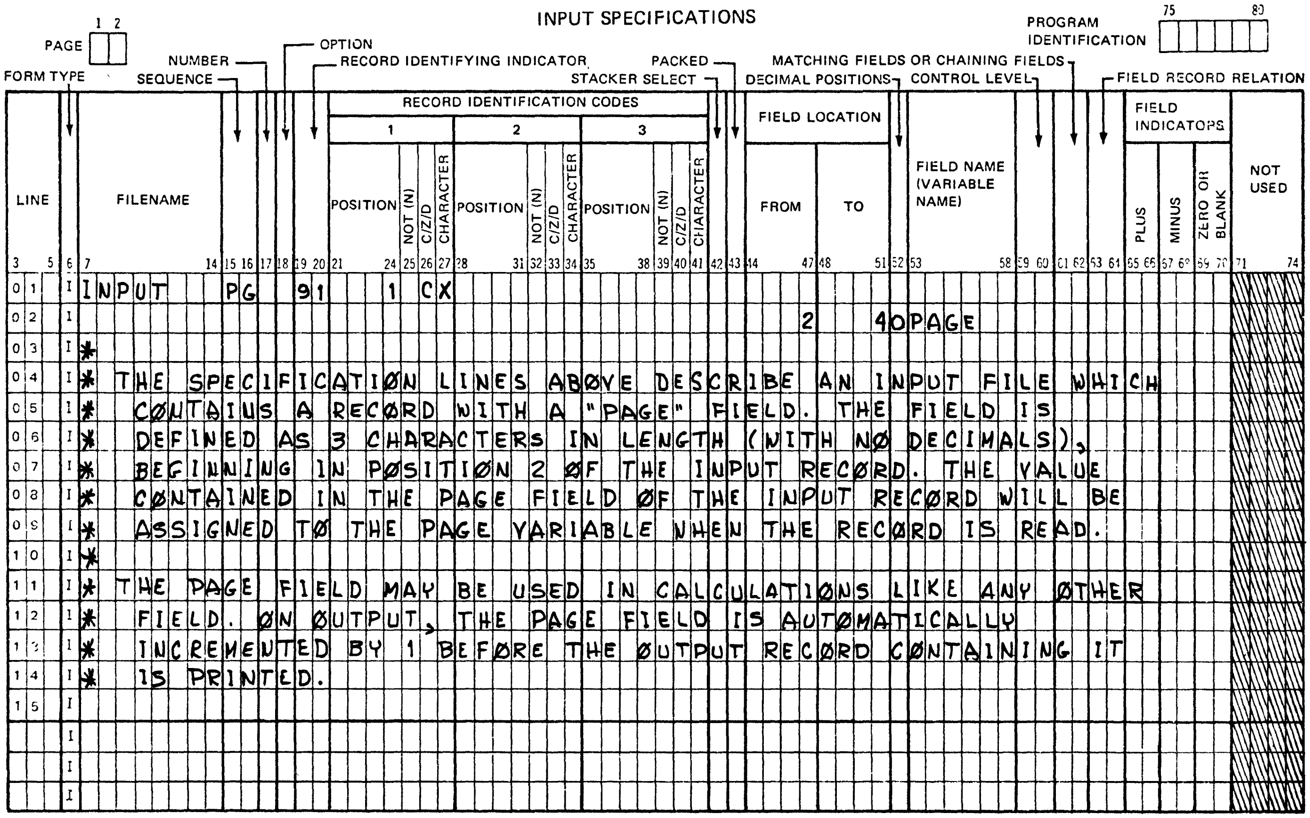


INPUT SPECIFICATIONS

```
  01 IFILENAM AA  21  80 CA
  02 I       ØR   31  80 CB
  03 I                                        1   5 FIELD1
  04 I                                       10  25 FIELD2
  05 I                                       41  472FIELD3
  06 I                                       60  69 FIELD4
  07 I
  08 I
  09 I
  10 I
  11 I
  12 I
  13 I
  14 I
  15 I
```

G14044

Figure 8-18. OR Relationship - Identical Fields Within Different Record Types



INPUT SPECIFICATIONS

PAGE 1 2 · PROGRAM IDENTIFICATION 75 80

FORM TYPE · NUMBER · SEQUENCE · OPTION · RECORD IDENTIFYING INDICATOR · PACKED · STACKER SELECT · MATCHING FIELDS OR CHAINING FIELDS · DECIMAL POSITIONS · CONTROL LEVEL · FIELD RECORD RELATION

LINE · FILENAME · RECORD IDENTIFICATION CODES 1 2 3 (POSITION · NOT (N) · C/Z/D · CHARACTER) · FIELD LOCATION (FROM · TO) · FIELD NAME (VARIABLE NAME) · FIELD INDICATORS (PLUS · MINUS · ZERO OR BLANK) · NOT USED

```
01 IINPUT   PG  91   1 CX
02 I                                        2   40PAGE
03 I*
04 I* THE SPECIFICATIØN LINES ABØVE DESCRIBE AN INPUT FILE WHICH
05 I*  CØNTAINS A RECØRD WITH A "PAGE" FIELD. THE FIELD IS
06 I*  DEFINED AS 3 CHARACTERS IN LENGTH (WITH NØ DECIMALS),
07 I*  BEGINNING IN PØSITIØN 2 ØF THE INPUT RECØRD. THE VALUE
08 I*  CØNTAINED IN THE PAGE FIELD ØF THE INPUT RECØRD WILL BE
09 I*  ASSIGNED TØ THE PAGE VARIABLE WHEN THE RECØRD IS READ.
10 I*
11 I* THE PAGE FIELD MAY BE USED IN CALCULATIØNS LIKE ANY ØTHER
12 I*  FIELD. ØN ØUTPUT, THE PAGE FIELD IS AUTØMATICALLY
13 I*  INCREMENTED BY 1 BEFØRE THE ØUTPUT RECØRD CØNTAINING IT
14 I*  IS PRINTED.
15 I
   I
   I
   I
```

G14045

Figure 8-19. PAGE Field Coding

59-60 CONTROL LEVEL

This field is used to assign control level indicators to primary or secondary files. Any field other than a whole array or a look-ahead field may be assigned a control level indicator, in which case it is known as a Control Field. Control fields are checked each program cycle for a change in information; when data in the field changes, a control break occurs. A group of records with the same information in the control field is known as a Control Group. Valid entries for this field are:

Entry	Definition
L1-L9	Control level field assigned.
Blank	No control level field assigned.

A control break occurs when a record containing a control field is read and the information in that control field is different from the information in the same control field of the previous record. When a control break occurs, the designated control level indicator turns ON, along with all control level indicators lower than it. For example, if control level indicator L5 is turned ON, L4, L3, L2, and L1 are also automatically turned ON. Control level indicator L0 is always ON and cannot be assigned to a control field. However, L0 can be used to condition total calculations or total output.

A control level indicator may be turned on or off by SETON or SETOF, or may be used as a record identifying indicator. However, in such cases, control level indicators lower than the one specified are not turned on or off automatically.

Control level indicators are used to condition operations, such as:

a. Calculations that must be performed when a control group changes (totals, etc.).

b. Operations that must be performed on the first card of a new control group.

c. Summary punching or total printing that must be performed for each control group.

Control level indicators can be used in input, calculation, and output specifications. The following rules must be observed when assigning control level indicators:

a. The same control level indicator may be used in different record types or files; however, the control fields associated with that indicator must be of the same length. (See figure 8-20).

b. Field names have no effect upon the control level indicator assigned; therefore, control fields in different record types may have both the same name and the same indicator assigned.

c. The maximum size of a control field is 255 characters.

d. Within one record type, control fields may overlap.

e. Numeric control fields are treated as though they had no decimal positions.

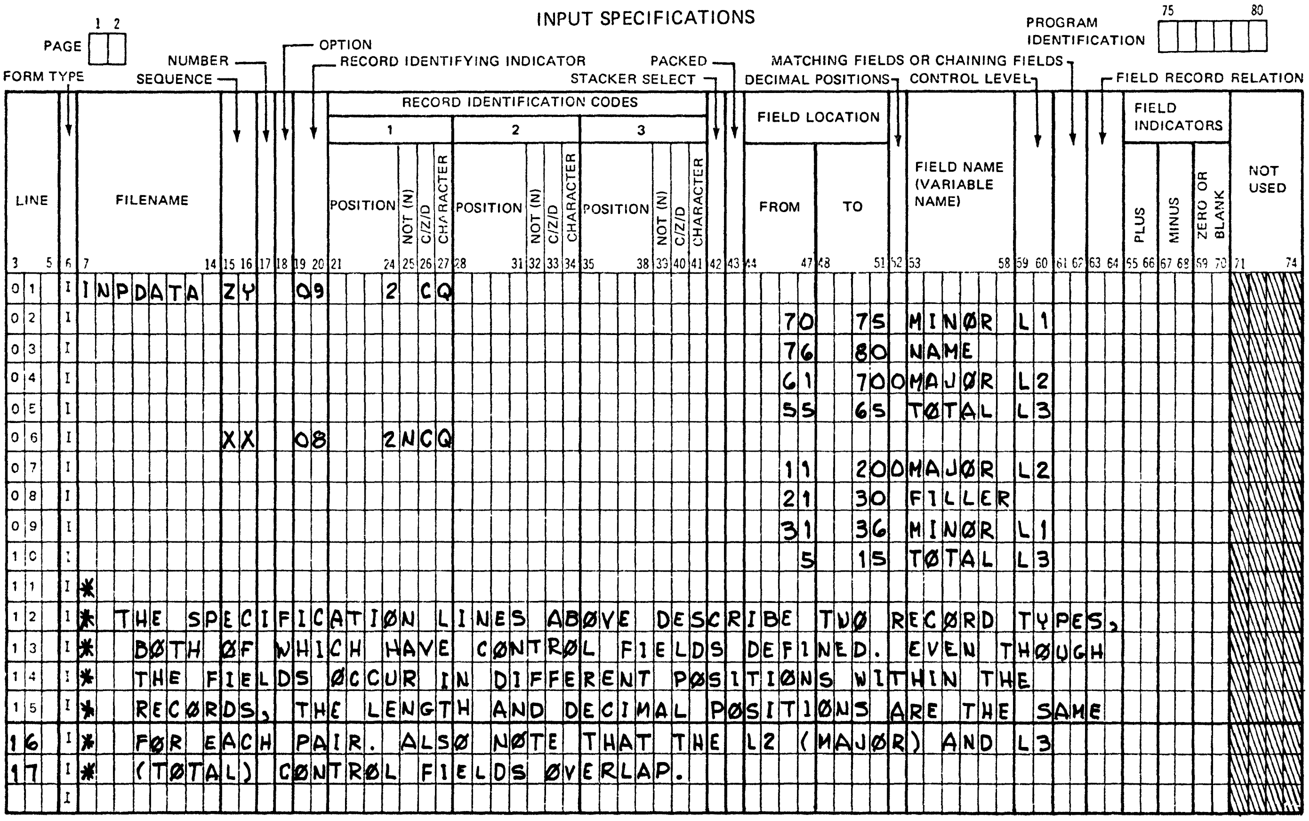
f. For numeric control fields, only the digit portion of each character is compared; thus, negative numbers are treated the same as positive numbers.

g. All control fields with the same control level indicator are considered numeric if any one of those fields is described as numeric.

h. Control levels need not be written in any special sequence, and gaps are permitted in the control levels assigned.

i. Control fields are initialized to binary zeros.

j. Total calculations and total output operations are bypassed until the first cycle following a cycle in which a record specifying control field is selected. This prevents a control break from occurring the first time a record with control



INPUT SPECIFICATIONS

```
01 IINPDATA ZY  09   2 CQ
02 I                                       70  75 MINØR L1
03 I                                       76  80 NAME
04 I                                       61  700MAJØR L2
05 I                                       55  65 TØTAL L3
06 I        XX  08   2NCQ
07 I                                       11  200MAJØR L2
08 I                                       21  30 FILLER
09 I                                       31  36 MINØR L1
10 I                                        5  15 TØTAL L3
11 I*
12 I* THE SPECIFICATIØN LINES ABØVE DESCRIBE TWØ RECØRD TYPES,
13 I*   BØTH ØF WHICH HAVE CØNTRØL FIELDS DEFINED. EVEN THØUGH
14 I*   THE FIELDS ØCCUR IN DIFFERENT PØSITIØNS WITHIN THE
15 I*   RECØRDS, THE LENGTH AND DECIMAL PØSITIØNS ARE THE SAME
16 I*   FØR EACH PAIR. ALSØ NØTE THAT THE L2 (MAJØR) AND L3
17 I*   (TØTAL) CØNTRØL FIELDS ØVERLAP.
   I
```

G14046

Figure 8-20. Control Fields in Two Record Types

fields is read (the input control fields would usually not be equal to the value contained in the initialized control fields).

k. Different record types in a file need not have the same number of control fields. However, the user must ensure that unwanted control breaks do not occur.

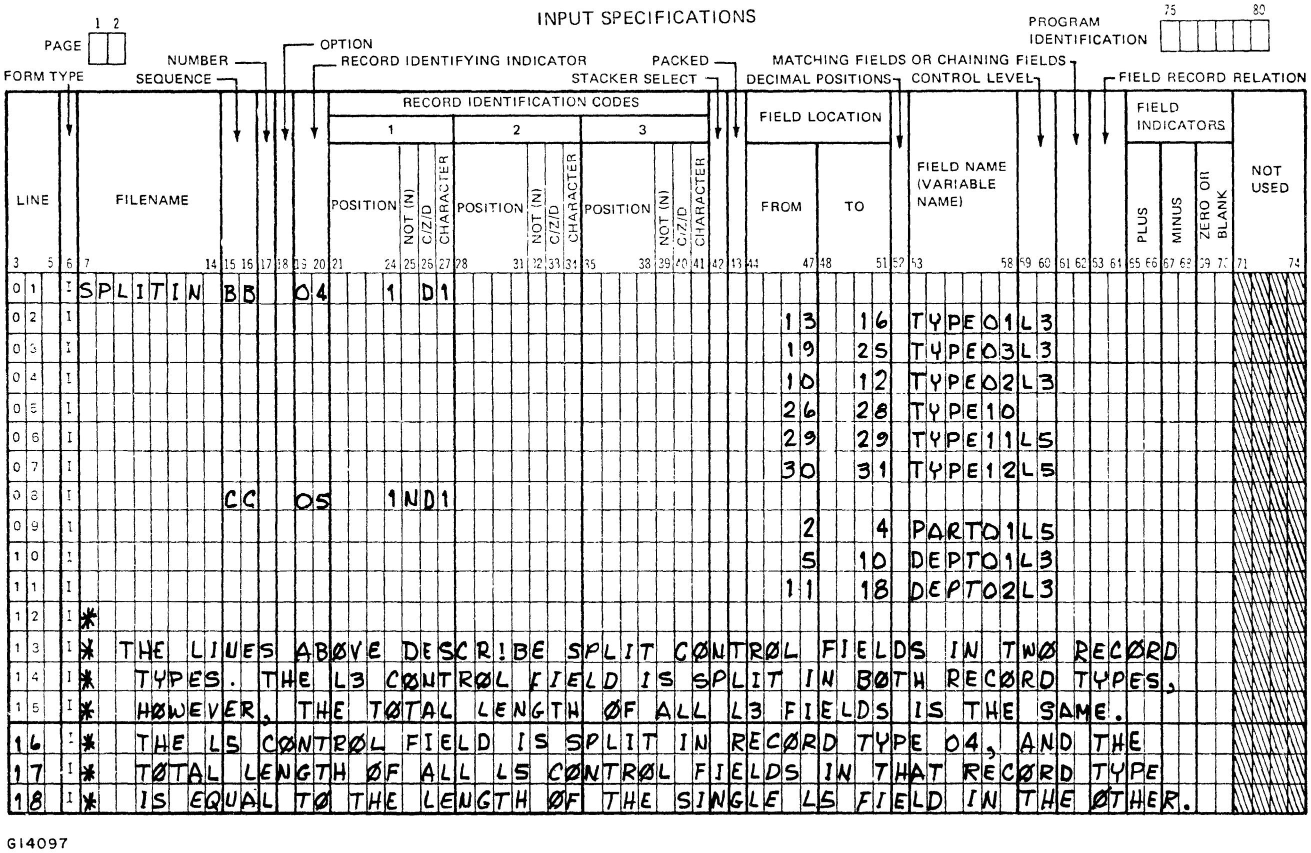
l. Control level indicators cannot be assigned to a binary field.

Split Control Fields

If the same control level indicator is assigned to more than one field within the same record type, the control field created is known as a Split Control Field. All fields so designated (those having the same control level within the same record type) are combined by the program in the order specified in the Input Specifications and are treated as one control field. Split control fields are illustrated in figure 8-21.

The following special rules must be observed for split control fields:

a. The same control level indicator may be used for split control fields in different record types if the field names used are different. The length of various portions of a split control


INPUT SPECIFICATIONS
PAGE
FORM TYPE
NUMBER
SEQUENCE
OPTION
RECORD IDENTIFYING INDICATOR
RECORD IDENTIFICATION CODES
POSITION
NOT (N)
C/Z/D
CHARACTER
PACKED
STACKER SELECT
DECIMAL POSITIONS
FIELD LOCATION
FROM
TO
MATCHING FIELDS OR CHAINING FIELDS
CONTROL LEVEL
FIELD NAME (VARIABLE NAME)
FIELD RECORD RELATION
FIELD INDICATORS
PLUS
MINUS
ZERO OR BLANK
NOT USED
PROGRAM IDENTIFICATION
LINE
FILENAME
G14097

```
  01 ISPLITIN BB  04   1 D1
  02 I                                       13  16 TYPE01L3
  03 I                                       19  25 TYPE03L3
  04 I                                       10  12 TYPE02L3
  05 I                                       26  28 TYPE10
  06 I                                       29  29 TYPE11L5
  07 I                                       30  31 TYPE12L5
  08 I        CC  05   1ND1
  09 I                                        2   4 PART01L5
  10 I                                        5  10 DEPT01L3
  11 I                                       11  18 DEPT02L3
  12 I*
  13 I* THE LIUES ABØVE DESCR!BE SPLIT CØNTRØL FIELDS IN TWØ RECØRD
  14 I*  TYPES. THE L3 CØNTRØL FIELD IS SPLIT IN BØTH RECØRD TYPES,
  15 I*  HØWEVER, THE TØTAL LENGTH ØF ALL L3 FIELDS IS THE SAME.
  16 I*  THE L5 CØNTRØL FIELD IS SPLIT IN RECØRD TYPE 04, AND THE
  17 I*  TØTAL LENGTH ØF ALL L5 CØNTRØL FIELDS IN THAT RECØRD TYPE
  18 I*  IS EQUAL TØ THE LENGTH ØF THE SINGLE L5 FIELD IN THE ØTHER.
```

Figure 8-21. Split Control Fields

field in one record type may be different than the corresponding portions in another record type; in fact, a control field may be split in one record type but not in another. However, the total length of the control fields (whether split or not) must be the same in both record types. For further information see FIELD RECORD RELATION (column 63-64).

b. If one portion of a split control field is numeric, the entire field is considered numeric.

c. Any one portion of a numeric split control field can not exceed the maximum size allowed for a numeric field, which is 15 characters. However, the total length of all fields assigned to one control level indicator (within each record type) can be as large as 255 characters.

d. A mixture of packed and unpacked control fields is allowed.

For example, consider that three different record types can have:

1. A control level specified with a field size of seven bytes.

2. Split control fields of three and four bytes.

3. A split control field consisting of two bytes containing three packed digits, one byte, and three bytes containing three unpacked digits.

e. No other specification lines may come between lines describing split control fields.

61-62 MATCHING FIELDS

This field is used to designate matching fields for multifile processing and sequence checking. Valid entries are:

Entry	Definition
Blank	No matching fields and no sequence checking specified.
M1-M9	Matching fields and sequence checking when two or more input, update, or combined files specify the same match fields. Sequence checking when only one input, update, or combined file specifies matching fields.

Matching Fields

Designation of matching fields allows comparison of records from a primary file with records of one or more secondary files to determine if the records match. A maximum of nine different fields can be specified by field designators M1-M9, which are used to specify which fields in each record are to be matched.

The matching record indicator, MR, is turned ON when the contents of the primary file match field is the same as the contents of the match field of any secondary file. The matching record indicator is set ON or OFF before detail calculations during the RPG program cycle.

M1-M9 are not indicators, but cause MR to be set ON when the specified records match. MR is used to condition those operations that are to be executed when the records match.

Rules for Matching Fields

The following rules must be observed when assigning matching field values:

a. All Match Fields must be in the same sequence during input, because sequence checking is automatically done on all fields designated as matching fields. A sequence error in any field will cause a program halt. When the system operator resumes program operation, the record in error is ignored and the next record from the same file is read.

b. If matching is used, it is not necessary for all primary and secondary files to have Match Fields. Neither must all record types within a file have Match Fields. But at least one record type from two files must have Match Fields specified if the files are ever to be matched.

c. All fields given the same matching field value (M1-M9) must be of the same length.

d. Overlapping of different Match Fields within one record type is allowed; however, the length of any individual Match Field must not exceed 255 characters.

e. All records to be matched must contain the same Match Fields (M1-M9); otherwise, a match may not be obtained.

f. When more than one Match Field is designated for a record type, all fields specified are combined in order by descending sequence of matching field values and are treated as one contiguous Match Field. The high order field is M9.

g. Split Match Fields are not allowed; thus, the same matching field value should not be used more than once in one record type, unless field record relation indicators are used.

h. Numeric Match Fields are treated as though they had no decimal positions.

i. For numeric Match Fields, only the digit portion of each character is compared; thus, negative numbers are treated the same as positive numbers.

j. All Match Fields with the same matching field value are considered numeric, if any one of the fields is numeric.

k. In order for the matching record indicator to be turned ON, the absolute values of the data in the match fields in the secondary file must equal the absolute values of the data of the match fields in the primary file. Therefore, when more than one matching field value is used for matching records, the contents of all match fields must match before MR can be turned ON.

l. Field names have no effect upon the matching field values assigned; therefore, Match Fields in different record types may have both the same name and the same matching field value assigned.

m. Whole arrays must not be designated as match fields.

n. Records in primary and secondary files without match fields are processed before records with match fields specified.

o. Match fields for each match field value must have the same length.

Multiple File Processing Using Matching Records

At the beginning of RPG program execution, one record is read into the input buffer of each primary and secondary file. One of these records is selected for processing according to the rules following this paragraph. After the selected record is processed, the next record is automatically read from the same file. During the next program cycle, the current record is compared with the records remaining from the previous cycle in order to select the next record.

Rules for Multiple File Processing Using Matching Records

The following rules apply to multiple file processing using matching records:

a. If the next record from any primary or secondary file has no match fields, then that record is selected.

b. If the records do not match and the records are in ascending sequence, then the record with the lowest match field value is selected. If records are in descending sequence, then the record with the highest match field value is selected.

c. If more than one record satisfies either of the rules described in items a and b, the record from the highest priority file is selected. The primary file is of highest priority, followed by the secondary files in the order they are specified on the File Description Specifications.

d. When a record from the primary file matches a record from the secondary file, the primary file record is processed first.

e. When more than one secondary file is declared, all matching records from a secondary file are processed before control is passed to the next secondary file.

f. Matching records allow the program to enter data from the primary record into the matching secondary record, since the primary record is processed first. Transfer of data from secondary records into matching primary records may be done through the use of look-ahead fields.

g. The MR indicator is ON during the processing of any record containing a match field, providing the current match field value originally occurred in a primary record. Otherwise, MR is in the OFF state.

h. MR, the matching record indicator, is turned OFF for one program cycle when a record selected by the FORCE operation is processed. If the next record not selected by the FORCE operation matches the last record with match fields specified, the MR indicator is turned ON.

i. If the primary file has an E in column 17 of the File Description Specification but the secondary files do not, any secondary file records that match the last primary record plus any interspersed secondary file records without match fields are processed before the LR indicator is turned ON.

When the coding shown in figure 8-15 is compiled and executed, the two matching fields are combined in the ascending order of the matching field values, M1 and M2. Notice that an End-of-File is specified for the secondary file. This causes the job to end after the last record in the file, TIMECDS, is read.

Processing of matching fields proceeds as follows:

a. When a record from the primary file matches a record from the secondary file, the primary file record is processed first.

b. When records do not match, the record with the lowest (ascending files) or highest (descending files) Match Field value is processed first.

c. A record type which has no matching field specification is processed immediately after the record it follows, and the MR indicator is not turned ON. If such a record is the first one in a file, it is processed first (even if it is not in the primary file).

d. Matching records allow the program to enter data from the primary record into the matching secondary record, since the primary record is processed first. Transfer of data from secondary records into matching primary records may be done through the use of look-ahead fields.

e. When additional secondary files are declared, all matching records are processed in one secondary file before passing control to the next secondary file. The precedence of the secondary files is determined by their order of appearance on the Input Specifications.

In figure 8-22 the two matching fields will be combined in the ascending order of the matching field values, M1 and M2. Notice that an End-of-File has been specified for the secondary file. This will cause the job to end after the last record in the file, TIMECDS, has been read.

The example shown in figure 8-23 is used to illustrate the order in which two matching fields will be processed. An End-of-File has been specified for the primary file.

The files (figure 8-23) will be processed in the following order:

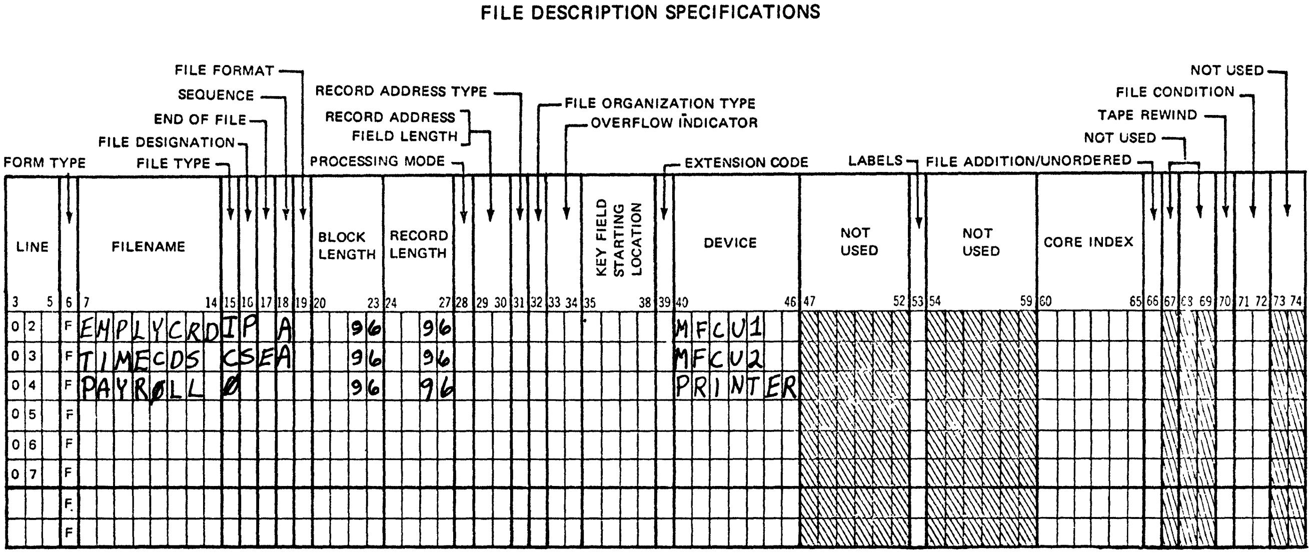
S1 Records with no matching fields are processed before records with matching fields regardless of file type.

P1
S2
S3 The MR indicator is ON. All matching secondary records are processed after the primary record.

P2

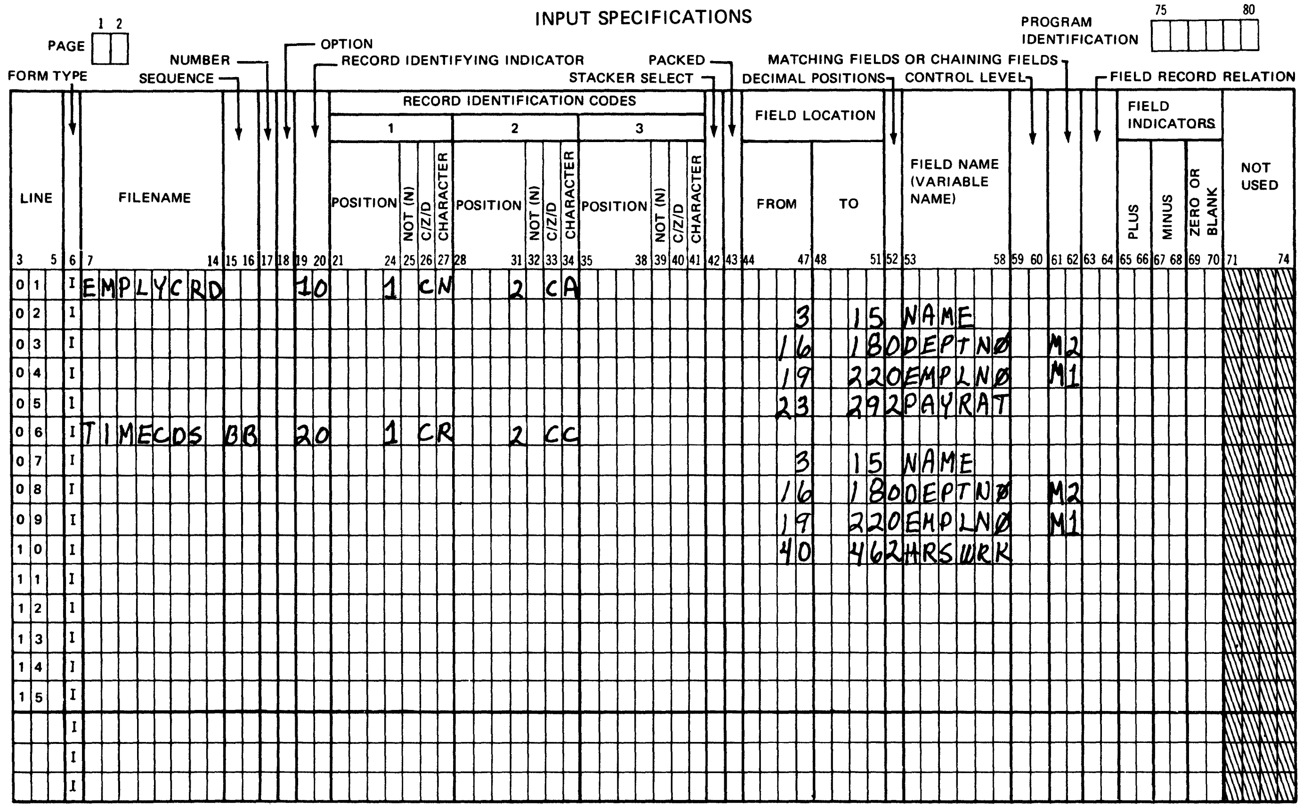
P3
S4 The MR indicator is ON.

P4



FILE DESCRIPTION SPECIFICATIONS

LINE (3-5)	FORM TYPE (6)	FILENAME (7-14)	FILE TYPE (15)	FILE DESIGNATION (16)	END OF FILE (17)	SEQUENCE (18)	FILE FORMAT (19)	BLOCK LENGTH (20-23)	RECORD LENGTH (24-27)	DEVICE (40-46)
02	F	EMPLYCRD	I	P		A		96	96	MFCU1
03	F	TIMECDS	C	S	E	A		96	96	MFCU2
04	F	PAYRØLL	Ø					96	96	PRINTER
05	F									
06	F									
07	F									
	F									
	F									



INPUT SPECIFICATIONS

LINE (3-5)	FORM TYPE (6)	FILENAME (7-14)	SEQUENCE (15-16)	RECORD IDENTIFYING INDICATOR (19-20)	1 POSITION (21-24)	1 C/Z/D (26)	1 CHARACTER (27)	2 POSITION (28-31)	2 C/Z/D (33)	2 CHARACTER (34)	FIELD LOCATION FROM (44-47)	FIELD LOCATION TO (48-51)	DECIMAL POSITIONS (52)	FIELD NAME (53-58)	MATCHING FIELDS (61-62)
01	I	EMPLYCRD		10	1	C	N	2	C	A					
02	I										3	15		NAME	
03	I										16	18	0	DEPTNØ	M2
04	I										19	22	0	EMPLNØ	M1
05	I										23	29	2	PAYRAT	
06	I	TIMECDS	BB	20	1	C	R	2	C	C					
07	I										3	15		NAME	
08	I										16	18	0	DEPTNØ	M2
09	I										19	22	0	EMPLNØ	M1
10	I										40	46	2	HRSWRK	
11	I														
12	I														
13	I														
14	I														
15	I														

G14048

Figure 8-22. Coding Matching Fields

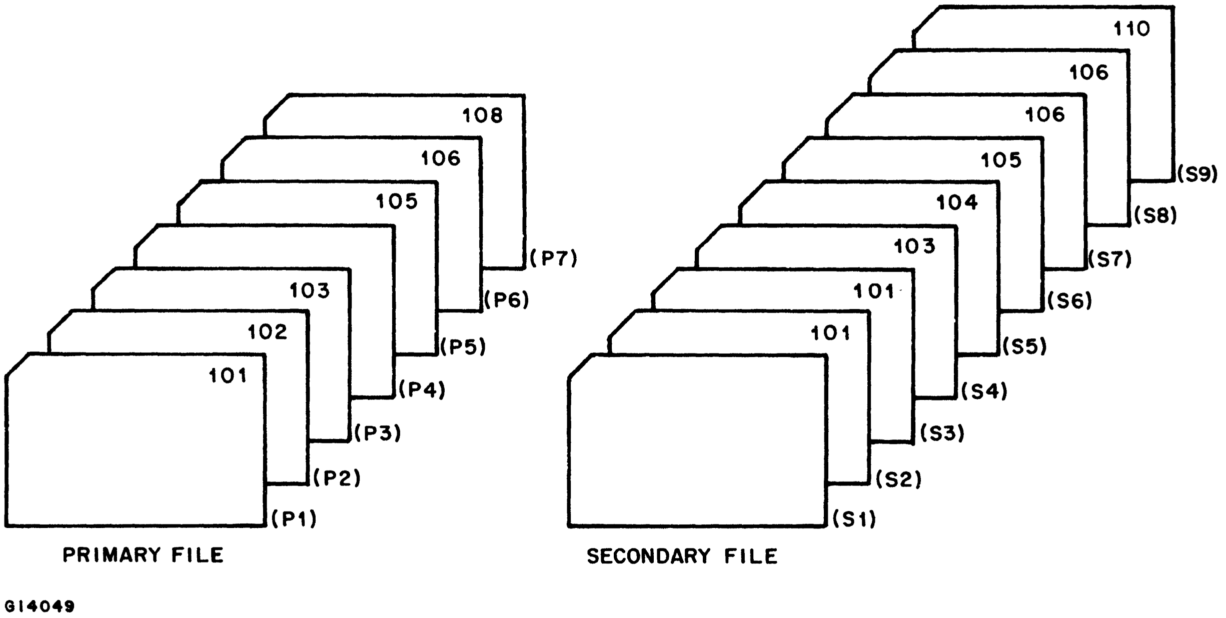


Figure 8-23. Record Selection From Two Matching Files

S5 When no records match, the record with the lowest sequence number is processed, regardless of file type.

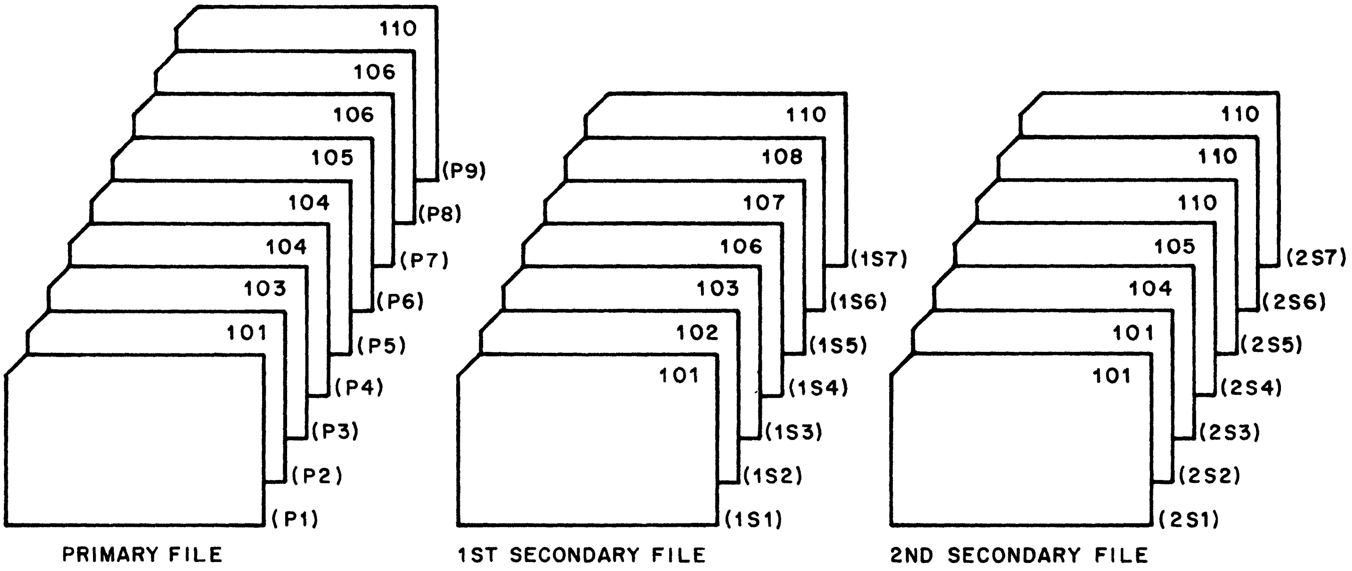
P5, S6 } The MR indicator is ON.

P6, S7, S8 } The MR indicator is ON.

P7 Last record processed.

S9 When an End-of-File condition is specified for a primary file, secondary records are not processed unless the MR indicator is turned ON.

The example shown in figure 8-24 illustrates the order in which records with three matching fields are processed.



G14050

Figure 8-24. Record Selection From Three Matching Files

The files (figure 8-24) will be processed in the following order:

P1

P2
1S1
2S1
2S2
} The MR indicator is ON since all Match Fields match.

1S2 When records do not match, the lowest matching field is processed next.

P3

1S3

P4

P5

2S3

P6

2S4

P7

P8

1S4

1S5

1S6

P9
1S7
2S5
2S6
2S7
} The MR indicator is ON. 2S7 is the last record processed.

Sequence Checking

Sequence checking of records within a file is performed when even matching field designators (M1-M9) are assigned. The sequence checking is performed in conjunction with matching records if two or more input, update, or combined files have match field designators specified.

If only one input, update, or combined file has an entry in columns 61-62, then sequence checking only is performed on the data in the fields to which M1-M9 have been assigned. A maximum of nine fields (M1-M9) within the record can be selected for sequence checking. When a record is encountered which has a field or fields out of sequence, the program halts. The system operator can cause program operation to be resumed, in which case the record in error is ignored and the next record is read from the same file.

The following rules must be observed when assigning matching field designators for sequence checking:

a. All fields designated for sequence checking must be in the same order, either ascending or descending.

b. When more than one field is designated for sequence checking, all fields specified are combined in order by ascending sequence of matching field designators (M1-M9) and are treated as one contiguous field.

c. Split sequence fields are not allowed; thus, the same matching field designation should not be used more than once in the record, unless field record relation indicators are used.

d. Numeric fields are treated as though they had no decimal positions.

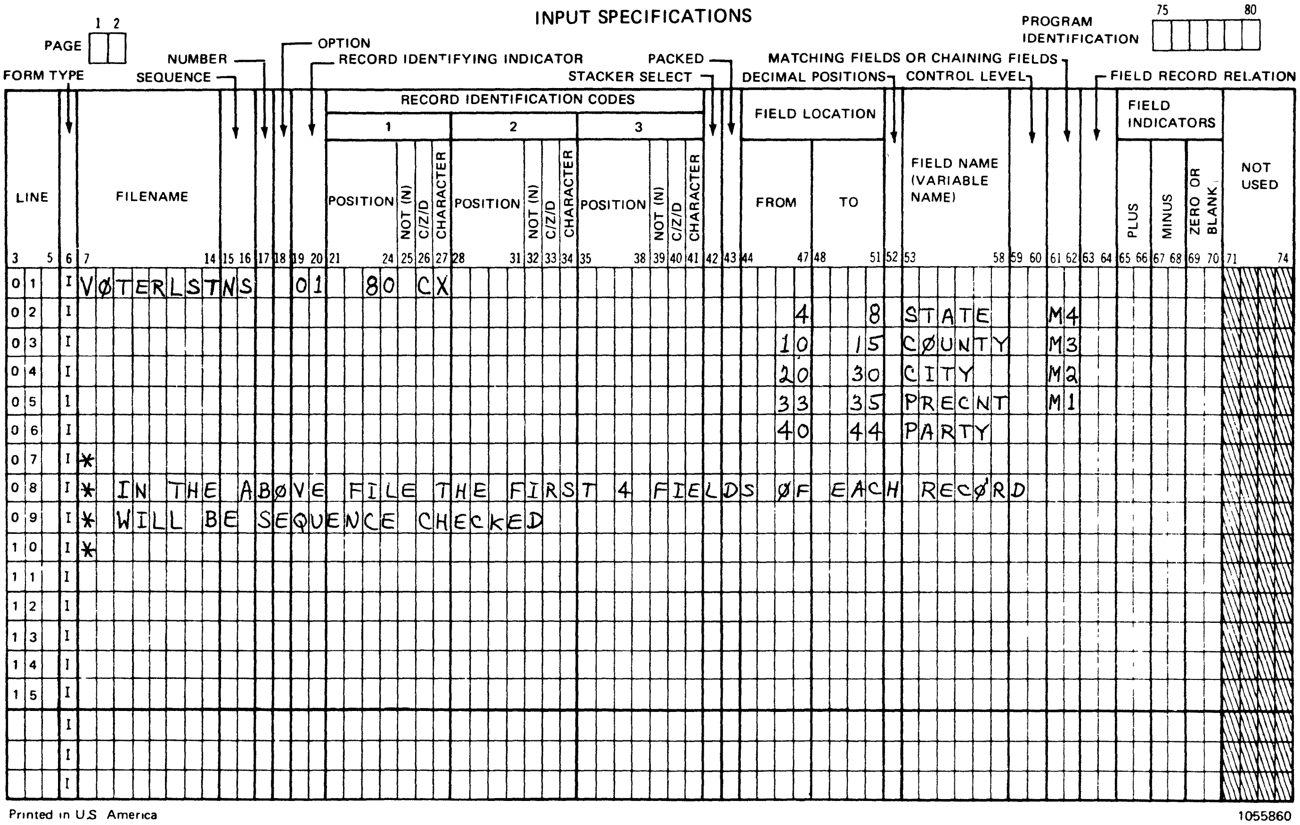
e. For numeric fields, only the digit portion of each character is compared; thus, negative numbers are treated the same as positive numbers.

f. All sequence fields are considered numeric if any one of the fields is numeric.

The example shown in figure 8-25 illustrates coding procedures for sequence checking.

63-64 FIELD RECORD RELATION

The assignment of field record relation indicators in columns 63-64 permits data to be made available for processing only if the conditions specified in the field record relation columns are satisfied. For example, if the indicator is ON, the data is made available just prior to detail calculations. If the indicator is OFF, the field remains in its previous condition.



INPUT SPECIFICATIONS

PAGE 1 2 · FORM TYPE · NUMBER · SEQUENCE · OPTION · RECORD IDENTIFYING INDICATOR · PACKED · STACKER SELECT · MATCHING FIELDS OR CHAINING FIELDS · DECIMAL POSITIONS · CONTROL LEVEL · FIELD RECORD RELATION · PROGRAM IDENTIFICATION 75 80

LINE · FILENAME · RECORD IDENTIFICATION CODES 1 2 3: POSITION, NOT (N), C/Z/D, CHARACTER · FIELD LOCATION: FROM, TO · FIELD NAME (VARIABLE NAME) · FIELD INDICATORS: PLUS, MINUS, ZERO OR BLANK · NOT USED

```
  01 IVØTERLSTNS  01  80 CX
  02 I                                     4   8 STATE   M4
  03 I                                    10  15 CØUNTY  M3
  04 I                                    20  30 CITY    M2
  05 I                                    33  35 PRECNT  M1
  06 I                                    40  44 PARTY
  07 I*
  08 I* IN THE ABØVE FILE THE FIRST 4 FIELDS ØF EACH RECØRD
  09 I* WILL BE SEQUENCE CHECKED
  10 I*
  11 I
  12 I
  13 I
  14 I
  15 I
     I
     I
     I
```

Printed in U.S. America

1055860

GI4047

Figure 8-25. Coding for Sequence Checking

The following rules should be observed when assigning field record relation indicators.

a. A field record relation indicator need not necessarily be the same as any record identifying indicator specified for the file.

b. Fields within one record type which specify the same field record relation indicator may be entered in any order; however, the most efficient data storage is obtained when they are written as a group on specification lines following one another.

c. All portions of a Split Control Field must be assigned the same field record relation indicator and must be written as a group on specification lines following one another.

If field record relation indicators are not to be assigned, this field should be left blank (see figure 8-26). The acceptable entries are as follows:

Entry	Definition
Blank	No field record relations.
01-99	Record identifying indicator assigned to a record type.
L1-L9	Control level indicator defined elsewhere.
MR	Matching record indicator.
U1-U8	External indicator defined elsewhere.
H0-H9	Halt indicator defined elsewhere.

Each of the entries is discussed in the following paragraphs.

01-99 Record Identifying Indicators

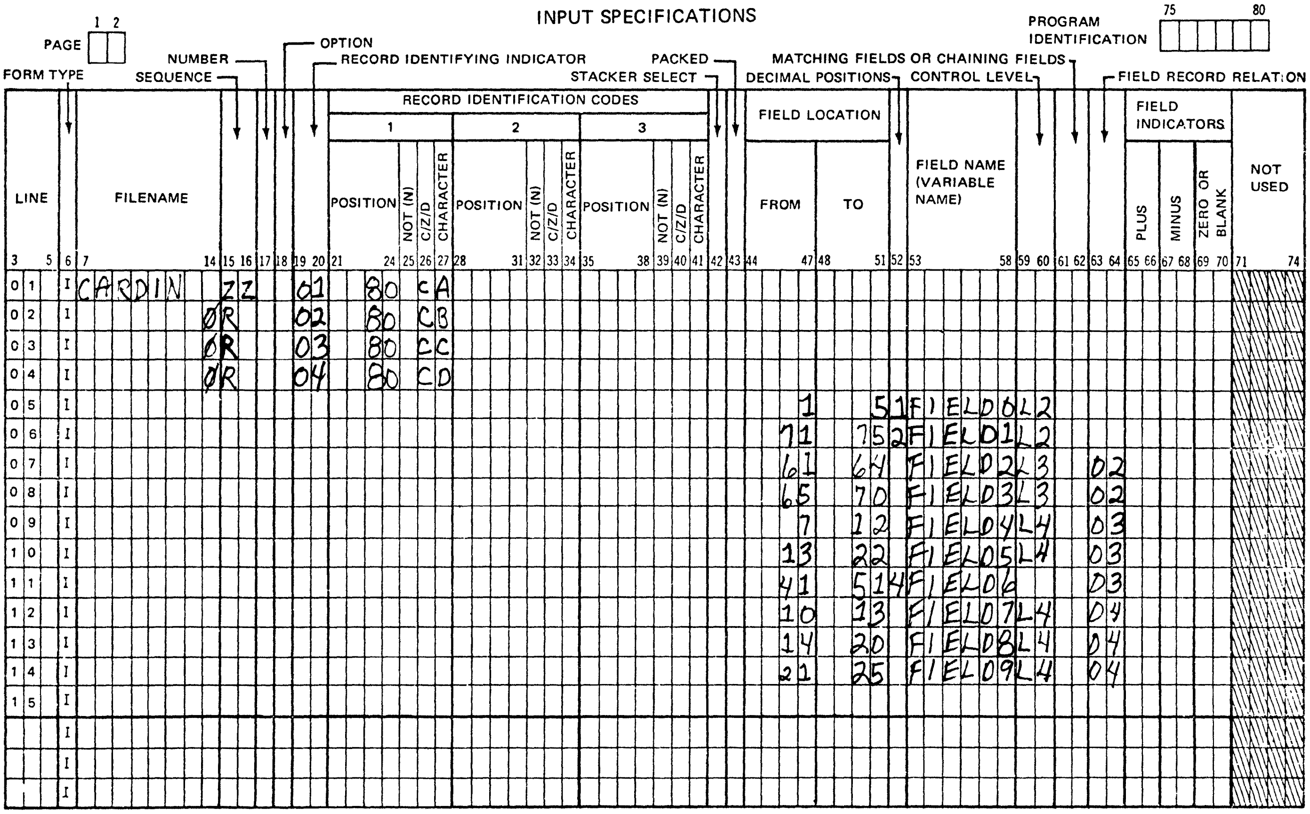
When several record types have been defined in an OR relationship, all fields defined apply to all record types. In many cases, however, not all of the record types defined have exactly the same fields. The FIELD RECORD RELATION field allows the programmer to specify that some fields apply only to certain record types and not to others. If the FIELD RECORD RELATION field is left blank, the associated field applies to all record types to which it is subordinate. However, by placing the same entry found in the Record Identifying Indicator field of one record type in the FIELD RECORD RELATION field, the field is identified as applying only to the corresponding record type.

Control Fields and Match Fields may also be related to a particular record type in an OR relationship by a FIELD RECORD RELATION entry.

When two or more Control Fields or two or more Match Fields have the same control level indicator or matching field value, respectively, only one of these may not have a field record relation indicator assigned. (This applies to a group of specifications, if it is a split control field specification.) Thus, specifications with field record relation indicators are used if that indicator is ON; when none of the field record relation indicators are ON, only the specification without any field record relation indicator is used.

L1-L9, MR Control Level or Matching Record Indicator

The use of control level or matching record indicators signifies that the data from the field is to be made available only if a control level break or matching record condition has occurred on this record.



INPUT SPECIFICATIONS

LINE	FORM TYPE	FILENAME	SEQUENCE	RECORD IDENTIFYING INDICATOR	POSITION	C/Z/D	CHARACTER	FIELD LOCATION FROM	FIELD LOCATION TO	DECIMAL POSITIONS	FIELD NAME (VARIABLE NAME)	CONTROL LEVEL	FIELD RECORD RELATION
01	I	CARDIN	ZZ	01	80	C	A						
02	I		ØR	02	80	C	B						
03	I		ØR	03	80	C	C						
04	I		ØR	04	80	C	D						
05	I							1	5	1	FIELD0	L2	
06	I							71	75	2	FIELD1	L2	
07	I							61	64		FIELD2	L3	02
08	I							65	70		FIELD3	L3	02
09	I							7	12		FIELD4	L4	03
10	I							13	22		FIELD5	L4	03
11	I							41	51	4	FIELD6		03
12	I							10	13		FIELD7	L4	04
13	I							14	20		FIELD8	L4	04
14	I							21	25		FIELD9	L4	04
15	I												
	I												
	I												
	I												

G14051

Figure 8-26. Field Record Relations - Using the OR Relationship

Restrictions:

a. A control level indicator should not be used on an input specification line to show the field record relation of a field that is designated in columns 59-60 of the same line as a part of, or the control group.

b. The matching record indicator (MR) should not be used on an input specification line to show the field record relation of a field that is designated in columns 61-62 of the same line as a part of, or all of, the matching record input control group.

To overcome these restrictions, the fields have to be defined twice, once to specify the control and/or matching fields, and again on another specification line (but with a different Variable Name) to specify the field record relations desired.

U1-U8 External Indicators

External indicators are used primarily to condition files in the File Description Specifications (see columns 71-72, FILE CONDITION). However, they may also be used to condition fields even though file conditioning is not specified. Data from the associated field will be accepted only when the specified indicator is ON.

HO-H9 Halt Indicators

The halt indicators are used to show a field/record relationship, and specify that the data in the field is to be made available only when the given halt indicator is ON. This would usually be used (with a record specified with the halt indicator in columns 19-20, RECORD IDENTIFYING INDICATOR) to load a signal pertaining to the reason or conditions of the halt operation.

65-70 FIELD INDICATORS

The FIELD INDICATOR field is composed of three subfields that allow the data of the input field to be tested as follows:

a. If the input field is defined as numeric, the data is tested for a positive, negative, or zero condition.

b. If the input field is defined as alphanumeric, the data is tested for greater than blank, less than blank, or blank, according to the EBCDIC collating sequence.

Valid entries are:

Entry	Definition
Blank	No field indicators used.
01-99	Field indicator.
HO-H9	Halt indicator.

The FIELD INDICATOR field is subdivided as described in the paragraphs that follow.

65-66 PLUS

Any valid indicator specified in this field will turn ON if the corresponding data field is greater than zero (numeric field only) or greater than blank (alphanumeric field).

67-68 MINUS

Any valid indicator specified in this field will turn ON if the corresponding data field is less than zero (numeric field only) or less than blank (alphanumeric field).

69-70 ZERO OR BLANK

Any valid indicator specified in this field will turn ON if the corresponding data field is zero (numeric field only) or blank (alphanumeric field).

The following rules must be observed when assigning and using field indicators:

a. All field indicators are OFF at the beginning of the program, and remain OFF until the condition being tested is satisfied on the input record just read.

   If the RPG I dialect is specified on the Control Card Specifications or the $ ZBINIT option card is included in an RPG II program, then any indicators from 01-99 which are used as zero/blank indicators are set ON at the beginning of program execution unless they are also used as record identifying indicators. All other field indicators are set OFF at the beginning of the program. Field indicators are set ON if:

   1. The condition being tested is satisfied by the input record just read, or

   2. In the case of zero/blank indicators which may have been set ON during the previous output cycle.

   If the condition being tested is not satisfied, the indicator is turned OFF.

   NOTE

   When the RPG I dialect option is specified, it can be overridden by including a $NZBINIT option card in the program.

b. A field may be assigned more than one indicator; however, only the indicator specified for the condition with a true result will be turned ON. All other indicators assigned to the field will be turned OFF.

c. The state of a field indicator assigned to fields in different record types is always determined by the last record selected.

d. The state of a field indicator assigned to more than one field within one record type is determined by the last field to which it is assigned.

e. When different field indicators are assigned to fields in different record types, a field indicator will remain ON (or OFF) until another record of the same type is selected.

f. If a halt indicator specified in the FIELD INDICATOR field is turned ON as a result of the corresponding condition being true, the program will halt <u>after</u> the input record which caused it to turn ON has been completely processed.

g. Field indicators cannot be assigned to whole arrays.

75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a complete description.

SECTION 9

CALCULATION SPECIFICATIONS AND OPERATION CODES

Calculation Specifications are used mainly to indicate the sequence and timing of the data manipulation that belongs to the following two broad categories:

a. Arithmetic, logical, and manipulative operations on defined data elements (fields), and

b. Input/output control of some kinds of files.

The Calculation Specifications describe the operations to be performed on the data and specify the order in which the operations are to be performed. Calculation Specifications are of three types, which are optional, but if they occur, must appear in the following order:

a. Detail calculations.

b. Total calculations.

c. Subroutines.

Subroutines must not be the only Calculation Specification entries, because subroutines can only be accessed from detail or total calculations.

Within each grouping, operations are performed in the order in which they are written.

Each specification line describes one operation, and is divided into three functional parts:

a. Conditions under which the operation is to be performed (columns 7-17).

b. The kind of operation to be performed and the data which is to be operated upon (columns 18-53).

c. Tests to be made upon the results of the operation (columns 54-59).

FIELD DEFINITIONS

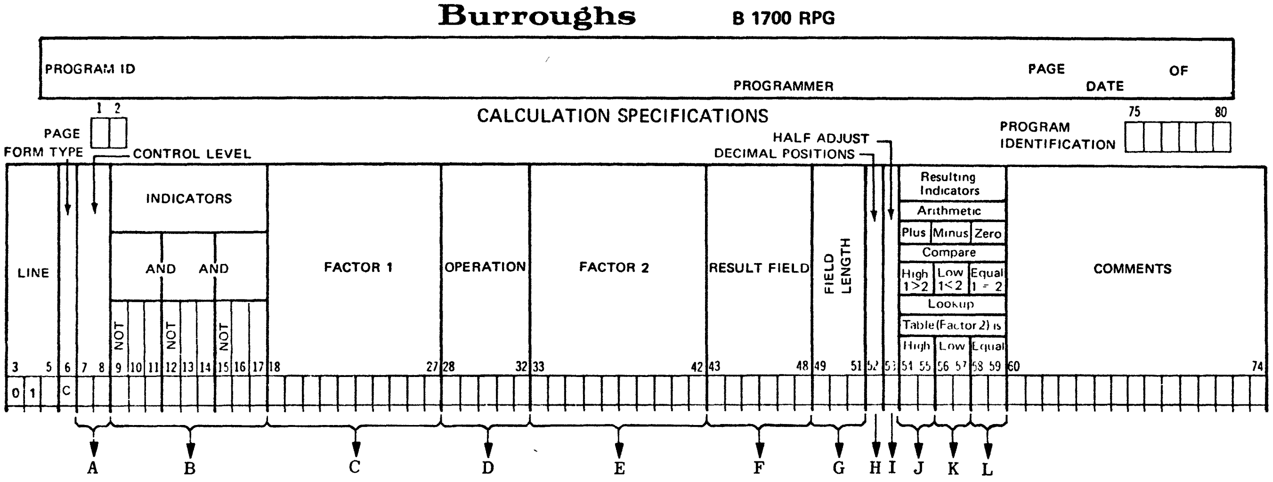
Figure 9-1 can be used in conjunction with the following field definitions for the Calculation Specification.

1-2 PAGE

Refer to Section 2 for a complete description.

3-5 LINE

Refer to Chapter 2 for complete description.



A. Contains one of the following control level indicators: L0-L9 (perform calculation at control break), LR (perform calculation after the last record), SR (calculation is part of a subroutine), AN, OR (establishes AND or OR relationships between indicators), or blank.

B. 9-17 Columns 10-11, 13-14, 16-17 may contain up to 3 indicators to condition a calculation operation. Entries: Blank, 01-99, L1-L9, LR-MR, H0-H9, U1-U8, 0A-0G, or OV Columns 9, 12, 15 specify that the following indicator is to be off. Entries: Blank or N.

C. 18-27 Contains either the name of any user or compiler defined field, an alphanumeric or numeric literal, any subroutine, or TAG name, or vector name, any special name, or blank. Entry must be left-justified.

D. 28-32 Specifies the type of operation to be performed. Entries: ADD, Z-ADD, SUB, Z-SUB, MULT, DIV, MVR, XFOOT, SQRT, MOVE, MOVEA, MOVEL, MLLZO, MHHZO, MLHZO, MHLZO, COMP, TESTN, TESTZ, BITON, BITOF, TESTB, SETON, SETOF, GOTO, TAG, TIME, ZIP, LOKUP, BEGSR, ENDSR, EXSR, FORCE, EXCPT, DSPLY, READ, CHAIN, RECV, SEND, SETLL, or DEBUG. Entry must be left-justified.

E. 33-42 Contains either the name of any user or compiler defined field, any alphanumeric or numeric literal, a subroutine name, a vector name, any special name, a GOTO operation label, a filename or blank. Entry must be left-justified.

F. 43-48 Specifies the name of the field, vector, or vector element that will be used to store the results of the operation. Entry must be alphanumeric and left-justified.

G. 49-51 Specifies the length of the result field. Entries: Blank or any decimal numeric.

H. 52 Specifies the number of decimal positions. If blank, entry is alphanumeric. Entries: Blank or 0-9.

I. 53 Specifies if the contents of the result field are to be rounded off. Entries: Blank or H.

J. 54-55 Entry is turned on if the result field is positive or FACTOR 1 is the highest in a compare operation, or FACTOR 2 is the highest in a lookup operation or a tested zone (TESTZ) is a plus-zone. Entries: 01-99, L1-L9, LR, H0-H9, 0A-0G, OV or blank.

K. 56-57 Entry is turned on if the result field is negative, or FACTOR 1 is the lowest in a compare operation, or FACTOR 2 is lowest in a lookup operation, or a tested zone (TESTZ) is a minus-zone. Entries: 01-99, L1-L9, LR, H0-H9, 0A-0G, OV or blank.

L. 58-59 Entry is turned on if the result field is zero, or FACTOR 1 is equal to FACTOR 2 in a compare or lookup operation, or a tested zone (TESTZ) is neither a plus- or minus-zone. Entries: 01-99, L1-L9, LR, H0-H9, 0A-0G, OV or blank.

G14052

Figure 9-1. Calculation Specifications Summary Sheet

6 FORM TYPE

A C must appear in this field.

7-8 CONTROL LEVEL

The valid entries for this field are listed below, and each entry is described in the paragraphs that follow.

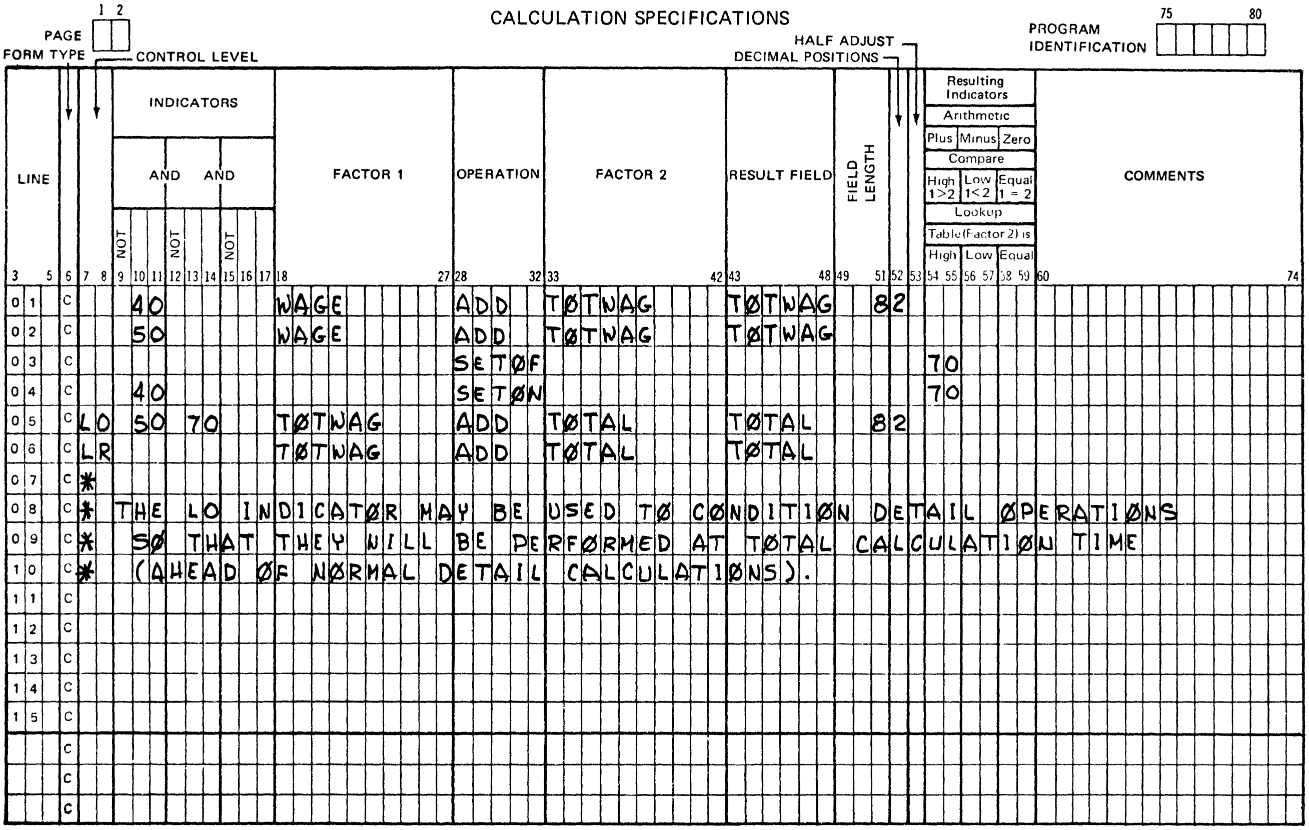
Entry	Definition
Blank	Calculation operation is part of detail calculations or may be part of a subroutine.
L0-L9	Calculation operation is performed in total calculations when the designated control break occurs or the specified indicator is set ON (L0 is always ON).
LR	Calculation operation is performed after the last record has been processed (or if set ON).
SR	Calculation operation is part of a subroutine (documentational not required for subroutines).
AN,OR	Establishes AND and OR relationship between lines of indicators.

An entry of L0-L9 or LR indicates that the operation is performed during total calculations when the specified indicator is ON. The first entry of L0-L9 or LR in columns 7-8 determines the start of total calculations. All detail calculations (if any are specified) must precede any total calculations. Any operations with blanks in columns 7-8 occurring before any total operation, or any BEGSR operation, are treated as detail operations, and are performed during detail calculations in every cycle depending upon the conditions specified in columns 9-17.

Subroutines, if used, must be specified after all detail and total lines. For a subroutine, column 7-8 may contain SR, OR, AN, or blank. The first BEGSR operation determines the start of subroutines.

L0-L9 Control level indicators L0-L9 are used to condition operations that are to occur at a control break. If a control level indicator is specified in columns 7-8, the operation described on the same specification line is done only when the designated indicator is ON. When a control break for a certain level occurs, all lower control level indicators also turn ON. However, when a control level indicator used as a record identifying indicator turns ON to indicate a specific record type, or when a control level indicator is turned ON by a SETON operation, all lower level indicators remain OFF. Control level indicators need not be specified in any particular order. Operations will be executed in the order in which they are specified, providing the control level indicator, columns 7-8, is ON.

The LO indicator is turned ON during every cycle of the program after detail output. If no other control level indicators are assigned, but it is desired to perform total calculation and total output operations, the LO indicator may be used to condition those operations. LO indicator coding is illustrated in figure 9-2.



CALCULATION SPECIFICATIONS

PAGE 1 2 □□ FORM TYPE — CONTROL LEVEL — HALF ADJUST — DECIMAL POSITIONS — PROGRAM IDENTIFICATION 75 80 □□□□□□

LINE 3-5	FORM TYPE 6	CONTROL LEVEL 7-8	INDICATORS 9-11	AND 12-14	AND 15-17	FACTOR 1 18-27	OPERATION 28-32	FACTOR 2 33-42	RESULT FIELD 43-48	FIELD LENGTH 49-51	DECIMAL POSITIONS 52	HALF ADJUST 53	Resulting Indicators Plus / High / High 54-55	Minus / Low / Low 56-57	Zero / Equal / Equal 58-59	COMMENTS 60-74
01	C		40			WAGE	ADD	TØTWAG	TØTWAG	8	2					
02	C		50			WAGE	ADD	TØTWAG	TØTWAG							
03	C						SETØF						70			
04	C		40				SETØN						70			
05	C	LO	50	70		TØTWAG	ADD	TØTAL	TØTAL	8	2					
06	C	LR				TØTWAG	ADD	TØTAL	TØTAL							
07	C	*														
08	C	*	THE LO INDICATØR MAY BE USED TØ CØNDITIØN DETAIL ØPERATIØNS													
09	C	*	SØ THAT THEY WILL BE PERFØRMED AT TØTAL CALCULATIØN TIME													
10	C	*	(AHEAD ØF NØRMAL DETAIL CALCULATIØNS).													
11	C															
12	C															
13	C															
14	C															
15	C															
	C															
	C															
	C															

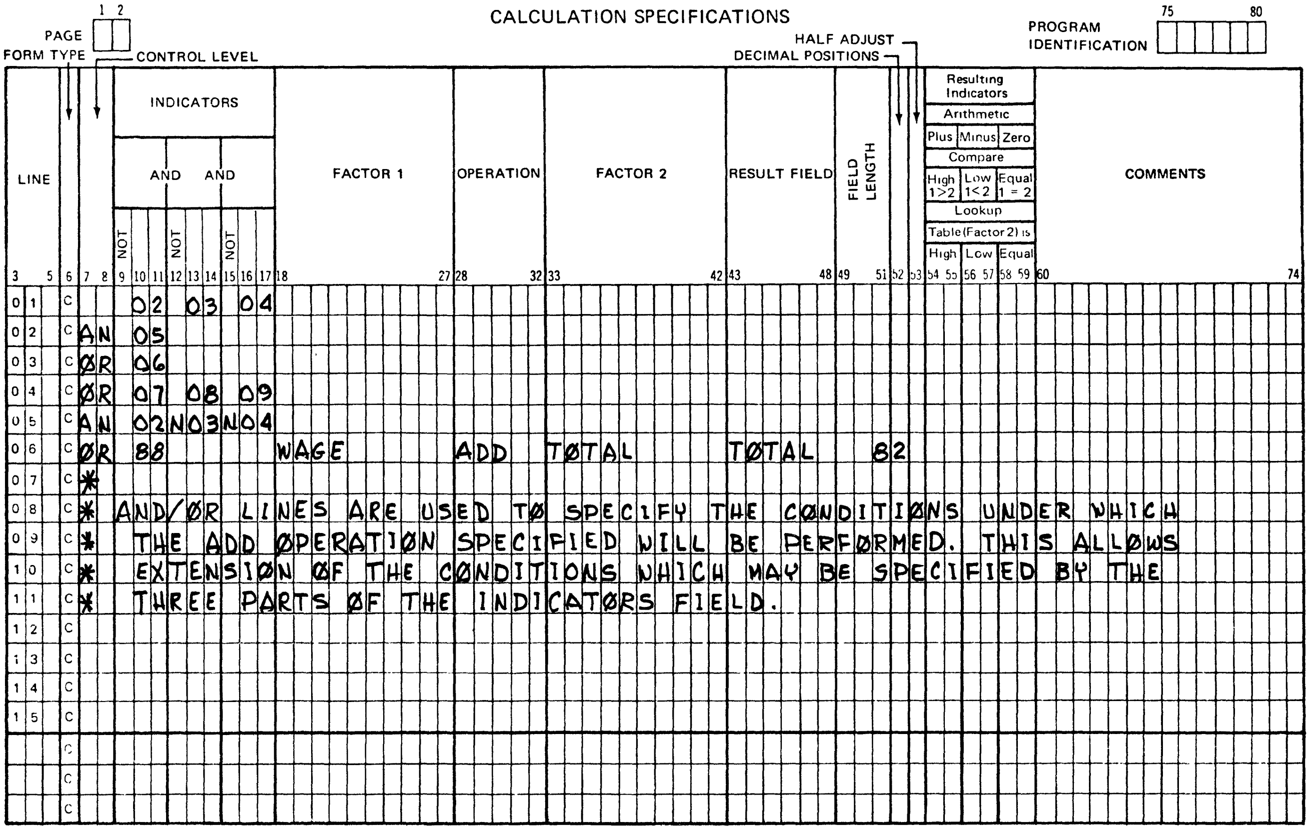
G14053

Figure 9-2. LO Indicator Coding Example

LR The LR indicator automatically turns ON when the last record of the appropriate file has been read and processed (all control level indicators L1-L9 also turn ON). This indicator is used to condition those operations that are to be performed at End-of-File. The LR indicator can also be turned ON by its specification as a calculation resulting indicator. In that case, if the LR indicator is turned ON, the lower control levels are not set ON as a result. When the LR indicator is used as a resulting indicator in Calculation Specifications, the operation should be conditioned by NLR to avoid the inadvertent resetting of the LR indicator.

SR The SR entry is not an indicator; rather, it is used to indicate that the specification on the same line is part of a subroutine. All subroutine lines must be specified after all other calculation lines. Subroutines must be specified after all detail and total lines are specified. For a subroutine specification line, columns 7-8 can contain SR, OR, AN or blank. The BEGSR operation determines the start of a subroutine.

AN-OR This field may also be used to specify that lines of indicators are in an AND or OR relationship. There is no limit on the number of AND or OR lines that may be specified; however, it is recommended that the user not exceed seven if compatibility with other Burroughs systems is desired. The last line of a group in an AND or OR relationship contains the Operation Code and all operands. All previous lines in the group must contain blanks in columns 18-59. The first line of a group may contain an L0-L9, LR, or SR entry if the entire group is conditioned by a control level indicator or is part of a subroutine. Each AND/OR line, and the previous line, must contain at least one indicator in columns 9-17.



CALCULATION SPECIFICATIONS

PAGE 1 2 □□ FORM TYPE CONTROL LEVEL HALF ADJUST DECIMAL POSITIONS PROGRAM IDENTIFICATION 75 □□□□□□ 80

LINE | INDICATORS (NOT AND NOT AND NOT) | FACTOR 1 | OPERATION | FACTOR 2 | RESULT FIELD | FIELD LENGTH | Resulting Indicators: Arithmetic Plus Minus Zero; Compare High 1>2 Low 1<2 Equal 1 = 2; Lookup Table (Factor 2) is High Low Equal | COMMENTS

```
01 C   02 03 04
02 CAN 05
03 CØR 06
04 CØR 07 08 09
05 CAN 02N03N04
06 CØR 88      WAGE      ADD  TØTAL     TØTAL   82
07 C*
08 C* AND/ØR LINES ARE USED TØ SPECIFY THE CØNDITIØNS UNDER WHICH
09 C*  THE ADD ØPERATIØN SPECIFIED WILL BE PERFØRMED. THIS ALLØWS
10 C*  EXTENSIØN ØF THE CØNDITIØNS WHICH MAY BE SPECIFIED BY THE
11 C*  THREE PARTS ØF THE INDICATØRS FIELD.
12 C
13 C
14 C
15 C
   C
   C
   C
```

G14054

Figure 9-3. AND/OR Relationship Coding Example

9-17 INDICATORS

This field is divided into three subfields such that up to three indicators on each line may be specified to condition a calculation operation. Each subfield is divided into two parts, as follows:

a. NOT (one column).

b. INDICATOR (two columns).

The NOT portion is used to specify that the associated indicator must be OFF in order for the operation to occur. If this condition is desired, an N must be entered in the NOT position; otherwise, the NOT portion must be left blank.

The INDICATOR portion is used to specify the indicator to be tested for ON (NOT = blank) or OFF (NOT = N). Only the following entries are allowed in this portion of the INDICATOR field:

Entry	Definition
Blank	Operation not conditioned by any indicator.
01-99	Operation conditioned by indicator used elsewhere in the program.
L0-L9	Operation conditioned by control level indicator previously assigned.
LR	Operation conditioned by last record indicator.
MR	Operation conditioned by matching record indicator.
H0-H9	Operation conditioned by halt indicator used elsewhere in the program.
U1-U8	Operation conditioned by external indicator previously defined.
OA-OG, OV	Operation conditioned by overflow indicator previously defined.

All three indicators on one line are in an AND relationship. All indicators on one line (or grouped lines), plus the Control Level Indicator (if used) must be ON or OFF as specified, in order for the associated operation to take place.

Each of the indicators is discussed in the paragraphs that follow.

Indicators 01-99

a. To condition calculation operations that are to be performed only for specific input record types (record identifying indicators).

b. To condition calculation operations that are to be performed only when an input field satisfies certain conditions (field indicators).

c. To condition calculation operations according to the results from previous operations that were specified in the Calculation Specifications (resulting indicators).

Control Level Indicators (L0-L9)

The control level indicators are normally defined in the CONTROL LEVEL field of the Input Specifications, but can be used as calculation resulting indicators. For either usage, the control level indicators entered in columns 9-17 signify an operation ~~to be~~ performed only during ~~the first~~ cycle and after a control break.

The following information is applicable when control level indicators are specified in the INDICATORS field:

a. If the operation is performed during detail calculations, it occurs only while processing the first record of the new group.

b. If the operation is performed during total calculations, it occurs only when the control break specified in columns 7-8 and the control break specified in the INDICATORS field are both satisfied. When both control breaks are satisfied, the last record of a control group has been processed.

c. If the operation is part of a subroutine and is called from detail calculations, it occurs only while processing the first record of the new group.

d. If the operation is part of a subroutine and is called from total calculations, the operation occurs only if the EXSR is of an equal or higher control level than the control level indicator specified in the INDICATORS field.

Last Record Indicator (LR)

The last record indicator is used to condition operations that are to be performed at end-of-job.

Matching Record Indicator (MR)

The matching record indicator is used to indicate whether an operation is to be performed only relative to the status of the matched records found.

The following rules must be observed when using the MR indicator:

a. When used during detail calculations, MR refers to the status of the match fields of the record last selected.

b. When used during total calculations, MR refers to the status of the match fields of the record prior to the last record selected.

c. When used in a subroutine that is called during detail calculations, MR refers to the status of the match fields of the record last selected. When used in a subroutine that is called during total calculations, MR refers to the status of the match fields of the record prior to the last record selected.

Halt Indicators (H0-H9)

Halt indicators previously assigned in the Field Indicators field of the Input Specifications or Resulting Indicators field of the Calculation Specifications can be used to condition operations that are to be performed only when an error condition occurs.

Since the program does not halt until the record in error has been completely processed, action must be taken to prevent reporting the erroneous results. By using the halt indicators in conjunction with an N entry in the NOT portion of the INDICATORS field, an operation can be inhibited when the specified halt indicator is ON.

External Indicators (U1-U8)

External indicators can be used to condition operations that are to be performed only when the specified indicator has been set at program execution time. Refer to the description of the FILE CONDITION field in the section on File Description Specifications for a complete description of external indicators.

Overflow Indicators (OA-OG, OV)

These indicators, previously assigned in the File Description Specifications, can be used to condition operations that are to be performed when the overflow line of a printer file has been reached.

18-27 FACTOR 1

This field is used to supply data to be operated on by the Operation Code specified in columns 28-32.

Allowable entries are:

a. The name of any field previously defined or to be defined later.

b. A literal (alphanumeric or numeric).

c. A label (tag name, subroutine name, END subroutine name).

d. A vector name or an element of a vector. A vector element is composed of the vector name, followed by a comma, followed by the desired index value.

e. The special words UDATE, UMONTH, UDAY, UYEAR, PAGE, and PAGEn.

Entries, including numeric literals, in this field must be left-justified.

28-32 OPERATION

This field is used to specify the proper operation to be performed using FACTOR 1 and FACTOR 2. The Operation Code must be left-justified in the field.

Operations are performed in the order in which they appear on the Calculation Specifications sheet; however, all operations conditioned by control level indicators specified in the CONTROL LEVEL field must appear after those operations not conditioned by control level indicators and prior to all subroutines.

Each of the operation codes is discussed in detail in the second portion of this section of the manual.

33-42 FACTOR 2

This field is used to supply data to be operated on by the Operation Code specified in columns 28-32. Allowable entries are:

a. The name of any field defined elsewhere in the program.

b. A literal (alphanumeric or numeric).

c. The name of a subroutine (EXSR operation only).

d. A vector name or an element of a vector. A vector element is composed of the vector name, followed by a comma, followed by the desired index value.

e. The special words UDATE, UMONTH, UDAY, UYEAR, PAGE, and PAGEn.

f. A label (GOTO Operation only).

g. A filename (DEBUG, DSPLY, CHAIN, READ, RECV, SEND or FORCE operation only).

Entries in this field must be left-justified.

43-48 RESULT FIELD

The use of this field is dependent upon the particular operation being performed. The entry in this field is used to name the field, vector, or vector element that is to be used to store the results of the operation specified on this specification line.

All entries in the RESULT FIELD must be left-justified. The entry can be the name of a vector or vector element defined elsewhere in the program, or the name of a field described elsewhere in the program or on this specification line. Special words, except those in the following list, can also be entered in the RESULT FIELD.

The following special words must never be used in the RESULT FIELD:

a. UDATE

b. UMONTH

c. UDAY

d. UYEAR

Also, literals must never be used in the RESULT FIELD.

Definition of a new field is also allowed, and is accomplished by entering the new field name, along with FIELD LENGTH and DECIMAL POSITIONS entries for that field. Refer to Section 2 for a complete description of field names.

49-51 FIELD LENGTH

This field is used to specify the length of a result field that has not been previously defined. Numeric and alphanumeric fields are limited to the size specified in Section 2.

If this entry is left blank the field must be defined elsewhere. This field must be blank if RESULT FIELD is blank. It is allowable to enter the length of a field that has been previously defined; however, the length and number of decimal positions specified must be the same as that previously defined.

Entries in this field must be right-justified, and leading zeros are optional.

52 DECIMAL POSITIONS

This field is used to specify the number of positions to the right of the implied decimal point in a numeric result field. If the RESULT FIELD is alphanumeric, or the FIELD LENGTH is blank, this field must be left blank. The valid entries are:

Entry	Definition
Blank	Alphanumeric field.
0-9	Number of decimal positions in a numeric field.

When defining numeric fields, the number of decimal positions must be the number of decimal specified; if the field only contains integral values, a zero must be entered to indicate no decimal positions.

The number of decimal positions specified must not exceed the length of the RESULT FIELD, as specified by the FIELD LENGTH entry.

53 HALF ADJUST

This field is used to indicate whether or not the contents of the RESULT FIELD are to be rounded. Rounding is accomplished by adding 5 (or -5 for negative values) to the digit to the right of the last decimal position of the RESULT FIELD. Then all digits to the right of the last decimal position are dropped. Valid entries for this field are:

Entry	Definition
Blank	Do not half adjust.
H	Half adjust.

Entries in this field are allowable only for certain arithmetic operations.

54-59 RESULTING INDICATORS

The indicators U1-U8, MR, or IP must not be entered in this field. Any other RPG indicator or blanks are valid entries.

Valid entries for this field are:

Entry	Definition
01-99	Any numeric indicator.
L1-L9	Any control level indicator.
LR	Last record indicator.
H0-H9	Any halt indicator.
OA-OG,OV	Any overflow indicator.

Indicators specified in these columns are set OFF immediately before the operation is performed. Immediately after the operation, indicators are set ON to indicate the result of the operations. The use of the result indicators is dependent on the type of operation specified.

Resulting Indicators must not be specified when the RESULT FIELD contains an array name, the only exception being a LOKUP operation.

Figure 9-4 illustrates the use of resulting indicators and various operation codes. The specific uses of the result fields is further described in the discussion on the various operation code types in this section.

60-74 COMMENTS

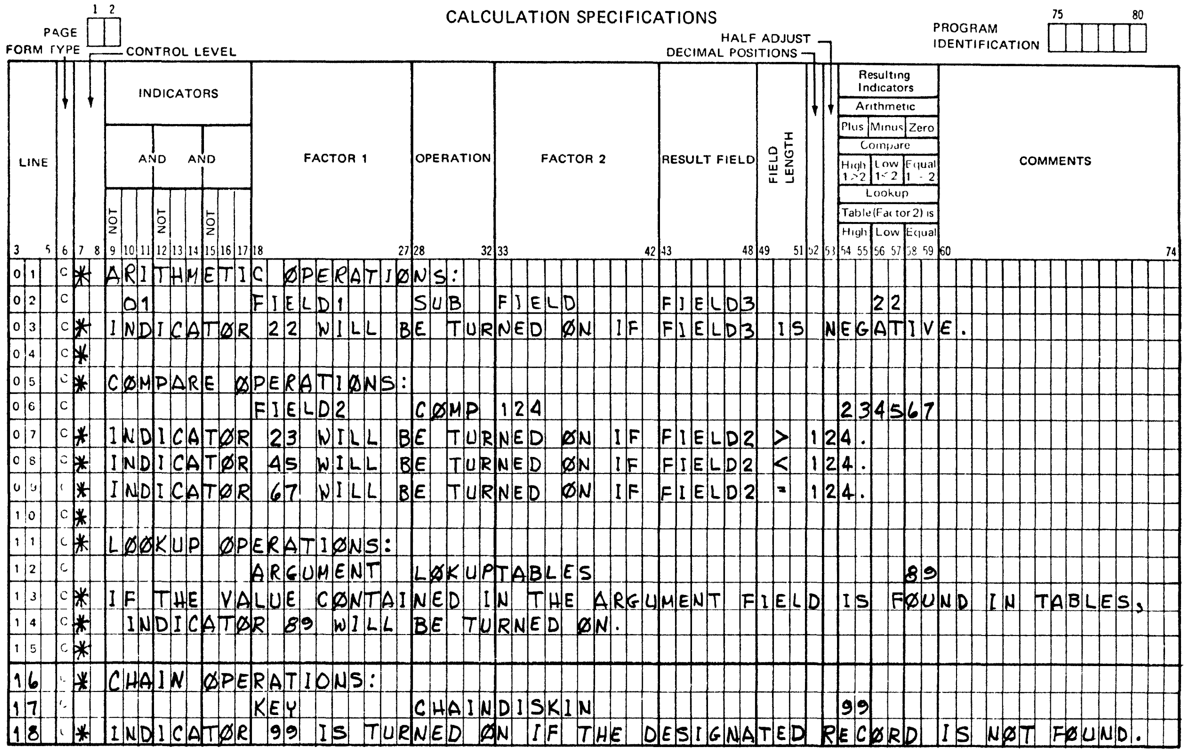
This field is available for inclusion of comments and documentary remarks, and may contain any valid EBCDIC characters.

75-80 PROGRAM IDENTIFICATION

Refer to Chapter 2 for a complete description.

OPERATION CODES

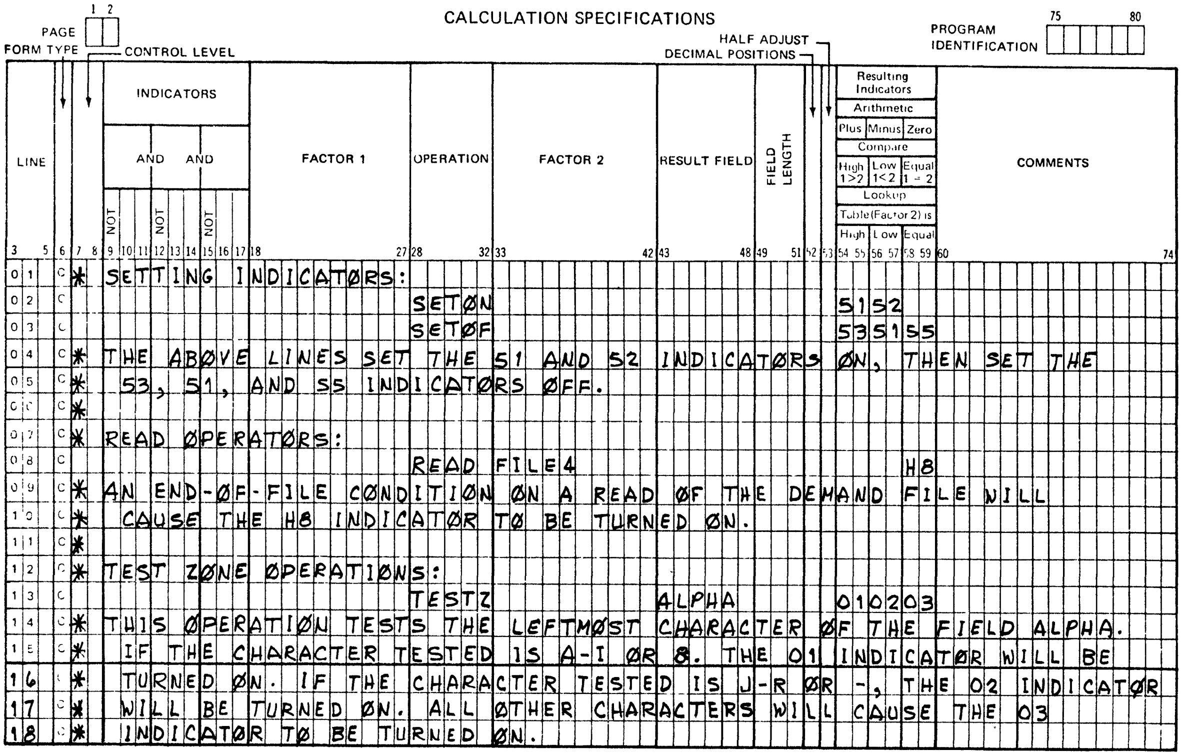
Various categories of Operation Codes are provided to allow data manipulation required by the RPG programmer. Operation Codes are entered only in the OPERATION field of the Calculation Specifications, and specify what type of



CALCULATION SPECIFICATIONS

PAGE 1 2 · FORM TYPE · CONTROL LEVEL · INDICATORS (AND, AND, NOT) · LINE · FACTOR 1 · OPERATION · FACTOR 2 · RESULT FIELD · FIELD LENGTH · DECIMAL POSITIONS · HALF ADJUST · Resulting Indicators: Arithmetic (Plus, Minus, Zero); Compare (High 1>2, Low 1<2, Equal 1 = 2); Lookup Table (Factor 2) is (High, Low, Equal) · COMMENTS · PROGRAM IDENTIFICATION 75 80

```
3 5 6 7 8 9 10 11 12 13 14 15 16 17 18   27 28  32 33      42 43  48 49 51 52 53 54 55 56 57 58 59 60   74
01 C* ARITHMETIC ØPERATIØNS:
02 C   01      FIELD1    SUB  FIELD     FIELD3       22
03 C* INDICATØR 22 WILL BE TURNED ØN IF FIELD3 IS NEGATIVE.
04 C*
05 C* CØMPARE ØPERATIØNS:
06 C           FIELD2    CØMP 124                  234567
07 C* INDICATØR 23 WILL BE TURNED ØN IF FIELD2 > 124.
08 C* INDICATØR 45 WILL BE TURNED ØN IF FIELD2 < 124.
09 C* INDICATØR 67 WILL BE TURNED ØN IF FIELD2 = 124.
10 C*
11 C* LØØKUP ØPERATIØNS:
12 C           ARGUMENT  LØKUPTABLES                       89
13 C* IF THE VALUE CØNTAINED IN THE ARGUMENT FIELD IS FØUND IN TABLES,
14 C*  INDICATØR 89 WILL BE TURNED ØN.
15 C*
16 C* CHAIN ØPERATIONS:
17 C           KEY       CHAINDISKIN                   99
18 C* INDICATØR 99 IS TURNED ØN IF THE DESIGNATED RECØRD IS NØT FØUND.
```



CALCULATION SPECIFICATIONS

PAGE 1 2 · FORM TYPE · CONTROL LEVEL · INDICATORS (AND, AND, NOT) · LINE · FACTOR 1 · OPERATION · FACTOR 2 · RESULT FIELD · FIELD LENGTH · DECIMAL POSITIONS · HALF ADJUST · Resulting Indicators: Arithmetic (Plus, Minus, Zero); Compare (High 1>2, Low 1<2, Equal 1 = 2); Lookup Table (Factor 2) is (High, Low, Equal) · COMMENTS · PROGRAM IDENTIFICATION 75 80

```
3 5 6 7 8 9 10 11 12 13 14 15 16 17 18   27 28  32 33      42 43  48 49 51 52 53 54 55 56 57 58 59 60   74
01 C* SETTING INDICATØRS:
02 C                     SETØN                     5152
03 C                     SETØF                     535155
04 C* THE ABØVE LINES SET THE 51 AND 52 INDICATØRS ØN, THEN SET THE
05 C*  53, 51, AND 55 INDICATØRS ØFF.
06 C*
07 C* READ ØPERATØRS:
08 C                     READ FILE4                    H8
09 C* AN END-ØF-FILE CØNDITIØN ØN A READ ØF THE DEMAND FILE WILL
10 C*  CAUSE THE H8 INDICATØR TØ BE TURNED ØN.
11 C*
12 C* TEST ZØNE ØPERATIØNS:
13 C                     TESTZ          ALPHA      010203
14 C* THIS ØPERATIØN TESTS THE LEFTMØST CHARACTER ØF THE FIELD ALPHA.
15 C*  IF THE CHARACTER TESTED IS A-I ØR &. THE 01 INDICATØR WILL BE
16 C*  TURNED ØN. IF THE CHARACTER TESTED IS J-R ØR -, THE 02 INDICATØR
17 C*  WILL BE TURNED ØN. ALL ØTHER CHARACTERS WILL CAUSE THE 03
18 C*  INDICATØR TØ BE TURNED ØN.
```

G14055

Figure 9-4. Resulting Indicators Coding Example

arithmetic, logical, or input/output operation is to be performed on the associated operands.

Table 9-1 summarizes each of the operation codes discussed in the following paragraphs.

ARITHMETIC OPERATIONS

In the descriptions of arithmetic operations in the following paragraphs, an indexed vector or a table name is valid anywhere that a field name is valid except as noted. Arithmetic operations are allowed only on numerically defined data items. FACTOR 1 and FACTOR 2 fields must contain the names of numeric fields or numeric literals. The RESULT FIELD must be numeric. All results are signed and decimal alignment is performed. If the RESULT FIELD is not large enough to hold the result of an arithmetic operation, the result is truncated at either or both ends following decimal alignment. If the RESULT FIELD is too large to hold the result of an arithmetic operation, then leading and trailing zeroes are inserted following decimal alignment.

FACTOR 1, FACTOR 2 and the RESULT FIELD may all be different fields or any two or all three may be the same field. HALF ADJUST may be specified for all operations except SQRT and MVR (i.e., DIV when followed by MVR). Resulting indicators may be specified for all operations provided the RESULT FIELD is not a whole array.

The use of result indicators in connection with arithmetic operations is as shown below:

a. Plus (Columns 54-55). Any indicator entered in this field will be turned on if the result of an arithmetic operation is positive.

b. Minus (Columns 56-57). Any indicator entered in this field will be turned on if the result of an arithmetic operation is negative.

c. Zero (Columns 58-59). Any indicator entered in this field will be turned on if the result of an arithmetic operation is zero.

Each of the arithmetic operations is discussed in the paragraphs that follow.

ADD This operation adds the contents of FACTOR 2 to the contents of FACTOR 1 and stores the sum in the RESULT FIELD. FACTOR 1 and FACTOR 2 are not affected by this operation, unless one of them is also designated as the RESULT FIELD. HALF ADJUST may be specified.

SUB This operation subtracts the contents of FACTOR 2 from the contents of FACTOR 1 and places the difference in the RESULT FIELD. FACTOR 1 and FACTOR 2 are not affected by this operation, unless one of them is also designated as the RESULT FIELD.

Note that subtracting two fields which are the same gives the same result as clearing the RESULT FIELD to zero, and can be used as a method of clearing fields to zero.

MULT This operation multiplies the contents of FACTOR 1 by the contents of FACTOR 2 and stores the product in the RESULT FIELD. FACTOR 1 and FACTOR 2 are not affected by this operation, unless one of them is also designated as the RESULT FIELD. HALF ADJUST may be specified.

Table 9-1: Summary of Operation Codes

Operation Type	Operation Code Columns 28-32	Definition	Control Level	Indicators	Factor 1	Factor 2	Result Field	Field Length	Decimal Position	Half Adjust	Resulting Indicators
Arithmetic Operations	ADD	Add Factor 2 to Factor 1	0	0	R	R	R	0	0	0	0
	Z-ADD	**Clear Result Field and add Factor 2**	0	0	B	R	R	0	0	0	0
	SUB	Subtract Factor 2 from Factor 1	0	0	R	R	R	0	0	0	0
	Z-SUB	Clear Result Field and subtract Factor 2	0	0	B	R	R	0	0	0	0
	MULT	Multiply Factor 1 by Factor 2	0	0	R	R	R	0	0	0	0
	DIV	Divide Factor 1 by Factor 2	0	0	R	R	R	0	0	0	0
	MVR	Move remainder of preceding division to a Result Field	0	0	B	B	R	0	0	B	0
	XFOOT	Sum elements of an array and put sum in Result Field	0	0	B	R	R	0	0	0	0
	SQRT	Derive the square root of Factor 2	0	0	B	R	R	0	0	B	B
Move Operation	MOVE	Move Factor 2 into Result Field, right-justified	0	0	B	R	R	0	0	B	0
	MOVEL	Move Factor 2 into Result Field, left-justified	0	0	B	R	R	0	0	B	0
	MOVEA	Moves data (left justified) into, or out of, an alphanumeric array.	0	0	B	R	R	B	B	B	B
Move Zone Operation	MLLZO	Move zone from low-order position of Factor 2 to low-order position of Result Field	0	0	B	R	R	0	0	B	B
	MHHZO	Move zone from high-order position of alphanumeric Factor 2 to high order of alphanumeric Result Field	0	0	B	R	R	0	B	B	B
	MLHZO	Move zone from low-order position of Factor 2 to high-order position of alphanumeric Result Field	0	0	B	R	R	0	B	B	B
	MHLZO	Move zone from high-order position of alphanumeric Factor 2 to low-order position of Result Field	0	0	B	R	R	0	0	B	B
Compare Operations	COMP	Compare Factor 1 to Factor 2	0	0	R	R	B	B	B	B	R
	TESTZ	**Tests the zone portion of the leftmost character of Result Field**	0	0	B	B	R	0	B	B	R
	TESTN	Tests a numeric value of the field specified by the Result Field.	0	0	B	B	R	0	B	B	R
Binary Field Operations	BITON	Set on specified bits	0	0	B	R	R	0	B	B	B
	BITOF	Set off specified bits	0	0	B	R	R	0	B	B	B
	TESTB	Test specified bits	0	0	B	R	R	0	B	B	R
Setting Indicators	SETON	Set one, two, or three specific indicators on	0	0	B	B	B	B	B	B	R
	SETOF	Set one, two, or three specific indicators off	0	0	B	B	B	B	B	B	R
Program Branching Operations	GOTO	Branch to another calculation specification line	0	0	B	R	B	B	B	B	B
	TAG	Identify the name in Factor 1 as a destination label to which GOTO may branch	0	B	R	B	B	B	B	B	B
Transfer Control Function	ZIP	Initialize System commands	0	0	B	R	B	B	B	B	B
Lookup Operations	LOKUP	Table or array lookup	0	0	R	R	0	0	0	B	R
Subroutines	BEGSR	Beginning of the subroutine	0	B	R	B	B	B	B	B	B
	ENDSR	End of the subroutine	0	B	0	B	B	B	B	B	B
	EXSR	Call to execute the subroutine	0	0	B	R	B	B	B	B	B
Program Control of Input and Output	FORCE	Forcing record to be read next	B	0	B	R	B	B	B	B	B
	EXCPT	Causes output	0	0	B	B	B	B	B	B	B
	DSPLY	A field is printed on the console printer and or data is entered via the console printer into a field	0	0	0	R	0	0	0	B	B
	READ	A record is read from a demand file	0	0	B	R	B	B	B	B	EI
	CHAIN	A record is read from a disk file	0	0	R	R	B	B	B	B	EI
Debug Operation	DEBUG	Aid in finding programming errors	0	0	0	R	0	B	B	B	B
Time	TIME	Time of day / time and day.	0	0	B	B	R	0	0	B	B
Data Comm.	SEND	Writes to the remote file specified in FACTOR 2.	0	0	B	R	B	B	B	B	0
	RECV	Reads from the remote file specified in FACTOR 2.	0	0	B	R	B	B	B	B	0
Process Within Limits	SETLL	Sets the lower limit for a demand file processed sequentially within limits.	0	0	R	R	B	B	B	B	B

NOTES

0 - Optional
R - Required
B - Blank

EI - Only the Equal Indicator columns may be used. The indicator will be turned on if end-of-file occurs on the demand file while reading or attempting to read.

Table 9-2 provides an example of the contents of the RESULT FIELD for the multiplication operation, showing various field lengths and decimal positions. In the example, all fields that are permitted are shown, although some contain incomplete results. Fields not permitted are blank; HALF ADJUST is not specified. Note that a field length of eight with five decimal positions gives all the significant digits without adding zeros either left or right.

DIV This operation divides the contents of FACTOR 1 by the contents of FACTOR 2 and places the quotient in the RESULT FIELD. FACTOR 1 and FACTOR 2 are not affected by this operation, unless one of them is also designated as the RESULT FIELD.

Any remainder resulting from the divide operation will be lost unless the next operation specified is the MOVE REMAINDER operation (MVR); if so, the result of the divide operation cannot be half adjusted. The RESULT FIELD cannot contain a whole array when an MVR is the next operation.

If FACTOR 2 is equal to zero, the operator is notified and the program discontinues.

MVR This operation moves the REMAINDER from a previous DIV operation to the designated RESULT FIELD. The MVR operation must immediately follow the DIV operation, and FACTOR 1 and FACTOR 2 must be left blank. The RESULT FIELD for the MVR operation must be the same length as FACTOR 2 for the DIV operation. Both the DIV and the MVR may be conditioned on indicators, which need not be the same for both operations. However, the programmer must ensure that the MVR operation is never performed without the DIV, otherwise the result of the MVR is undefined. HALF ADJUST must not be specified.

The following considerations should be made when specifying the size of the RESULT FIELD:

a. The number of significant decimal places in the REMAINDER is the larger of:

   1) The number of decimal positions in FACTOR 1 of the previous DIV operation.

   2) The sum of the decimal positions in FACTOR 2 and the RESULT FIELD of the previous DIV operation.

b. The maximum integer positions in the REMAINDER equals the integer positions in FACTOR 2 of the previous DIV operation.

c. The RESULT FIELD must not be a whole array.

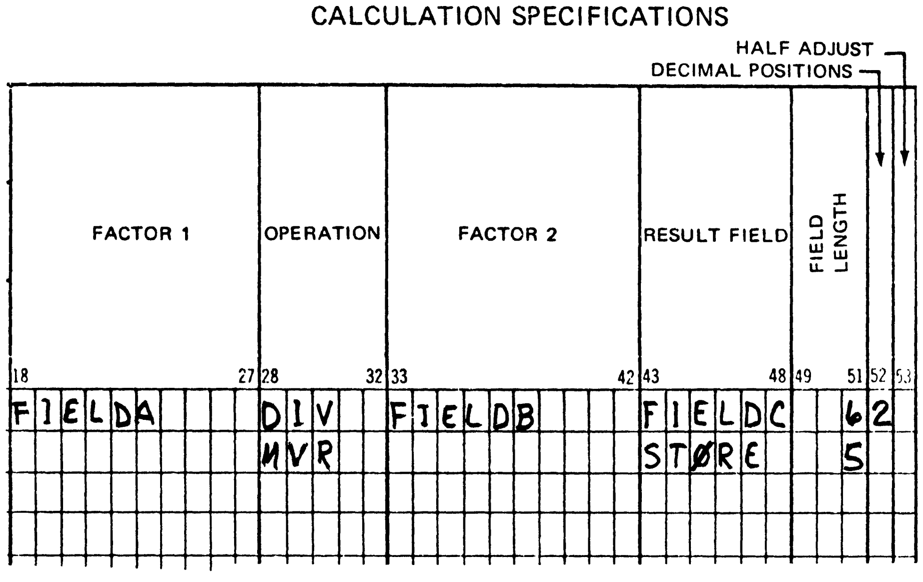
Figure 9-5 provides a DIV and MVR coding example.

Table 9-2. RESULT FIELD Contents for Various Field Lengths and Decimal Positions - MULT Operation

	Decimal Positions									
Field Length	0	1	2	3	4	5	6	7	8	9
1	2	.6								
2	92	2.6	.61							
3	192	92.6	2.61	.611						
4	0192	192.6	92.61	2.611	.6111					
5	00192	0192.6	192.61	92.611	2.6111	.61116				
6	000192	00192.6	0192.61	192.611	92.6111	2.61116	.611160			
7	0000192	000192.6	00192.61	0192.611	192.6111	92.61116	2.611160	.6111600		
8	00000192	0000192.6	000192.61	00192.611	0192.6111	192.61116	92.611160	2.6111600	.61116000	
9	000000192	00000192.6	0000192.61	000192.611	00192.6111	0192.61116	192.611160	92.6111600	2.61116000	.611160000
10	0000000192	000000192.6	00000192.61	0000192.611	000192.6111	00192.61116	0192.611160	192.6111600	92.61116000	2.611160000

NOTE

RESULT FIELD contents for the multiplication operation: 89.67 MULT 2.148.



CALCULATION SPECIFICATIONS

FACTOR 1 18–27	OPERATION 28–32	FACTOR 2 33–42	RESULT FIELD 43–48	FIELD LENGTH 49–51	DECIMAL POSITIONS 52	HALF ADJUST 53
FIELDA	DIV	FIELDB	FIELDC	6	2	
	MVR		STØRE	5		

G14056

Figure 9-5. DIV and MVR Coding Example

SQRT The operation derives the SQUARE ROOT of the contents of FACTOR 2 and places it in the RESULT FIELD. FACTOR 1 must be left blank.

To obtain reasonable precision from SQRT, the following points should be observed:

a. For every digit left of the decimal place in the RESULT FIELD, there should be two digits left of the decimal placed in FACTOR 2.

b. For every digit right of the decimal point in the RESULT FIELD, there should be two digits right of the decimal point in FACTOR 2.

The result of the SQRT operation is automatically adjusted; therefore, a HALF ADJUST entry for SQRT is not allowed.

If the FACTOR 2 operand is negative, the program will display a "SQRT" error message and halt. The operator may resume processing, in which case the RESULT FIELD will set to zero.

Table 9-3 provides an example of the contents of the RESULT FIELD for the square root operation. In the example, all fields that are permitted are shown, although some contain incomplete results. Fields not permitted are blank.

XFOOT This operation is used to crossfoot (sum) the elements of a numeric array. All the elements of the array specified by FACTOR 2 are summed and the total placed in the RESULT FIELD. FACTOR 1 must be left blank. The RESULT FIELD must not be a whole array.

Resulting indicators may be specified. If the RESULT FIELD is an element of the array named in FACTOR 2, the value of that element before the XFOOT operation is used in obtaining the total. HALF ADJUST may be specified.

Table 9-3. RESULT FIELD Contents for Various Field Lengths and Decimal Positions - SQRT Operation

Field Length	Decimal Positions								
	0	1	2	3	4	5	6	7	8
1	5	.7							
2	55	4.7	.72						
3	055	54.7	4.72	.721					
4	0055	054.7	54.72	4.721	.7213				
5	00055	0054.7	054.72	54.721	4.7213	.72130			
6	000055	00054.7	0054.72	054.721	54.7213	4.72130			
7	0000055	000054.7	00054.72	0054.721	054.7213	54.72130	.721305		
8	00000055	0000054.7	000054.72	00054.721	0054.7213	054.72130	4.721305	.7213048	
9	000000055	00000054.7	0000054.72	000054.721	00054.7213	0054.72130	54.721305	4.7213048	.72130482
10	0000000055	000000054.7	00000054.72	0000054.721	000054.7213	00054.72130	054.721305	54.7213048	4.72130482
11	00000000055	0000000054.7	000000054.72	00000054.721	0000054.7213	000054.72130	0054.721305	054.7213048	54.72130482
12	000000000055	00000000054.7	0000000054.72	000000054.721	00000054.7213	0000054.72130	00054.721305	0054.7213048	054.72130482
13	0000000000055	000000000054.7	00000000054.72	0000000054.721	000000054.7213	00000054.72130	000054.721305	00054.7213048	0054.72130482
14	00000000000055	0000000000054.7	000000000054.72	00000000054.721	0000000054.7213	000000054.72130	0000054.721305	000054.7213048	00054.72130482
15	000000000000055	00000000000054.7	0000000000054.72	000000000054.721	00000000054.7213	0000000054.72130	00000054.721305	0000054.7213048	000054.72130482

NOTE

RESULT FIELD contents for the square root operation: SQRT 2994.42.

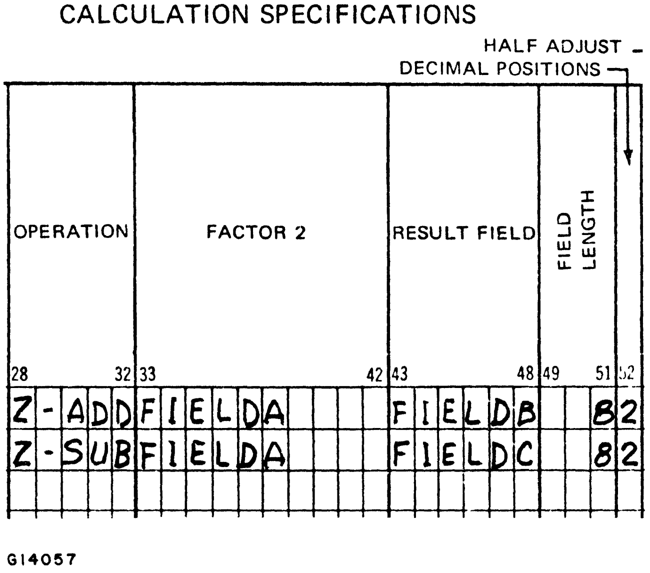
Z-ADD This operation sets the RESULT FIELD to zeros, and adds the contents of FACTOR 2 to the RESULT FIELD. FACTOR 1 must be left blank.

FACTOR 2 is not affected by this operation unless it is named in the RESULT FIELD. HALF ADJUST may be specified.

Z-SUB This operation sets the RESULT FIELD to zeros, then subtracts the contents of FACTOR 2 from the RESULT FIELD. FACTOR 1 must be left blank. This operation is used to change the sign of the field designated by FACTOR 2. HALF ADJUST may be specified.

Note that this is the same as Z-ADD, but the sign is changed.

In the example shown in figure 9-6, if the contents of FIELDA equals 678.9321, then the contents of FIELDB will equal 000678.93, and the contents of FIELDC will equal -000678.93.



CALCULATION SPECIFICATIONS

OPERATION 28–32	FACTOR 2 33–42	RESULT FIELD 43–48	FIELD LENGTH 49–51	DECIMAL POSITIONS 52	HALF ADJUST
Z-ADD	FIELDA	FIELDB	8	2	
Z-SUB	FIELDA	FIELDC	8	2	

G14057

Figure 9-6. Z-ADD and Z-SUB Coding Example

MOVE OPERATIONS

MOVE operations transfer the contents of the FACTOR 2 field to the RESULT FIELD. FACTOR 2 is not affected by the MOVE operation. FACTOR 1 must be left blank. Resulting indicators can be specified to indicate the status of RESULT FIELD (plus, minus, or zero/blank) in the same manner as field indicators on Input Specifications. If FACTOR 2 and the RESULT FIELD are the same length, MOVE and MOVEL operate in the same manner. FACTOR 2 can be a field, vector, vector element, or literal; however, a RESULT FIELD can only be a field, vector, or vector element.

An array name may be entered in FACTOR 2 as long as the RESULT FIELD also specifies an array. In this case, the designated move operation will be performed on each element of the array designated in FACTOR 2, with the result being placed in the corresponding elements of the array designated in the RESULT FIELD. The operation is terminated when the end of the shorter array is reached. If the RESULT FIELD is an array and FACTOR 2 is not, then

DECIMAL POSITIONS ARE IGNORED WHEN MOVING DATA FROM ONE NUMERIC FIELD TO ANOTHER. FOR EXAMPLE, X is a field w/ 3 decimals and Y is a field w/ 2 decimals, a MOVE of 10.000 from X to Y yields 100.00 in Y.

the designated move operation will be performed on FACTOR 2, with the result being placed in all elements of the RESULT FIELD array.

For the purpose of MOVE, MOVEL, the "sign" of the variable is defined as (1) the algebraic sign if the variable is numeric, or (2) the low order zone if the variable is alphanumeric.

Move operations may be specified between fields of different data types. This will convert an alphanumeric to a numeric and vice versa. Decimal points are ignored in all move operations. HALF ADJUST may not be specified with MOVE or MOVEL.

MOVE This operation moves characters from FACTOR 2 to the RESULT FIELD, starting with the rightmost character and continuing until either the source field is exhausted, or the destination field is filled. The sign of FACTOR 2 is moved to the sign position of the RESULT FIELD.

When an alphanumeric-to-numeric move is specified, the digit portion of each character is moved to the RESULT FIELD. Blanks are transferred as zeros. The zone portion of the rightmost alphanumeric character is used as the sign of the RESULT FIELD.

When a numeric-to-alphanumeric move is specified, each digit is converted to its corresponding internal character code as it is moved to the RESULT FIELD. The sign of the numeric field is placed in the zone portion of the rightmost character. If the numeric field is positive, then the zone portion of the rightmost character contains the systems standard positive sign (hexadecimal C). If the numeric field is negative, then the zone portion of the rightmost character contains the systems standard negative sign (hexadecimal D). If writing to the printer, characters A thru I, or ? are printed in the rightmost character of the field. The characters A thru I represent digits 1 thru 9, respectively. The question mark character (?) represents digit 0. Refer to Appendix B for more information concerning the hexadecimal values for the B 1800/B 1700 character set.

MOVE operations are shown in table 9-4.

MOVEL This operation moves, left-justified, characters from FACTOR 2 to the RESULT FIELD, starting with the leftmost character and continuing until either the source field is exhausted, or the destination field is filled.

When a numeric to alphanumeric move is specified, each digit is converted to its corresponding internal character code as it is moved to the RESULT FIELD.

When an alphanumeric-to-numeric move is specified, only the digit portion of each character is moved to the RESULT FIELD. Blanks are transferred as zeros.

The sign is transferred only if the RESULT FIELD is numeric and not greater in length than FACTOR 2 or if the result is alphanumeric and equal in length to FACTOR 2. The sign of FACTOR 2 is the algebraic sign if it is a numeric data item or low order zone if FACTOR 2 is alphanumeric.

MOVEL operations are shown in table 9-5.

Table 9-4. MOVE Operations

RESULT FIELD LARGER THAN FACTOR 2.

	FACTOR 2		RESULT FIELDS	
Alphanumeric	A B 4 S K A B 4 S K	Before MOVE Operation After MOVE Operation	1 2 3 4 5 6 7 8 9 1 2 3 4 A B 4 S K	Alphanumeric
Alphanumeric	A B 4 S K A B 4 S K	Before MOVE Operation After MOVE Operation	+1 2 3 4 5 6 7 8 9 -1 2 3 4 1 2 4 2 2	Numeric
Numeric	-9 8 7 6 5 4 3 -9 8 7 6 5 4 3	Before MOVE Operation After MOVE Operation	+1 2 3 4 5 6 7 8 9 -1 2 9 8 7 6 5 4 3	Numeric
Numeric	+9 8 7 6 5 4 3 +9 8 7 6 5 4 3	Before MOVE Operation After MOVE Operation	A B C D E F G H I A B 9 8 7 6 5 4 3	Alphanumeric

RESULT FIELD SMALLER THAN FACTOR 2.

	FACTOR 2		RESULT FIELD	
Alphanumeric	G E B K L M 4 S K G E B K L M 4 S K	Before MOVE Operation After MOVE Operation	5 6 7 8 4 L M 4 S K	Alphanumeric
Alphanumeric	G E B K L M 4 S K G E B K L M 4 S K	Before MOVE Operation After MOVE Operation	+5 6 7 8 4 -3 4 4 2 2	Numeric
Numeric	-9 8 7 6 5 4 3 -9 6 7 6 5 4 3 7/0	Before MOVE Operation After MOVE Operation	+5 6 7 8 9 -7 6 5 4 3 5/0	Numeric
Numeric	-9 8 7 6 5 4 3 -9 8 7 6 5 4 3	Before MOVE Operation After MOVE Operation	A B C D E 7 6 5 4 L	Alphanumeric

FACTOR 2 AND RESULT FIELD SAME LENGTH.

	FACTOR 2		RESULT FIELD	
Alphanumeric	A B 4 S K A B 4 S K	Before MOVE Operation After MOVE Operation	5 6 7 8 9 A B 4 S K	Alphanumeric
Alphanumeric	A B 4 S K A B 4 S K	Before MOVE Operation After MOVE Operation	+5 6 7 8 9 -1 2 4 2 2	Numeric
Numeric	-9 8 7 6 5 -9 8 7 6 5	Before MOVE Operation After MOVE Operation	+1 2 3 4 5 -9 8 7 6 5	Numeric
Numeric	-8 7 6 5 4 -8 7 6 5 4	Before MOVE Operation After MOVE Operation	A B S D E 8 7 6 5 M	Alphanumeric

LETTER K = MINUS 2   LETTER L = MINUS 3   LETTER M = MINUS 4

Table 9-5. MOVEL Operations

Factor 2		Result Field		Characters	Sign
Type	Length	Type	Length	Transferred	Transferred
Numeric	Equal	Numeric	Equal	Digits	To sign position
Numeric	Shorter	Numeric	Longer	Digits	No transfer
Numeric	Longer	Numeric	Shorter	Digits	To sign position
Numeric	Equal	Alphanumeric	Equal	Digits	To rightmost zone
Numeric	Shorter	Alphanumeric	Longer	Digits	To rightmost zone
Numeric	Longer	Alphanumeric	Shorter	Digits	No transfer
Alphanumeric	Equal	Numeric	Equal	Digits	To sign position
Alphanumeric	Shorter	Numeric	Longer	Digits	No transfer
Alphanumeric	Longer	Numeric	Shorter	Digits	To sign position
Alphanumeric	Equal	Alphanumeric	Equal	Bytes	To rightmost zone
Alphanumeric	Shorter	Alphanumeric	Longer	Bytes	To rightmost zone
Alphanumeric	Longer	Alphanumeric	Shorter	Bytes	No transfer

MOVEA The MOVEA (MOVE ARRAY) operation moves the data from the left-most position of the field specified in FACTOR 2 to the leftmost position of the field specified in the RESULT FIELD. When the end of either FACTOR 2 or the RESULT FIELD is reached, data transfer is terminated. When the RESULT FIELD is longer than FACTOR 2, the remainder of the data in the RESULT FIELD remains unchanged. The data contained in FACTOR 2 is never changed.

The MOVEA operation code may be used with either vector type, tables, or arrays.

It is possible to move the following with MOVEA:

a. Adjacent array elements into a field.

b. A field into adjacent array elements.

c. Adjacent elements in one array into the corresponding elements of another array.

d. The array in either FACTOR 2 and/or the RESULT FIELD can be indexed. The index can be either absolute or variable.

    1. For moves from FACTOR 2, the data transfer begins at the element indicated by the value of the index.

    2. For moves to the RESULT FIELD, the data reception begins at the element indicated by the value of the index.

e. A literal to a field.

f. A literal to an array.

g. A literal to an array element.

h. A table name or element when referenced by FACTOR 2 or the RESULT FIELD.

i. An array can be moved to itself by referencing the same array in both the RESULT FIELD and FACTOR 2.

Requirements for MOVEA Operation

The following requirements must be observed for the MOVEA operation:

a. The data format of both FACTOR 2 and the RESULT FIELD must be alphanumeric.

b. The operation code, FACTOR 2, and the RESULT FIELD must contain entries.

c. Control levels and conditioning indicators (columns 7-17) may contain valid entries. The remaining portion of the specification line must be left blank.

Restrictions for MOVEA Operations

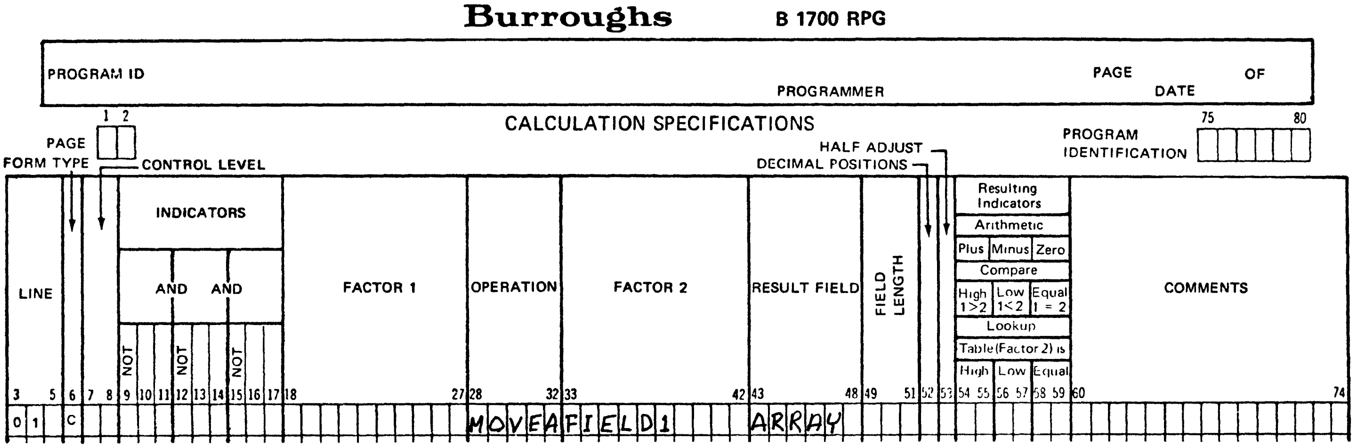
The following restrictions must be observed when using the MOVEA operation:

a. FACTOR 2 or the RESULT FIELD cannot reference numeric data.

b. The RESULT FIELD cannot contain a literal.

Coding Examples for MOVEA Operation

The six figures (figures 9-6a-f) on the following pages show the use of the MOVEA operation and the contents of FACTOR 2 and RESULT FIELD before and after execution of the MOVEA.



Burroughs B 1700 RPG

PROGRAM ID | PROGRAMMER | PAGE OF | DATE

CALCULATION SPECIFICATIONS

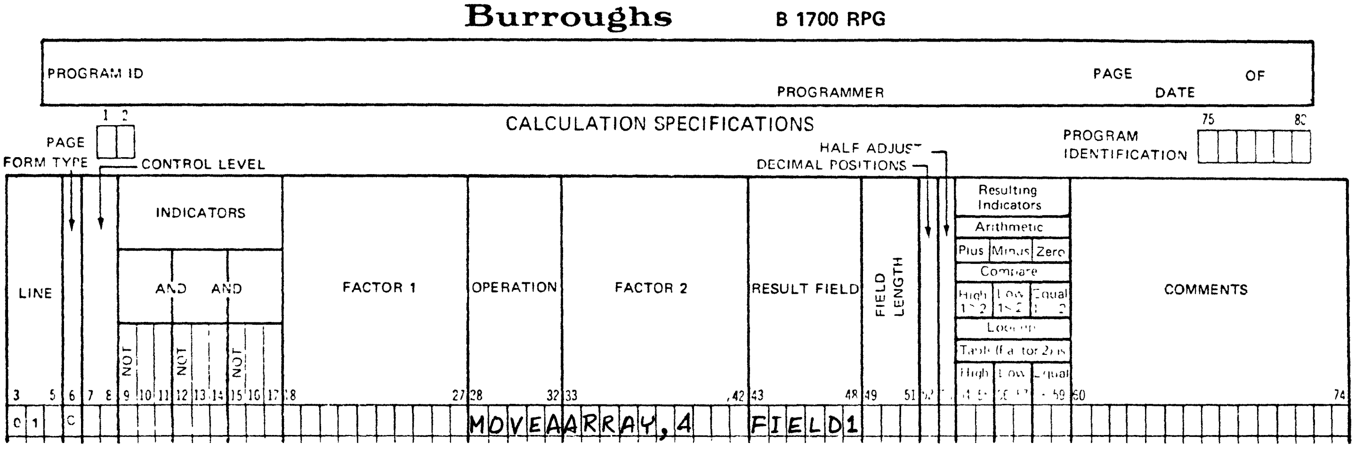
LINE	FORM TYPE	CONTROL LEVEL	INDICATORS	FACTOR 1	OPERATION	FACTOR 2	RESULT FIELD	FIELD LENGTH	DECIMAL POSITIONS	HALF ADJUST	Resulting Indicators	COMMENTS
01	C				MOVEA	FIELD1	ARRAY					

FIELD1		ARRAY			
1 2 3 4 5 6 7 8	BEFORE MOVEA	A B C	D E F	G H I	J K L
1 2 3 4 5 6 7 8	AFTER MOVEA	1 2 3	4 5 6	7 8 I	J K L

Figure 9-6a. Field to Array MOVE - No Indexing

FIELD1 is an eight-character alphanumeric field.

ARRAY is an alphanumeric array containing four elements of three characters each.



Burroughs B 1700 RPG

PROGRAM ID | PROGRAMMER | PAGE OF | DATE

CALCULATION SPECIFICATIONS

LINE	FORM TYPE	CONTROL LEVEL	INDICATORS	FACTOR 1	OPERATION	FACTOR 2	RESULT FIELD	FIELD LENGTH	DECIMAL POSITIONS	HALF ADJUST	Resulting Indicators	COMMENTS
01	C				MOVEA	ARRAY,4	FIELD1					

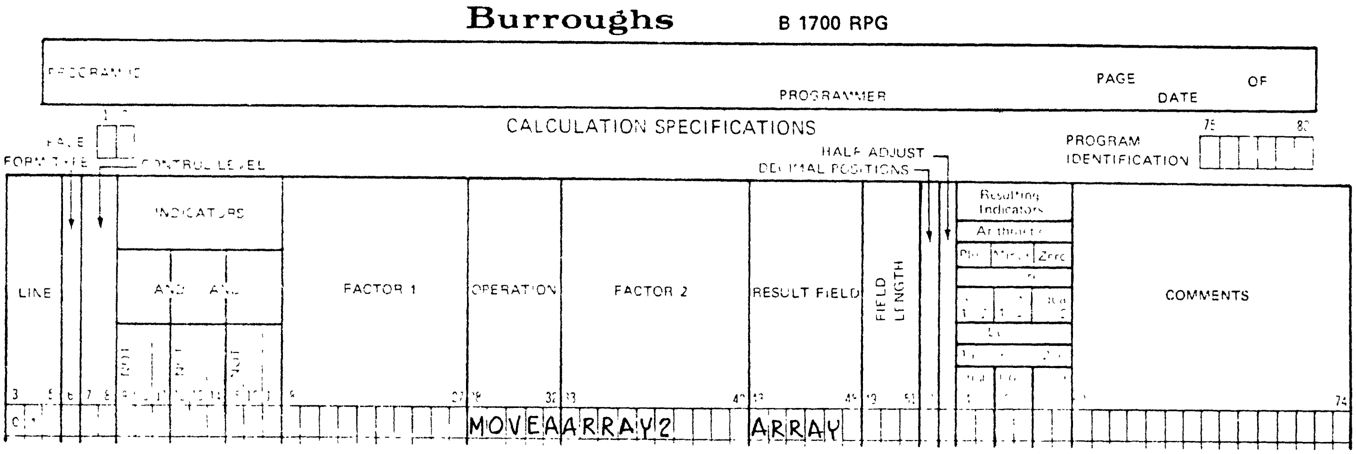
ARRAY					FIELD1
A B C	D E F	G H I	J K L	BEFORE MOVEA	1 2 3 4 5 6 7 8
A B C	D E F	G H I	J K L	AFTER MOVEA	J K L 4 5 6 7 8

G12047

Figure 9-6b. Array to Field MOVE - Array Indexed by Literal

ARRAY is an alphanumeric array containing four elements of three characters each.

FIELD1 is an eight-character alphanumeric field.



Burroughs B 1700 RPG

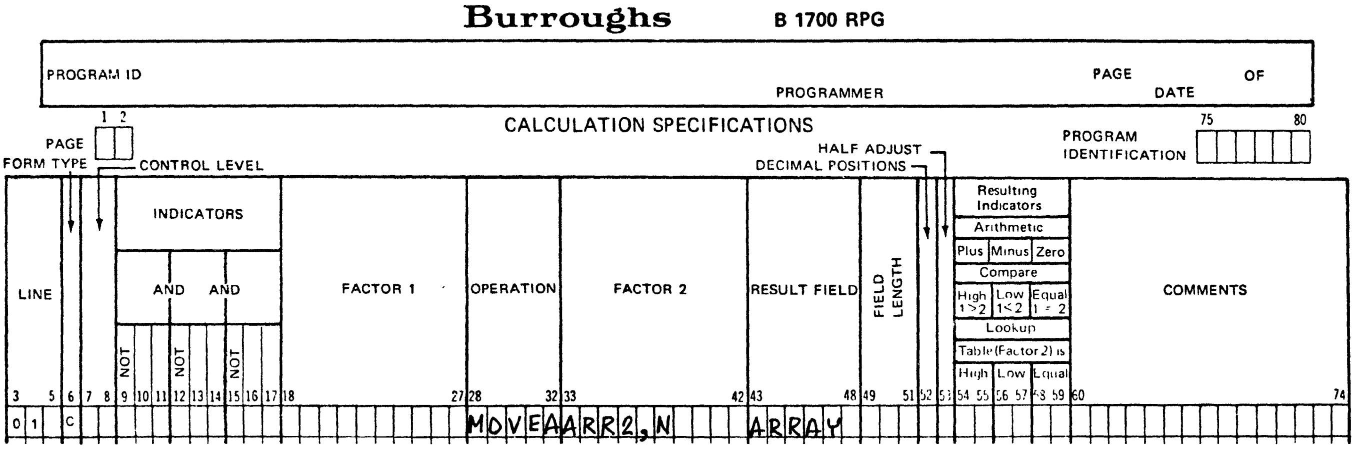
CALCULATION SPECIFICATIONS

LINE	FORM TYPE	FACTOR 1	OPERATION	FACTOR 2	RESULT FIELD
	C		MOVEA	ARRAY2	ARRAY

ARRAY2				ARRAY			
1 2 3	4 5 6	7 8 9	BEFORE MOVEA	A B C	D E F	G H I	J K L
1 2 3	4 5 6	7 8 9	AFTER MOVEA	1 2 3	4 5 6	7 8 9	J K L

Figure 9-6c. Array to Array MOVE - No Indexing

ARRAY2 and ARRAY have different numbers of elements, but all elements are of the same length.



Burroughs B 1700 RPG

PROGRAM ID PROGRAMMER PAGE OF DATE

CALCULATION SPECIFICATIONS

LINE	FORM TYPE	FACTOR 1	OPERATION	FACTOR 2	RESULT FIELD
01	C		MOVEA	ARR2,N	ARRAY

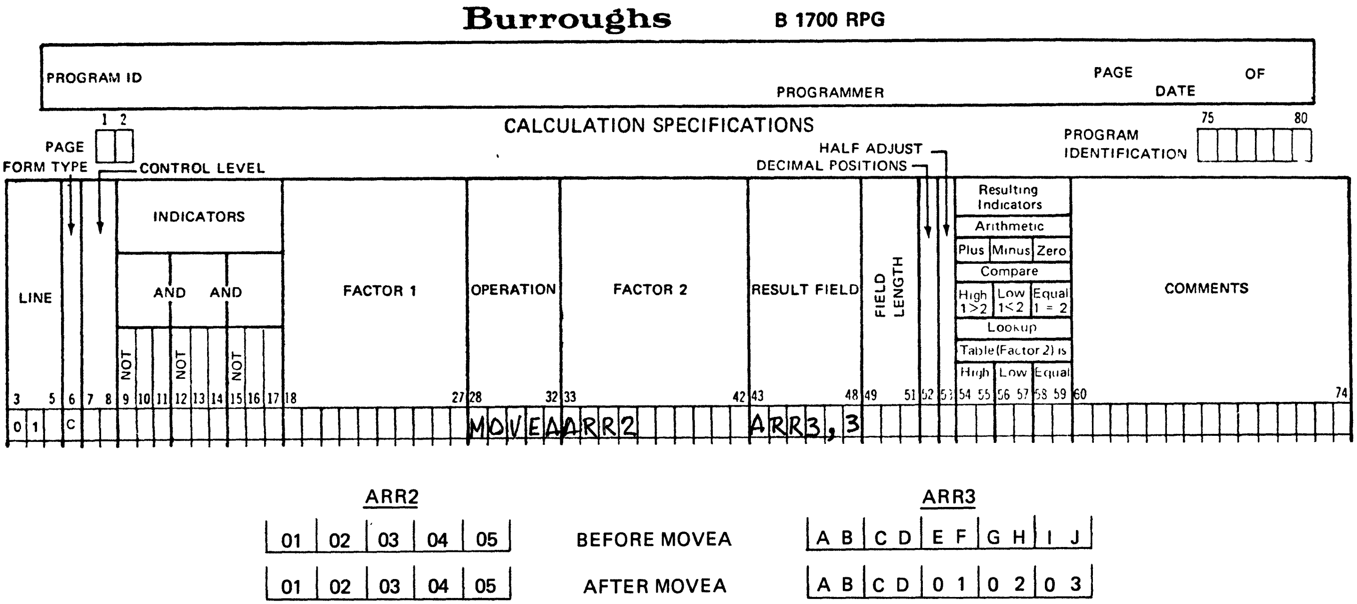
ARR2						ARRAY			
0 1	2 3	4 5	6 7	8 9	BEFORE MOVEA	A B C	D E F	G H I	J K L
0 1	2 3	4 5	6 7	8 9	AFTER MOVEA	6 7 8	9 E F	G H I	J K L

G12049

Figure 9-6d. Array to Array MOVE - FACTOR 2 Indexed

FACTOR 2 is indexed by a variable (N) whose value is 4.

ARR2 and ARRAY have elements of different lengths.



Burroughs B 1700 RPG

PROGRAM ID PROGRAMMER PAGE OF DATE

CALCULATION SPECIFICATIONS

PROGRAM IDENTIFICATION

LINE	FORM TYPE	CONTROL LEVEL	INDICATORS	FACTOR 1	OPERATION	FACTOR 2	RESULT FIELD	FIELD LENGTH	DECIMAL POSITIONS	HALF ADJUST	Resulting Indicators	COMMENTS
01	C				MOVEA	ARR2	ARR3,3					

ARR2

01	02	03	04	05	BEFORE MOVEA
01	02	03	04	05	AFTER MOVEA

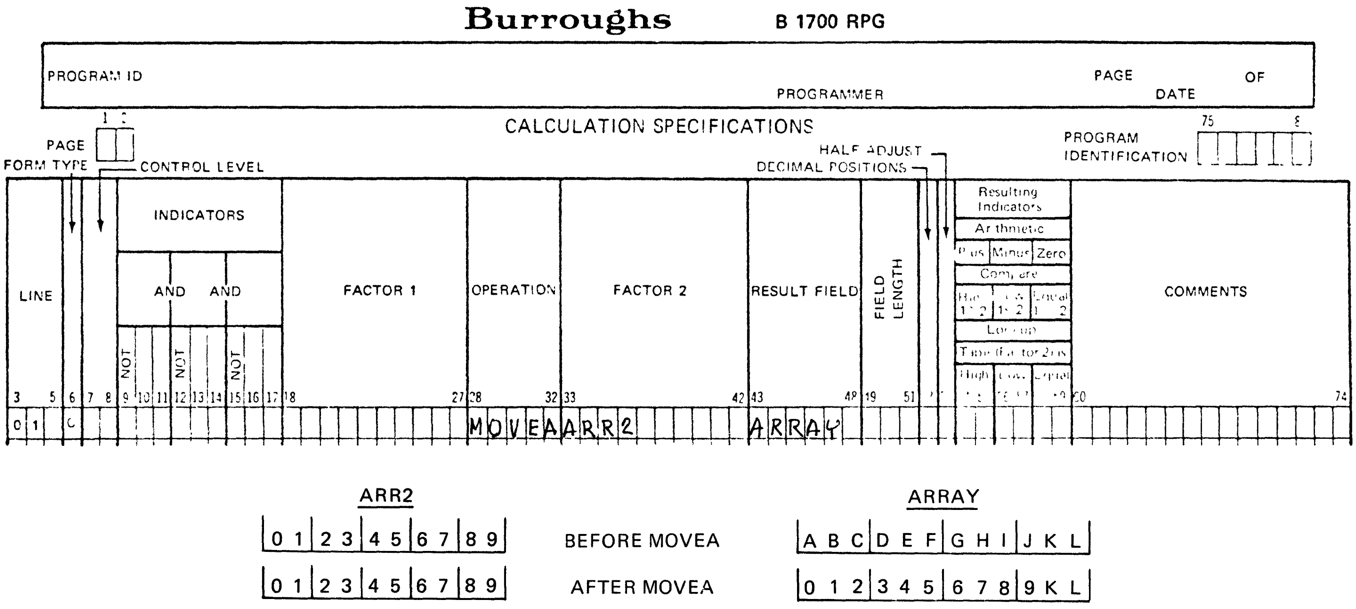
ARR3

BEFORE MOVEA	A B	C D	E F	G H	I J
AFTER MOVEA	A B	C D	0 1	0 2	0 3

G12050

Figure 9-6e. Array to Array MOVE - RESULT FIELD Indexed

RESULT FIELD is indexed by a literal.



Burroughs B 1700 RPG

PROGRAM ID PROGRAMMER PAGE OF DATE

CALCULATION SPECIFICATIONS

PROGRAM IDENTIFICATION

LINE	FORM TYPE	CONTROL LEVEL	INDICATORS	FACTOR 1	OPERATION	FACTOR 2	RESULT FIELD	FIELD LENGTH	DECIMAL POSITIONS	HALF ADJUST	Resulting Indicators	COMMENTS
01	C				MOVEA	ARR2	ARRAY					

ARR2

0 1	2 3	4 5	6 7	8 9	BEFORE MOVEA
0 1	2 3	4 5	6 7	8 9	AFTER MOVEA

ARRAY

BEFORE MOVEA	A B C	D E F	G H I	J K L
AFTER MOVEA	0 1 2	3 4 5	6 7 8	9 K L

Figure 9-6f. Array to Array MOVE - No Indexing

ARR2 and ARRAY have elements of different lengths.

MOVE ZONE OPERATIONS

Move zone operations are used to move the zone portion of only one character. FACTOR 2 is not affected by this operation, and only the zone portion of one character of the RESULT FIELD is affected. FACTOR 1 must be left blank. HALF ADJUST and resulting indicators must not be specified.

An array name may be entered in FACTOR 2, as long as the RESULT FIELD also specifies an array. In this case, the designated move zone operation will be performed on each element of the array designated in FACTOR 2, with the result being placed in the corresponding elements of the array designated in the RESULT FIELD. The operation is terminated when the end of the shorter array is reached.

If the RESULT FIELD is an array, but FACTOR 2 is not, the zone from FACTOR 2 is moved into the appropriate position in all elements of the array.

For the purpose of move zone operations, the low-order zone portion of a variable is defined as:

a. The algebraic sign if the variable is numeric, or

b. The low-order zone if the variable is alphanumeric.

A high-order zone must only refer to alphanumeric field. For a numeric field, only the sign can be referenced.

The systems standard positive sign is forced into all positive numeric fields that are the result of a move zone operation. When the zone portion of a positive numeric field is moved to the zone portion of an alphanumeric field by a move zone operation, the sign byte of the result field contains alphanumeric 0-9. When the zone of a negative numeric field is moved to the zone of an alphanumeric field by a move zone operation, the sign of the result field contains the systems standard negative sign (D).

Move zone operations are shown in table 9-6.

Table 9-6. Move Zone Operations

Instruction	Contents of FACTOR 2	RESULT FIELD	Definition
MHHZO	Alpha	Alpha	Move high-order zone portion of FACTOR 2 to high-order zone of RESULT FIELD
MHLZO	Alpha	Alpha	Move high-order zone portion of FACTOR 2 to low-order zone of RESULT FIELD.
	Alpha	Numeric	Move high-order zone portion of FACTOR 2 to the sign position of RESULT FIELD.
MLHZO	Alpha	Alpha	Move low-order zone portion of FACTOR 2 to high-order zone of RESULT FIELD.
	Numeric	Alpha	Move sign position of FACTOR 2 to high-order zone portion of RESULT FIELD.
MLLZO	Alpha	Alpha	Move low-order zone of FACTOR 2 to low-order zone of RESULT FIELD.
	Alpha	Numeric	Move low-order zone portion of FACTOR 2 to sign position of RESULT FIELD.
	Numeric	Alpha	Move sign position of FACTOR 2 to low-order zone portion of RESULT FIELD.
	Numeric	Numeric	Move sign position of FACTOR 2 to sign position of RESULT FIELD.

Each of the move zone operations is discussed in the following paragraphs.

MHHZO This operation moves the zone from the high-order (leftmost) position of FACTOR 2 to the high-order (leftmost) position of the RESULT FIELD. Both FACTOR 2 and RESULT FIELD must be alphanumeric (see figure 9-7).

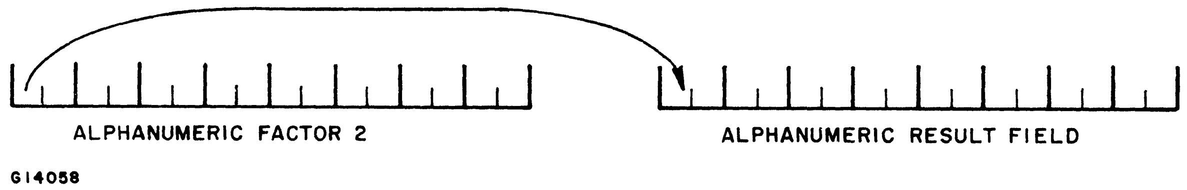


Figure 9-7. MHHZO Move Zone Operation

MHLZO If FACTOR 2 is alphanumeric and the RESULT FIELD is alphanumeric, this operation moves the zone from the high-order (leftmost) position of FACTOR 2 to the low-order (rightmost) position of the RESULT FIELD. If FACTOR 2 is alphanumeric and the RESULT FIELD is numeric, this operation moves the high-order (leftmost) position of FACTOR 2 to the sign position of the RESULT FIELD. FACTOR 2 must be alphanumeric (see figure 9-8).

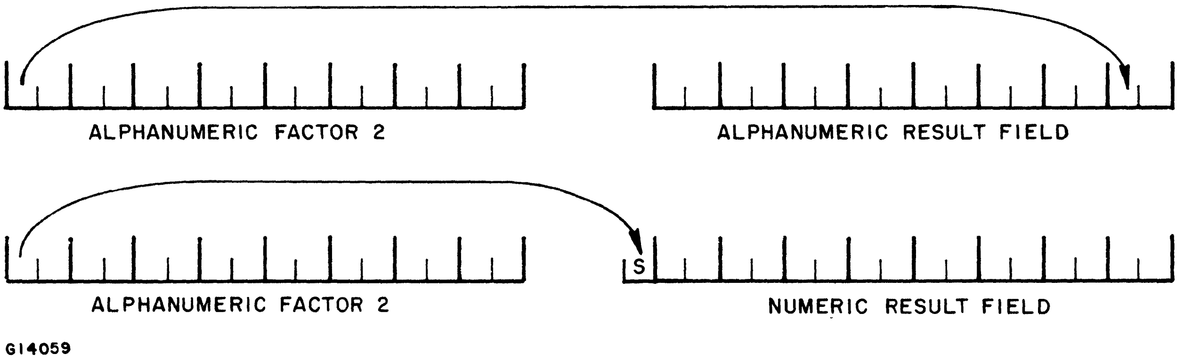
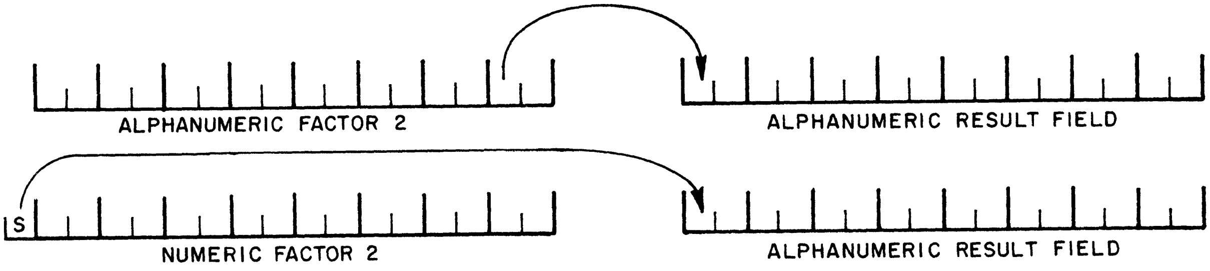


Figure 9-8. MHLZO Move Zone Operations

MLHZO If FACTOR 2 is alphanumeric and the RESULT FIELD is alphanumeric, this operation moves the zone from the low-order (rightmost) position of FACTOR 2 to the high-order (leftmost) position of the RESULT FIELD. If FACTOR 2 is numeric and the RESULT FIELD is alphanumeric, this operation moves the zone from the sign position of FACTOR 2 to the high-order (leftmost) position of the RESULT FIELD. The RESULT FIELD must be alphanumeric (see figure 9-9).



G14060

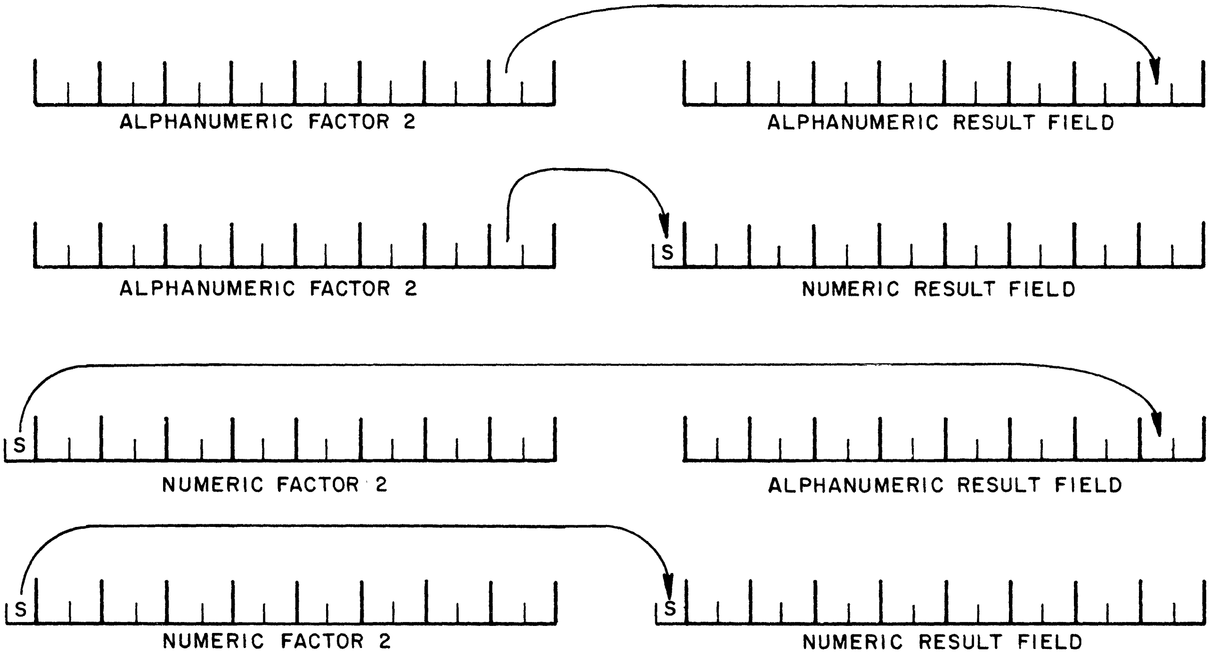
Figure 9-9. MLHZO Move Zone Operations

MLLZO If FACTOR 2 is alphanumeric and the RESULT FIELD is alphanumeric, this operation moves the zone from the low-order (rightmost) position of FACTOR 2 to the low-order (rightmost) position of the RESULT FIELD.

If FACTOR 2 is alphanumeric and the RESULT FIELD is numeric, this operation moves the zone from the low-order (rightmost) position of FACTOR 2 to the sign position of the RESULT FIELD.

If FACTOR 2 is numeric and the RESULT FIELD is alphanumeric, this operation moves the zone from the sign position of FACTOR 2 to the low-order (rightmost) position of the RESULT FIELD.

If FACTOR 2 is numeric and the RESULT FIELD is numeric, this operation moves the zone from the sign position of FACTOR 2 to the sign position of the RESULT FIELD (see figure 9-10).



G14061

Figure 9-10. MLLZO Move Zone Operations

COMPARE OPERATIONS

These operations are used to test specified fields for certain conditions. The results of these operations are shown by the setting of the specified RESULTING INDICATOR (01-99, L1-L9, LR, H0-H9, OA-OG, OV). FACTOR 1 and FACTOR 2 are not affected by the operation. At least one RESULTING INDICATOR must be specified. The RESULT FIELD must be blank. HALF ADJUST must not be specified. Neither FACTOR 1 or FACTOR 2 may be a whole array.

COMP This operation compares FACTOR 1 with FACTOR 2, causing the RESULTING INDICATORS to be set as follows:

a. HIGH (columns 54-55) - FACTOR 1 > FACTOR 2

b. LOW (columns 56-57) - FACTOR 1 < FACTOR 2

c. EQUAL (columns 58-59) - FACTOR 1 = FACTOR 2

Both fields must be of the same data type.

Comparison of Numeric Fields

The comparison of numeric fields is based on their respective values considered purely as signed numeric quantities. The length of the fields, in terms of digits, is not itself significant. Both fields are automatically aligned on their decimal points, and leading or trailing zeros are supplied, as needed, to make the lengths identical.

Comparison of Alphanumeric Fields

The comparison of alphanumeric fields is accomplished by placing FACTOR 1 and FACTOR 2 left-justified in work areas of equal length, and padding the rightmost positions of the shorter field with blanks. A character-to-character comparison is then made beginning at the leftmost position of each field. Moving to the right, the comparison continues until:

a. An unequal condition is found between corresponding characters, or

b. The end of the fields is reached.

When the unequal condition is found, if the character that is highest in the collating sequence is in FACTOR 1, the high (1>2) indicator is turned ON, if specified. If the character that is highest in the collating sequence is in FACTOR 2, the low (1<2) indicator is turned ON, if specified. Otherwise, the equal (1=2) indicator is turned ON, if specified.

When the end of the fields is reached, a condition of equal is indicated.

TESTN The test numeric operation tests the contents of the alphanumeric field, indexed vector name, or unindexed table name specified in the RESULT FIELD (columns 43-48) for numeric characters or blanks. A whole array can not be specified. If all characters in the alphanumeric field specified are numeric, the indicator specified in columns

54-55 (high) is set ON. Each character in the field must be unsigned, with the exception of the low-order character, which must contain:

a. The systems standard positive or negative sign, or

b. Be unsigned, or

c. Contain a valid plus or minus character to be considered numeric.

If the alphanumeric field specified contains numeric characters and leading blanks, the indicator specified in columns 56-57 (low) is set ON. The indicator specified in columns 58-59 (equal) is set on ON if the field contains all blanks. The same indicator can be specified for more than one condition and is set if the condition exists.

The TESTN operation determines the sign of the low-order (right-most) character within an alphanumeric field as follows:

Character Sign	Character Being Tested
Plus	A thru I, +0, +
Minus	J thru R, -0, -
Unsigned	0 thru 9

The TESTN operation determines the character status of the other characters within an alphanumeric field, with the exception of the low-order character, as follows:

Character Status	Character Being Tested
Numeric	0 thru 9
Non-numeric	All other characters

NOTE

> The TESTN operation applies to alphanumeric fields only. If an arithmetic operation is to be performed, then this field must be moved to a numeric field.

Table 9-7 lists the various results of the TESTN operation for a 4-byte alphanumeric field. A "b" indicates a blank character.

Table 9-7. Results of the TESTN Operation Code

	Indicator Results		
Contents of Alphanumeric Field	Columns 54-55	Columns 56-57	Columns 58-59
ABCD	OFF	OFF	OFF
bABC	OFF	OFF	OFF
bbbA	OFF	ON	OFF
Abbb	OFF	OFF	OFF
123D	ON	OFF	OFF
b12C	OFF	ON	OFF
bb1C	OFF	ON	OFF
bbbb	OFF	OFF	ON
1234	ON	OFF	OFF

TESTZ This operation tests the ZONE portion of the leftmost character of the RESULT FIELD, setting the specified RESULTING INDICATOR (01-99, L1-L9, LR, H0-H9, OA-OG, OV) to the results of the test. FACTOR 1 and FACTOR 2 must be blank. HALF ADJUST must not be specified. The RESULT FIELD must not be a whole array.

If the RESULT FIELD is alphanumeric, the character under test will set a specific RESULT INDICATOR according to the following:

Character Under Test	RESULTING INDICATOR Set
A-I, &	PLUS (columns 54-55)
J-R, -	MINUS (columns 56-57)
All Others	ZERO (columns 58-59)

If the RESULT FIELD is numeric, the PLUS (columns 54-55) or MINUS (columns 56-57) indicator will be set according to the sign of the field. A ZERO indicator must not be specified for a numeric field. Note that TESTZ is essentially a zone portion test and will differentiate between +0 and -0. If the programmer wishes to test for greater than, less than, or equal to zero, he should use a compare (COMP opcode) against zero, instead of TESTZ.

BINARY FIELD OPERATIONS

Three operation codes, BITON, BITOF, and TESTB, are provided to set and test individual bits. The individual bits can be used as switches in a program.

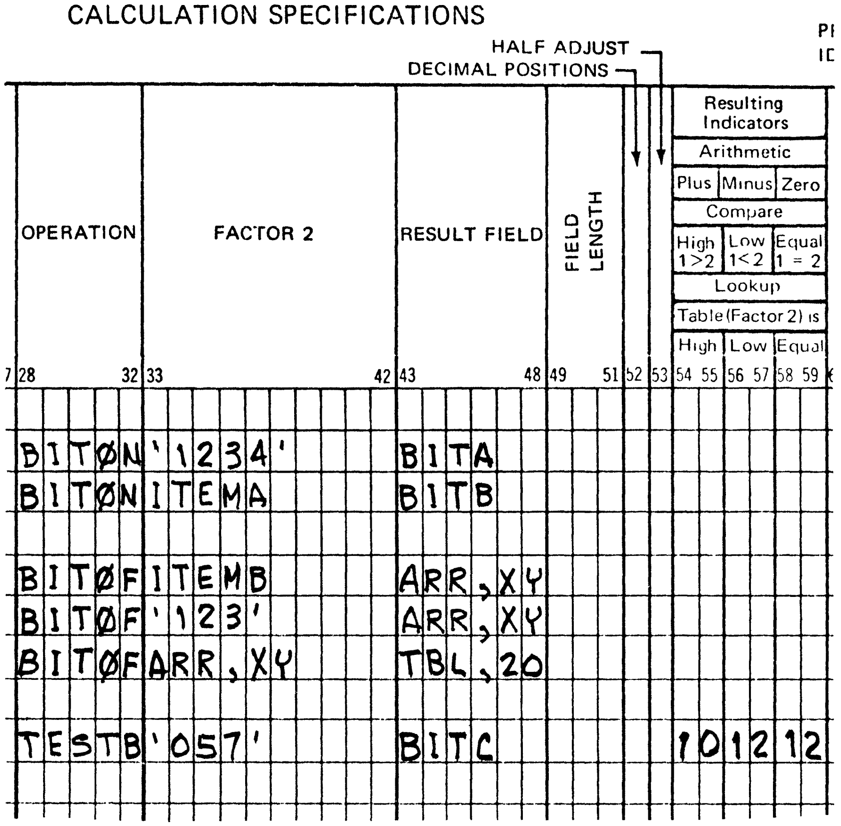
In binary field operations the operation codes BITON, BITOF, or TESTB are used. FACTOR 2 may contain either of the following:

a. Bit values 0-7. One or more bits (maximum of eight) may be set ON, set OFF, or tested per operation. The bits are numbered from left to right and are enclosed in apostrophes. The order of specification of the bits is not restricted. Bits not specified in FACTOR 2 are not changed.

b. The name of a one-position, alphanumeric field or table, or array element. In this case, the bits which are on in the field or array or table element are set ON, set OFF, or tested in the RESULT FIELD; bits which are not ON are not affected.

BITON This operation code causes bits identified in FACTOR 2 to turn ON (set to one) in a previously defined field named as the RESULT FIELD. The operation code BITON must appear in columns 28-32. Conditioning indicators can be used in columns 7-17. Any entry under FIELD LENGTH must be 1.

FACTOR 1, DECIMAL POSITIONS, HALF ADJUST, and RESULTING INDICATORS are not used with the BITON operation (see figure 9-11).


CALCULATION SPECIFICATIONS
HALF ADJUST
DECIMAL POSITIONS
PI
IC
Resulting Indicators
Arithmetic
Plus Minus Zero
Compare
High 1>2 Low 1<2 Equal 1 = 2
Lookup
Table (Factor 2) is
High Low Equal

OPERATION 28-32	FACTOR 2 33-42	RESULT FIELD 43-48	FIELD LENGTH 49-51	52	53	54 55	56 57	58 59
BITØN	'1234'	BITA						
BITØN	ITEMA	BITB						
BITØF	ITEMB	ARR,XY						
BITØF	'123'	ARR,XY						
BITØF	ARR,XY	TBL,20						
TESTB	'057'	BITC				10	12	12

G14062

Figure 9-11. Binary Field Operations Coding Examples

BITOF This operation code causes bits identified in FACTOR 2 to turn OFF (set to zero) in a previously defined field named as the RESULT FIELD.

The operation code BITOF must appear in columns 28-32. All other specifications are the same as those for the BITON operation (see figure 9-11).

TESTB This operation code causes bits identified in FACTOR 2 to be tested for an ON or OFF condition in the previously defined field named as a RESULT FIELD, setting the specified RESULTING INDICATOR to the result of the test. All other specifications are the same as those for BITON and BITOF.

At least one RESULTING INDICATOR must be used with the TESTB operation; as many as three can be named for one operation. Two indicators may be the same for one TESTB operation, but not three. If FACTOR 2 contains bits which are all OFF, no RESULTING INDICATORS are turned ON. RESULTING INDICATORS have the meanings described in the following paragraphs.

Columns 54-55

An indicator in these columns is turned ON if each bit specified in FACTOR 2 is OFF (0) in the Result Field.

Columns 56-57

An indicator in these columns is turned ON if two or more bits were tested and found to be of mixed status; that is, some bits ON and other bits OFF. It is important to ensure that the field named in FACTOR 2 contains more than one bit which is ON if an indicator appears in columns 56-57.

Columns 58-59

An indicator in these columns is turned ON if each bit specified in FACTOR 2 is ON (1) in the Result Field.

The following explanations refer to figure 9-11.

a. Bits 1234 are turned ON in the field named BITA.

b. Bits that are ON in the field named ITEMA will cause the corresponding bits in the field named BITB to be turned ON.

c. Bits that are OFF in the field named ITEMB will cause the corresponding bits in the array element ARR,XY to be turned OFF. Then bits 1, 2, 3 will be turned off in array element ARR,XY, and finally the bits that are off in ARR,XY will be turned off in the array element TBL,20.

d. If bits 0, 5, and 7 are OFF in the field named BITC, indicator 10 will be turned ON. If bits 0, 5 and 7 are of mixed status in the field named BITC, indicator 12 will be turned ON. If bits 0, 5, and 7 are ON in the field named BITC, indicator 12 will be turned ON.

SETTING INDICATORS

Up to three indicators may be set ON or OFF with one operation. These may be entered in the RESULTING INDICATORS field. FACTOR 1, FACTOR 2, HALF ADJUST, and the RESULT FIELD must be left blank. The following rules must be observed when setting indicators:

a. The following indicators may not be set ON or OFF: 1P, MR, LO, or U1-U8.

b. Setting a control level indicator (L1-L9) ON or OFF does not affect any other control level indicator.

c. All control level and record identifying indicators, except LO are automatically turned OFF after detail output operations are completed, regardless of any previous SETON or SETOF operations.

d. If any halt indicators (HO-H9) are set ON and are not turned OFF before the detail output operations finish, the program will halt.

e. If the LR indicator is turned ON by a SETON operation which is conditioned by a control level indicator (columns 7-8 on Calculations Specifications form), the program will stop after all total output operations are completed. If the LR indicator is turned ON by a SETON operation which is not conditioned by a control level indicator, the program stops after the next total output operation is completed.

SETON This operation sets the indicators entered in the RESULTING INDICATORS field ON.

SETOF This operation sets the indicators entered in the RESULTING INDICATORS field OFF.

PROGRAM BRANCHING OPERATIONS

Operations within the Calculation Specifications are normally performed in the order in which they are written. Branching operations allow variation of the order of operation; thus, conditional branching and repetitive operations are possible.

GOTO This operation causes the program to branch (GOTO) to some other instruction rather than "falling through" to the next sequential operation. Branching both forward and backward is allowed. Branching into or out of subroutines is not permitted. Branching from total calculations to detail calculations, though permitted in RPG I, may produce unexpected results when used in RPG II.

FACTOR 2 must contain a label (which must be defined elsewhere in the program as a TAG) which must follow the rules for formation of labels as described in Section 2. Columns 18-27 and 39-59 must be blank.

TAG This operation is used to identify the point where a GOTO operation will branch. FACTOR 1 must contain a unique label (TAG) which must follow the rules for formation of labels as described in Section 2.

Conditioning by a control level indicator in columns 7-8 is permissible if the TAG is part of TOTAL CALCULATIONS. The INDICATOR field (columns 9-17) must be left blank. Columns 33-59 and 24-27 must be blank.

TRANSFER CONTROL FUNCTION

This operation causes the MCP to execute a control instruction within the operating RPG object program. The MCP control instructions which may be specified are system dependent. Refer to the B 1700 System Software Operational Guide, Form No. 1068731.

ZIP The operation code ZIP causes the MCP to execute the control instruction contained in FACTOR 2. FACTOR 2 may be a field name or vector name, previously defined, or a meaningful alphanumeric literal enclosed in apostrophes. The operation may be conditioned by the conditioning indicator. All other fields must be left blank. The information contained in FACTOR 2 must be a valid MCP Control Statement. The contents of FACTOR 2 must not exceed a maximum of 511 alpha characters.

ZIP may be used for programmatic scheduling of object programs contained in the Disk Directory, or ZIP may be used to accomplish any of the MCP control functions performed through the console printer (SPO) or card reader.

In the example shown in figure 9-12, the field DATA contains the alphanumeric information, "EX JOB10". EX JOB10 is a control instruction. When the priority for JOB10 is recognized once memory space becomes available, the MCP will retrieve JOB10 from the Disk Directory and place it in the MIX for subsequent operation.

The program containing the ZIP operation will proceed to the next sequential instruction following the ZIP operation, without waiting for the execution of the program JOB10.

The example of a literal in FACTOR 2, shown in figure 9-13, will cause the same action.

LOOKUP OPERATIONS

LOKUP

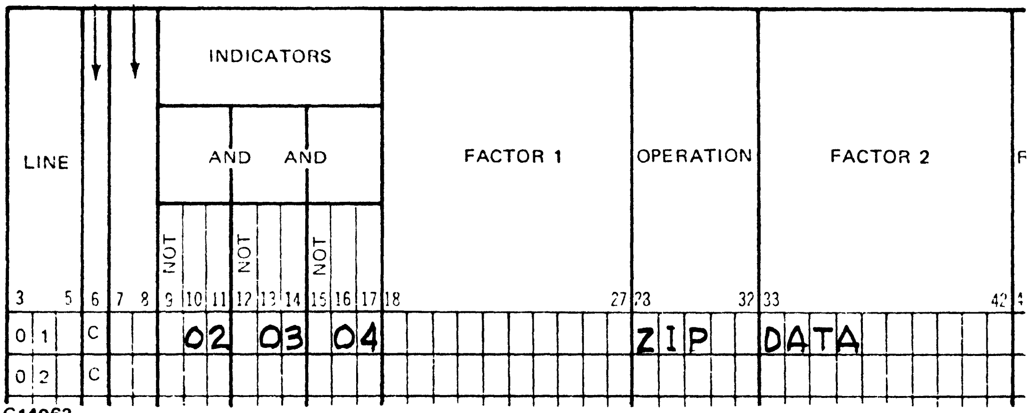
This operation code is used to search a table or array for a particular data item. The table or array name is entered in FACTOR 2. The search word is entered in FACTOR 1.

Search Word

The search word, sometimes called the search argument, is the data item for which the program tries to find a match in the table or array named in FACTOR 2.

The Search Word in FACTOR 1 may be:

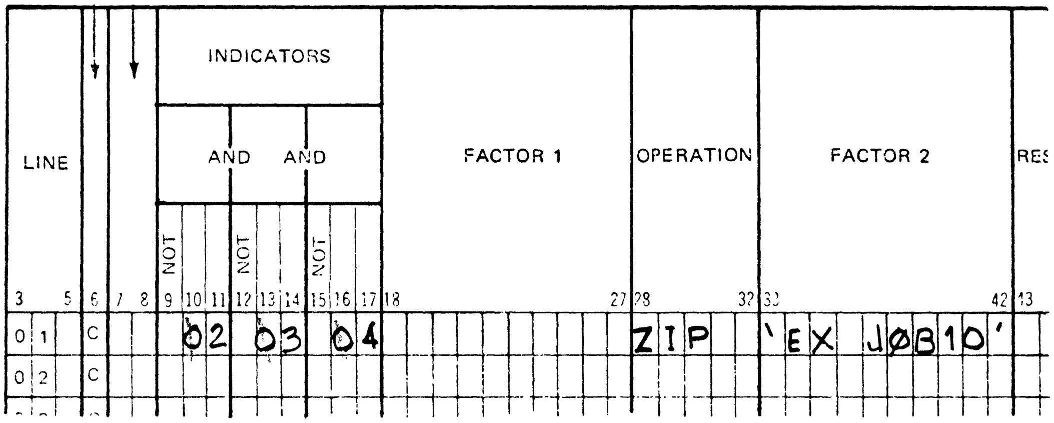
a. An alphameric or numeric literal.



LINE			INDICATORS (NOT / AND NOT / AND NOT)	FACTOR 1	OPERATION	FACTOR 2	F
3 5	6	7 8	9 10 11 12 13 14 15 16 17	18 27	28 32	33 42	4
0 1	C		02 03 04		ZIP	DATA	
0 2	C						

G14063

Figure 9-12. The ZIP Operation Using a Field Name



LINE			INDICATORS (NOT / AND NOT / AND NOT)	FACTOR 1	OPERATION	FACTOR 2	RES
3 5	6	7 8	9 10 11 12 13 14 15 16 17	18 27	28 32	33 42	43
0 1	C		02 03 04		ZIP	'EX JØB10'	
0 2	C						

G14064

Figure 9-13. The ZIP Operation Using a Literal

b. A field name.

c. A table name.

A table element.

d. An array element.

When FACTOR 1 references a table name, it refers to the element in the table that was last selected in a previous LOKUP operation. It does not refer to the whole table.

When FACTOR 1 references a table element, it refers to the actual table element and not necessarily to the element in the table that was last selected in a previous LOKUP operation.

Resulting Indicators

Resulting indicators must be used with the LOKUP operation code. Those resulting indicators are used to specify the type of search to be performed. For example:

a. When an indicator is assigned to EQUAL (columns 58-59), the table or array is searched for an item equal to the search word.

b. When an indicator is assigned to HIGH (columns 54-55), the table or array is searched for an item that is nearest to, yet higher in sequence than, the search word. When the search is successful the indicator is turned ON.

c. When an indicator is assigned to LOW (columns 56-57), the table or array is searched for an item that is nearest to, yet lower in sequence than, the search word. When the search is successful the indicator is turned ON.

Indicator Assignment

At least one resulting indicator must be assigned, and up to two resulting indicators may be specified if desired. However, when two resulting indicators are specified, one of them must be assigned to EQUAL. Resulting indicators may be assigned as follows:

EQUAL

EQUAL, HIGH

EQUAL, LOW

HIGH

LOW

A LOKUP operation code causes a syntax error when:

a. No resulting indicator is specified.

b. Indicators are assigned to both HIGH and LOW.

Rules for LOKUP

a. At least one resulting indicator must be assigned.

b. The search word and the data items in the table or array must be of the same data type (alpha or numeric), and of the same length.

c. Decimal points are ignored for numeric data.

d. A table or array should be searched for HIGH, LOW, HIGH EQUAL or LOW EQUAL only if it is specified as ordered on the Extension Specifications.

e. When multiple indicators are assigned, one of them must be assigned to EQUAL, with EQUAL taking precedence.

For example: Search for HIGH or EQUAL, the EQUAL condition takes precedence.

```
                          >  =
'5'   LOKUPARY,X          20 21
```

The array contains: 1 2 3 4 5 6 7

Indicator 21 will turn on and X points to element 5, containing "5".

f. Resulting indicators are turned ON only when the search is successful, and likewise, are turned OFF for an unsuccessful search.

g. When a LOKUP operation is performed on a table or array, the search starts with:

    1. The first element of the table or array when FACTOR 2 contains an unsubscripted table or array name.

    2: The element reference when FACTOR 2 contains a subscripted table or array name.

h. A subscripted table name cannot be specified in the result field for a LOKUP operation.

i. An array name can not be specified in the result field for a LOKUP operation.

j. Short tables or arrays should only be searched for an EQUAL condition because the compiler-supplied "filler" may cause the LOKUP to produce unexpected results.

k. Any search for other than an EQUAL condition (LOW or HIGH, or LOW and EQUAL, or HIGH and EQUAL) is allowed only if the vector was specified as having ascending or descending sequence on the Extension Specifications. The search algorithm assumes that the vector is in the specified sequence; if the user allows the vector to get out of sequence, the results may be unpredictable or not what the user anticipated.

Single Table LOKUP

When the LOKUP operation code is used to search for an item in a single table, entries must be made in FACTOR 1 (search word). FACTOR 2 (table to be searched), and at least one resulting indicator must be assigned to designate the type of search. In addition, control level and conditioning indicators may be used.

When a table item is found that satisfies the type of search specified (HIGH, LOW, EQUAL), a copy of that data item is placed in a special hold field, and the appropriate resulting indicator is turned ON. With each successful search, the contents of the table element is placed in the special hold field, thereby destroying the data that was previously there. When the search is unsuccessful, the contents of this field are left unchanged.

Two Table LOKUP

When the LOKUP operation code is used with two related tables, only one table is actually searched. Entries must be made in FACTOR 1 (search word), FACTOR 2 (table to be searched), the result field which must contain the name of the related table from which data will be made available, and at least one resulting indicator must be assigned to designate the type of search. In addition, control level and conditioning indicators may be used.

When using LOKUP with related tables, the two tables should be capable of holding an equal number of entries. Whenever the searched table (FACTOR 2) is longer than the related table (Result field), the search is terminated when the end of the shorter table is reached.

Referencing Unsubscripted Table Entries

When a unsubscripted table name is used in any operation except LOKUP, the table name refers to the data item that was placed in the special hold field by the last successful LOKUP. This allows a program to use table items when performing calculations.

When an unsubscripted table name is used as the result field in operations other than a LOKUP, the data item that was placed in the special hold field is changed by the calculation operation. In addition, the actual corresponding table item in the table is also changed. This is one way that the contents of a table can be changed during program execution.

When an unsubscripted table name is used as the search word (FACTOR 1) of a LOKUP operation, the contents of the special hold field is used. This allows a program to use the result of the last successful search as the search word or argument for other LOKUP operations.

When an unsubscripted table name is used in an operation prior to a successful LOKUP, the contents of the special hold field is the first data item in the table.

Referencing Subscripted Table Entries

When a subscripted table name is used in any operation, the data item referenced is the actual entry in the table. The contents of the special hold area are not affected in any way.

LOKUP With An Array

The LOKUP specifications for an array are the same as those specified for a table, except that when an array is specified in a two-vector LOKUP, the array name must appear in FACTOR 2, and only an unsubscripted table name can be used in the result field.

The following items pertain to array handling:

a. Arrays do not have special hold fields; therefore, when a search is successful, the indicators only reflect that the data item is in the array. The data item is not immediately available to the program.

b. When referencing an array, if only the array name is used (no subscript), the search begins with the first element in the array.

c. When referencing an array and a subscripted array name is used, the search begins at the element specified by the subscript. The subscript may be a numeric field name or a literal. When a search is successful, the appropriate indicator will turn ON. The type of subscript used will have the following effect:

a. When the subscript is a literal, a successful LOKUP operation will only verify that the data item is in the array. The actual data item or its location within the array will not be available to the program.

b. When the subscript references a numeric field name, a successful LOKUP operation will verify that the data item is in the array. In addition, the element number of the array in which the data item is contained will be automatically placed in the field that originally contained the beginning subscript. The data can now be made available to the program by again referencing the array using the same subscript field.

When the LOKUP operation is unsuccessful, the results are as follows:

a. For an unsubscripted array all indicators are reset.

b. For an array subscripted with a literal all indicators are reset. The literal is not affected.

c. For an array subscripted by a numeric field name, all indicators are reset, and the value of the subscript is set to 1.

The algorithm used for the LOKUP operation code is shown in table 9-8. The array or table type references the A or D entries in columns 45 and/or 57 of the Extension Specifications. These entries determine the sequence of the data items, which is either ascending (A) or descending (D). A LOKUP operation for any search relation other than EQUAL requires that the table or array must be in either ascending or descending sequence.

The LOKUP search relation references the entries made in columns 54 through 59 of the Calculation Specifications. The entries made in these columns specify the type of search to be performed.

LOKUP Search Relation	Definition
LSS	Less than
LEQ	Less than or equal to
EQL	Equal
GEQ	Greater than or equal to
GTR	Greater than

The algorithm references the type of search actually performed by the interpreter for each type of LOKUP operation.

Example:

An ascending table is to be searched for an item that is less than (LSS) a particular value. The table is searched for the first item that is equal to

(EQL) or greater than (GTR) the particular value. If the search is successful, the interpreter then references the table item immediately preceding the found item.

When a table or array is not specified as being in sequence, the only LOKUP operation that is permitted is a search for EQUAL. Each element in the table is compared with the data item specified for an EQUAL comparison.

Table 9-8 follows:

Table 9-8. LOKUP Algorithm

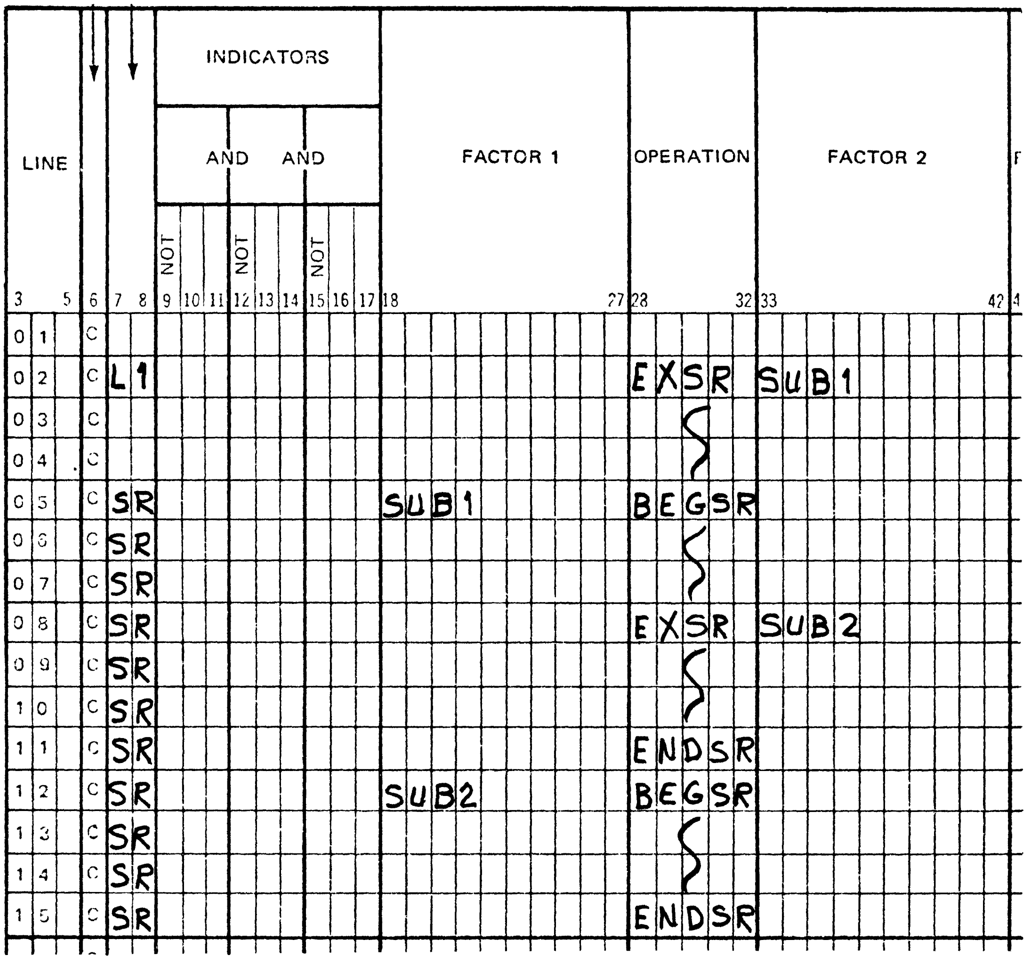
Array or Table Type	LOKUP Search Relation	Algorithm of Actual Search Performed by Interpreter
Ascending	LSS	GTR or EQL
Ascending	LEQ	GTR
Ascending	EQL	EQL
Ascending	GEQ	GEQ
Ascending	GTR	GTR
Descending	LSS	LSS
Descending	LEQ	LEQ
Descending	EQL	EQL
Descending	GEQ	LSS
Descending	GTR	LSS or EQL

SUBROUTINES

Subroutine operation codes are used only to delimit the beginning and end of a subroutine, or to call a subroutine for execution from some point in the program. Specification lines within a subroutine must contain the entries SR, OR, AN, or blank in the CONTROL LEVEL field (columns 7-8), and all subroutines must be the last operations specified in the Calculation Specifications. The recursive use of subroutines is allowed but not recommended, for example, the operation EXSR SUB1 within the SUB1 subroutine. Subroutines may not be nested in the Source Language, e.g., BEGSR cannot appear in a subroutine. Refer to figure 9-14.

BEGSR This operation code is used to indicate the beginning of a subroutine. FACTOR 1 must contain the name of the subroutine, which must follow the rules for formation of labels as described in Section 2. Columns 33-59, and 24-27 must be blank.

ENDSR This operation code is used to indicate the end of a subroutine. FACTOR 1 may contain a label. This label is used like a TAG, as a point for a GOTO operation within the subroutine to branch, thus



LINE 3-5	6	7 8	INDICATORS NOT 9 10 11	AND NOT 12 13 14	AND NOT 15 16 17	FACTOR 1 18-27	OPERATION 28-32	FACTOR 2 33-42	F
01	C								
02	C	L1					EXSR	SUB1	
03	C								
04	C								
05	C	SR				SUB1	BEGSR		
06	C	SR							
07	C	SR							
08	C	SR					EXSR	SUB2	
09	C	SR							
10	C	SR							
11	C	SR					ENDSR		
12	C	SR				SUB2	BEGSR		
13	C	SR							
14	C	SR							
15	C	SR					ENDSR		

G14068

Figure 9-14. Subroutine Coding Examples

allowing an exit from different points within the subroutine. Columns 33-59 and 24-27 must be blank.

EXSR This operation causes execution of a subroutine, and may appear anywhere within the Calculation Specification. When execution of the subroutine is completed, control returns to the next line following the EXSR operation. The EXSR operation can be used within a subroutine.

The EXSR operation may be conditioned by an indicator, allowing the subroutine to be executed (called) only when all the conditions are satisfied. FACTOR 2 must contain the name of the subroutine being called, which must be the same name entered in FACTOR 1 of a BEGSR operation. FACTOR 1 and columns 39-59 must be left blank.

PROGRAMMED CONTROL OF INPUT AND OUTPUT

Within the normal B 1700 RPG program cycle, a record is read, calculations are performed (using the data from that input record), and an output record is written. The CHAIN, DSPLY, EXCPT, READ, RECV, and SEND operations allow greater control over input and output, providing the capability to read and write records at times other than those normally available as part of the RPG program cycle. The FORCE and SETLL operations permit some programmatic control over the selection of records for processing.

CHAIN The CHAIN operation provides a method of accessing a data file on disk in an order other than the physical sequence of the records. CHAIN can be specified anywhere in the Calculation Specifications in columns 28-33.

When the CHAIN operation code is specified, the following requirements must be satisfied:

a. FACTOR 1 must contain a key specifier which cannot be a whole array.

b. FACTOR 2 must contain the name of the file the program is chaining to.

c. An indicator should be specified in columns 54-55 to turn ON if the record is not found. If it is desired that the systems operator be informed when no record is found, the field can be left blank.

When the CHAIN operation code is specified, the following fields must be left blank:

a. RESULT FIELD.

b. FIELD LENGTH.

c. HALF ADJUST.

d. DECIMAL POSITIONS.

e. RESULTING INDICATORS - EQUAL

When the CHAIN operation code is specified, optional entries are allowed in the following fields:

a. CONTROL LEVEL INDICATOR.

b. CONDITIONING INDICATORS.

When the chained file is conditioned with an external indicator, the CHAIN operation should be conditioned with the same external indicator. If this is not done, the CHAIN operation is automatically suppressed when the external indicator is OFF in order to avoid changing the status of the indicator specified in columns 54-55.

The following paragraphs describe how the CHAIN operation functions for each of the applicable file types.

Chained Indexed Files

The field length of FACTOR 1 must be the same as the key length specified for the file in the File Description Specifications. Note that if FACTOR 1 is a literal, leading zeroes or trailing blanks may be required to make the field length and key length equal. The filename entered in FACTOR 2 must have been defined in the File Description Specifications as a chained indexed disk file. If the file named in FACTOR 2 was defined in the File Description Specifications with P or N in column 31, then FACTOR 1 must be numeric.

Input Files. The value of the data specified in FACTOR 1 is compared against the index in order to locate the data record whose key field contains the same value as FACTOR 1. All specified record identifying indicators and field indicators are set appropriately.

If an equal condition is not found in the index, the indicator specified in columns 54-55 is switched ON. If no indicator is specified, the systems operator is notified by a message on the console printer.

Update Files. A chained indexed update file is processed in the same manner as chained indexed input files, with the following exception: When a record is found, the record location is retained. This allows the record to be written back into the same location from which it was read without having to CHAIN again.

Output Files. Output files are not allowed.

Chained Direct Files

FACTOR 1 must be numeric and FACTOR 2 must be defined in the File Description Specifications as a chained direct file.

Record Identification in Direct Files. When chaining to a direct file, if the key is equal to or greater than 1 and equal or less than the total number of records in the file, there will always be a "found" condition. Such a situation can present problems. In a file with a potential of more records than have actually been written, there are gaps in the file. For example, records one, five, and 1000 are all that a 1000-record, direct file currently contains. Under these conditions, when chaining to relative record number six, a record found condition implies only that the key is greater than 0 and less than or equal to the potential number of records in the file.

To ensure that the record being read is, in fact, a record and not an empty location, the user must take care to design proper record identifying codes in the direct files.

Input Files. The value of the data specified in FACTOR 1 must be a numeric (literal or field name) integer between 1 and n, where n is not greater than the total number of records in the file, and is the relative record number of the desired record.

If the key (numeric integer) is within the bounds of the file, the record will be found. Refer to Record Identification in Direct Files in this section for additional information. If the key is beyond the bounds of the file, the indicator specified in columns 54-55 is turned ON; however, if no such indicator is specified, the operator is notified by a message on the console printer.

Update Files. The chained direct update file is processed in the same manner as the chained direct input file, with the following exception: When a record is found, the record location is retained. Therefore, the record can be written back into the location from which it was read without chaining again.

Output Files. In order to create a direct file, it must be specified in the File Description Specifications as output and chained in columns 15-16. At program execution time, when the key (relative record number) is used to chain, the record location specified is made available for the writing of the record as defined on the Output-Format Specifications. Refer to Record Identification in Direct Files in this section for additional information.

The highest key (relative record number) that is written when the file is first created determines the maximum file size.

Chained Sequential Files

Any disk file can be accessed as a sequentially created disk file by providing FACTOR 1 of the chain with relative record numbers for keys. To accomplish this, the disk file must also be defined on the File Description Specifications as a chained direct file, regardless of how it was originally created. However, this is an unusual procedure as it requires extensive knowledge of the location within the file of particular records.

Input Files. Chained sequential input files are handled the same as chained direct input files. Refer to the subsection titled Chained Direct Files in this section for detailed information.

Update Files. Chained sequential update files are handled the same as chained direct update files. Refer to the subsection titled Chained Direct Files in this section for detailed information.

DSPLY This operation causes data to be displayed on the console printer (SPO) or provides for certain low volume data entries to be entered by the systems operator through the console printer. The file name of the file assigned to the console printer must be entered in FACTOR 2. The HALF ADJUST and RESULTING INDICATORS fields must be left blank. Control level and conditioning indicators may be assigned. FACTOR 1 is optional and may be used to name a data item, which may be a field name or a vector element, or to specify a literal (numeric or alphanumeric). The RESULT FIELD is optional and may be used to specify the name of a data item (field name or vector element). If the RESULT FIELD is used, the systems operator must respond on the console printer. If both FACTOR 1 and the RESULT FIELD are blank, a syntax error is emitted.

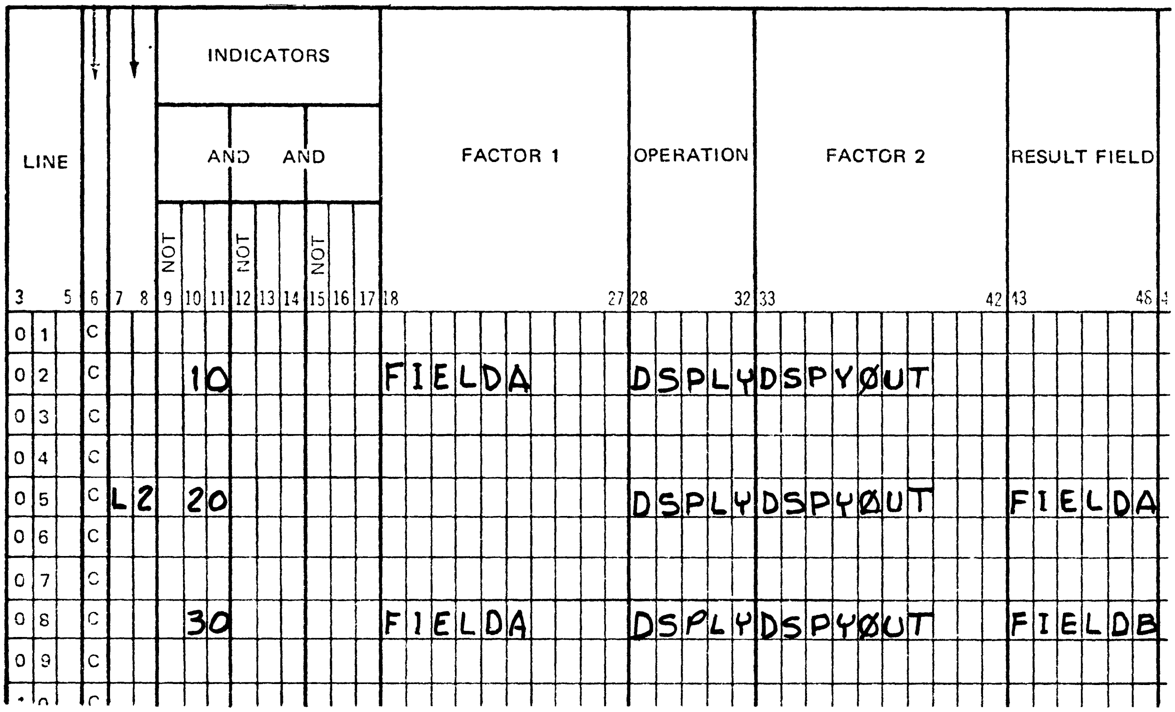
The DSPLY operates in the following manner:

a. If the RESULT FIELD is blank and FACTOR 1 contains a data item, the data item is printed on the console printer and the program proceeds to the next operation. (See figure 9-18, line 2.)

b. If the RESULT FIELD contains a data item and FACTOR 1 is blank or contains a data item, the data is printed on the console printer and an ACCEPT message will be generated by the MCP, to which the systems operator must respond. The contents of the response will be placed in the field specified in the RESULT FIELD. For the format of the ACCEPT message, refer to the <u>B 1800/B 1700 System Software Operational Guide</u>, Form No. 1068731. (See figure 9-15, line 5.)

c. If both the RESULT FIELD and FACTOR 1 contain data items, the data from both is printed on the console printer and an ACCEPT message is generated by the MCP, to which the systems operator must respond. The contents of the response are placed in the field specified in the RESULT FIELD. (Refer to figure 9-15, line 8).

d. Display output is crunched by the MCP.



LINE 3 5	6	7 8	INDICATORS AND NOT 9 10 11	AND NOT 12 13 14	NOT 15 16 17	FACTOR 1 18 27	OPERATION 28 32	FACTOR 2 33 42	RESULT FIELD 43 48
01	C								
02	C		10			FIELDA	DSPLY	DSPYØUT	
03	C								
04	C								
05	C	L2	20				DSPLY	DSPYØUT	FIELDA
06	C								
07	C								
08	C		30			FIELDA	DSPLY	DSPYØUT	FIELDB
09	C								
10	C								

GI4069

Figure 9-15. DSPLY Operation Coding Examples

EXCPT This operation allows EXCEPTION records to be written during calculations. Every time the EXCPT operation is executed, all lines in the Output-Format Specifications with a TYPE entry (column 15) of E will be written, dependent on conditioning indicators. If this operation is specified, Exception Time Output Specifications are required. The EXCPT operation may have CONTROL LEVEL and conditioning INDICATORS assigned; all other fields must be left blank.

FORCE This operation enables selection of the file from which the next record is to be taken for processing. This specification overrides the normal record selection process that occurs during input. The FORCE operation does not actually read any records during calculations, but only selects the file which will supply the next record for processing.

The force operation enables programmatic selection of the file from which the next record is to be taken for processing, and overrides the normal record selection process that occurs during input. The FORCE operation only selects the file which is to supply the next record for processing, and does not actually read any records during calculations.

The FORCE operation applies only to input, update, or combined files designated as PRIMARY or SECONDARY files. FACTOR 2 must contain the name of the file to be forced; all other fields (except conditioning INDICATORS) must be left blank. This operation must not occur within total calculations or in subroutines.

The MR indicator is always off while a forced record is being processed, and the forced record is treated as if it had no match fields specified.

The programmer should exercise care when specifying more than one FORCE operation during the same program cycle. If more than one FORCE is conditioned to occur, only the last one affects record selection, and all of the others are ignored. If the forced file is at end-of-file, normal record selection determines the next record to be selected for processing.

READ This operation is used to cause a record to be read from a demand file during calculations. This differs from the FORCE operation because FORCE causes input during the <u>next</u> program cycle, whereas the READ operation causes input during the <u>current</u> program cycle. It also differs from the CHAIN operation, because CHAIN is used to read records randomly, whereas the READ operation is used to read records sequentially.

FACTOR 2 must contain the name of the file to be read. The FACTOR 1, RESULT FIELD, FIELD LENGTH, DECIMAL POSITIONS, and HALF ADJUST fields must be left blank. The READ operation may be conditioned by CONTROL LEVEL or conditioning INDICATORS in columns 7-17.

Whenever the READ operation is specified, an indicator should be specified in columns 58-59 of the Calculation Specifications. The indicator is turned ON when an end-of-file condition occurs for the indexed file. Any subsequent attempt to read from the same file causes the indicator to be turned ON. When no indicator is used and an end-of-file condition occurs, the program automatically displays an REOF message on the console printer. The systems operator can then enter either STOP or GO on the console printer. If STOP is entered, the program terminates as if LR had occurred. If GO is entered, the program continues.

It is good programming policy to always use an indicator in columns 58-59 with the READ operation code.

The following rules must be observed when using the READ operation:

a. Only demand files designated as INPUT, UPDATE, or COMBINED may be read by the READ operation.

b. Sequence-checking in the Input Specifications is not allowed for demand files.

c. Control levels, matching fields, and look-ahead fields are not allowed for demand files.

d. If a demand file is conditioned by an external indicator (U1-U8) which is not set, the READ operation will be ignored (the End-of-File indicator in columns 58-59 will not turn ON).

e. When READ operations are performed on demand files for which sequential within limits processing has been specified, the entire file can be read as many times as desired even though end-of-file has been reached. Refer to the subsection titled Sequential Within Limits in Section 4 for additional information about that type of file processing.

RECV This operation is used to read messages from REMOTE data communication files during calculations. Only files specified as REMOTE and designated as demand in the File Description Specifications can be accessed with the RECV operation.

The RECV operation is similar to READ but permits the user to specify indicators in columns 54-59 to cause the following to be reported to the RPG program:

a. Exception conditions.

b. Incomplete I/O operations.

c. End-of-file condition.

FACTOR 2 must contain the name of the REMOTE file to be read. The FACTOR 1, RESULT FIELD, FIELD LENGTH, DECIMAL POSITIONS, and HALF ADJUST fields must be left blank.

Indicators can be specified in columns 54-59 and are described as follows:

Columns 54-55

An indicator can be specified in columns 54-55 to cause exception conditions to be reported. The indicator is turned on only when the following conditions are satisfied:

a. The established number of retries per read operation is attempted by the network controller without success, and

b. The TERMINATE ERROR statement is included in the NDL network controller and is invoked during the attempted RECV operation.

NOTE

> When a TERMINATE ERROR condition occurs, the incoming message to the RPG program is not processed.

When a specified indicator in columns 54-55 is turned on, there are several possibilities that can be programmed by the user. Some of these possibilities are:

a. Re-issue a RECV operation for the same terminal.

b. Make an entry in a log file.

c. Display a message on the console printer to notify the B 1700 operator.

d. Stop attempting to read from a terminal if repeated errors occur, or go to end-of-job.

If no indicator is specified in columns 54-55 and a TERMINATE ERROR condition occurs, the message is not processed and the RPG program is not informed of the error unless the program has previously cleared the record identifying indicators.

Columns 56-57

This field can be used to specify the type of read discipline desired for a RECV operation. If an indicator is specified in columns 56-57, incomplete I/O operations are reported to the RPG program. If no message is queued on an attempted RECV operation, incomplete I/O is reported and control is returned to the RPG program.

If no indicator is specified in columns 56-57 and a RECV operation is executed, control is not returned to the RPG program until a message is read.

Following are several items that could be programmed for depending upon the results of incomplete I/O reporting:

a. Notify the operator that specific terminals do not respond.

b. Send inquiry messages to the terminals.

c. Have the RPG program perform other functions before attempting to read the message again.

d. Determine if there is activity on the data communications transmission lines.

Columns 58-59.

This field is used to indicate if end-of-file conditions are to be reported to the RPG program. The end-of-file indicator can be turned on by:

a. Entry of the Quit Controller (QC) input message on the console printer by the B 1700 operator, or

b. And end-of-file message is placed in the queue assigned to the file by a controlling Message Control System (MCS).

SEND This operation is used to write exception records (messages) to REMOTE data communications files during calculations. Only files specified as device type REMOTE and designated as demand in the File Description Specifications can be accessed with the SEND operation.

FACTOR 2 must contain the name of the REMOTE file to which the message is to be sent. The FACTOR 1, RESULT FIELD, FIELD LENGTH, DECIMAL POSITIONS, and HALF ADJUST fields must be left blank.

The SEND operation is similar to EXCPT but permits the user to specify indicators in columns 56-59 to cause the following to be reported to the RPG program:

a. Incomplete I/O operations.

b. End-of-file conditions if an invalid key is used.

The indicators that can be specified in columns 56-59 are described as follows:

Columns 56-57.

This field can be used to specify the type of write discipline desired for a SEND operation. If an indicator is specified in columns 56-57 and the SEND operation is attempted when the message queue is full, incomplete I/O condition is reported to the RPG program. The output message queue size is controlled by the network controller.

Several actions that could be taken following incomplete I/O reporting are:

a. The RPG program can send the message again.

b. The B 1800/B 1700 operator can be notified that a terminal doesn't respond to inquiry.

c. Stop attempting to send messages to a particular terminal.

Columns 58-59.

This field can be used to indicate if end-of-file conditions are to be reported. If an indicator is specified in columns 58-59, end-of-file is reported for output and combined (input and output) files if an invalid key is used for a SEND operation.

If no indicator is specified in columns 58-59, end-of-file is not reported and use of an invalid key results in the display of a DS or DP message on the console printer.

Prior to the first SEND operation, a value must be moved to the station number field (columns 22-27 on the Telecommunications Specification Line). This value is three characters long and represents the relative station number in the file of logical stations defined in the Network Controller program.

Prior to every SEND operation, a value must be moved to message length field (columns 28-33 on the Telecommunications Specification line). This value is four characters long and represents the largest ending position value defined on the Output Specification line for the REMOTE file.

Example Program:

The following program sends the message 'HI THERE' to the remote station from which the program was executed:

```
00100H

00200FPRIMARY IPE F 180 180                DISK
00300FREMFILE O   F19201920                REMOTE

00400TREMFILE  001002STATONMESLEN

00500IPRIMARY AA  01
00600I                                           1   1 DUMMY

00700C                      MOVE '001'       STATON  3
00800C                      MOVE '0010'      MESLEN  4
00900C                      SEND REMFILE

01000OREMFILE E        01
011000                                      10 'HI THERE'
```

SETLL When this operation is specified in the Calculation Specifications, any indexed file that is designated as a demand file can be processed sequentially within limits.

WARNING

> An indexed file designated as a demand file cannot be processed sequentially within limits by both a limits file and the SETLL operation code within the same program.

When SETLL is specified, the absolute value contained in a field or literal is used to establish the lower limit (bound) of the record key for the indexed file that is to be processed sequentially within limits. The upper limit (bound) automatically defaults to the highest record key that exists in the file.

FACTOR 1 must contain a field name, a vector element, a table name, or a literal representing the value of the lower limit being set. A whole array is not allowed. The type (alphanumeric or numeric) and length of FACTOR 1 should be equal to the type and length of the key for the file named as FACTOR 2.

FACTOR 2 must contain the name of the file for which the lower limit is to be set. The named file must be specified on the File Description Specifications as an indexed demand file processed within limits, but must not occur on Extension Specifications. For example, no record address file can be associated with this file.

Columns 43-59 must be blank. Columns 7-17 can be used in the normal way.

Whenever the indexed demand file named as FACTOR 2 is read using the READ operation, records are read sequentially by key. The SETLL operation sets the lower key limit for the file. The next record read is the record whose key is next in sequence but greater than the lower key limit. Successive records are read in ascending key sequence until either end-of-file occurs or another SETLL operation is executed.

Refer to Section 4 of this manual for additional information concerning SETLL and file processing within limits.

DEBUG OPERATION

The DEBUG operation is a special-purpose function which simplifies the location and correction of errors in an RPG Program.

DEBUG This operation causes records to be written during calculations. These records contain specific information which may be helpful in locating errors in the program. DEBUG operations may appear in as many places as needed within the Calculation Specifications. Every time the DEBUG operation is executed, one or more fixed-format records will be written to an output device. One record contains a list of all indicators which are ON at the time the DEBUG operation is specified; the other record(s) shows the contents of any one field.

DEBUG operations will be compiled into the object program only if the DEBUG field of the Control Card contains a 1 (column 15). All DEBUG operations are syntax checked during compilation.

The file named must be defined as a sequential output file on the File Specifications. All DEBUG output must go to the same file. FACTOR 2 must contain the name of the output file on which the records are written.

FACTOR 1 is optional, and may contain a literal or field name to identify the particular DEBUG operation being executed.

FACTOR 1 must not be a whole array, and the size of the field must not exceed eight positions. The literal or the value of the designated field is written as part of the output records.

The DEBUG operation may be conditioned by indicators in columns 7-17. Columns 49-59 must be left blank. RESULT FIELD is optional, but if specified it must be a field, vector, or indexed vector whose contents are written on the second and subsequent records.

DEBUG Operation Output

One or more records will be written as output from every DEBUG operation. The first record is always written; the second and subsequent records will be written only if the RESULT FIELD contains an entry. See figure 9-16 for an example of the output produced by the DEBUG operation.

SOURCE PROGRAM

```
  1 0101 H         1


  2 0102 FCARDIN  IPE    80  80              READER
  3 0103 FPRINTOUTO     132 132              PRINTER


  4 0201 E                       ARY         5  6 2


  5 0301 ICARDIN  AA  01
  6 0302 I                                            1   62ARDATA      020304
  7 0303 I                                            9  100IX          05
  8 0304 I                                           13  80 CMNT


  9 0401 C   01 05              MOVE ARDATA    ARY,IX
S 10 0302 C            ≥1≥      DEBUGPRINTOUT  ARY
 11 0403 C                      SETON                    758085
 12 0404 C            ≥2≥      DEBUGPRINTOUT  ARDATA
 13 0405 C                      SETOF                    80
 14 0406 C                      DEBUGPRINTOUT
 15 0407 C                      SETOF                    758505
 16 0408 C                      DEBUGPRINTOUT


 17 0501 OPRINTOUTD  2     01
 18 0502 O                                         9 ≥ELEMENT $≥
 19 0503 O                         IX     Z    11
 20 0504 O                                        33 ≥ OF ARRAY ≥≥ARY≥≥ LOADED≥
 21 0505 O                                        40 ≥ WITH: ≥
 22 0506 O                         ARDATA    48 ≥  0 .  ≥
 23 0507 O                         CMNT     120
```

OUTPUT LISTING

```
 DEBUG- 1          INDICATORS ON-01 02 05
 ARY   ,0001  000136
 ARY   ,0002  000000
 ARY   ,0003  000000
 ARY   ,0004  000000
 ARY   ,0005  000000
 DEBUG- 2          INDICATORS ON-01 02 05 75 80 85
 ARDATA       000136
 DEBUG-            INDICATORS ON-01 02 05 75 85
 DEBUG-            INDICATORS ON-01 02
ELEMENT $ 1 OF ARRAY ≥ARY≥ LOADED WITH:      1.36    LOAD ARY,1 WITH    1.36

 DEBUG- 1          INDICATORS ON-01 04 05
 ARY   ,0001  000136
 ARY   ,0002  000000
 ARY   ,0003  000000
 ARY   ,0004  000000
 ARY   ,0005  000000
 DEBUG- 2          INDICATORS ON-01 04 05 75 80 85
 ARDATA       000000
 DEBUG-            INDICATORS ON-01 04 05 75 85
 DEBUG-            INDICATORS ON-01 04
ELEMENT $ 2 OF ARRAY ≥ARY≥ LOADED WITH:      0.00    LOAD ARY,2 WITH    0.00

 DEBUG- 1          INDICATORS ON-01 02 05
 ARY   ,0001  000136
 ARY   ,0002  000000
 ARY   ,0003  123456
 ARY   ,0004  000000
 ARY   ,0005  000000
 DEBUG- 2          INDICATORS ON-01 02 05 75 80 85
 ARDATA       123456
 DEBUG-            INDICATORS ON-01 02 05 75 85
 DEBUG-            INDICATORS ON-01 02
ELEMENT $ 3 OF ARRAY ≥ARY≥ LOADED WITH:  1234.56    LOAD ARY,3 WITH 1234.56

 DEBUG- 1          INDICATORS ON-01 02 05
 ARY   ,0001  000136
 ARY   ,0002  000000
 ARY   ,0003  123456
 ARY   ,0004  999999
 ARY   ,0005  000000
 DEBUG- 2          INDICATORS ON-01 02 05 75 80 85
 ARDATA       999999
 DEBUG-            INDICATORS ON-01 02 05 75 85
 DEBUG-            INDICATORS ON-01 02
ELEMENT $ 4 OF ARRAY ≥ARY≥ LOADED WITH:  9999.99    LOAD ARY,4 WITH 9999.99

 DEBUG- 1          INDICATORS ON-01 02 05
 ARY   ,0001  000136
 ARY   ,0002  000000
 ARY   ,0003  123456
 ARY   ,0004  999999
 ARY   ,0005  654321
 DEBUG- 2          INDICATORS ON-01 02 05 75 80 85
 ARDATA       654321
 DEBUG-            INDICATORS ON-01 02 05 75 85
 DEBUG-            INDICATORS ON-01 02
ELEMENT $ 5 OF ARRAY ≥ARY≥ LOADED WITH:  6543.21    LOAD ARY,5 WITH 6543.21
```

G14070

Figure 9-16. DEBUG Operation Output

The first record written is in the following format:

Record Position	Entry
2-7	DEBUG
9-17	Blank
18-31	The words INDICATORS ON-
32-any position depending on the number of indicators ON	List of indicators that are ON, separated by blanks.

TIME

The TIME operation provides the system time. The system date is provided as an option. System time and system date, as referred to here, are the dynamic time and date provided by the system, and may change during the execution of an RPG program.

To use this operation, enter the operation code TIME in columns 28-32 of the Calculation Specifications. The Result Field, columns 43-48, must contain the name of a six or twelve digit numeric field which may be defined elsewhere in the program or in columns 49-52 of the Calculation Specifications. The numeric field named in columns 43-48 may not be a whole array. FACTOR 1, FACTOR 2, HALF ADJUST, and the Resulting Indicators must remain blank.

After this operation is performed:

a. A six-digit Result Field will contain the system time in the format hhmmss, where hh represents hours, mm represents minutes, and ss represents seconds.

b. A twelve-digit Result Field will contain both the system time and the system date, with the format of the system date being dependent on the entry in the Inverted Print field (column 21) of the Control Card Specification.

   If the Inverted Print field contains a blank, the Result Field is provided in the format hhmmssmmddyy.

   If the Inverted Print field contains a D, I, or J, the Result Field will be in the format hhmmssddmmyy, where hh represents hours, mm represents minutes, ss represents seconds, mm represents month, dd represents day, and yy represents year.

SECTION 10
OUTPUT-FORMAT SPECIFICATIONS

The Output-Format Specifications state where a record is to be written. Within the file where the record is to be written, these specifications also define when to write, what data is to be written, and the location and format of the data within the record.

Depending on the peripheral equipment available, RPG can be used to write on line printers, character printers, tapes, disks, cards and remote devices. While any of these devices can be used to create new files or records, disks are suitable for changing a record in place without changing the location of the record in the file. Therefore, the Output-Format Specifications can describe all of, or part of, a record.

The Output-Format Specifications are used to define where to write. Device in the File Description Specifications names the peripheral and filename associates a uniquely devised mnemonic with that device.

The Output-Format Specifications are functionally divided into the following two sections:

a. Record description, consisting of columns 7-31.

b. Field description, consisting of columns 23-70.

RECORD DESCRIPTION SECTION

The record description section contains the following information:

a. Where the record is to be written - filename, columns 7-14.

b. When the record is to be written - file type, column 15.

   1. Detail output, enter D. Enter H for heading information.

   2. Total output, enter T.

   3. Exception output, enter E. Exception output is written only as a result of the EXCPT operation code and may be executed during either detail calculations or total calculations.

c. Output conditioning indicators, columns 23-31. These indicators are used in parallel with the H, D, T, and E of column 15 further control when the record is to be written.

FIELD DESCRIPTION SECTION

The field descriptions contain the following information:

a. When to write the data on the record. Output conditioning indicators, columns 23-31.

b. What data to write on the record.

1. Variable name, columns 32-37.
2. Edit codes, column 38, punctuate the data written.
3. Blank after, column 39, clears the field after the write.
4. Constant or edit word, columns 45-70, punctuates and/or specifies constant information to be written in the record.

c. Relative location where the data is to be written in the record-end position, columns 40-43.

d. Format of data to be written in the record.

1. Edit codes, column 38, punctuate numeric variables.
2. Packed, column 44, may specify packed format of numeric variables.
3. Binary, column 44. May specify binary format of numeric variables.
4. Constant or edit word, columns 45-70, specifies constant data and punctuation and constants, if required, of numeric variables.

Since several of the Output-Format Specification fields have multiple usages which are conflicting, they are discussed in the text of this section.

Figure 10-1 illustrates and describes the Output-Format Specifications form.

The first Output Specification must be a record type description. Field description entries must start one line below the associated record type descriptions. A warning is emitted if a record type description has no associated field descriptions.

FIELD DEFINITIONS

Refer to figure 10-1 in conjunction with the following field definitions for the Output-Format Specifications.

1-2 PAGE

Refer to Section 2 for complete description.

3-5 LINE

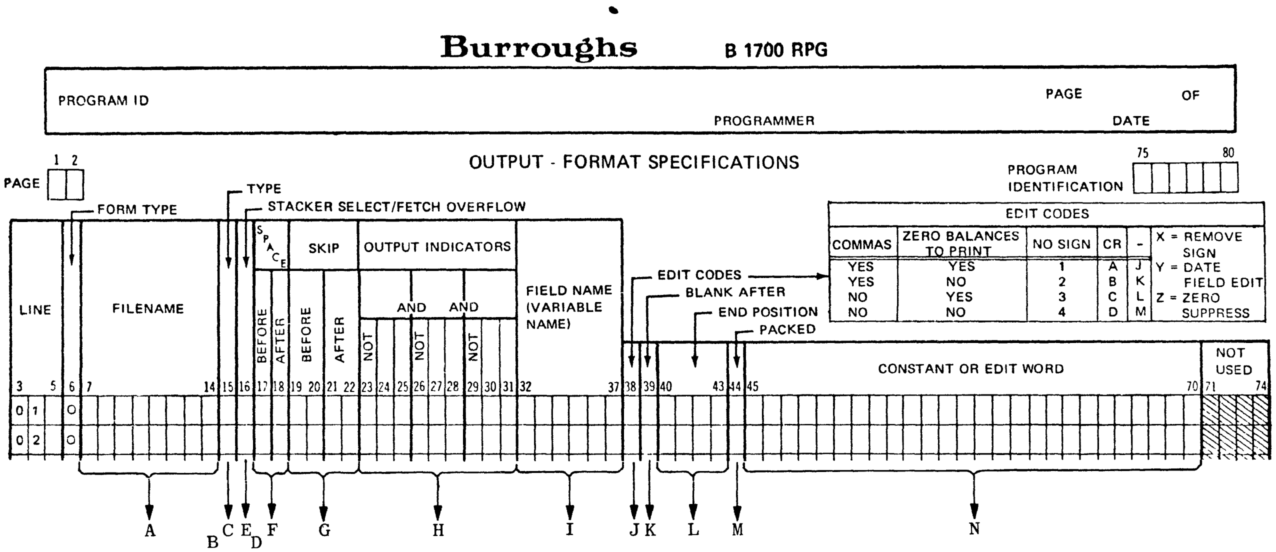
Refer to Section 2 for complete description.

6 FORM TYPE

An O must appear in this field.

7-14 FILENAME

This field is used to identify the file to which the subsequent record type and field description entries belong. The file specified must have been previously described on the File Description Specifications form as output, update, combined, or input, with an A in column 66 on the File Description Specifications and ADD specified in columns 16-18 of the Output-Format Specifications. Every output file described



A. 7-14 Contains a filename specified in the File Description Specifications.

B. 14-16 Puts the output indicators in an AND or OR relationship. Entries: AND or OR.

C. 15 Specifies the type of output record to be written. Entries: H, D, T or E.

D. 16-18 Specifies if a record is to be added to an indexed sequential file. Entries: Blank or ADD.

E. 16 Specifies (1) which stacker the output card is to be placed or (2) that the overflow routine is to be invoked. Entries: Blank, F, or 1-N (N=number of stackers).

F. 17-18 Specifies forms spacing, before or after printing, for printer output files. Entries: 0 or blank, or 1-9.

G. 19-22 Specifies forms skipping, before or after printing, for printer output files. Entries: 0-99, A0-A9, B0-B2, or blank.

H. 23-31 Output Indicators:

23, 26, 29 Indicates if the output indicator in columns 24-25, 27-28, or 30-31, must be ON or OFF. Entries: Blank or N.

24-25, 27-28, 30-31 Contains a previously defined indicator which is to condition output. Entries: 01-99, L0-L9, LR, MR, H0-H9, U1-U8, OA-OG, OV, 1P, or blank.

G14071

I. 32-37 Contains a previously defined field name or vector or one of the special field names PAGE, PAGE through PAGEn, *PLACE, *PRINT, UDATE, UMONTH, UYEAR, UDAY, JDATE.

J. 38 Specifies the editing for a numeric output field when not using an edit word. Entries: Blank, 1, 2, 3, 4, A, B, C, D, J, K, L, M, X, Y, or Z.

K. 39 Specifies if a variable is to be reset after the output operation is finished. Entries: Blank or B.

L. 40-43 Specifies the location of a field within an output record. Entries: 1-N (N=maximum record length) right-justified or an * in column 40.

M. 44 Specifies that an output field is to be written in alphanumeric, packed or binary format. Entries: Blank, P, or B.

N. 45-70 Contains constants and/or edit words, used to format and punctuate output records, enclosed in apostrophes and left-justified. Entries: Any valid RPG character (see text for uses of characters with special meanings).

Figure 10-1. Output-Format Specifications Summary Sheet

in the File Description Specifications should also be described on the Output-Format Specifications form, but this is not required.

If this entry is blank on a record type description, the filename of the previous record is assumed. The first record type description must not have a blank filename entry.

For update files, only the fields to be changed must be specified in an output record. The remainder of the update record remains unchanged.

15 TYPE

This field is used to specify when an output record is to be written. These records contain such information as the following:

a. Heading records contain such information as page headings, and are treated as if they were detail records.

b. Detail records usually contain some type of data obtained directly from input records and calculation operations. Detail records are written once during every cycle, depending upon conditioning indicators.

c. Total records usually contain totals accumulated from a group of input records. Total records are written only during cycles in which a control break occurs, depending upon conditioning indicators.

d. Exception records are written during calculation time, through use of the EXCPT or SEND operation codes.

Valid entries for this field are:

Entry	Definition
H	Heading records.
D	Detail records.
T	Total records.
E	Exception records (written during calculation time).

Record types can be specified in any order. However, if all output indicators are satisfied, the sequence of output is the order specified on the Output-Format Specifications, as follows:

a. Heading and detail output. Each cycle including IP output on the first cycle only.

b. Exception outnut. During total calculation except during the first cycle.

c. Total output. Each cycle except the first.

d. Overflow output. Each cycle when a paper overlow condition has occurred.

e. Exception output. During detail calculations.

First page (1P indicator ON) output is heading or detail output conditioned with the 1P indicator, and only occurs at the beginning of program execution because the 1P indicator is permanently turned OFF after the first detail output.

Overflow output is heading, detail, or total output records conditioned with an overflow indicator.

For ease of writing and subsequent program maintenance, one of the following two methods is usually used for determining output record sequence:

a. The records within one file are specified, beginning with heading records and continuing with detail records, total records, and exception records.

b. The programmer can specify headings for all files, detail records for all files, total records for all files, and exception records for all files.

The two methods of determining the sequence of output specifications can be varied as required, so that the programmer can ensure the proper physical sequence of the records in the files.

Exception records can be specified for a combined file, although this is not recommended for files other than REMOTE. Exception output records conditioned on L0-L9 or total output must not be specified for primary or secondary update files.

16-18 RECORD ADDITION

If a record is to be added to an input, output, or update file, the following conditions apply:

a. The file must be a disk file.

b. The corresponding File Description Specification must have an A in column 66.

c. The word ADD must be entered in columns 16-18 of the record type description.

d. ADD must not be specified on an AND/OR line.

16 STACKER SELECT/FETCH OVERFLOW

This field may be used to specify:

a. The stacker into which the output or combined file card is placed after it is processed, or

b. That the overflow routine can be called for possible execution prior to printing the record specified by this line.

Valid entries for this field are:

Entry	Definition
Blank	Cards automatically go to default stacker.
Numeric Entry	Stacker into which card type is stacked.
F	Fetch overflow (printer files only).

STACKER SELECT

Output card files can only be stacker selected on the Output-Format Specification.

Combined card files can be stacker selected on either the Input Specifications or the Output-Format Specifications. At program execution time, if a record appears that is selected in both the Input and Output Specifications, it is selected according to the Output-Format Specification. The programmer should try to avoid such instances because the results may be confusing.

Stacker selection on the basis of matching records should be specified only for detail output lines, because the card is selected prior to total time.

If the numeric entry specifying stacker selection is greater than the potential of the attached hardware device, the record is selected to the normal (default) stacker.

Record types identified by OR lines can be stacker selected to a special stacker by an entry in the field. However, if the stacker select field is left blank, the record type selected by the OR line goes to the default stacker (stacker select entry on the previous line is not assumed). AND lines can not have a stacker select entry.

FETCH OVERFLOW

If the printing of a line could cause overflow, leaving insufficient space on the page to print the remaining detail, total output lines or lines conditioned by the overflow indicator, FETCH OVERFLOW should be specified (F in column 16).

When the overflow line is reached, the same sequence of events always occurs. These events are described in detail in the subsection titled Printer File Handling in this section of the manual. Briefly, remaining detail lines, total lines, and overflow-lines (lines conditioned by the overflow indicator) are printed on the page following the occurrence of overflow.

If, however, it is desired to print overflow lines ahead of the usual time, a FETCH OVERFLOW routine may be specified. This may be initiated any time after the overflow line has been reached. When overflow is caused in this manner, the following actions take place:

a. All total lines conditioned by the overflow indicator are printed. If skipping is specified on these lines, it occurs as defined by the user.

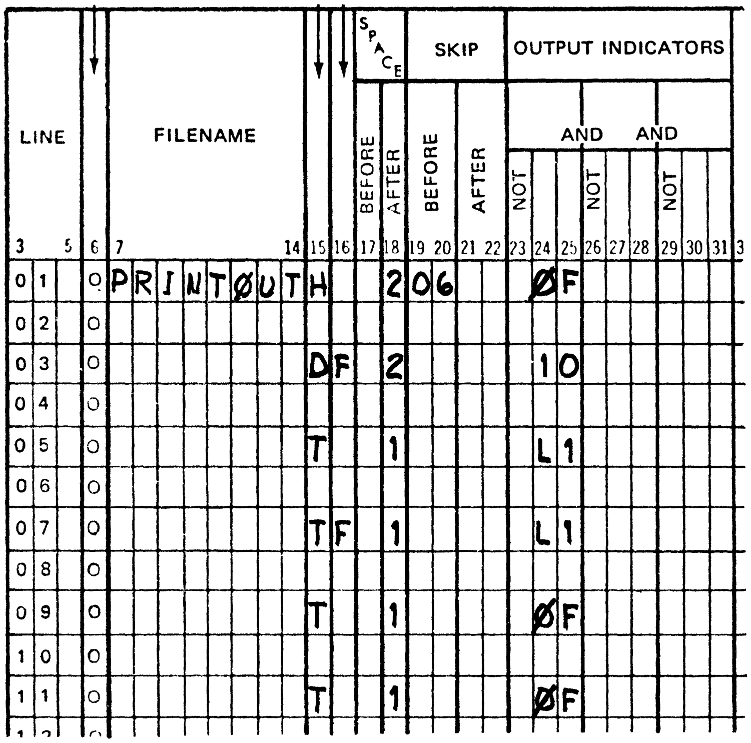
b. Heading and detail output lines conditioned by the overflow indicator are printed. If skipping is specified on these lines, it occurs as defined by the user.

c. The line that fetched the overflow routine is printed.

d. Any detail and/or total lines left to be printed in the current program cycle are printed.

For the printer file, an F in column 16 on the Output Format Specifications specifies that the overflow routine will be fetched. An F can be specified for any total, detail, or exception line that is not conditioned by an overflow indicator.

If a line causes the overflow indicator to turn ON, the next line containing an F in column 16 will cause the execution of the overflow routine. When that is complete, normal printing will resume with the statement that fetched the overflow routine (see figure 10-2).



LINE		FILENAME			SPACE		SKIP		OUTPUT INDICATORS					
											AND		AND	
					BEFORE	AFTER	BEFORE	AFTER	NOT		NOT		NOT	
3 5	6	7 14	15	16	17	18	19 20	21 22	23	24 25	26	27 28	29	30 31
0 1	O	PRINTØUT	H			2	06			ØF				
0 2	O													
0 3	O		D	F		2				10				
0 4	O													
0 5	O		T			1				L1				
0 6	O													
0 7	O		T	F		1				L1				
0 8	O													
0 9	O		T			1				ØF				
1 0	O													
1 1	O		T			1				ØF				
1 2	O													

G14072

Figure 10-2. FETCH OVERFLOW Coding Example

In the example in figure 10-2, if the printing of line 03 causes the overflow indicator to turn on, output lines will occur in the following order:

05 is printed on the same page if indicator L1 is on.

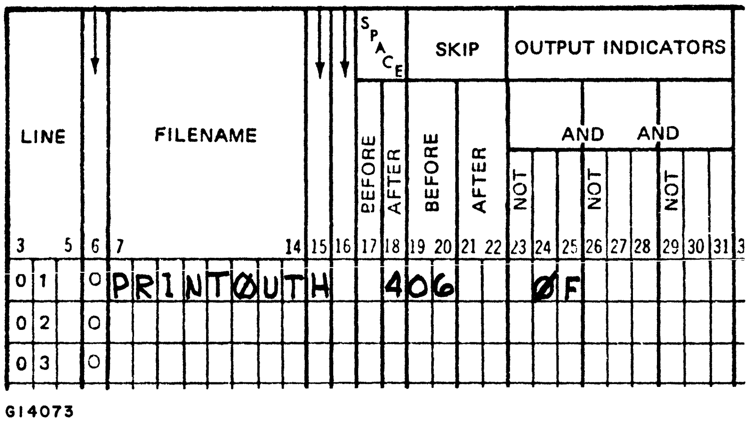
07 fetches overflow.

09 and 11 are printed.

01 is printed after skipping to a new page.

07 is printed if indicator L1 is on.

Forms will not automatically advance to a new page. It is the programmer's responsibility to specify a skip to the first printing line on a new page. This skip to Top-of-Page must be specified on a line conditioned by the overflow indicator (see figure 10-3).



LINE 3-5	6	FILENAME 7-14	15	16	SPACE BEFORE 17	SPACE AFTER 18	SKIP BEFORE 19-20	SKIP AFTER 21-22	OUTPUT INDICATORS NOT 23	24-25	AND NOT 26	27-28	AND NOT 29	30-31
01	O	PRINTØUT	H			4	06			ØF				
02	O													
03	O													

G14073

Figure 10-3. Overflow Indicator with Skip Specified

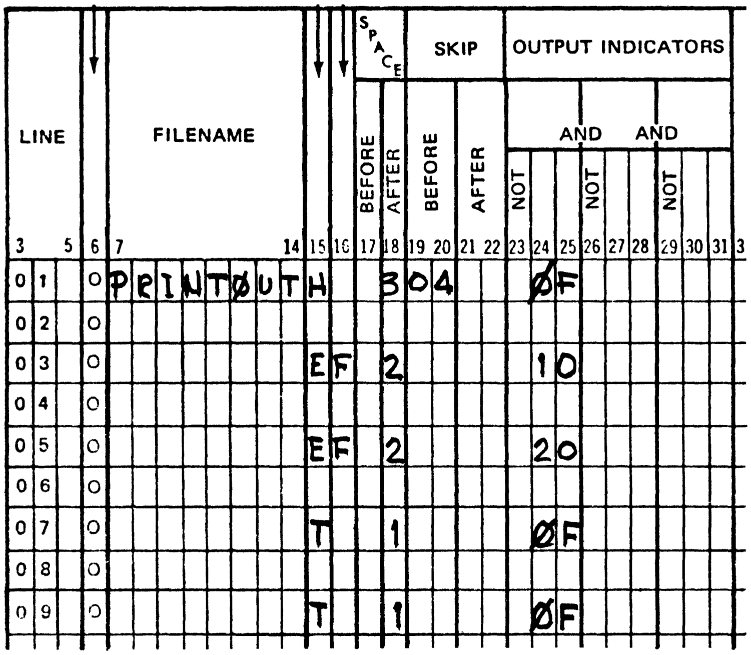
Fetch may be specified for an OR line, but not for an AND line. Fetched overflow can be specified for any detail, total, or exception line, except those conditioned by an overflow indicator.

When more than one printer file is used, fetch pertains only to the overflow lines associated with the file in which the record specifying fetch is defined.

Exception Lines

Since an overflow indicator cannot be specified on an exception output line (E in column 15), FETCH OVERFLOW may be used to cause overflow output at exception time.

The use of fetch overflow (figure 10-4) will cause the heading, detail, and total overflow lines to be printed when the overflow line has been passed (if the conditioning indicators of the fetch line are satisfied). The user may also force overflow by setting ON the appropriate overflow indicator (using the SETON operation code) prior to issuing the EXCPT operation code.



LINE		FILENAME			SPACE		SKIP		OUTPUT INDICATORS		
					BEFORE	AFTER	BEFORE	AFTER	NOT	AND NOT	AND NOT
3 5	6	7 14	15	16	17	18	19 20	21 22	23 24 25	26 27 28	29 30 31
0 1	O	PRINTØUT	H			3	04		ØF		
0 2	O										
0 3	O		E	F		2			10		
0 4	O										
0 5	O		E	F		2			20		
0 6	O										
0 7	O		T			1			ØF		
0 8	O										
0 9	O		T			1			ØF		

G14074

Figure 10-4. FETCH OVERFLOW with Exception Output Coding Example

17-18 SPACE

This field is used to specify forms spacing for printer output files. It is divided into two subfields such that spacing BEFORE or AFTER printing may be specified. If both fields (SPACE and SKIP) are blank, single spacing after printing is assumed.

Valid entries for this field are:

Entry	Definition
Blank	No spacing specified.
0	Space Suppress.
1-9	Space 1-9 lines, as specified.

Refer to the subsection titled Printer File Handling in this section of the manual for additional information.

19-22 SKIP (RPG II DIALECT)

This field is used to specify forms skipping for printer output files. It is divided into two subfields such that skipping BEFORE or AFTER printing may be specified. Valid entries for this field are:

Entry	Definition
0-99	Line number for skip-0-99.
A0-A9	Line number for skip-100-109.
B0-B2	Line number for skip-110-112.
Blank	No skipping specified.

Refer to the subsection titled Printer File Handling in this section of the manual for additional information.

A skip entry must not be greater than the form length as specified in the Line Counter Specifications or as defaulted.

If both skipping and spacing are specified on the same line, the operations are performed in the following order:

a. SKIP BEFORE.

b. SPACE BEFORE.

c. SKIP AFTER.

d. SPACE AFTER.

The user should exercise his option to ensure skipping and spacing AFTER printing since it is more efficient than skipping and spacing BEFORE printing.

Different SPACE and SKIP entries may be specified for OR lines. If all these entries are blank for an OR line, spacing and skipping are done according to the specifications on the preceding line. If any entries are made for an OR line, all desired spacing and skipping must be specified. SPACE and SKIP entries are not permitted on AND lines. Spacing and skipping must not be specified for other than printer files.

23-31 OUTPUT INDICATORS

This field is divided into three subfields such that up to three indicators on each line may be specified to condition an output operation. Each subfield is divided into two parts as follows:

a. NOT (one column).

b. INDICATOR (two columns).

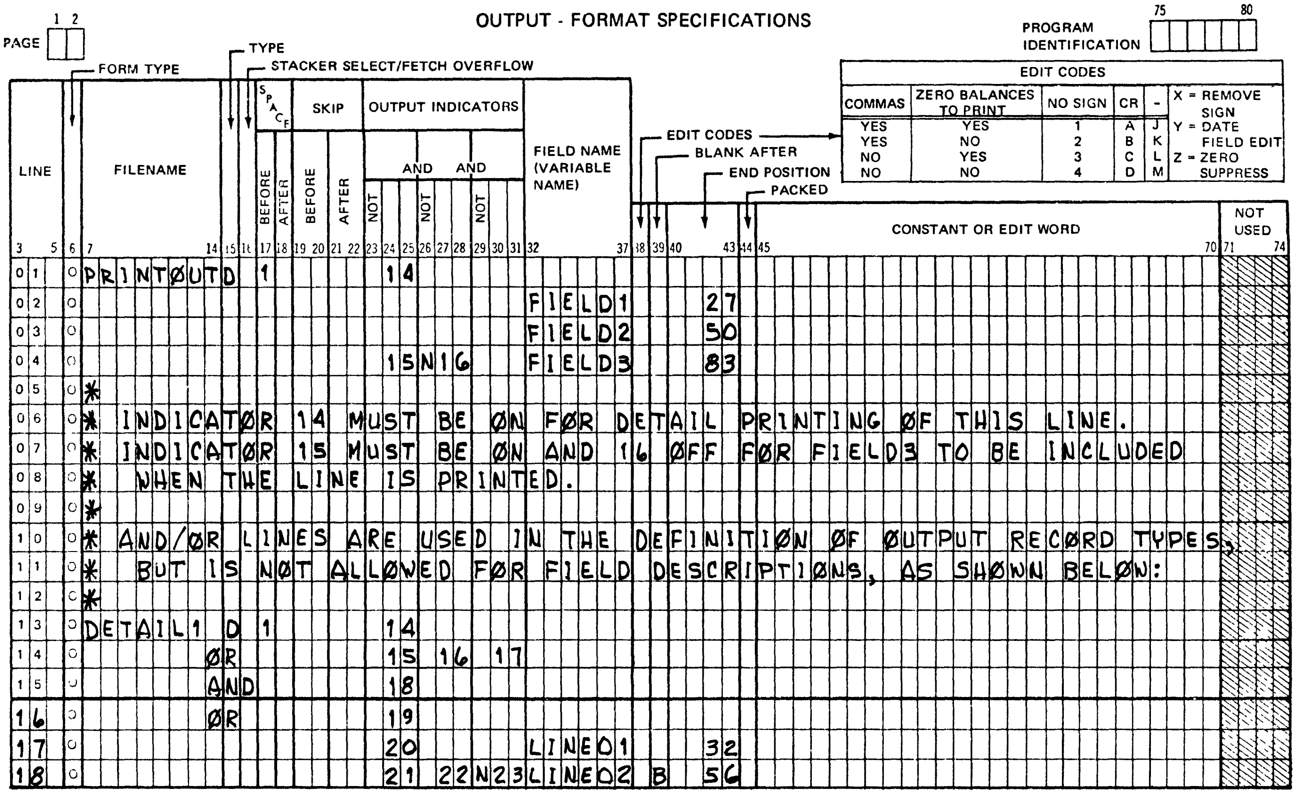
The NOT portion is used to specify that the associated indicator must be OFF in order for the operation to occur. If this condition is desired, an N must be entered in the NOT portion. Otherwise, the NOT portion must be left blank. No output line should be conditioned by all negative indicators (at least one of the indicators used should be positive). If all negative indicators condition a heading or detail line, the line is printed at the beginning of the program cycle when 1P lines are written.

The INDICATOR portion is used to specify the indicator to be tested for ON (NOT = blank) or OFF (NOT = N). The following entries are allowed in this portion of the INDICATORS field:

Entry	Definition
Blank	Operation not conditioned by an indicator.
01-99	Operation conditioned by indicator used elsewhere in the program.
L0-L9	Operation conditioned by control level indicator previously assigned.
LR	Operation conditioned by last record indicator.
MR	Operation conditioned by matching record indicator.
H0-H9	Operation conditioned by halt indicator used elsewhere in the program.
U1-U8	Operation conditioned by external indicator previously set.
OA-OG, OV	Operation conditioned by overflow indicator previously set.
1P	Operation conditioned by first page indicator.

All three indicators on one line are in an AND relationship. All indicators on one line (or grouped lines) must be ON or OFF as specified in order for the associated operation to take place.

An indicator specified on the line describing the record type will condition the entire output record. An indicator used to condition a field within the record is placed on the same line as the field description (see figure 10-5).



OUTPUT - FORMAT SPECIFICATIONS

PAGE 1 2 PROGRAM IDENTIFICATION 75 80

EDIT CODES

COMMAS	ZERO BALANCES TO PRINT	NO SIGN	CR	-	
YES	YES	1	A	J	X = REMOVE SIGN
YES	NO	2	B	K	Y = DATE FIELD EDIT
NO	YES	3	C	L	Z = ZERO SUPPRESS
NO	NO	4	D	M	

LINE
FORM TYPE
FILENAME
TYPE
STACKER SELECT/FETCH OVERFLOW
SPACE
SKIP
BEFORE
AFTER
OUTPUT INDICATORS
AND
NOT
FIELD NAME (VARIABLE NAME)
EDIT CODES
BLANK AFTER
END POSITION
PACKED
CONSTANT OR EDIT WORD
NOT USED

```
  01 OPRINTØUTD 1      14
  02 O                         FIELD1    27
  03 O                         FIELD2    50
  04 O                 15N16   FIELD3    83
  05 O*
  06 O* INDICATØR 14 MUST BE ØN FØR DETAIL PRINTING ØF THIS LINE.
  07 O* INDICATØR 15 MUST BE ØN AND 16 ØFF FØR FIELD3 TO BE INCLUDED
  08 O*   WHEN THE LINE IS PRINTED.
  09 O*
  10 O* AND/ØR LINES ARE USED IN THE DEFINITIØN ØF ØUTPUT RECØRD TYPES,
  11 O*   BUT IS NØT ALLØWED FØR FIELD DESCRIPTIØNS, AS SHØWN BELØW:
  12 O*
  13 ODETAIL1 D 1      14
  14 O       ØR        15 16 17
  15 O       AND       18
  16 O       ØR        19
  17 O                 20      LINE01    32
  18 O                 21 22N23LINE02 B  56
```

G14075

Figure 10-5. Output Indicators Coding Example

The following paragraphs describe the uses of each of the output indicators.

01-99 Numeric indicators may be used as follows:

a. To condition output operations that are only to be performed for specific input record types (record identifying indicators).

b. To condition output operations that are to be done when an input field meets certain conditions (field indicators).

c. To condition output operations according to the results of previous operations in the Calculation Specifications (resulting indicators).

L0-L9 Control level indicators may be used to condition output operations that are to be performed only on the first record of a new control group.

The L0 indicator remains ON during the entire program. If no other control level indicators are assigned, but it is desired to perform total calculation and total output operations, the L0 indicator may be used to condition those operations.

LR The last record indicator is used to condition output operations that are to be performed at End-of-Job.

MR The matching records indicator is used to condition output operations that are to be performed only when matching input records are found.

H0-H9 Halt indicators previously assigned in the FIELD INDICATORS field (Input Specifications) or the RESULTING INDICATORS field (calculation Specifications) may be used to condition output operations that are to be performed only when an error condition occurs.

Since the program does not halt until after the record in error has been completely processed, some operations must be prevented in order to avoid erroneous output. By using halt indicators in conjunction with an N in the NOT portion of the INDICATORS field, an operation may be inhibited when the specified halt indicator is ON.

U1-U8 If an output file is specified as conditioned by an external indicator in the File Description Specifications (EXTERNAL INDICATORS field), every output record described for the file should be conditioned by the same external indicator, otherwise the object program will still build the output record (and perform any blank after operations specified) but must suppress the write operation.

OA-OG,OV Overflow indicators previously assigned in the OVERFLOW INDICATOR field (File Description Specifications) may be used to condition output operations that are to be performed when the overflow line on a printer file has been reached.

Overflow indicators which have not been previously assigned in the File Description Specifications (except OF and OV, which are compiler defined) may not be used in the Output Format Specifications. Forms advancing at End-of-Page are handled automatically if no overflow indicators are assigned to the file. Any specification line not conditioned by an overflow indicator which designates a skip to the next page turns OFF all overflow indicators before the skip takes place.

Overflow indicators must not be used to condition exception records but may condition fields within exception records.

No more than one overflow indicator may be associated with a group of output indicators in an AND or OR relationship, and it must be the same indicator assigned to the file in the File Description Specifications.

The overflow line can be sensed when printing total or detail time output. If an overflow indicator is being used to condition output lines, the following steps occur when the overflow line is sensed during total time output:

a. The overflow indicator turns on.

b. The remaining total lines not conditioned by overflow are printed.

c. All total lines conditioned by overflow are printed. If skipping is specified on these lines, it occurs as defined by the user.

d. Heading and detail output lines conditioned by overflow are printed. If skipping is specified on these lines, it occurs as defined by the user.

e. Heading and detail lines not conditioned by overflow are printed.

f. The overflow indicator turns off.

If the overflow line is sensed during detail time output, the following steps occur:

a. The overflow indicator turns on.

b. The remaining detail lines not conditioned by overflow are printed.

c. All total lines not conditioned by overflow are printed.

d. All total lines conditioned by overflow are printed. If skipping is specified on these lines, it occurs as defined by the user.

e. Heading and detail output lines conditioned by overflow are printed. If skipping is specified on these lines, it occurs as defined by the user.

f. Heading and detail lines not conditioned by overflow are printed.

g. The overflow indicator turns off.

When using the overflow indicator to condition overflow printing, remember:

a. Overflow indicators may be turned on and off by the operation codes SETON and SETOF.

b. Spacing past the overflow line causes the overflow indicator to turn on.

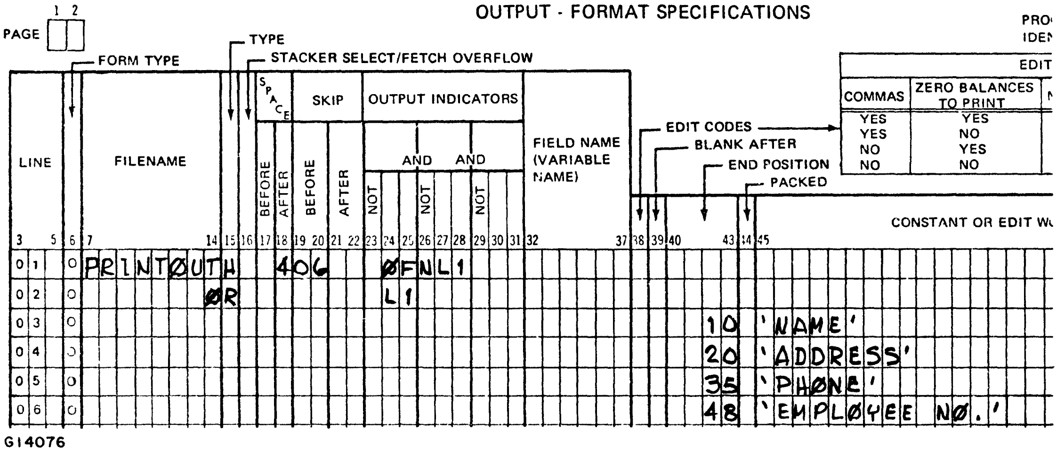
c. Spacing or skipping past the overflow line to any line on the new page does not turn the overflow indicator on.

d. A skip to a new page specified on a line not conditioned by an overflow indicator causes the overflow indicator to turn off.

Control Level Indicators With Overflow Indicators

If it is desired to have headings identifying the type of information on each page or each page to contain information from only one control group, control level indicators may be used in conjunction with overflow indicators. Together they condition when headings and/or group information are to be printed.

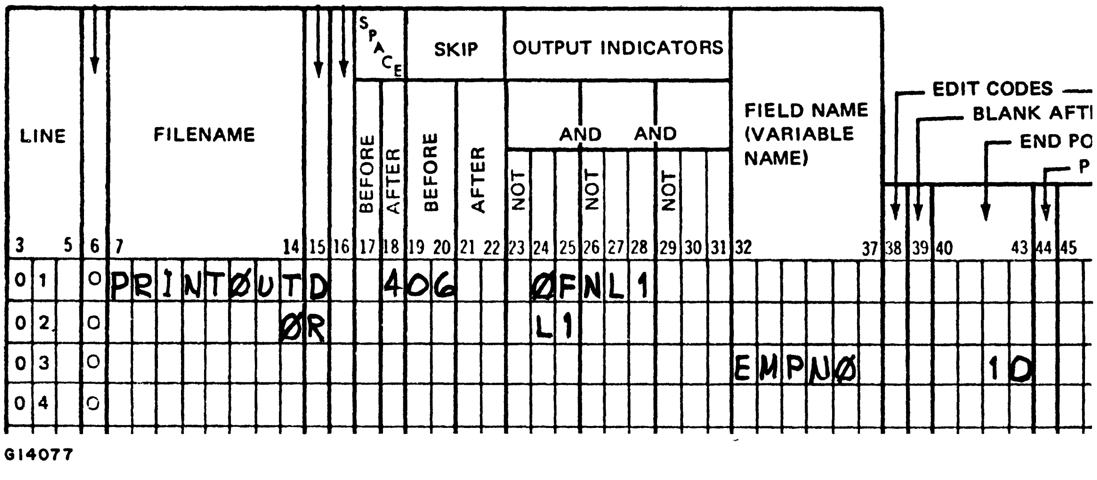
In the example in figure 10-6, the control level indicator L1 is used in conjunction with the overflow indicator in order to print headings on every page. Line 01 allows the headings to be printed at the top of a new page only when overflow occurs. Line 02 allows printing of headings on a new page only at the beginning of a control group (L1). In this way, duplicate headings caused by both L1 and OF being ON at the same time will not occur.


OUTPUT - FORMAT SPECIFICATIONS
PAGE
FORM TYPE
TYPE
STACKER SELECT/FETCH OVERFLOW
SPACE
SKIP
OUTPUT INDICATORS
LINE
FILENAME
BEFORE
AFTER
AND
NOT
FIELD NAME (VARIABLE NAME)
EDIT CODES
BLANK AFTER
END POSITION
PACKED
COMMAS
ZERO BALANCES TO PRINT
CONSTANT OR EDIT W
G14076

```
01 O PRINTØUTH  406    ØFNL1
02 O         ØR        L1
03 O                                10 'NAME'
04 O                                20 'ADDRESS'
05 O                                35 'PHØNE'
06 O                                48 'EMPLØYEE NØ.'
```

Figure 10-6. Using Control Level Indicators With Overflow Indicators

Figure 10-7 shows the necessary coding for the printing of certain fields on every page.


SPACE
SKIP
OUTPUT INDICATORS
LINE
FILENAME
BEFORE
AFTER
AND
NOT
FIELD NAME (VARIABLE NAME)
EDIT CODES
BLANK AFTER
END PO
G14077

```
01 O PRINTØUTD  406    ØFNL1
02 O         ØR        L1
03 O                              EMPNØ     10
04 O
```

Figure 10-7. Coding for the Printing of Certain Fields

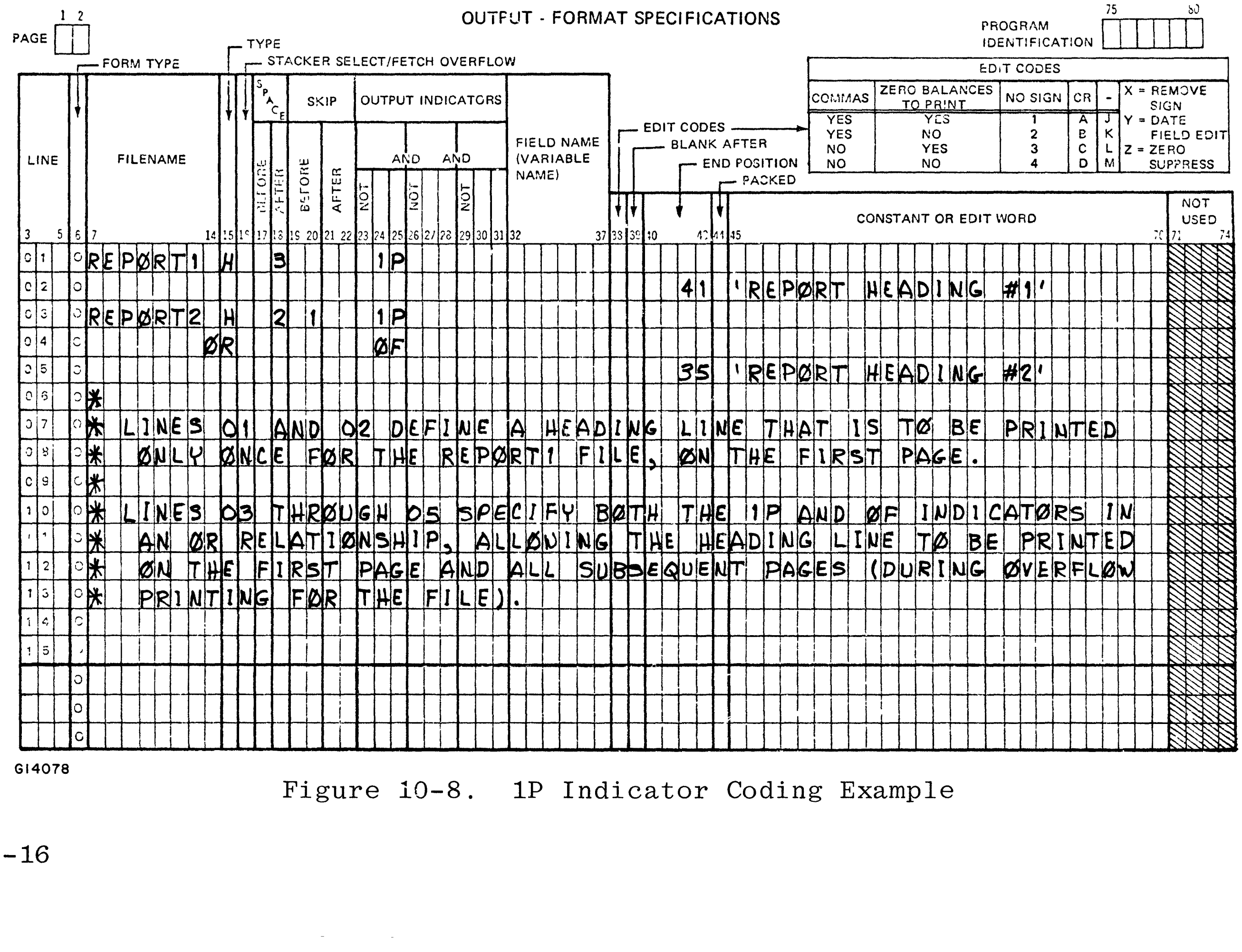
1P The first page indicator is used to condition HEADING and DETAIL printer output before the first record is processed. The 1P indicator is turned on by the RPG program and is set off after detail-time output and before the first record is processed. It cannot be used to condition calculations, output at total time, or output at exception time.

All lines conditioned by the 1P indicator are written even before the first record from any input file is processed. Therefore, do not condition output fields (except PAGE, UPDATE, UYEAR, UMONTH, UDAY, and JDATE) which are based upon data from input records by the 1P indicator.

The 1P indicator may also be used in an OR relationship with the overflow indicators OA-OG, OV. This allows printing the same HEADING or DETAIL information on all pages.

The 1P indicator is invalid for update and combined files.

When forms are first inserted in the printer, they may not always be in perfect alignment. Sometimes several lines must be printed to determine the correct positioning of the form. Specifying the number 1 in column 41 of the Control Card Specification, allows the option of repeatedly printing the first line conditioned by the 1P indicator. Each time the 1P line is printed, the program


OUTPUT - FORMAT SPECIFICATIONS
PAGE
PROGRAM IDENTIFICATION
FORM TYPE
TYPE
STACKER SELECT/FETCH OVERFLOW
LINE
FILENAME
SPACE
BEFORE
AFTER
SKIP
OUTPUT INDICATORS
AND
NOT
FIELD NAME (VARIABLE NAME)
EDIT CODES
BLANK AFTER
END POSITION
PACKED
CONSTANT OR EDIT WORD
NOT USED
COMMAS
ZERO BALANCES TO PRINT
NO SIGN
CR
X = REMOVE SIGN
Y = DATE FIELD EDIT
Z = ZERO SUPPRESS

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

```
01 OREPØRT1 H  3     1P
02 O                                   41 'REPØRT HEADING #1'
03 OREPØRT2 H  2 1   1P
04 O       ØR        ØF
05 O                                   35 'REPØRT HEADING #2'
06 O*
07 O* LINES 01 AND 02 DEFINE A HEADING LINE THAT IS TØ BE PRINTED
08 O*   ØNLY ØNCE FØR THE REPØRT1 FILE, ØN THE FIRST PAGE.
09 O*
10 O* LINES 03 THRØUGH 05 SPECIFY BØTH THE 1P AND ØF INDICATØRS IN
11 O*   AN ØR RELATIØNSHIP, ALLØWING THE HEADING LINE TØ BE PRINTED
12 O*   ØN THE FIRST PAGE AND ALL SUBSEQUENT PAGES (DURING ØVERFLØW
13 O*   PRINTING FØR THE FILE).
14 O
15 O
```

G14078

Figure 10-8. 1P Indicator Coding Example halts allowing the operator to reposition the form, if needed. To continue processing, enter on the CONSOLE (SPO): < program number > AXGO.

AND/OR If it is necessary to specify more than three indicators to condition an output operation, an AND line may be used. The word AND must be entered in columns 14-16, and the additional indicators entered in their respective fields. The conditions specified for all indicators in an AND relationship must be met before the associated output operation will take place.

OR lines (OR in columns 14-15) may be used to group indicators such that only one of the conditions specified must be met for the associated output operation to take place. Both AND and OR lines may be used together to condition an output record (but not a field). A maximum of three indicators in an AND relationship (on one line) may be used to condition a field. See figure 10-5 for an example of the usage of AND and OR lines.

There is no limit on the number of AND or OR lines that may be specified. However, it is recommended that the user not exceed 20 if compatibility with other Burroughs systems is desired. There must be at least one indicator on an AND/OR line and on the preceding line.

32-27 FIELD NAME (VARIABLE NAME)

This field is used to assign an identifier (name) to an output data field. The identifier used must have been previously defined in the Input Specifications (VARIABLE NAME field), the Extension Specifications (VECTOR NAME field), or the Calculation Specifications (RESULT FIELD field). Also, any special words may be used. A separate line must be used for each field description. Fields may be listed in any order within each record type, since their location is determined by the entry in the END POSITION fields. If fields overlap, only the last field specified appears intact in the output record (the exact results will depend upon the degree of overlap).

Special Words

The following special words are reserved for use as variable names:

PAGE through PAGEn
UDATE
UMONTH
UDAY
UYEAR
JDATE
*PLACE
*PRINT

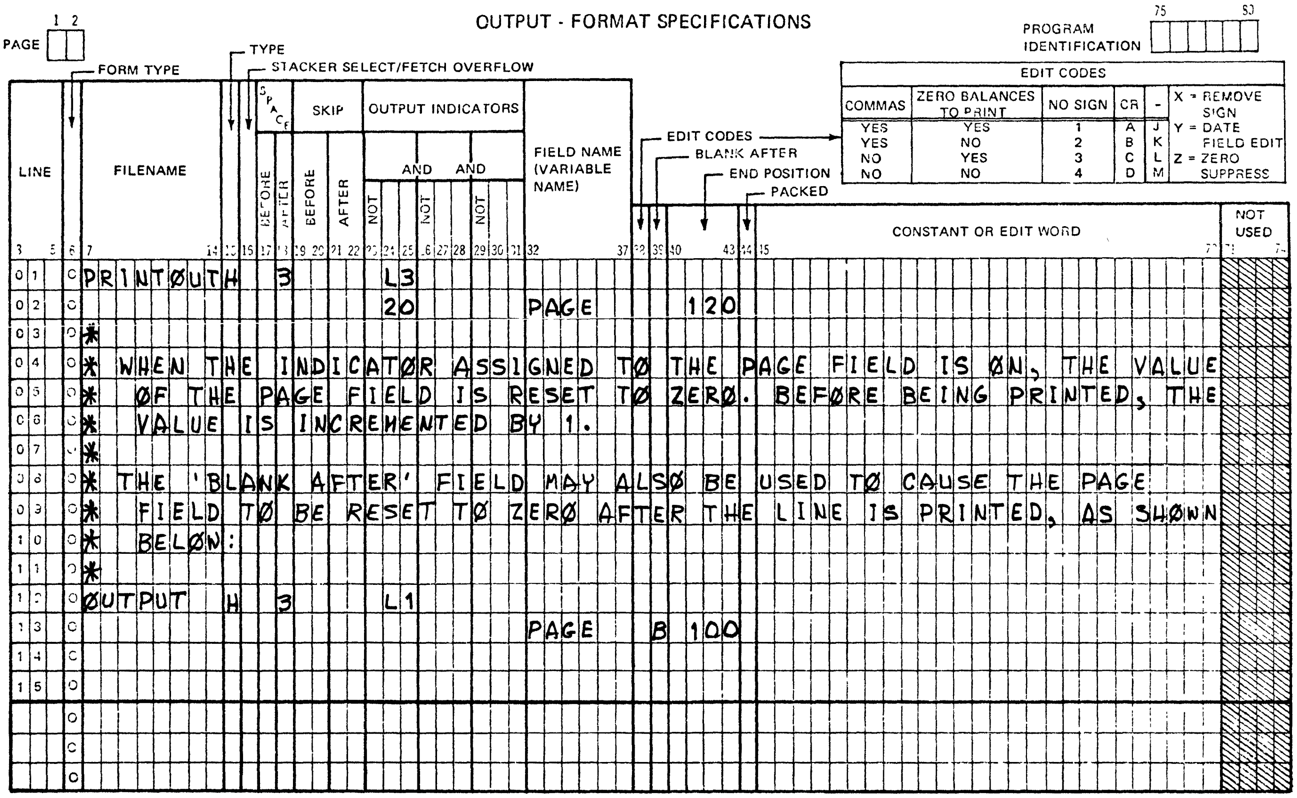
Each special word has a specification defined usage, as described in the following paragraphs:

Page Fields

When page numbering is to be done on output, the special words PAGE through PAGEn are used to indicate that automatic page numbering is to be performed. When a PAGE field is named in this field without being previously defined in the Input or Calculation Specifications, it is assumed to be four characters in length with no decimal positions. On output, leading zeros are suppressed and the sign is not printed unless an edit word or edit code is specified.

The page number begins at zero (unless otherwise specified), and is automatically incremented by one each time before the page field is written.

The page number may be reset at any point during the program by setting the PAGE field to zero before it is printed (see figure 10-9). This may be accomplished in two ways:


OUTPUT - FORMAT SPECIFICATIONS
PAGE
PROGRAM IDENTIFICATION
FORM TYPE
TYPE
STACKER SELECT/FETCH OVERFLOW
LINE
FILENAME
SPACE
BEFORE
AFTER
SKIP
BEFORE
AFTER
OUTPUT INDICATORS
NOT
AND
NOT
AND
NOT
FIELD NAME (VARIABLE NAME)
EDIT CODES
BLANK AFTER
END POSITION
PACKED
CONSTANT OR EDIT WORD
NOT USED

EDIT CODES						
COMMAS	ZERO BALANCES TO PRINT	NO SIGN	CR	-	X = REMOVE SIGN Y = DATE FIELD EDIT Z = ZERO SUPPRESS
YES	YES	1	A	J	
YES	NO	2	B	K	
NO	YES	3	C	L	
NO	NO	4	D	M	

```
  01 OPRINTØUTH  3     L3
  02 O                 20      PAGE     120
  03 O*
  04 O* WHEN THE INDICATØR ASSIGNED TØ THE PAGE FIELD IS ØN, THE VALUE
  05 O*   ØF THE PAGE FIELD IS RESET TØ ZERØ. BEFØRE BEING PRINTED, THE
  06 O*   VALUE IS INCREMENTED BY 1.
  07 O*
  08 O* THE 'BLANK AFTER' FIELD MAY ALSØ BE USED TØ CAUSE THE PAGE
  09 O*   FIELD TØ BE RESET TØ ZERØ AFTER THE LINE IS PRINTED, AS SHØWN
  10 O*   BELØW:
  11 O*
  12 OØUTPUT  H  3     L1
  13 O                         PAGE   B 100
  14 O
  15 O
```

G14079

Figure 10-9. PAGE Specification Coding Example

a. Use the BLANK AFTER specification to cause the field to be cleared to zero after printing, or

b. Assign an OUTPUT INDICATOR to the PAGE field. If the indicator is ON, the field will be set to zero before normal incrementation takes place.

The same PAGE through PAGEn entry may be used for two different output files, but this is not recommended. PAGE fields are not restricted to printer files.

Date Fields (UDATE, UMONTH, UDAY, UYEAR and JDATE)

Five special words allow the program to obtain the current value of the date as supplied through the B 1800/B 1700 System. The following rules apply to date fields.

a. UDATE gives a 6-digit numeric field in one of the following formats (depending upon the entry in the INVERTED PRINT field of the Control Card):

   1) Domestic (MMDDYY), or

   2) International (DDMMYY).

b. The other three fields, UMONTH, UDAY, and UYEAR, each gives a 2-digit field representing the month, day, and year, respectively.

c. These fields must not be changed by any operations within the program; thus, they are usually used only in compare, test, and output operations.

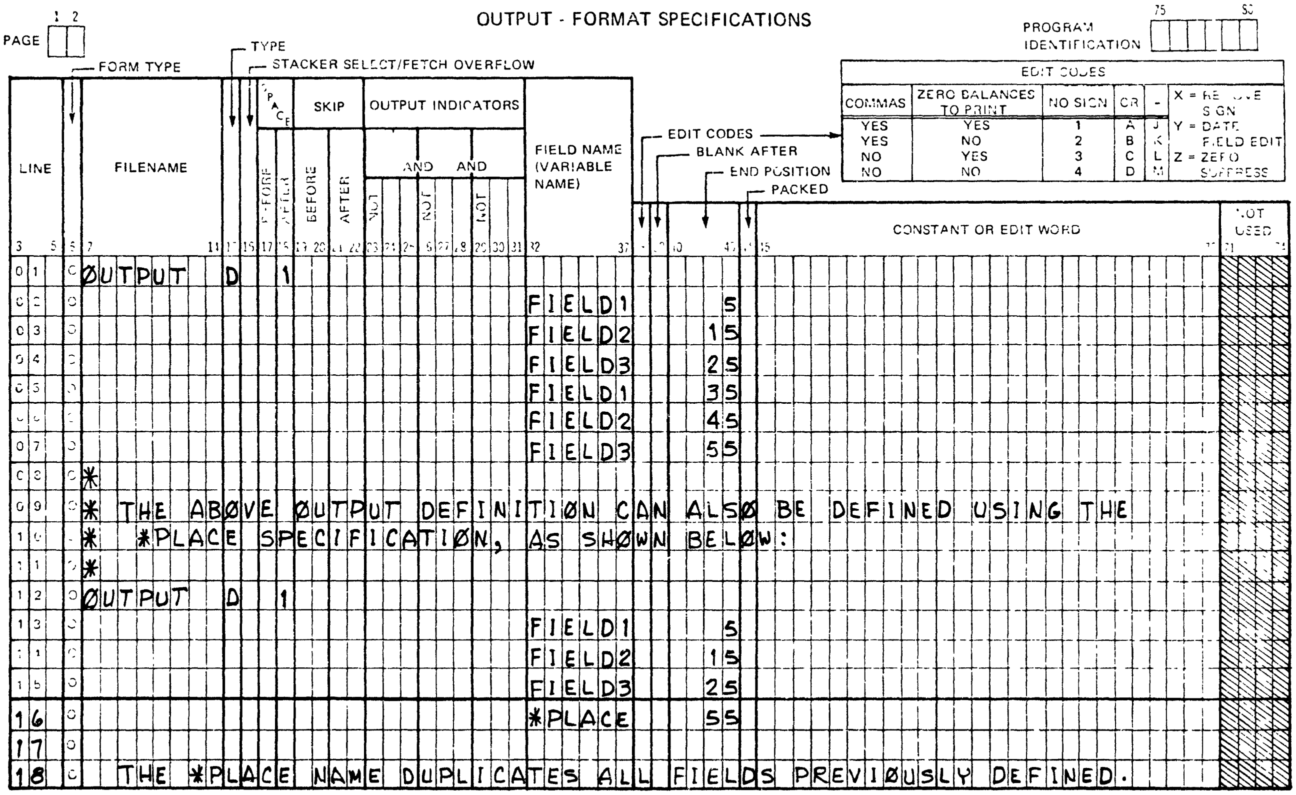
d. JDATE is a special word reserved for accessing the Julian date as obtained from the system. JDATE can be used anywhere that UDATE can be used. JDATE can not be changed in any way by the RPG program. JDATE is five digits long with zero decimal places and has the format YYDDD.

*PLACE Specification

The special word *PLACE allows writing of the same field or fields in more than one place in an output record without having to specify the field names and end positions more than once. The designated fields are written in the same relative positions ending in the position specified for the *PLACE entry. It is possible to obtain the same results in two ways (see figure 10-10).

a. Define each field and its corresponding end position for every time it is to appear in the output records, or

b. Use the special word *PLACE.



OUTPUT - FORMAT SPECIFICATIONS

PAGE · PROGRAM IDENTIFICATION · FORM TYPE · TYPE · STACKER SELECT/FETCH OVERFLOW · LINE · FILENAME · SPACE BEFORE AFTER · SKIP BEFORE AFTER · OUTPUT INDICATORS NOT AND NOT AND NOT · FIELD NAME (VARIABLE NAME) · EDIT CODES · BLANK AFTER · END POSITION · PACKED · CONSTANT OR EDIT WORD · NOT USED

EDIT CODES

COMMAS	ZERO BALANCES TO PRINT	NO SIGN	CR	-	
YES	YES	1	A	J	X = REMOVE SIGN
YES	NO	2	B	K	Y = DATE FIELD EDIT
NO	YES	3	C	L	Z = ZERO SUPPRESS
NO	NO	4	D	M	

```
01 OØUTPUT D 1
02 O                        FIELD1     5
03 O                        FIELD2    15
04 O                        FIELD3    25
05 O                        FIELD1    35
06 O                        FIELD2    45
07 O                        FIELD3    55
08 O*
09 O* THE ABØVE ØUTPUT DEFINITIØN CAN ALSØ BE DEFINED USING THE
10 O*  *PLACE SPECIFICATIØN, AS SHØWN BELØW:
11 O*
12 OØUTPUT D 1
13 O                        FIELD1     5
14 O                        FIELD2    15
15 O                        FIELD3    25
16 O                        *PLACE    55
17 O
18 O  THE *PLACE NAME DUPLICATES ALL FIELDS PREVIOUSLY DEFINED.
```

G14080

Figure 10-10. *PLACE Specification Coding Example

Both methods will produce identical results, but use of the *PLACE entry saves extra coding.

The following rules must be observed when using the *PLACE specification:

a. All fields within the record type written above the *PLACE entry are repeated according to the *PLACE specification, not just the one line above.

b. An end position must be given for every *PLACE specification.

c. An additional *PLACE entry (on a separate line) must be used every time the fields are to be repeated.

d. *PLACE must be specified after the field names which are to be placed in different positions on the line; *PLACE must not be specified on the first field description line for a record.

e. The end position specified for *PLACE should be at least twice the highest previously specified end position. If enough space is not allowed for all fields to be printed again, overlapping will occur, with the *PLACE output overlapping the previous characters.

f. The end position specified for *PLACE must not be lower than the highest previously specified field end position.

g. The leftmost position of the fields to be moved by the *PLACE specification is always assumed to be position 1.

h. When *PLACE is specified for card output, the fields and constants named above will be repunched. Any printed output on the cards will not be reprinted unless an * is entered in column 40 of the same line as *PLACE.

i. Only the conditioning indicators (columns 23-31), FIELD NAME (columns 32-37) and END POSITION (columns 40-43) may have entries.

*PRINT Specification

The special word *PRINT is used to cause card interpreting after punching (for card files only). Printing is done at the top of the cards in the same column positions as the fields are punched. The *PRINT specification must be used only once for each record and appears after all fields on the card which are to be printed. The *PRINT specification may be conditioned by indicators in the OUTPUT INDICATORS field; all other fields must be left blank (see figure 10-11).

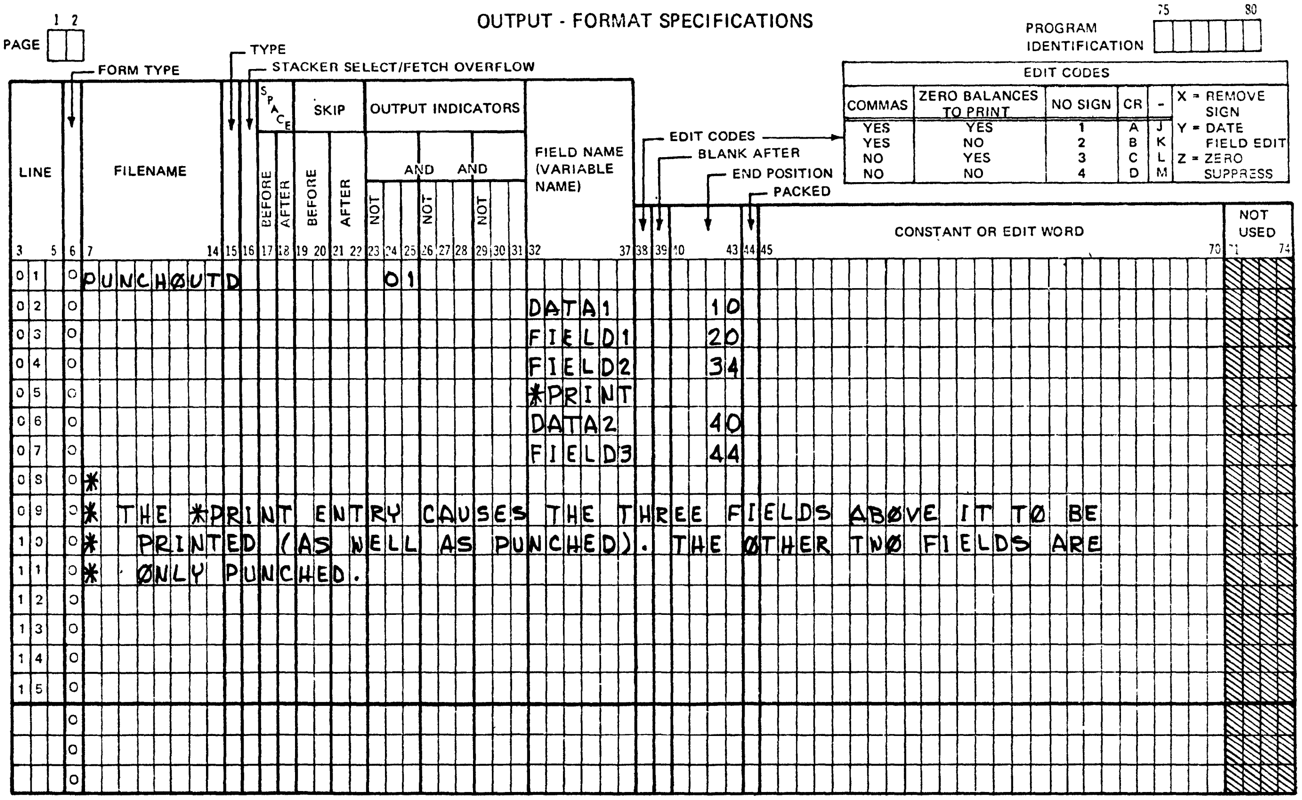
Having the *PRINT specification print the corresponding field exactly as punched is not always desirable.

To print the fields in positions other than would be assigned by the *PRINT specifications:

a. Enter the field name to be printed in the VARIABLE NAME field, and

b. Enter an asterisk in column 40, and

c. Enter the end position for the field in columns 41-43 (limited to a maximum of 128), right-justified; leading zeros not required.



OUTPUT - FORMAT SPECIFICATIONS

PAGE □□ (1 2)

PROGRAM IDENTIFICATION □□□□□□ (75 80)

FORM TYPE · TYPE · STACKER SELECT/FETCH OVERFLOW · SPACE · SKIP · OUTPUT INDICATORS · FIELD NAME (VARIABLE NAME) · EDIT CODES · BLANK AFTER · END POSITION · PACKED · CONSTANT OR EDIT WORD · NOT USED

EDIT CODES

COMMAS	ZERO BALANCES TO PRINT	NO SIGN	CR	-	
YES	YES	1	A	J	X = REMOVE SIGN
YES	NO	2	B	K	Y = DATE FIELD EDIT
NO	YES	3	C	L	Z = ZERO SUPPRESS
NO	NO	4	D	M	

```
LINE   FILENAME                         FIELD NAME      END POSITION
3 5 6 7                                 32    37    40 43
01  OPUNCHØUTD        01
02  O                         DATA1        10
03  O                         FIELD1       20
04  O                         FIELD2       34
05  O                         *PRINT
06  O                         DATA2        40
07  O                         FIELD3       44
08  O*
09  O* THE *PRINT ENTRY CAUSES THE THREE FIELDS ABØVE IT TØ BE
10  O*   PRINTED (AS WELL AS PUNCHED). THE ØTHER TWØ FIELDS ARE
11  O*   ØNLY PUNCHED.
12  O
13  O
14  O
15  O
    O
    O
    O
```

G14081

Figure 10-11. *PRINT Specification Coding Example

38 EDIT CODES

This field is used to specify editing of an unpacked numeric output field. Editing operations have been provided, for which it is not necessary to write an edit word. These operations and their corresponding edit codes are summarized in the EDIT CODES portion of the Output-Format Specifications sheet.

Only unpacked numeric fields can be edited, and if an edit code is specified, the Constant or Edit Word field must be left blank, except when the check protect or floating dollar sign is required. In that case, enter an asterisk (*) or dollar sign ($), respectively, in column 45-47. Edit codes X, Y, and Z may never be used with the asterisk or dollar sign. The Y edit code must only be used with field lengths of three to six digits.

When an entire array is to be edited with an edit code, the following points must be taken into account to avoid specifying the end position less than the total size of the output data:

a. Two spaces are inserted to the left of each element, and

b. Commas, decimal points, and minus signs occupy one character each. The credit sign (CR) occupies two characters.

The following are the edit code rules:

a. Conditioning indicators on the record description line and the field description line determine when the field is written.

b. The variable name specified in columns 32-37 must be previously defined as:

   1. An entire numeric array, or

   2. A single element of a numeric Table or Array, or

   3. A numerically defined field, or

   4. A numerically defined special word.

c. The use of an edit code in column 38 will cause various punctuation of the numeric field specified in columns 32-37.

d. The use of B in BLANK AFTER, column 39, clears the field after output but does not affect any editing specified on this line.

e. End Position, columns 40-43, defines where the last character of the edited output is to appear.

NOTE

Syntax errors occur if the total length in bytes of the variable and all inserted decimals, commas, and status reporting is greater than the length in bytes from the specified end position to position one of the output record.

f. Packed, column 44, must be blank.

g. The Constant or Edit Word portion of this specification line must remain blank, except for the following two exceptions:

   1. If the check protect asterisk is required, enter an asterisk in columns 45-47.

2. If the floating dollar sign is required, enter a dollar sign in columns 45-47.

NOTE

The check protect asterisk and the floating dollar sign may never be used with edit codes X, Y, or Z.

h. Edit codes A, B, J, K, 1, and 2 provide the comma option. Commas will only be inserted in the edited output if significant non-zero integers exceed three in number.

i. Edit codes A, C, J, L, 1, and 3 provide the zero balance option. This option provides that at least one zero will be written if the value of the field is zero. The number of zeroes written is determined by the decimal positions assigned when the field was defined, as shown in the following example:

Decimal Position	Zero Balance Output
0	0
1	.0
2	.00
9	.000000000

j. A sign or symbol for a negative number may be requested or ignored.

1. When requested, A, B, C, and D provide CR for negative status report and J, K, L, and M provide - for negative status report.

2. When ignored, 1, 2, 3, and 4 provide for no status report.

k. Edit code X provides for removal of the plus sign from the field with no zero suppression.

l. Edit code Z provides for removal of all signs and the suppression of leading zeroes.

m. Edit code Y provides for editing in date field format. The variable must be no less than three digits and no more than six digits in length.

n. An entire array can be edited with one edit code. Include the following when specifying the end position:

1. One space for each character of the array, plus

2. Two spaces to be inserted to the left of each element, plus

3. One space for each comma that may be inserted, plus

4. One space for each decimal that will be inserted, plus one of the following unless using edit codes 1, 2, 3, or 4.

   (a) One space for each minus sign with each element, or

   (b) Two spaces for each CR sign with each element.

Table 10-1. Edit Codes Table

Edit Code	Commas	Decimal Point	Zero Suppress	Sign For Negative Balance	Printout on Zero Balance*		
					International I	International J	Domestic United Kingdom
1	Yes	Yes	Yes	No Sign	,00 or 0	0,00 or 0	.00 or 0
2	Yes	Yes	Yes	No Sign	Blanks	Blanks	Blanks
3		Yes	Yes	No Sign	,00 or 0	0,00 or 0	.00 or 0
4		Yes	Yes	No Sign	Blanks	Blanks	Blanks
A	Yes	Yes	Yes	CR	,00 or 0	0,00 or 0	.00 or 0
B	Yes	Yes	Yes	CR	Blanks	Blanks	Blanks
C		Yes	Yes	CR	,00 or 0	0,00 or 0	.00 or 0
D		Yes	Yes	CR	Blanks	Blanks	Blanks
J	Yes	Yes	Yes	-	,00 or 0	0,00 or 0	.00 or 0
K	Yes	Yes	Yes	-	Blanks	Blanks	Blanks
L		Yes	Yes	-	,00 or 0	0,00 or 0	.00 or 0
M		Yes	Yes	-	Blanks	Blanks	Blanks
X**							
Y***			Yes				
Z			Yes				

NOTES

*Zero balances for the International format are printed or punched in two ways, depending on the entry made in column 21 of the Control Card Specifications.

**The X code removes the sign.

***The Y code suppresses the leftmost zero only. The Y code edits a three to six digit field according to the following pattern:

nn/n
nn/nn
nn/nn/n
nn/nn/nn

39 BLANK AFTER

This field is used to specify that a variable is to be reset after the output operation is finished. Alphanumeric fields will be cleared to blanks and numeric fields will be cleared to zeros. Constants and UDATE, UDAY, UMONTH, and UYEAR must not be specified with BLANK AFTER. Valid entries for this field are:

Entry	Definition
Blank	Field is not to be cleared.
B	Variable is to be cleared after being moved to the output buffer.

Entering a B in column 39 specifies that the field named in columns 32-37 is to be set to zero or blank if the field is numeric or alphanumeric, respectively. This is to be done immediately after output to the designated record work area.

When an indicator is assigned to a field to test for zero or blank in the Input or Calculation Specifications and the same field is used on Output Specifications with

BLANK AFTER, then that zero/blank indicator will be affected as follows:

RPG II Dialect

DEFAULT:	No effect on zero/blank indicator.
$ BAZBON:	Specified indicator is turned ON after the field is blanked during the output operations.

RPG I Dialect

DEFAULT:	Specified indicator is turned ON after the field is blanked during the output operation.
$NBAZBON:	No effect on zero/blank indicator.

RPG I And RPG II Dialects

If more than one indicator is assigned to the same field as a ZERO/BLANK indicator (on different Input or Calculation Specifications), then only the first one assigned is turned ON when the field is blanked out. When an array or array elements are output with BLANK AFTER, zero/blank indicators are not changed.

When the specified field is a table name, the element that satisfied the last successful LOKUP operation is blanked or zeroed.

40-43 END POSITION

This field is used to specify the location of a field within an output record. Only the location of the rightmost position is specified; the number of positions to the left of that location is determined by the length specified previously for the field. Space must also be left to allow for any editing symbols (edit codes or edit words) that may be entered. The end position must not exceed the record length specified for the file,except in the special case where an * is entered in column 40.

To print fields on a card in a position other than the one assigned by using the *PRINT specification, enter an asterisk in column 40 and proceed as follows:

a. Enter the field name to be printed in the Variable Name field, or enter a literal in columns 45-70. A variable can be edited by means of an edit code or edit word.

b. Enter the end position for the field in columns 41-43, right-justified. Leading zeroes are not required.

Table 10-2 Illustrates the results obtained when the various edit codes are used to edit a two decimal position field whose value is a negative (-3.12). The Output Specifications specify an ending position of 12.

Table 10-2. Effect of Edit Codes on End Position

Edit Codes	Output Print Positions								
	5	6	7	8	9	10	11	12	13
Unedited				-*	0	3	1	2	
1					3	•	1	2	
2					3	•	1	2	
3					3	•	1	2	
4					3	•	1	2	
A			3	•	1	2	C	R	
B			3	•	1	2	C	R	
C			3	•	1	2	C	R	
D			3	•	1	2	C	R	
J				3	•	1	2	-	
K				3	•	1	2	-	
L				3	•	1	2	-	
M				3	•	1	2	-	
X				0	0	3	1	2	
Y			0	/	3	1	/	2	
Z						3	1	2	

* - Represents a negative 0.

44 PACKED OR BINARY

This field is used to specify that a numeric output field is to be written in packed decimal or binary format. Packed decimal and binary fields should not be printed because the resultant printout will be meaningless.

Valid entries for this field are:

Entry	Definition
Blank	Field is to be written in alphanumeric or unpacked decimal format.
P	Field is to be written in packed decimal format.
B	Field is to be written in binary format.

When the letter B is specified to indicate binary format, the length of the numeric field that is to be written cannot exceed nine bytes, and the following rules apply:

a. Numeric output fields that have a length of one digit to four digits inclusive are converted to 2-byte binary fields.

b. Numeric output fields that have a length of five digits to nine digits inclusive are converted to 4-byte binary fields.

c. Output in binary format is only allowed for disk, tape, or 80-column card devices.

d. Arrays, tables, array elements, and numeric fields can be specified.

Column 44 must be left blank:

a. When an asterisk (*) appears in column 40 of the same Output Specification line.

b. For fields that precede an *PRINT specification for a card file.

c. For fields that precede an *PLACE specification for a card or printer file.

d. For alphanumeric fields.

e. When a constant is specified.

f. When an edit code is specified.

g. When an edit word is specified.

45-70 CONSTANT OR EDIT WORD

Constants or Edit Words are entered in this field. A description of each is provided in the paragraphs that follow.

Constants

Constants are literals usually used for such things as page headings, and contain information that is not changed by an operation. Constants will appear exactly as written in the Output-Format Specifications, in the position specified. The following rules must be observed when forming constants:

a. The VARIABLE NAME entry must be left blank.

b. Constants must be left-justified, enclosed in apostrophes, and no more than 24 characters in length.

c. If an apostrophe is to appear in the constant, two consecutive apostrophes must be coded for each apostrophe required. In reference to the end position, the two apostrophes are to be counted as one.

d. Numeric data may be used as a constant, but must still be enclosed in apostrophes. Note that this is an alphanumeric literal composed of numeric characters.

The example in figure 10-12 uses the Line Printer Spacing Form to illustrate the coding of output constants.

Edit Words

Edit words give a program more adaptability than edit codes when editing numeric data fields. These data fields are made more meaningful and easier to read by inserting one or more of the following characters:

a. Comma (,)

b. Slash (/)

c. Decimal point (.)

d. Dollar sign ($)

e. Asterisk protection (*)

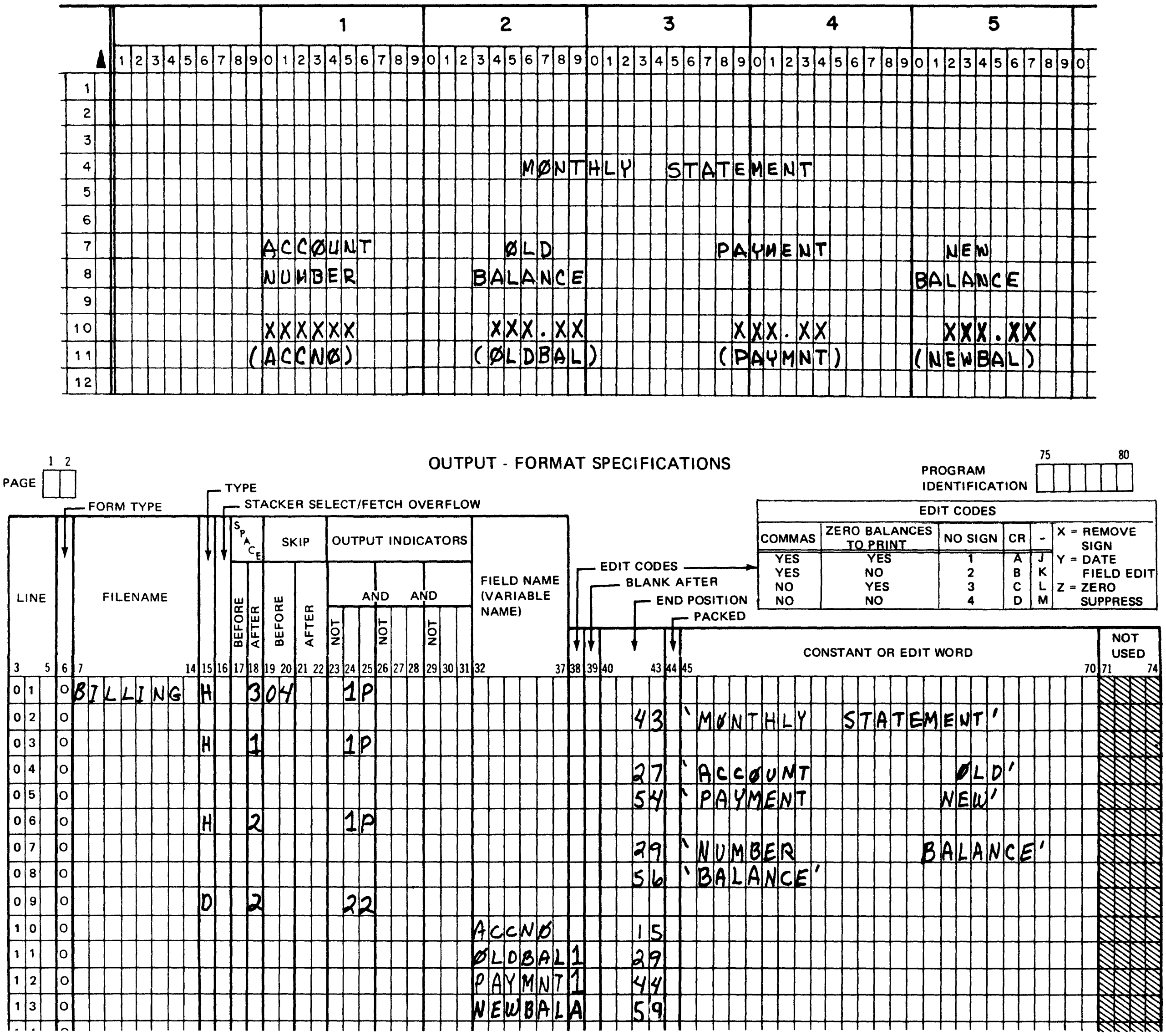
f. Zero (0)

g. Credit field identifier (CR)

h. Negative field identifier (-)

i. Constants

When a numeric data field is to be edited, an edit word (sometimes called an edit mask) must be placed in columns 45-70 of the Output-Format Specifications. The edit word must begin in Column 26, be enclosed in apostrophes, and be no more than 24 characters in



```
                         MØNTHLY  STATEMENT

         ACCØUNT        ØLD          PAYMENT       NEW
         NUMBER       BALANCE                     BALANCE

         XXXXXX        XXX.XX         XXX.XX       XXX.XX
        (ACCNØ)       (ØLDBAL)       (PAYMNT)    (NEWBAL)
```

OUTPUT - FORMAT SPECIFICATIONS

PAGE □□ PROGRAM IDENTIFICATION □□□□□□

Commas	Zero Balances to Print	No Sign	CR	-	
YES	YES	1	A	J	X = REMOVE SIGN
YES	NO	2	B	K	Y = DATE FIELD EDIT
NO	YES	3	C	L	Z = ZERO SUPPRESS
NO	NO	4	D	M	

Line	Form Type	Filename	Type	Space Before	Space After	Skip Before	Skip After	Output Indicators	Field Name	Edit Codes	Blank After	End Position	Packed	Constant or Edit Word
01	O	BILLING	H		3	04		1P						
02	O											43		'MØNTHLY STATEMENT'
03	O		H		1			1P						
04	O											27		'ACCØUNT ØLD'
05	O											54		'PAYMENT NEW'
06	O		H		2			1P						
07	O											29		'NUMBER BALANCE'
08	O											56		'BALANCE'
09	O		D		2			22						
10	O								ACCNØ			15		
11	O								ØLDBAL	1		29		
12	O								PAYMNT	1		44		
13	O								NEWBAL	A		59		

G14082

Figure 10-12. Coding of Output Constants

length. The first apostrophe must be coded in column 45, and the ending apostrophe must be coded in the column following the last character of the edit word.

Edit Word Rules

The following rules must be observed when using edit words:

a. Column 38 must be blank when edit words are used.

b. A numeric data field name entry must be made in columns 32-37.

c. An end position must be specified in columns 40-43.

d. Edit words must be enclosed in apostrophes and must not exceed 24 characters in length.

e. Any valid, printable RPG character can be used in an edit word. The following characters have special uses when placed in specific positions within the edit word:

b̸	blank	,	comma
0	zero	CR	credit symbol
$	dollar sign	*	asterisk
.	decimal point	&	ampersand
-	minus sign		

f. When an edit word is used, automatic suppression of all high-order zeroes within the data field occurs unless a zero or asterisk is specified in the edit word. When zero is specified, the last leading zero in the field is replaced by a blank (b̸). When an asterisk is specified, the last leading zero in the field is replaced by an asterisk.

g. The number of replaceable characters in an edit word should be equal to the length of the field that is to be edited, with the following exceptions:

1. When the floating dollar sign is used, an extra space may be left in the edit word to ensure that the dollar sign is printed when the numeric data field is full.

2. An extra space may be left in the edit word when zero suppression is not wanted. All other editing will be performed.

The replaceable characters and their meanings are:

Replaceable Character	Meaning
0	Zero suppression.
*	Asterisk fill.
	Blank.
$	Floating dollar sign (if it appears immediately to the left of zero suppress character).

h. Fixed dollar signs, decimal points, commas, ampersands, negative signs (CR and -), and constant information are not replaceable characters.

i. Any zeroes or asterisks following the left-most zero or asterisk are treated as constants and are not replaceable characters.

j. Any constant to the left of the zero suppression stop character, except the floating dollar sign ($), will be suppressed unless a significant digit precedes the constant.

k. When a blank edit word is used, all leading zeroes are suppressed and any sign in the unedited field is removed. Negative values are not identified.

l. When no edit word is used, the data in the output record has the same format as the unedited data. When the numeric data field contains a negative number, the low-order sign position will be printed as an alphabetic character (J-R) depending upon the value of the numeric data field.

Zero Suppression

The elimination of high-order zeroes when printing a numeric data field is known as zero suppression. Either full, none, or limited zero suppression can be accomplished by the use of an edit word.

To create the proper edit word a programmer must know:

a. The maximum size of the field to be edited.

b. The number of positions to be zero suppressed.

Full Zero Suppression Coding. The following are recommended procedures for full zero suppression coding:

a. The edit word length is equal to the maximum size of the field to be edited, or

b. Enter a zero in the rightmost position of the edit word, and

c. Leave the remaining position in the edit word blank.

When full zero suppression is the only editing function desired, it is more efficient to use the Z edit code by making the appropriate entry in column 38 of the Output-Format Specifications.

No Zero Suppression Coding. The following is a recommended method of coding for no zero suppression, but does not apply to RPG I:

a. Add one to the maximum size of the field to be edited,

b. Enter a zero in the leftmost position of the edit word, and

c. Leave the remaining positions in the edit word blank.

Figure 10-13 illustrates some zero suppression examples:

SOURCE DATA	CONSTANT OR EDIT WORD	OUTPUT RECORD
	45 70	
0000000005	`'  ,   ,  0.  '`	bbbbbbbbbb.05
00000000	`'  ,   , 0 -'`	bbbbbbbbb0b
-00000000123	`'   ,   ,   .  -'`	bbbbbbbbbb1.23-
-0000123456	`'         0'`	bbbb123456
+0000123456	`'0         '`	b000123456
0000000000	`'$       0  &CR GRØSS'`	$bbbbbbbb00bbbbGROSS
0000000000	`' ,   ,   ,   -ØLD BAL'`	bbbbbbbbbbbbbbOLD BAL
000000	`' ,  0.  '`	bbbbb.00
000000	`' ,   .0 '`	bbbbbbb0
0000000000	`'  ,   , 0 *  '`	bbbbbbbbb0*00
001234	`'0,   ,0  '`	b,012,034
-000000015	`' ,   ,   .  -'`	bbbbbbbbbb15-
000000005	`' ,   , 0 .  -'`	bbbbbbbb0.05b

G14084

Figure 10-13. Zero Suppression Coding Examples

Limited Zero Suppression Coding. The following conditions will accomplish limited zero suppression when:

a. The edit word length is equal to the maximum size of the field to be re-edited.

b. Enter a zero in the last position that is to be zero suppressed by the edit word. Allow all other positions in the edit word to remain blank.

Editing with a Comma, Decimal Point, and Fixed Dollar Sign.

When editing with a comma, decimal point, or fixed dollar sign, note the following:

a. The edit word length is equal to the maximum size of the field to be edited plus the number of editing characters that are to be inserted.

b. When a zero suppress "zero" is not included in the edit word, full suppression is done automatically.

c. All commas and decimal points to the left of a zero suppress "zero" will be replaced by blanks when they are not preceded by a significant non-zero digit.

d. The fixed dollar sign will never be suppressed.

Editing with an Asterisk

This special character is used when it is necessary to protect printed data from being tampered with by unauthorized persons. This is accomplished by printing asterisks (8) in the print positions where zeroes and commas have been suppressed. This particular technique is frequently referred to as check protection.

Asterisk fill/check protect follows the same rules as zero suppression. Zero suppression replaces all leading zeroes and commas with blanks, while asterisk fill/check protect replaces these characters with asterisks.

Figure 10-14 shows examples of check-protect or asterisk-fill editing.

SOURCE DATA	CONSTANT OR EDIT WORD																										OUTPUT RECORD
	45																									70	
0000123456	'			,				,		*		.			&	-	'										*****1,234.56bb
-0000123456	'	*										'															*000123456
+1234567890	'	*										'															1234567890
-0000123456	'										*	'															****123456
0000001234	'			,			*	,				*			'												******,012*34
012345	'	&		,	*		0	,				'															***120,345

G14086

Figure 10-14. Asterisk Fill Coding Examples

Editing with a Floating Dollar Sign

A floating dollar sign is frequently used when writing checks. This method of check protection places a dollar sign immediately to the left of the first significant non-zero digit in the data field or the decimal point. Unnecessary commas and zeroes to the left of the dollar sign are replaced by spaces in the output area.

The following rules apply to editing with a floating dollar sign:

a. An extra space must be left in the edit word for the floating dollar sign because a full print position is required for the dollar sign when the field is full.

b. The floating dollar sign is placed immediately to the left of the "zero" used to signify zero suppression.

c. The floating dollar sign should never be placed immediately to the left of a decimal point.

Figure 10-15 illustrates examples of dollar sign coding with before and after images.

SOURCE DATA	CONSTANT OR EDIT WORD (45–70)	OUTPUT RECORD
+0000000005	`\   ,   ,$0 .  -'`	bbbbbbbbb$0.05b
-0012345678	`\   ,   , $0.  CR**'`	bbb$123,456.78CR**
0000000000	`\$  ,   ,  0.  '`	$bbbbbbbbbb.00
+0000123456	`\$          &- NET'`	$bbbb123456bbbNET
-0000123456	`\$          &- NET'`	$bbbb123456b-bNET
0000123456	`\$0         - NET'`	$000123456bbNET
-0000123456	`\       $0  &CR'`	bbbb$123456bCR
-1234567890	`\       $0  &CR'`	$1234567890bCR
0000000005	`\   ,   ,$0 .  &NET'`	bbbbbbbbb$0.05bNET
0000000005	`\   ,   , $0.  '`	bbbbbbbbbb$.05
-1234567890	`\   ,   , $0.  -'`	$12,345,678.90-
-0001234567	`\   ,   , $0.  CR'`	bbbb$12,345.67CR
0000001234	`\   , $0,   .  -SALES'`	bbbbbb$,012.34bSALES
00123456	`\ ,  $,0   . '`	bbb1$,234.56

G14085

Figure 10-15. Dollar Sign Coding Examples

Editing Negative Numbers

Negative numbers can be represented in the following two ways:

a. The dash or minus sign (-).

b. The credit symbol (CR).

These symbols that designate negative values are placed on the right side of the edit word. The dash requires one additional position in the output area, while the credit symbol requires two additional positions.

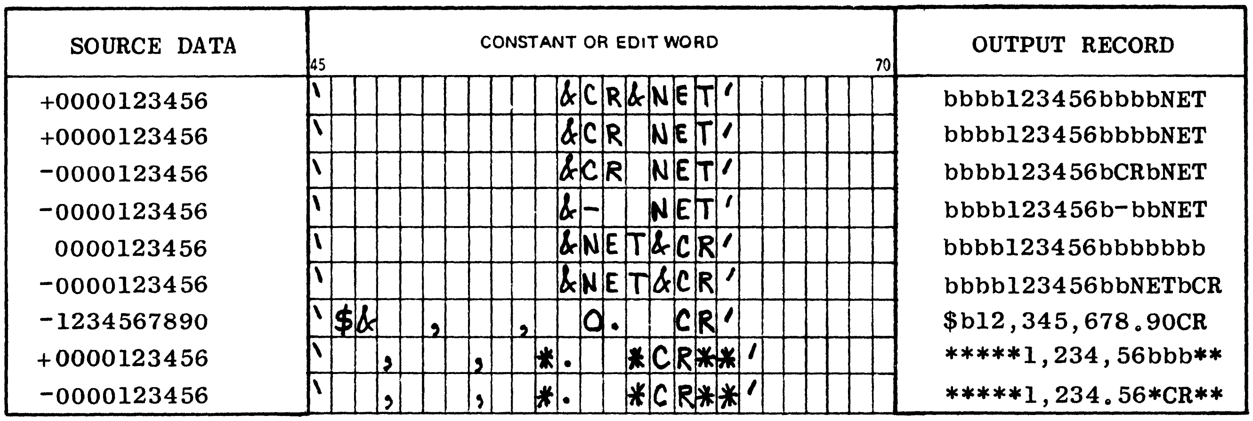
When the field being edited is positive in value, the negative symbols are not printed. If a number is negative and the edit word does not contain negative editing symbols, the number is edited as a positive number.

Figure 10-16 illustrates examples of negative sign coding.

Editing with a Slash

The slash (/) is primarily used to edit date fields, and can be used to provide separation of the day, month, and year in order to make the printed date more readable. As with all edit words, automatic zero suppression will be performed.

When date field editing is desired, it is more efficient to use the "Y" edit code entry in column 38 if the field is from 3 to 6 digits in length.



SOURCE DATA	CONSTANT OR EDIT WORD (45–70)	OUTPUT RECORD
+0000123456	`\          &CR&NET'`	bbbb123456bbbbNET
+0000123456	`\          &CR NET'`	bbbb123456bbbbNET
-0000123456	`\          &CR NET'`	bbbb123456bCRbNET
-0000123456	`\          &-  NET'`	bbbb123456b-bbNET
0000123456	`\          &NET&CR'`	bbbb123456bbbbbbb
-0000123456	`\          &NET&CR'`	bbbb123456bbNETbCR
-1234567890	`\$&  ,   ,  0.  CR'`	$b12,345,678.90CR
+0000123456	`\  ,   ,  *.  *CR**'`	*****1,234,56bbb**
-0000123456	`\  ,   ,  *.  *CR**'`	*****1,234.56*CR**

G14087

Figure 10-16. Negative Signs Coding Examples

Editing to Remove the Sign

When no edit word is specified and a numeric field is printed and/or punched, the output will have a zone overscore over the low-order digit (right-signed) or the most significant digit (left-signed) of the field. This zone represents the sign of the field. When the field is positive, the letters A through I will be printed and/or punched. If the field is negative, the letters J through R will be printed and/or punched. The zone can be eliminated by:

a. Positive signs:

   1. An edit word that contains any or none of the edit characters.

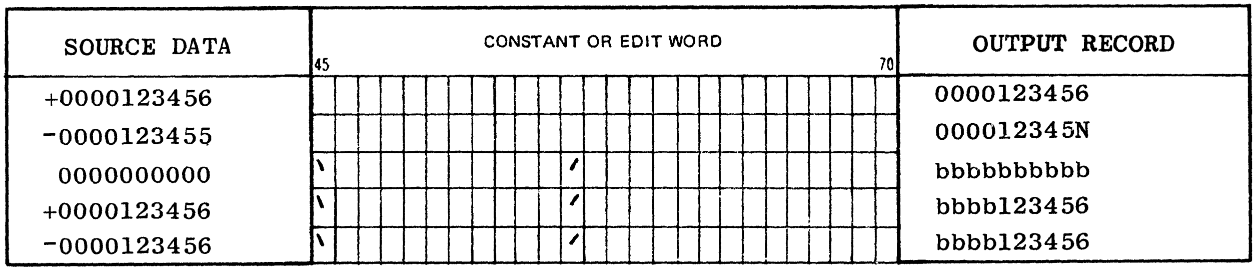
   2. The "X" edit code entry in column 38. This is the most efficient way of removing the positive sign when no other editing is desired.

b. Negative sign:

   1. An edit word that contains any or none of the edit characters.

For negative fields, an edit word must be used to remove the sign.

Figure 10-17 illustrates examples of sign removal.



SOURCE DATA	CONSTANT OR EDIT WORD (45–70)	OUTPUT RECORD
+0000123456		0000123456
-0000123455		000012345N
0000000000	`\          '`	bbbbbbbbbb
+0000123456	`\          '`	bbbb123456
-0000123456	`\          '`	bbbb123456

G14083

Figure 10-17. Sign Removal Coding Examples

Ampersand

The ampersand is used to provide a blank space in the edit word when RPG II dialect is specified. Refer to figure 10-18 for ampersand coding examples using RPG II dialect.

The ampersand appears as an ampersand when RPG I dialect is specified. Refer to figure 10-19 for ampersand coding examples using RPG I dialect.

SOURCE DATA	CONSTANT OR EDIT WORD 45 70	OUTPUT RECORD
-1234567890	`'$&  ,   , 0 .  &- GRØSS'`	$b12,345,678.90b-bGROSS
-1234567890	`'$&  ,   , 0 .  &-NET&PAY'`	$b12,345,678.90b-NET PAY
-0000123456	`'$&  ,   ,  *.  CR'`	$******1,234.56CR
+000002	`'  0 LBS.&  ØZ.TARE&-'`	bbb0LBS.b02bbbbbbbbb
031274	`'  &  &  &LATER'`	b3b12b74bLATER

G14088

Figure 10-18. Ampersand Coding Examples (RPG II Dialect)

SOURCE DATA	CONSTANT OR EDIT WORD 45 70	OUTPUT RECORD
0000123456	`'          &&PRØFIT'`	bbbb123456&&PROFIT
0000123456	`'          &CR&NET'`	bbbb123456bbb&NET
1234567890	`'$&  ,   , 0 .  &-NET&PAY'`	$b12,345,678.90b-NET&PAY

G14089

Figure 10-19. Ampersand Coding Examples (RPG I Dialect)

Editing and Constants

The edit characters have been described are the standard characters used for editing. However, it is possible to use any character in an edit word. The specified character will be printed on the output report in the position specified whenever a non-zero character has been printed to the left of the specified character.

Figure 10-20 illustrates the use of constants in the edit word.

SOURCE DATA	CONSTANT OR EDIT WORD 45 70	OUTPUT RECORD
0000123456	`'          &&PRØFIT'`	bbbb123456bbPROFIT
-0000123456	`'  ,   ,   .  &CR  NET'`	bbbbb1,234.56bCRbbNET
0000123456	`'  ,   ,   DØLLARS  CENTS'`	bbbbb1,234DOLLARS56CENTS
000000	`' ,   DØLLARS  CENTS'`	bbbbbbbbbbbbbbCENTS
000000	`' , 0 DØLLARS  CENTS&CR'`	bbbb0DOLLARS00bbbbbbbb
-000002	`'   0LBS.  ØZ.TARE&-'`	bbbbLBS.020Z.TAREb-
063369690	`'0  -  -    '`	b63-36-9690
0042	`'0 HRS.  MINS.&Ø''CLØCK'`	b0HRS.42MINS.b0'CLOCK
093074	`'  -  -  &LATER'`	b9-30-74bLATER
093074	`'  /  /  '`	b9/30/74

G14090

Figure 10-20. Examples of Constants in the Edit Word

75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a complete description.

EXAMPLES OF EDIT CODES AND EDIT WORDS - RPG II Dialect

Examples of the use of edit words and edit codes follows. Examples 1 and 2 are exceptions, as they are unedited.

The column on the left contains the number of the example.

The center column contains:

a. The source data.

b. The edit word or edit code used.

c. The edited data.

The Column on the right contains a brief description of the information in the center column.

Example Number	Edit Word or Edit Code	Description
1	0000123456 no edit word +000123456	With no edit word, the sign position is not converted to a numeric character. The first character is plus zero (+).
2	-0000123456 no edit word !000123456	With no edit word, the sign position is not converted to a numeric character. The first character is minus zero (!).
3	0000000000 ' '	Blank edit word. Suppresses leading zeroes.
4	+0000123456 ' ' 123456	Blank edit word. Suppresses leading zeroes.
5	-0000123456 ' ' 123456	Blank edit word. Suppresses leading zeroes.
6	0000000005 ' , , 0. ' .05	This is the usual approach to editing an amount field. The decimal appears between the dollars and cents. The commas mark each three positions left of the decimal and the zero suppression character (0) immediately left of the decimal causes zero suppression to the left.

7	00000000 ' , , 0 -' 0	The zero suppression begins where the zero suppression character is specified and continues left.
8	-00000000123 ' , , . -' 1.23-	The status portion is printed when the field is negative. Since no zero suppress character is specified, all insignificant characters are suppressed.
9	-0000123456 ' 0' 123456	The zero suppress specified for this position performs no function. The sign is lost as there is no status specified in the edit word.
10	+0000123456 '0 ' 000123456	The zero suppress specified for this position performs no function. The sign is lost as there is no status specified in the edit word.
11	0000000000 '$ 0 &CR GROSS' $ 0 CR GROSS	From the right of the $ to the left of the & is 10 characters. Eight zeros are suppressed. All zero fields are positive.
12	0000000000 ' , , , -OLD BAL' OLD BAL	All 10 zeros are suppressed. Only the extension portion of the edit word is output.
13	000000 ' , 0. ' .00	This is the usual approach to editing an amount field. The decimal appears between the dollars and cents. The commas mark each three positions left of the decimal and the zero suppression character (0) immediately left of the decimal causes zero suppression to the left.
14	000000 ' , .0 ' 0	All insignificant zeros and punctuation are suppressed.
15	0000000000 ' , , 0 * ' 0*00	The first zero suppress character is the zero. The * is treated as a constant.
16	001234 '0, ,0 ' ,012,034	Only the first zero of the source data is suppressed. The second zero of the edit word is treated as a constant.

17

```
   -000000015
' ,   ,   .  _'
           15-
```

Zero suppression of all insignificant zeros to the decimal point. Punctuation is suppressed, also.

18

```
    000000005
' ,   , 0 .  -'
         0.05
```

Zero suppression of all insignificant zeros to the decimal point. Punctuation is suppressed, also.

19

```
    0000000005
'  ,   ,$0 .  -'
         $0.05
```

Zero suppress with floating dollar sign. The dollar sign will move as far right as the zero suppress character.

20

```
    -0012345678
'   ,  ,  $0.  CR**'
$123,456.78CR**
```

Floating dollar sign with negative amount. Note that the edit word has one more replaceable position than the length of the source data.

21

```
    0000000000
'$ ,   ,  0.   '
 $         .00
```

This is not a floating dollar sign as it does not immediately precede the zero suppress character. All superfluous punctuation is suppressed.

22

```
 +0000123456
'$          &- NET'
 $    123456   NET
```

Zero suppression with constant $ and an extension portion.

23

```
 -0000123456
'$          &- NET'
 $    123456 - NET
```

Same as example 22 with status portion reporting negative.

24

```
 +0000123456
'$0         - NET'
  $000123456  NET
```

Only high-order position zero suppression. Floating $.

25

```
 -0000123456
'       $0   &CR'
     $123456  CR
```

Floating dollar sign with the status portion reporting negative.

26

```
 -1234567890
'       $0  &CR'
 $1234567890 CR
```

Floating dollar sign with the status portion reporting negative.

27

```
     0000000005
'   ,   ,$0 .  &NET'
          $0.05 NET
```

Floating dollar sign.

```
28           0000000005          Floating dollar sign.
        '   ,   , $0.  '
                   $.05

29          -1234567890          Note again that in order for the
                                 $ to appear, the edit word had to
        '  ,    , $0. -'         have one more replaceable charac-
         $12,345,678.90-         ter than the length of the
                                 variable data.

30          -0001234567          Note again that in order for the
        '   ,   , $0.  CR'       $ to appear, the edit word had
                                 to have one more replaceable
             $12,345.67CR        character than the length of the
                                 variable data.

31           0000001234          Same as example 29 with an exten-
        '   , $0,   .  -SALES'   sion.  Note that the placement of
                                 the zero suppress may cause non-
               $,012.34 SALES    esthetic results.

32            00123456           Since the $ and the 0 are
        '  ,  $,0  .  '          separated by the comma, the $
                                 is a constant in the body.
             1$,234.56

33          0000123456           Check protect asterisk.  Sup-
        '  ,   , * .  &-'        presses insignificant zeros and
                                 punctuation, filling each posi-
         *****1,234.56           tion with *.

34      -0000123456              The edit word is 10 characters
        '*         '             long but the * occupies one
                                 position.
         *000123456

35      +1234567890              * is a replaceable character.
        '*         '             When there are no insignificant
                                 characters, it is replaced.
         1234567890

36      -0000123456              Check protect asterisk.  Sup-
        '         *'             presses insignificant zeros and
                                 punctuation, filling each posi-
         ****123456              tion with *.

37          0000001234           Same as example 33.  The second *
        '  , *,    *  '          is a constant in the body.  Non-
                                 esthetic.
         ******,012*34

38           012345              The & is a replaceable character.
        '& ,* 0,   '             (See example 45.)  The * fills
                                 all insignificant positions.  The
         ***120,345              0 is a constant in the body.
```

```
39      +0000123456               Ten position field into ten po-
        '          &CR&NET'       sition body.  The ampersands are
             123456    NET        for spacing only and are not
                                  printed.

40      +0000123456               Ten position field into ten po-
        '          &CR NET'       sition body.  The ampersands
             123456    NET        are for spacing only and are
                                  not printed.

41      -0000123456               Same as example 40 with status
        '          &CR NET'       reporting negative.
             123456 CR NET

42      -0000123456               Same as example 40 with status
        '          &- NET'        reporting negative.
             123456 - NET

43       0000123456               The extension, NET, does not out-
        '          &NET&CR'       put because the status reports
             123456               positive.  NET is interpreted by
                                  the generator as part of the
                                  status, not the extension.  Re-
                                  verse the order (CR&NET) and the
                                  word will function properly.

44      -0000123456               Proof of 43.
        '          &NET&CR'
             123456 NET CR

45           -1234567890          Constant $. & for spacing only.
        '$&  ,   ,   .  CR'       Status reports negative.
         $ 12,345,678.90CR

46         +0000123456            First * is check protect.  Sec-
        '  ,   ,  *.  *CR**'      ond * is part of status.  Third
         *****1,234.56   **       and fourth * are extension.

47         -0000123456            Same as example 46 with status
        '  ,   ,  *.  *CR**'      reporting negative.
         *****1,234.56*CR**

48           -1234567890          Constant $. & for spacing.  Zero
        '$&  ,   , 0 .  &-GROSS'  suppress.  & for spacing.  Status
         $ 12,345,678.90 -GROSS   reports negative.  Extension con-
                                  tains GROSS.

49           -1234567890          Same as example 48.  Extension
        '$&  ,   , 0 .  &-NET PAY' contains NET PAY.
         $ 12,345,678.90 -NET PAY
```

50

```
      -0000123456
  '$&   ,    ,  *.  CR'
   $******1,234.56CR
```

Constant $. Replaceable &. Check protect & substituted for replaceable characters, insignificant zeros and punctuation.

51

```
      +000002
  '  0 LBS.&  OZ.TARE&-'
      0LBS. 02
```

Supress first three zeros of source data. Output fourth zero. Output the constant, LBS. Space across the &. Insert the 02. Report the status as positive. For more meaningful output, see example 51a.

51a

```
      +000002
  '  0 LBS.&  -  OZ.  TARE'
     0LBS. 02    OZ.  TARE
```

The significant difference between this example and the previous one is that what is meant to be the extension is located after the status portion.

52

```
    031274
  '  &  &  &LATER'
   3 12 74 LATER
```

Special zero suppression is not specified so zeros are automatically suppressed. The ampersands force the source data to be spaced. No status.

53

```
  0000123456
  '         &&PROFIT'
       123456  PROFIT
```

Automatic zero suppression. No status. Extension.

54

```
    -0000123456
  '  ,   ,  .  &CR NET'
       1,234.56 CR NET
```

Automatic zero suppression. Status reports negative. Extension.

55

```
             0000123456
  '   ,   ,   DOLLARS  CENTS'
          1,234DOLLARS56CENTS
```

Automatic zero suppression. Insert significant punctuation. Space across constants until the end of the source data is reached.

56

```
           000000
  ' ,    DOLLARS  CENTS'
                  CENTS
```

Automatic zero suppression also suppresses insignificant constants in the body of the edit word. CENTS is the extension.

57

```
           000000
  ' , 0 DOLLARS  CENTS&CR'
     0DOLLARS00
```

Suppress three zeros of the source data. Since the status reports positive, CENTS is not printed. Extension must be after status, if there is a status portion.

58

```
      -000002
  '    0LBS.  OZ.TARE&-'
       LBS.02OZ.TARE -
```

Same as example 51. See example 51a for more readable output.

59	063369690 '0 - - ' 63-36-9690	One method of editing a social security number. If the high order zero is not to be suppressed, see example 59a.
59a	063369690 '0 - - ' 063-36-9690	When the high-order zero is not to be suppressed, specify the edit word with one extra position in the first portion, and zero suppress that position.
60	0042 '0 HRS. MINS.&0''CLOCK' 0HRS.42MINS. 0'CLOCK	Zero suppress high-order position only. Since the apostrophe is used to delimit edit words and constants, if printing the apostrophe is desired, it must be specified two times for each time it is to appear in the output. Example 51a could be profitably used here for greater readability.
61	093074 ' - - &LATER' 9-30-74 LATER	Same type editing as used in example 59 with an extension portion added. Automatic zero suppression.
62	093074 ' / / ' 9/30/74	Another method of editing a date field.
63	093074 Y Edit Code 9/30/74	Y is used for editing date fields. The largest field that may be edited with the Y is six digits.
64	09307 Y Edit Code 9/30/7	Y edit of a five-digit date field.
65	0930 Y Edit Code 9/30	Y edit of a four-digit date field.
66	093 Y Edit Code 9/3	Y edit of a three-digit date field.
67	+0000000000 1 Edit Code 0	1 edit of ten-position field. Commas and zero balances to print. No sign.

68	+0000000000 2 Edit Code	2 edit of ten-position field. Commas to print. No zero balance or sign.
69	+0000000000 A Edit Code 0	A edit of ten-position field. Commas and zero balance to print. CR for status report.
70	+0000000000 B Edit Code	B edit of ten-position field. Commas to print. No zero balance. CR for status report.
71	+0000000000 X Edit Code 0000000000	X edit of ten-position field. Remove plus sign from field.
72	+0000000000 Z Edit Code	Z edit of ten-position field. Zero suppress.
73	-0000000000 1 Edit Code 0	1 edit of ten-position field. Commas and zero balance to print. No sign.
74	-0000000000 2 Edit Code	2 edit of ten-position field. Commas to print. No zero balance or sign.
75	-0000000000 A Edit Code 0	A edit of ten-position field. Commas and zero balance to print. CR for status report.
76	-0000000000 B Edit Code	B edit of ten-position field. Commas to print. No zero balance. CR for status report.
77	-0000000000 J Edit Code 0	J edit of ten-position field. Commas and zero balance to print. - for status report.
78	-0000000000 K Edit Code	K edit of ten-position field. Commas to print. No zero balance. - for status report.
79	-0000000000 X Edit Code 0000000000	X edit of ten-position field. Remove plus sign from field. Since all zero value fields are treated as positive, the X edit code does not work in this case.
80	-0000000000 Z Edit Code	Z edit of ten-position field. Zero suppress.

81	+0000000000 1 Edit Code -00	1 edit of ten-position field, with two implied decimals. Commas and zero balance to print. No sign.
82	+0000123456 1 Edit Code 1,234.56	1 edit of ten-position field with two implied decimals. Commas and zero balance to print. No sign.
83	+0000123456 2 Edit Code 1,234.56	2 edit of ten-position field with two implied decimals. Commas to print. No zero balance. No sign.
84	+0000123456 3 Edit Code 1234.56	3 edit of ten-position field with two implied decimals. Zero balance to print. No commas. No sign.
85	+0000123456 4 Edit Code 1234.56	4 edit of ten-position field with two implied decimals. No commas. No zero balance. No sign.
86	+0000123456 A Edit Code 1,234.56	A edit of ten-position field with two implied decimals. Commas and zero balance to print. CR for status report.
87	+0000123456 B Edit Code 1,234.56	B edit of ten-position field with two implied decimals. Commas to print. No zero balance. CR for status report.
88	+0000123456 C Edit Code 1234.56	C edit of ten-position field with two implied decimals. No commas. Zero balance to print. CR for status report.
89	+0000123456 D Edit Code 1234.56	D edit of ten-position field with two implied decimals. No commas. No zero balance. CR for status report.
90	+0000123456 J Edit Code 1,234.56	J edit of ten-position field with two implied decimals. Commas and zero balances to print. - for status report.
91	+0000123456 K Edit Code 1,234.56	K edit of ten-position field with two implied decimals. Commas to print. No zero balance. - for status report.
92	+0000123456 L Edit Code 1234.56	L edit of ten-position field with two implied decimals. No commas. Zero balance to print. - for status report.

93	+0000123456 M Edit Code 1234.56	M edit of ten-position field with two implied decimals. No commas. No zero balance. - for status report.
94	+0000123456 X Edit Code 0000123456	X edit of ten-position field with two implied decimals. Remove plus sign.
95	+0000123456 Z Edit Code 123456	Z edit of ten-position field with two implied decimals. Zero suppress.
96	-0000123456 1 Edit Code 1,234.56	1 edit of ten-position field with two implied decimals. Commas and zero balances to print. No sign.
97	-0000123456 2 Edit Code 1,234.56	2 edit of ten-position field with two implied decimals. Commas to print. No zero balance. No sign.
98	-0000123456 3 Edit Code 1234.56	3 edit of ten-position field with two implied decimals. No commas. Zero balance to print. No sign.
99	-0000123456 4 Edit Code 1234.56	4 edit of ten-position field with two implied decimals. No commas. No zero balance. No sign.
100	-0000123456 A Edit Code 1,234.56CR	A edit of ten-position field with two implied decimals. Commas and zero balance to print. CR for status report.
101	-0000123456 B Edit Code 1,234.56CR	B edit of ten-position field with two implied decimals. Commas to print. No zero balance. CR for status report.
102	-0000123456 C Edit Code 1234.56CR	C edit of ten-position field with two implied decimals. No commas. Zero balance to print. CR for status report.
103	-0000123456 D Edit Code 1234.56CR	D edit of ten-position field with two implied decimals. No commas. No zero balance. CR for status report.
104	-0000123456 J Edit Code 1,234.56-	J edit of ten-position field with two implied decimals. Commas and zero balance to print. - for status report.

105	-0000123456 K Edit Code 1,234.56-	K edit of ten-position field with two implied decimals. Commas to print. No zero balance. - for status report.
106	-0000123456 L Edit Code 1234.56-	L edit of ten-position field with two implied decimals. No commas. Zero balance to print. - for status report.
107	-0000123456 M Edit Code 1234.56-	M edit of ten-position field with two implied decimals. No commas. No zero balance. - for status report.
108	-0000123456 X Edit Code !000123456	X edit of ten-position field with two implied decimals. Remove plus sign. Since the sign is negative, the sign position is output as the graphic representation of the minus sign and digit (in this case, ! or minus zero).
109	-0000123456 Z Edit Code 123456	Z edit of ten-position field with two implied decimals. Zero suppress.

Printer File Handling

An important objective of most RPG programs is the generation of a printed report. To help facilitate the production of an easily readable report with minimum programming, special features and functions have been incorporated into Burroughs B 1700 RPG.

Page Formatting

The RPG I and RPG II dialects can both use the Line Counter Specifications and/or carriage control tapes to control paper movement through the printer. However, there are several differences in usage and specification between the two dialects.

RPG I Dialect. The RPG I dialect uses the skip to channel option. Line Counter Specifications are required and must contain at least line-channel equations for channel 1 and channel 12.

The overflow line is the line specified in the line channel equations for channel 12. Up to 12 line-channel equations can be specified on the Line Counter Specifications form.

When a different forms length is desired, the length is entered in columns 15-19 and FL is entered in columns 18-19 of the Line Counter Specifications. Line-Channel equations are required for channels 1 and 12. Up to nine additional line-channel equations can be specified.

Whenever a channel number is specified in columns 19-22 of the Output-Format Specifications, there must be a corresponding line-channel equation entry in the Line Counter Specifications. At compile time, when a skip to channel is found that does not have a corresponding line channel equation, a syntax message is emitted. Neither syntax error nor warning is issued if a line-channel equation is defined but is not referenced in the Output-Format Specifications.

RPG II Dialect. In RPG II dialect, Line Counter Specifications are optional. The default value for form length is 66 lines and the default value for the overflow line is line 60. When these default values are acceptable, as many as 12 line-channel equations can be specified on the Line Counter Specification entry. If the defaults are not desirable, then both the form length and overflow line must be entered in the first two sections of the Line Counter Specification and only 10 line-channel equations can be specified on the remainder of the specification line.

When the line-channel equations are supplied, skip to line will be performed as skip to channel. If no line-channel equation is supplied, skip to line is performed by spacing, in the usual manner.

Entries in columns 19-22 of the Output-Format Specifications are treated as line numbers.

End of Page

The physical end of a page is reached when the paper becomes positioned in the overflow area. For RPG I this is the area from channel 12 to 1 line prior to channel 1. In RPG II it is the first line after the overflow line to 1 line prior to line 1.

When End-Of-Page (overflow area) is sensed during a printer operation, one of the following must occur:

a. Automatic skipping is specified when an overflow indicator is not entered in columns 33-34 of the File Description Specifications. Since the indicator is not defined, it may not be used to condition calculations or output.

   Automatic skipping moves the paper from the overflow area to the top of the next page before resuming print operations.

   No overflow bookkeeping is available to the user.

   For RPG I the paper is moved from the overflow area to Channel 1.

   For RPG II the paper is moved from the overflow area to line 6.

b. Continuous printing is specified when an overflow indicator is entered in columns 33-34 of the File Description Specifications but is not used to condition printer record output. The overflow indicator is defined and may be used in Calculation Specifications. If used on the Output-

Format Specifications, the overflow indicator must condition only field descriptions of printer files.

c. Overflow operations are specified when an overflow indicator is entered in columns 33-34 of the File Description Specifications and that indicator is used to condition record output in the print file. Using the overflow indicator to condition printer file record output defines those lines of print that are to be written during overflow output. It is the user's responsibility to initiate skipping based on overflow, as this is not done automatically.

When overflow operations are specified, normal overflow output may be printed when the overflow output routine in the RPG cycle is reached. If fetching was specified (F in column 16 of the record description line on the Output-Format Specifications), overflow output may be printed immediately: Before the record described on the line containing the F in column 16 is printed. This is fetched overflow output.

Overflow Indicators

Overflow indicators must be used only with printers, and are not assigned by default. The assignment of overflow indicators determines the type of paper motion and overflow handling that will occur.

a. Overflow indicators not assigned and not used result in automatic skipping. In RPG II this is a skip to line 6. In RPG I this is a skip to channel 1.

b. Overflow indicators assigned but not used result in printing over the perforation.

c. Overflow indicators assigned and used result in normal overflow handling.

<u>RPG I Dialect.</u> An overflow indicator will turn on:

a. When a line is actually printed in the overflow area, or

b. When, as the result of using the overflow indicator as a calculation resulting indicator, the overflow indicator is set on.

An overflow indicator will turn off when:

a. The overflow indicator is used as a calculation resulting indicator, and the operation turns it off.

b. The conditions of the overflow turn off routine are satisfied. Refer to figure 10-21.

<u>RPG II Dialect.</u> An overflow indicator will turn on when:

a. The destination of a skip before or a skip after is in the overflow area.

b. The destination of space before or a space after is in the overflow area.

c. A line is actually printed in the overflow area.

d. The use of the overflow indicator as a calculation resulting indicator turns it on.

An overflow indicator will turn off when:

a. The destination of a skip before or a skip after is a new page, and the record is not conditioned by the overflow indicator.

b. The overflow indicator is used as a calculation resulting indicator, and the operation turns it off.

c. The conditions of the overflow turn-off routine are satisfied. Refer to figure 10-21.

The overflow indicator is not set off if the new page is reached by spacing.

Overflow

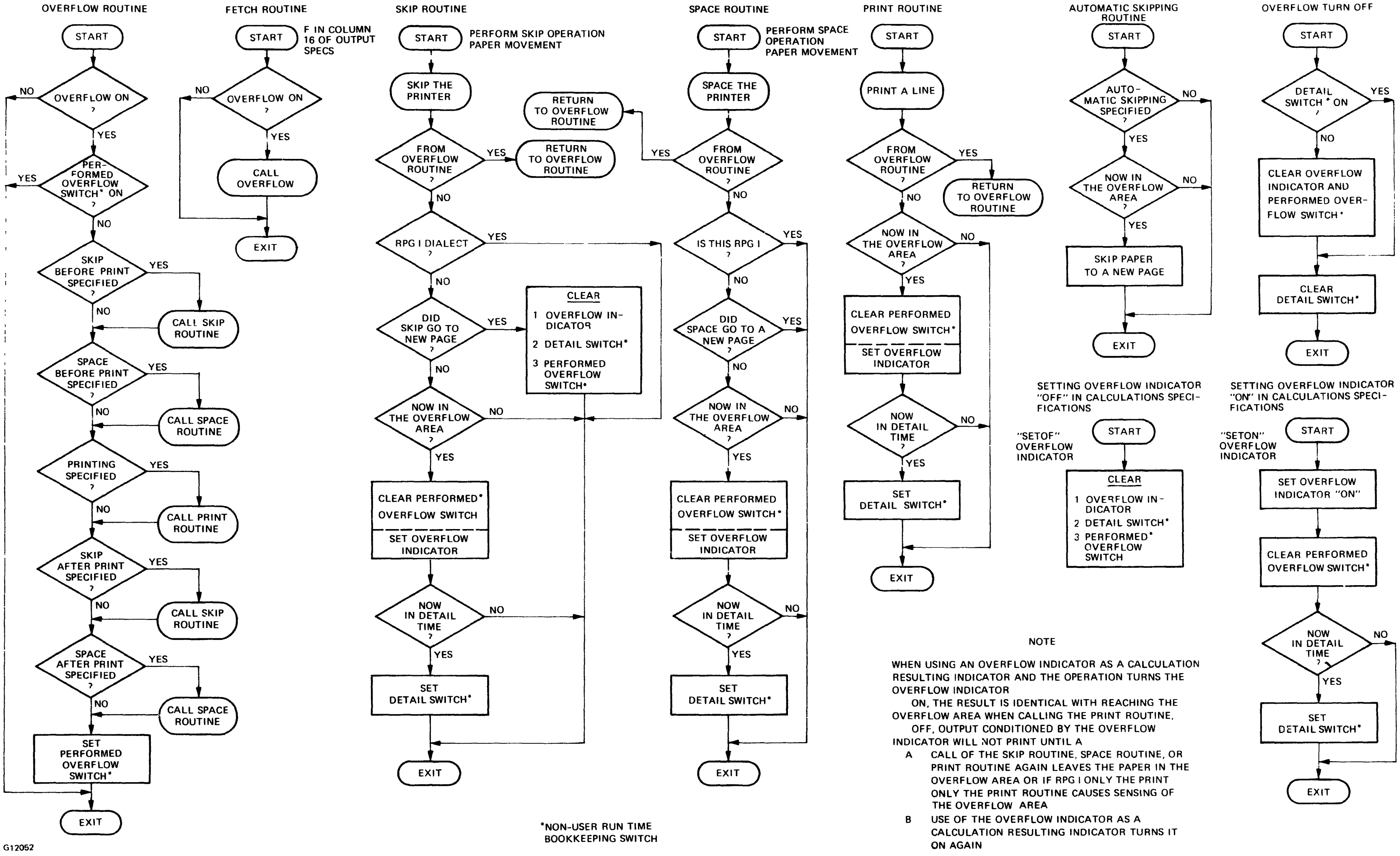
The principles used for handling overflow are:

a. Overflow will be performed only once for any single physical overflow, whether fetched or performed during the normal part of the cycle when overflow processing is performed (For example, after total output).

b. The RPG program distinguishes between overflow indicators which are set during the detail part of the cycle and those which are set during the total part of the cycle. If set during detail, overflow indicators do not get turned off after detail output, but are maintained for one full cycle until the next time around. If an overflow indicator went on during detail and overflow was fetched, the reason overflow doesn't get performed twice (once for the fetch and once at normal overflow processing), is because overflow will only be performed once for each physical overflow.

   If the program overflow specifications leave the printer in the overflow area, the cycle will not erroneously do overflow processing again, but remembers that it has already been performed for the condition that caused overflow. To do overflow processing again, the program must cause another physical overflow by printing again in the overflow area.

c. Overflow indicators can also be controlled in the Calculation Section when used as resulting indicators. When an overflow indicator is set on in the Calculation Section, the effect is exactly the same as if a physical overflow condition had occurred on the printer, with the cycle also distinguishing whether it has been turned on in detail or


OVERFLOW ROUTINE
START
OVERFLOW ON ?
NO
YES
PER-FORMED OVERFLOW SWITCH* ON ?
YES
NO
SKIP BEFORE PRINT SPECIFIED ?
YES
CALL SKIP ROUTINE
NO
SPACE BEFORE PRINT SPECIFIED ?
YES
CALL SPACE ROUTINE
NO
PRINTING SPECIFIED ?
YES
CALL PRINT ROUTINE
NO
SKIP AFTER PRINT SPECIFIED ?
YES
CALL SKIP ROUTINE
NO
SPACE AFTER PRINT SPECIFIED ?
YES
CALL SPACE ROUTINE
NO
SET PERFORMED OVERFLOW SWITCH*
EXIT
G12052
FETCH ROUTINE
START
F IN COLUMN 16 OF OUTPUT SPECS
OVERFLOW ON ?
NO
YES
CALL OVERFLOW
EXIT
SKIP ROUTINE
START
PERFORM SKIP OPERATION PAPER MOVEMENT
SKIP THE PRINTER
FROM OVERFLOW ROUTINE ?
YES
RETURN TO OVERFLOW ROUTINE
NO
RPG I DIALECT ?
YES
NO
DID SKIP GO TO NEW PAGE ?
YES
NO
CLEAR
1 OVERFLOW IN-DICATOR
2 DETAIL SWITCH*
3 PERFORMED OVERFLOW SWITCH*
NOW IN THE OVERFLOW AREA ?
NO
YES
CLEAR PERFORMED* OVERFLOW SWITCH
SET OVERFLOW INDICATOR
NOW IN DETAIL TIME ?
NO
YES
SET DETAIL SWITCH*
EXIT
SPACE ROUTINE
START
PERFORM SPACE OPERATION PAPER MOVEMENT
SPACE THE PRINTER
RETURN TO OVERFLOW ROUTINE
YES
FROM OVERFLOW ROUTINE ?
NO
IS THIS RPG I ?
YES
NO
DID SPACE GO TO A NEW PAGE ?
YES
NO
NOW IN THE OVERFLOW AREA ?
NO
YES
CLEAR PERFORMED OVERFLOW SWITCH*
SET OVERFLOW INDICATOR
NOW IN DETAIL TIME ?
NO
YES
SET DETAIL SWITCH*
EXIT
PRINT ROUTINE
START
PRINT A LINE
FROM OVERFLOW ROUTINE ?
YES
RETURN TO OVERFLOW ROUTINE
NO
NOW IN THE OVERFLOW AREA ?
NO
YES
CLEAR PERFORMED OVERFLOW SWITCH*
SET OVERFLOW INDICATOR
NOW IN DETAIL TIME ?
NO
YES
SET DETAIL SWITCH*
EXIT
AUTOMATIC SKIPPING ROUTINE
START
AUTO-MATIC SKIPPING SPECIFIED ?
NO
YES
NOW IN THE OVERFLOW AREA ?
NO
YES
SKIP PAPER TO A NEW PAGE
EXIT
SETTING OVERFLOW INDICATOR "OFF" IN CALCULATIONS SPECI-FICATIONS
"SETOF" OVERFLOW INDICATOR
START
CLEAR
1 OVERFLOW IN-DICATOR
2 DETAIL SWITCH*
3 PERFORMED* OVERFLOW SWITCH
OVERFLOW TURN OFF
START
DETAIL SWITCH * ON ?
YES
NO
CLEAR OVERFLOW INDICATOR AND PERFORMED OVER-FLOW SWITCH*
CLEAR DETAIL SWITCH*
EXIT
SETTING OVERFLOW INDICATOR "ON" IN CALCULATIONS SPECI-FICATIONS
"SETON" OVERFLOW INDICATOR
START
SET OVERFLOW INDICATOR "ON"
CLEAR PERFORMED OVERFLOW SWITCH*
NOW IN DETAIL TIME ?
NO
YES
SET DETAIL SWITCH*
EXIT

NOTE

WHEN USING AN OVERFLOW INDICATOR AS A CALCULATION RESULTING INDICATOR AND THE OPERATION TURNS THE OVERFLOW INDICATOR

ON, THE RESULT IS IDENTICAL WITH REACHING THE OVERFLOW AREA WHEN CALLING THE PRINT ROUTINE.

OFF, OUTPUT CONDITIONED BY THE OVERFLOW INDICATOR WILL NOT PRINT UNTIL A

A CALL OF THE SKIP ROUTINE, SPACE ROUTINE, OR PRINT ROUTINE AGAIN LEAVES THE PAPER IN THE OVERFLOW AREA OR IF RPG I ONLY THE PRINT ONLY THE PRINT ROUTINE CAUSES SENSING OF THE OVERFLOW AREA

B USE OF THE OVERFLOW INDICATOR AS A CALCULATION RESULTING INDICATOR TURNS IT ON AGAIN

*NON-USER RUN TIME BOOKKEEPING SWITCH

Figure 10-21. Printer Overflow Operations

or total. When an overflow indicator is set off in the Calculation Section, the effect is exactly the same as if overflow processing had just been performed.

A composite flow chart of printer overflow operations is shown in figure 10-21.

Sequence Of Printed Output Operations When All Conditioning Indicators Are Satisfied:

a. If skip before is specified, call SKIP ROUTINE.

b. If space before is specified, call SPACE ROUTINE.

c. Call PRINT ROUTINE.

d. If skip after is specified, call SKIP ROUTINE.

e. If space after is specified, call SPACE ROUTINE.

f. If fetching is specified for this line of print, items a through e are done once for the fetch (call FETCH OVERFLOW), and again for the line to be printed.

g. After all detail output, call OVERFLOW TURN OFF ROUTINE.

h. After all total output, if the last record indicator (LR) is off, call either:

   1. The OVERFLOW ROUTINE, or

   2. If automatic skipping is specified, call the AUTOMATIC SKIPPING ROUTINE.

Table 10-3. Effect of Printer Operations on Overflow Indicators Destination

	RPG I	RPG II
Overflow Area		
Skip	No effect	TURN ON
Space	No effect	TURN ON
Print	TURN ON	TURN ON
New Page		
Skip	No effect	On a line not conditioned by an overflow indicator TURN OFF
Space	No effect	No effect
Print	No effect	No effect
Overflow Area	Channel 12 to 1 line prior to channel 1.	First line after overflow line to last line prior to line 1.
Automatic Skip	Occurs when an overflow indicator is not assigned in the File Description Specification for a printer and is not used in the rest of the program.	
	Paper is moved from overflow area to channel 1.	Paper is moved from overflow area to line six.

Normal Overflow Output. When the overflow area is sensed (and the indicator comes on), and normal overflow output is specified, the following sequence of events takes place. If the overflow area is sensed during detail calculation exception output:

a. Complete remaining detail calculations.

b. Perform detail and heading output not conditioned on the overflow indicator.

c. Read the next record.

d. If a control break occurred, perform total calculations.

e. Perform necessary total output not conditioned on the overflow indicator.

f. Perform total output conditioned on the overflow indicator.

g. Perform heading and detail output conditioned on the overflow indicator.

h. Perform detail calculations.

i. Perform heading and detail output not conditioned on the overflow indicator.

j. Turn off the overflow indicator.

When the overflow area is sensed during detail output, the sequence is item b through item j.

When the overflow area is sensed during total calculation exception output, the sequence is item d through item j.

Figure 10-22 provides a graphic view of when the overflow indicator is turned on or off and when printing is performed.

Fetched Overflow Output. When the overflow indicator is sensed and fetched overflow output is specified, the following sequence of events takes place. Before printing the record specified with F in column 16 of the Output-Format Specifications:

a. Perform total output conditioned by the overflow indicator.

b. Perform heading and detail output conditioned by the overflow indicator.

c. Print the record with the F in column 16 that caused the overflow routine to be fetched.

If the overflow area was sensed during detail calculation exception output, the remainder of the sequence is:

a. Perform the remainder of the detail calculation exception output, if any.

b. Finish the remainder of the detail calculations.

c. Perform detail output not conditioned on the overflow indicator.

d. Read the next record.

e. If a control break has occurred (item d), perform total calculations.

f. Perform total output lines not conditioned on the overflow indicator.

g. Do not perform overflow output. For exceptions, refer to the following subsection titled Multiple Output Lines.

h. Perform detail calculations.

i. Perform detail output.

j. Turn off the overflow indicator.

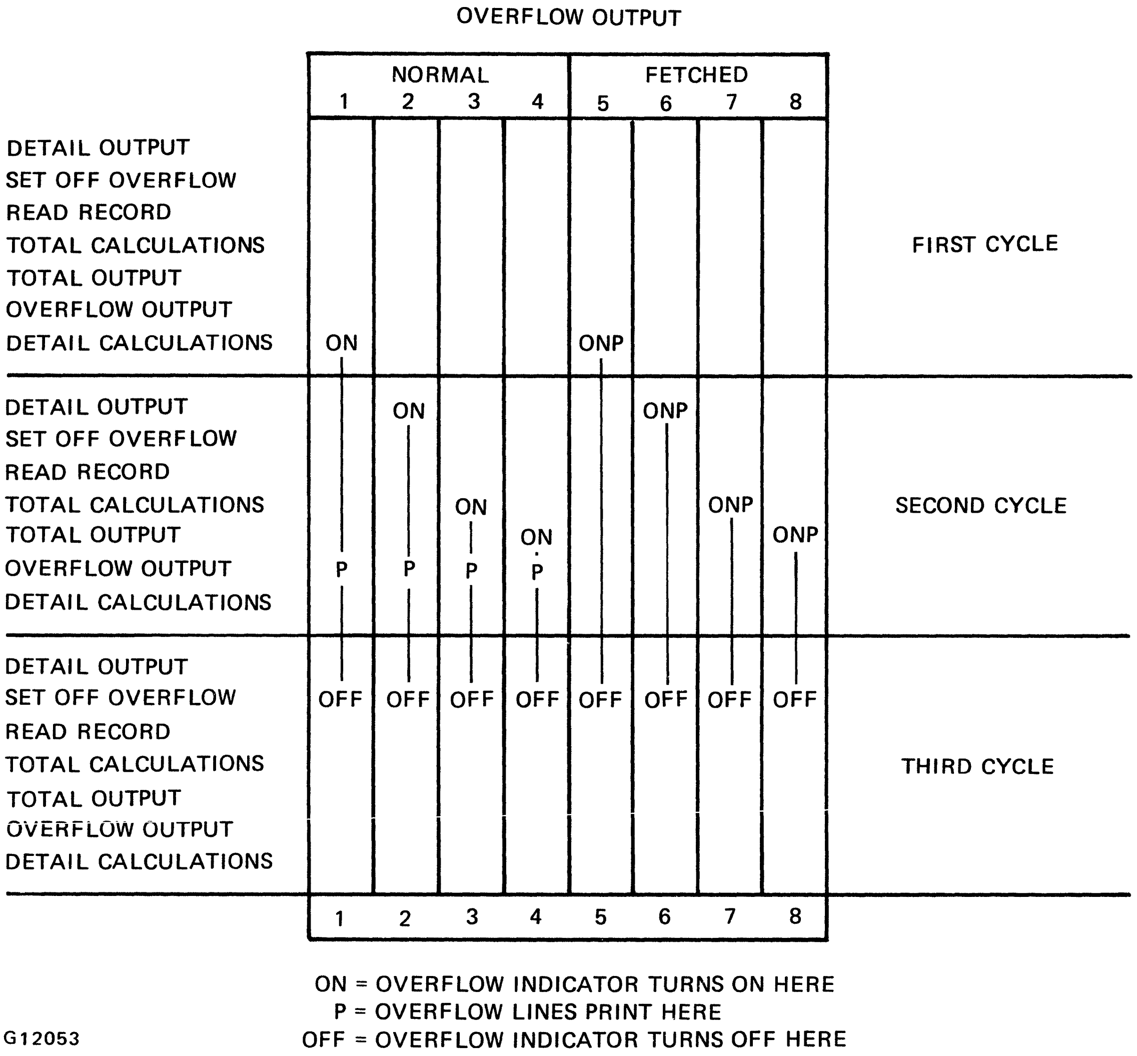


Figure 10-22. Bar Schematic of Printer Overflow Operations

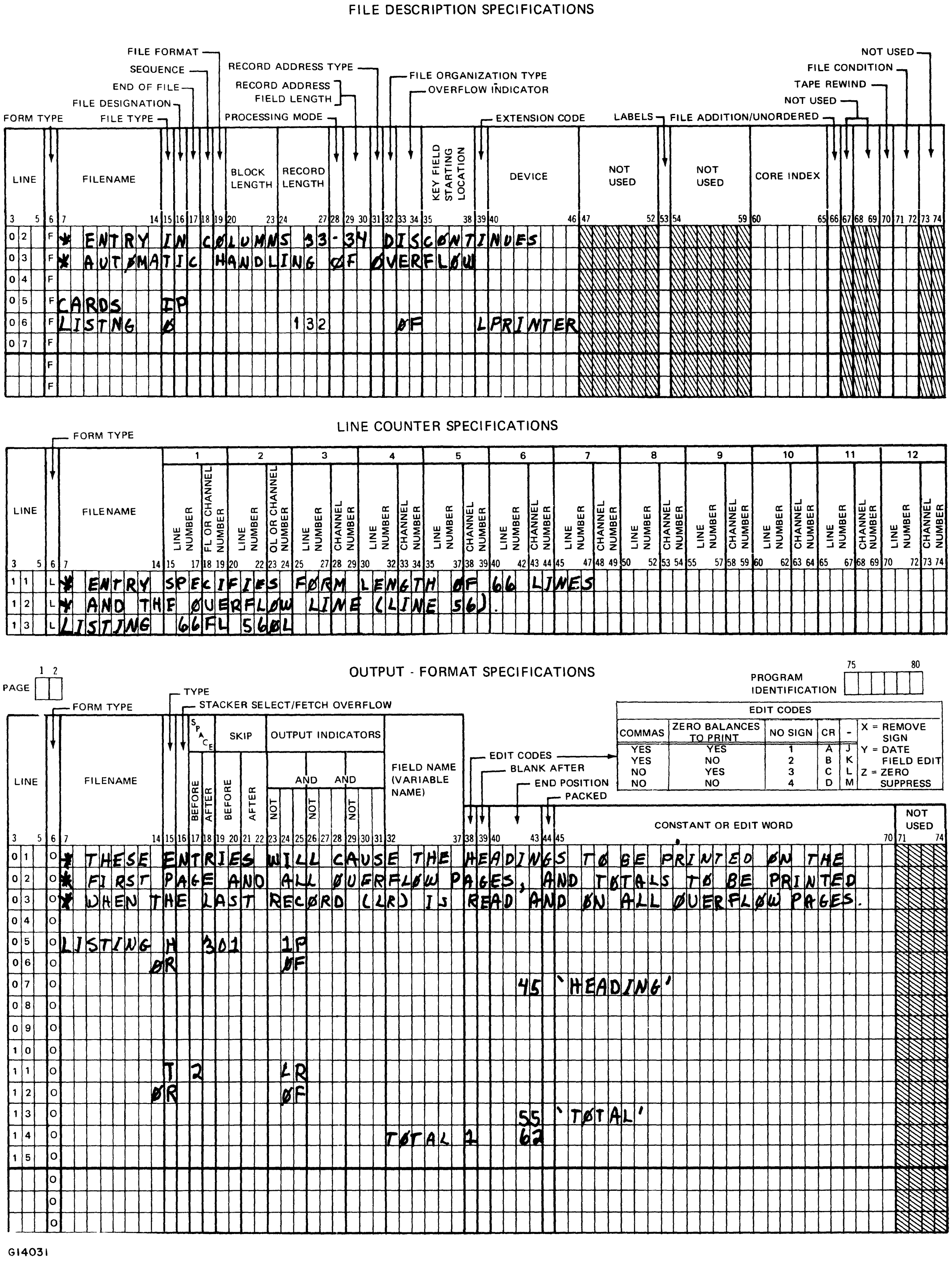
If the overflow area was sensed during detail output, the remainder of the sequence is item c through item j.

If the overflow area was sensed during total calculation exception output, the remainder of the sequence is item e through j.

If the overflow area was sensed during total output, the remainder of the sequence is item f through item j.

Multiple Output Lines. When multiple output lines have been specified for any printer, it is possible to continue printing, either before or after overflow output, and again arrive in the overflow area. If this happens, all overflow output may occur again.

Examples of overflow coding are shown in figures 10-23 and 10-24.



FILE DESCRIPTION SPECIFICATIONS

```
02 F* ENTRY IN CØLUMNS 33-34 DISCØNTINUES
03 F* AUTØMATIC HANDLING ØF ØVERFLØW
04 F
05 FCARDS   IP
06 FLISTNG  Ø         132     ØF     LPRINTER
07 F
```

LINE COUNTER SPECIFICATIONS

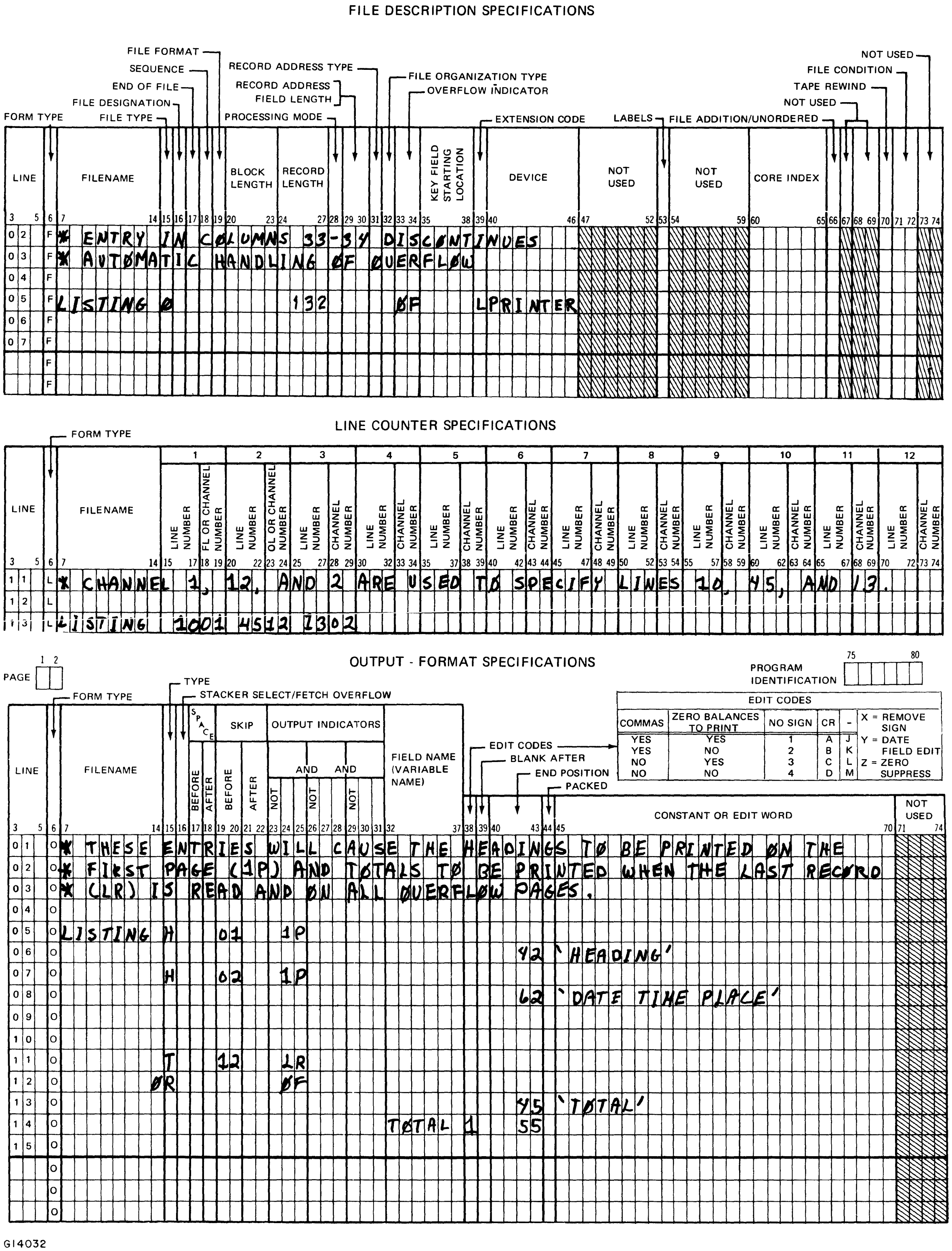
```
11 L* ENTRY SPECIFIES FØRM LENGTH ØF 66 LINES
12 L* AND THE ØVERFLØW LINE (LINE 56).
13 LLISTING  66FL 56ØL
```

OUTPUT - FORMAT SPECIFICATIONS

```
01 O* THESE ENTRIES WILL CAUSE THE HEADINGS TØ BE PRINTED ØN THE
02 O* FIRST PAGE AND ALL ØVERFLØW PAGES, AND TØTALS TØ BE PRINTED
03 O* WHEN THE LAST RECØRD (LR) IS READ AND ØN ALL ØVERFLØW PAGES.
04 O
05 OLISTING H  301   1P
06 O       ØR        ØF
07 O                                   45 `HEADING'
08 O
09 O
10 O
11 O        T 2      LR
12 O       ØR        ØF
13 O                                   55 `TØTAL'
14 O                         TØTAL 1   62
15 O
```

G14031

Figure 10-23. Overflow Coding Example 1 - RPG II


FORM TYPE
LINE
FILENAME
FILE TYPE
FILE DESIGNATION
END OF FILE
SEQUENCE
FILE FORMAT
BLOCK LENGTH
RECORD LENGTH
PROCESSING MODE
RECORD ADDRESS FIELD LENGTH
RECORD ADDRESS TYPE
FILE ORGANIZATION TYPE
OVERFLOW INDICATOR
KEY FIELD STARTING LOCATION
EXTENSION CODE
DEVICE
NOT USED
LABELS
FILE ADDITION/UNORDERED
CORE INDEX
NOT USED
TAPE REWIND
FILE CONDITION
NOT USED
LINE NUMBER
FL OR CHANNEL NUMBER
OL OR CHANNEL NUMBER
CHANNEL NUMBER
PAGE
TYPE
STACKER SELECT/FETCH OVERFLOW
SPACE
BEFORE
AFTER
SKIP
OUTPUT INDICATORS
AND
NOT
FIELD NAME (VARIABLE NAME)
EDIT CODES
BLANK AFTER
END POSITION
PACKED
CONSTANT OR EDIT WORD
PROGRAM IDENTIFICATION

FILE DESCRIPTION SPECIFICATIONS

```
02 F* ENTRY IN CØLUMNS 33-34 DISCØNTINUES
03 F* AUTØMATIC HANDLING ØF ØVERFLØW
04 F
05 FLISTING Ø         132     ØF     LPRINTER
06 F
07 F
   F
   F
```

LINE COUNTER SPECIFICATIONS

```
11 L* CHANNEL 1, 12, AND 2 ARE USED TØ SPECIFY LINES 10, 45, AND 13.
12 L
13 LLISTING  1001 4512 1302
```

OUTPUT - FORMAT SPECIFICATIONS

EDIT CODES					
COMMAS	ZERO BALANCES TO PRINT	NO SIGN	CR	-	X = REMOVE SIGN Y = DATE FIELD EDIT Z = ZERO SUPPRESS
YES	YES	1	A	J	
YES	NO	2	B	K	
NO	YES	3	C	L	
NO	NO	4	D	M	

```
01 O* THESE ENTRIES WILL CAUSE THE HEADINGS TØ BE PRINTED ØN THE
02 O* FIRST PAGE (1P) AND TØTALS TØ BE PRINTED WHEN THE LAST RECØRD
03 O* (LR) IS READ AND ØN ALL ØVERFLØW PAGES.
04 O
05 OLISTING H   01   1P
06 O                                   42 `HEADING'
07 O        H   02   1P
08 O                                   62 `DATE TIME PLACE'
09 O
10 O
11 O        T   12   LR
12 O       ØR        ØF
13 O                                   45 `TØTAL'
14 O                         TØTAL  1  55
15 O
   O
   O
   O
```

G14032

Figure 10-24. Overflow Coding Example 2 - RPG I

SECTION 11

DOLLAR CARD SPECIFICATIONS

Dollar Card Specifications accommodate various extensions to the B 1700 RPG Language, which cannot be handled on the other specification forms. Dollar Cards also allow certain compiler-control options to be set or reset during compilation.

Dollar Cards may appear anywhere within the source deck, as required. The standard Dollar Card Specifications form may be used or options can be coded on the other specification sheets at the points where they will be placed in the source deck.

FIELD DEFINITIONS

Refer to figure 11-1 in conjunction with the following field definitions for the Dollar Card Specifications.

1-2 PAGE

Refer to Section 2 for a complete description.

3-5 LINE

Refer to Section 2 for a complete description.

6 FORM TYPE

This field may be left blank or used to contain the form type of the specification in which the option is to be inserted.

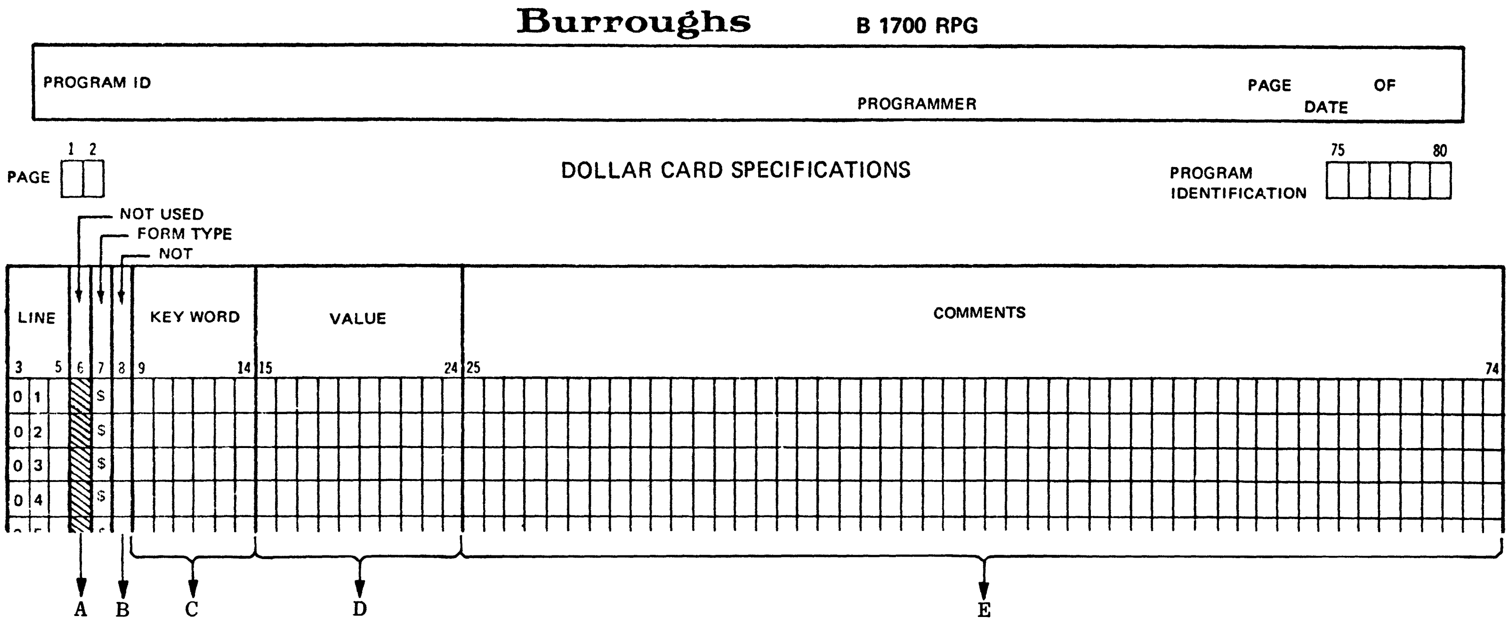
7 $ OPTION

A $ sign must appear in this field.

8 NOT

This field is used to specify that the option entered in the KEY WORD field is set ON (NOT = blank) or OFF (NOT = N). Certain options cannot be turned OFF; these are indicated under the KEY WORD entry description. Valid entries for this field are:

Entry	Definition
Blank	Specified option is "set".
N	Specified option is "reset".



A. 6 May contain the form type of the specification in which the option is to be inserted. Entries: Blank, F, E, L, I, C, or O.

B. 8 Specifies if the option entered in the KEY WORD field is to be set ON or OFF. Entries: Blank or N.

C. 9-14 Names the option to be used, left-justified. For entries, see the descriptions in the text.

D. 15-24 Used to specify a value to be associated with the entry in the KEY WORD field. Not always required. Entries: Alphanumeric, left-justified and numeric, right-justified.

E. 15-74 Used for documenting and commenting.

G14091

Figure 11-1. Dollar Card Specifications Summary Sheet

9-14 KEY WORD

This field is used to name the option to be set or reset (according to entry in NOT field). The option name must be entered left-justified in the field. Options fall into three categories: file identification extensions, RPG extensions, and compiler-directing options.

The following B 1700 dollar options may or may not be valid on other systems and vice versa. Unrecognized dollar cards will be ignored, but a warning will be emitted.

File Identification Extensions

The following extensions are used to specify the external names for files described in the File Description Specifications. They must immediately precede the file in the File Description Specifications to which they refer. None of the options may be reset.

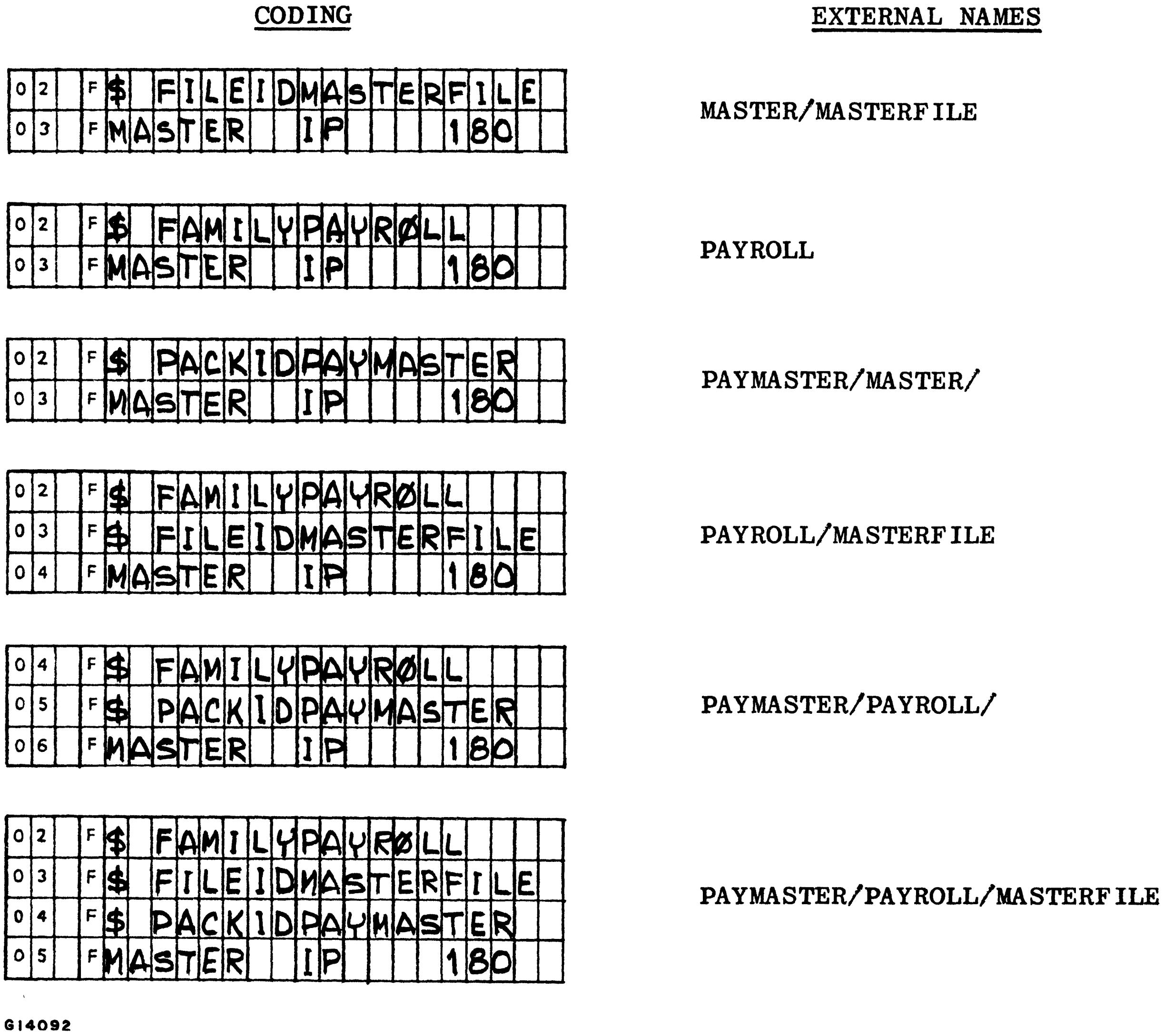
$ PACKID	Specifies the pack name of a disk file. Similar to $ FAMILY and $ FILEID. Default of blank name and MCP assumes systems pack. This entry should be included to ensure correct handling of the file by the MCP.
$ FAMILY	Specifies the external main directory family name associated with the file. The value field contains the name (one to 10 characters left-justified).
$ FILEID	Specifies the external file ID associated with the file. The value field contains the name (one to 10 characters left-justified).
$ DISKID	Same as $ PACKID.

An external name may be in one of the following four forms:

family name
family name/file ID
disk pack ID/family name/file ID
disk pack ID/family name/

The internal name of each file is the filename assigned in the File Description Specifications.

Unless the FAMILY specification is used, the family name will be the same as the internal filename, that is, the filename specified on the File Description Specification. Therefore, if none of the file identification extensions are used the internal and external name will be the same, i.e., the family name. Figure 11-2 illustrates how the insertion of the different file identification extensions before a file called MASTER affects that file's external name.



```
CODING                                  EXTERNAL NAMES

02 F$ FILEIDMASTERFILE                  MASTER/MASTERFILE
03 FMASTER  IP   180

02 F$ FAMILYPAYRØLL                     PAYROLL
03 FMASTER  IP   180

02 F$ PACKIDPAYMASTER                   PAYMASTER/MASTER/
03 FMASTER  IP   180

02 F$ FAMILYPAYRØLL
03 F$ FILEIDMASTERFILE                  PAYROLL/MASTERFILE
04 FMASTER  IP   180

04 F$ FAMILYPAYRØLL
05 F$ PACKIDPAYMASTER                   PAYMASTER/PAYROLL/
06 FMASTER  IP   180

02 F$ FAMILYPAYRØLL
03 F$ FILEIDMASTERFILE                  PAYMASTER/PAYROLL/MASTERFILE
04 F$ PACKIDPAYMASTER
05 FMASTER  IP   180
```

G14092

Figure 11-2. Coding File Identification Extensions - Dollar Sign Options

RPG Extensions

The following options may appear only within the File Description Specifications, except RSIGN, and must immediately precede the specification line describing the file to which they apply. None of the following operations may be reset except RSIGN and ONEPAK.

$ AREAS — Specifies the maximum number of areas to be allocated for the file (disk files only). The VALUE field contains an integral value (right justified, leading zeros optional). A system-dependent default value of 25 is assumed unless specified by use of this dollar option. Maximum number of areas that may be assigned is 105.

$ RPERA — Specifies the maximum number of logical records that will be written in each disk area. The VALUE field contains an integral value (right-justified, leading zeros optional). The default value assigned is the first multiple of the records-per-block that is equal to or greater than 500.

$ OPEN — The use of this option results in the explicit open of a file at Beginning-of-Job time. A $ OPEN card must appear before the File Specification card for each file that is to be explicitly opened at BOJ time.

The default condition is as follows:

> At BOJ time, the primary and all secondary files are opened, as well as all input or update indexed files. Demand and output files are implicitly opened upon the first read from, or write to, them.

$ CLOSE — The use of this option forces a serial input file to remain open until the program reaches End-of-Job. A $ CLOSE card must appear before the File Specification card for each file that is to remain open until EOJ. The default is for each serial input file to be explicitly closed upon encountering the End-of-File.

$ AAOPEN — Is a file OPEN time option used to set a bit in the program's code file File Parameter Block for this file and to allocate all disk space areas at the time the file is opened.

$ ONEPAK — Specifies that this particular file must be contained on one disk.

$ CYL — Allocates file areas starting on an integral cylinder boundary.

$ DRIVE — Allocates a physical drive to that particular file. Applies only to Systems disks. VALUE field must be 0-15. Option may not be reset and is not related to PACKID.

$ REFORM — Input and update disk files are assumed to have the record and block sizes specified in the disk file header, regardless of those specified in the File Specification card. The use of the $ REFORM option causes the record and block sizes declared in the File Specification card to override those in the disk file header. For example, the DEFAULT bit in the File Parameter Block is reset.

A $ REFORM card must appear before each input or update disk file where it is necessary to override the record and block sizes specified in the disk file header.

$ REORG Specifies a specialized method of sorting indexed files and will be invoked at End-of-Job. The REORG feature only sorts the additions and then merges them, in place, into the master file. This method of sorting should decrease the sort time and the temporary disk area required. The VALUE field (columns 25-46) must contain the external file identifier of indexed file. The VALUE field has the following naming convention:

disk pack ID/family name/file ID

The use of the $ REORG option results in the following program execution at EOJ:

INDEXED FILE - RPG/REORD

"B" INDEXED FILE - RPG/REORG

Information deleted

$ RSIGN Indicates to the compiler the location of the sign of external numeric data items. When set, all signs are assumed to be in the low-order (rightmost) position of the field; when reset, all signs are assumed to be in the high-order (leftmost) position of the field. This option may be set and reset at different points in the source program, allowing different fields to have different sign positions. If the option is used, it will override the sign position specified in the Control Card Specifications.

$ DNAME The value field contains an alpha mnemonic for the physical hardware device to which it refers.

$ CHECK Causes all patching records to be sequence checked. Syntax errors are generated for any patch record that is out of sequence.

If the $ CHECK option is not specified, then sequence warnings are issued. The warning is denoted by an "S" appearing to the left of the record's sequence number on the output listing.

$ MERGE Causes the following records in the patch file to be merged with an existing source file. The source file is from a source other than a card reader. The source file is expected to reside on tape or disk (disk is default). The internal file name is SOURCE and external file name is SOURCE. The default blocking option is set.

If it is desirable to change the external file name or input device from disk to tape, then a label equation card must be used. The $ NEW option can be used with the $ MERGE option to create a new output source file with patches.

The sequence fields are used to control the merge sequence. If the sequence field of a patch record and disk record match, then the patch record replaces the disk record; otherwise, patch records are merged with the tape or disk file. The existing source file remains unchanged.

A "P" is appended to the left of the record's sequence field on the source listing, specifying that the image is a patch record.

$ NEW Creates new output source files. If $ MERGE option is used, then the patches are included in the new source file. The new source file is created on disk or tape (disk is default). Internal file name is NEWSOURCE and external file name is NEWSOURCE. If it is desirable to change the external file name or input device from disk to tape, then label equations cards must be used.

$ SEQ Starts sequencing the subsequent source lines on the output listing. If $ NEW option is used, then the subsequent new source file is sequenced. The starting sequence number begins with the number contained in the sequence field (columns 1-5) of the $ SEQ option. If the sequence field is blank, the starting sequence number begins with zero. The subsequent records have sequence numbers which are incremented by the value contained in columns 22-24 of the $ SEQ option. If columns 22-24 are blank, then 010 is assigned.

$ DMSNAM Specifies that the compiler print the DMS library files used for the compile. This option is set on by DEFAULT.

$ DIGIT8 Causes RPG indexed files to use eight-digit relative record numbers in the tag file. The default is six digits. With this option, RPG indexed files are compatible with COBOL indexed files. The $ DIGIT8 compiler option must be specified prior to each file's File Description Specification for which an eight-digit relative record number tag file is required.

$ SECURE Causes RPG programs to access and create single-named tag files when the program is executed using the USERCODE/PASSWORD pair implemented on the B 1800/B 1700 file security system.

Only one of the following four $ RPG Extensions may be used per file specification:

$ PTAPE Modifies READER or PUNCH to apply to a paper tape reader.

$ TAPE7 Modifies a tape device to 7-track.

$ TAPE9 Modifies a tape device to 9-track.

$ CASSET Modifies a tape device to a cassette.

Compiler-Directing Options

The following options may appear anywhere within the B 1700 RPG Source Program. These options direct the compiler to perform specific functions. All of the following options may be "reset".

$ LIST — Specifies that the compiler produce a single spaced output listing of the source statements with the error or warning messages. This option is set "on" by default. Resetting to "off" will not inhibit the errors or warning messages from printing.

$ LOGIC — Specifies that the compiler produce a single-spaced listing of each source specification line, followed immediately by an intermediate code used to generate RPG S-code. The listing is produced after the NAMES listing (if the NAMES option is set), and does not include addresses or bit configurations, but only the opcodes and logical operands of the program.

$ MAP — Specifies that the compiler produce a single-spaced listing, detailing the program's memory utilization. The MAP listing is produced after the LOGIC listing (if the LOGIC option is set).

$ NAMES — Specifies that the compiler is to produce a single-spaced listing of all assigned indicators, file names, and field names. The attributes associated with each file and field are also listed. The NAMES listing is produced immediately after the normal source input listing.

$ BAZBON — Specifies that if an indicator is assigned to a field to test for ZERO or BLANK in the Input or Calculation Specifications and the same field is used in the Output Specifications with a BLANK AFTER designation, that indicator will be turned ON after the field is blanked during the output operations. The maximum number of associations is 200. Should a N (NOT) be specified in column 8, the indicator will be turned OFF, overriding the original RPG 1 or RPG II specifications. The default values are as follows:

RPG I	BAZBON is set
RPG II	BAZBON is reset

$ ZBINIT — Specifies that all ZERO or BLANK indicators are initialized ON at Beginning-of-Job. The maximum number of associations is 200. If a N (NOT) is specified in column 8, then the ZERO or BLANK indicators are initialized OFF regardless of the specifications for RPG I or RPG II. The default values are as follows:

RPG I	ZBINIT is set
RPG II	ZBINIT is reset

$ STACK — If infrequent stack overflow conditions occur during program execution, the user may change the stack size of the resultant program. This should be used only when a legitimate STACK overflow condition has occurred (i.e., nested subroutines more than 8 deep). The default stack size is 313 bits which will allow 8 entries in the stack. To increase the stack size, add 39 bits to the default size of 313 for each additional stack entry.

$ XREF — Allows the RPGXRF file to be created during compilation for use as input to the RPG/XREF program. The XREF option must be placed at the beginning of the RPG source program, prior to the first File Specification or Control Card Specification. At completion of the compilation, the RPG/XREF program is automatically executed. The program generates the cross-reference listing.

$ PARMAP — Produces a single-spaced listing of the compiler-generated paragraph names, source statement numbers, and actual segment displacements of the emitted code. This listing may be used to relate to the LOGIC listing.

$ SUPR — Specifies that the compiler is to suppress all warning messages from the source program listing. (Error messages still print.)

$ XMAP — Specifies that the compiler print a single-spaced listing of all the code generated, complete with actual bit configurations and addresses. Combined with the listing produced by the LOGIC option, complete information about the generated code of the program is available. The XMAP listing is produced after the MAP listing if the MAP option is set.

$ LIBR — Copies source records from a library file located on tape or disk to a new or merging file. This option can reference different library files. Columns 15-24 contain the Pack-Id; columns 25-34 the Multi-File-Id; columns 35-44 the File-Id; columns 45-49 the starting sequence number in the library file (optional); and columns 50-54 the ending sequence number in the library file (optional).

An "L" is appended to the record image and is placed to the left of record's sequence field in the output listing. The "L" specifies that the record is from a library source file.

$ NEWID — Causes the six character identifier specified in columns 15-20 of the $ NEWID option to be placed in columns 75-80 of the subsequent lines in the output listing. If the $ NEW option is used, then the six-character identifier is also placed in columns 75-80 of the subsequent source records in the new source file.

$ VOID — Deletes records of the source file on the output listing. If the $ NEWID option is used, then the records are deleted in the new source file. Source lines or records are deleted up to and including the five-digit void limit. If the void limit field (columns 20-24) is blank, then only the source record with sequence number matching the sequence field $ VOID option is deleted.

$ PAGE — Causes the RPG compiler to eject a page on the output source listing. This option can appear anywhere in the RPG source program and is especially useful for separating subroutines on the output source listing.

15-24 VALUE

This field is used to specify a value to be associated with the option entered in the KEY WORD field. Not all options require a value; those that do are so designated in the individual descriptions of each option (KEY WORD field).

All alphanumeric values must be entered left-justified in the VALUE field. All numeric values must be entered right-justified in the VALUE field; leading zeros are optional.

25-74 COMMENTS

This field is available for inclusion of comments and documentary remarks, and may contain any valid EBCDIC characters.

75-80 PROGRAM IDENTIFICATION

Refer to Chapter 2 for a complete description.

SECTION 12

COMPILER OPERATION

The Burroughs RPG Compiler is an integral part of the B 1700 software system. It is treated in many respects as merely another program to be executed and can be multiprogrammed with other programs, or with itself.

SOURCE INPUT

An input card file labeled RPG/CARD is the only source required by the compiler. All source input must be punched in the EBCDIC character set unless it is translated on input.

If it is desired that the input file be assigned to a device other than the card reader, then this may be achieved by using the FILE statement. This, of course, means that the device from which the source file is read should have the same record length (80 characters) and blocking factor (1) as the device from which the compiler expects to read the file. An RPG/VECTOR file assigned through the FILE statement should have a record length of 96 and a blocking factor of 1.

CONTROL CARD SYNTAX

The format of the MCP control cards is as follows:

```
                                     ┌┌TO  LIBRARY┐┐
    ? COMPILE <program-name> WITH RPG ││FOR SYNTAX ││  <control-attributes> ...
                                     │└    SAVE   ┘│
                                     └             ┘

    ? CHARGE <charge-number>

    ? MEMORY <memory-size>

    ? PRIORITY <priority-number>

    ? FILE <internal-file-name> <file-attributes>

    ? DATA RPG/CARD

      source specification cards

    ? END

   ┌? DATA RPG/VECTOR            ┐
   │                             │
   │  compile-time table cards   │
   │                             │
   └? END                        ┘
```

All of the above control cards are fully discussed in the B 1700 System Software Operational Guide, Form No. 1068731. The format of the four options of the COMPILE statement are discussed here.

COMPILE

This is a "compile and go" operation. If the compilation is error-free, the MCP schedules the object program for execution. The program will not be entered into the Disk Directory, and must be recompiled to be used again. The "compile and go" is the default option of the COMPILE statement. COMPILE may be abbreviated as CO.

COMPILE TO LIBRARY

This option will leave the program object file on disk and will enter the program-name into the Disk Directory after an error-free compilation. The program is not scheduled for execution. LIBRARY may be abbreviated as LI.

COMPILE SAVE

This option combines the execute and library options. The MCP will enter the program-name into the Disk Directory and will leave the object program file on disk. The MCP will also schedule the program for execution after an error-free compilation. The program remains in the Disk Directory. SAVE may be abbreviated as SA.

COMPILE FOR SYNTAX

This option provides a diagnostic listing as the only output. This option does not enter the program-name into the Disk Directory or leave the program object file on disk. SYNTAX may be abbreviated as SY.

The following sample deck could be used to compile a source program contained in punched card to library:

```
? COMPILE PROGRAM/TEST1 WITH RPG TO LIBRARY
? CHARGE 12345
? DATA RPG/CARD
  source specification cards
? END
```

The following sample deck could be used to compile a source program contained on magnetic tape for syntax:

```
? COMPILE AAA/BBB/TEST2 WITH RPG FOR SYNTAX
? FILE CARDS NAME = RPG/SOURCE DEFAULT TAPE
? END
```

NOTE

For the effect of the DEFAULT option, refer to the B 1700 System Software Operational Guide, Form No. 1068731.

VECTOR FILE INPUT

The vector input files consist of compile-time and pre-execution-time vector files, both of which are described in the paragraphs that follow.

COMPILE-TIME VECTOR FILES

If the program requires that table or array files be included during compilation, the compiler requires an input card file labeled RPG/VECTOR. This file may be input from a medium other than cards, through use of the FILE statement. Card input may be punched in either the BCD (96 column card) or EBCDIC (80 column card) character set.

Vector files must be entered in the same order as specified in the Extension Specifications, and each file must contain exactly the number of records specified. No separators define the end of one vector and the beginning of the next; records are read into one vector until that vector is full. Filling of the next vector is begun from subsequent records.

The following sample deck could be used to compile a source program (with compile-time vector input) contained on cards to library:

```
? COMPILE AAA/TEST3 WITH RPG TO LIBRARY
? DATA RPG/CARD
  source specification cards
? END
? DATA RPG/VECTOR
  table file cards
? END
```

PRE-EXECUTION-TIME VECTOR FILES

If the program requires table or array files to be included at the beginning of object program execution, the program will require card files with the labels as specified by the program in the File Description and Extension Specifications.

LABEL EQUATION CARD

The label equation card is used to change a compiler file name in order to avoid duplication of file names in the disk directory when operating in a multiprogramming environment, or to access a file name other than the default file names in the RPG compiler.

Table 12-1 lists the files that are used by the RPG compiler for source input and compilation output.

Table 12-1. Files Used for Source Input and Compilation Output

Internal File Name	External File Name	Description
CARDS	RPG/CARD	Input file from the card reader.
LINE	RPG/LIST	Source output listing to the line printer.
NEWSOURCE	NEWSOURCE	Output file to disk for a NEW source file when the $ NEW option is used.
SOURCE	SOURCE	Input file from disk when $ MERGE option is used.
TABCRD	RPG/VECTOR	Input file for tables from the card reader.

Label equation cards use the FILE statement. Refer to the B 1800/B 1700 Systems, System Software Operational Guide, Form Number 1068731, for additional information. The format of the FILE statement is:

```
? FILE <internal-file-name>  NAME=<file-identifier>;
```

The FILE statement must immediately follow a COMPILE, EXECUTE, DYNAMIC, or MODIFY statement.

The following sample deck could be used to compile a source program which merges a patch file on cards with a SOURCE program located on disk. The SOURCE program's name is PROGRAM1 and the OBJECT program's name is OBJ.PROG1.

```
? COMPILE OBJ.PROG1 WITH RPG TO LIBRARY
? FILE SOURCE NAME PROGRAM1
? DATA RPG/CARD
  $ MERGE card followed by patch card source statements
? END
? DATA RPG/VECTOR
  table file cards
? END
```

SECTION 13

DATA MANAGEMENT SYSTEM SPECIFICATIONS

RPG DATA MANAGEMENT FACILITY

The RPG data management system provides the ability to process a DMSII data base using conventional RPG syntax and concepts.

A data base is constructed by a DASDL compilation. The DASDL compiler, using a description of the data base (DASDL source statements), produces a data base dictionary file which contains information about each structure described within the data base.

Data sets, sets, and subsets are the structures which make up the data base. A data set description in DASDL specifies the logical structure of a file. Set and subset descriptions in DASDL specify the logical structures of indexes or index tables (paths) that are used in storage and retrieval of the data contained in a data set.

Data sets and sets on the outermost level of the description are disjoint data sets and disjoint sets respectively. A data base must contain at least one disjoint data set.

In order to effectively use the RPG data management system, understanding how the data base structures relate to the concepts of RPG files is important.

RPG DATA MANAGEMENT FILES

The basic structure in a data base is the disjoint data set. A data set is similar to the RPG concept of a file in that a data set contains the actual records. Unlike RPG, a record can contain not only items of information but other data sets (embedded data sets).

Additionally, records within a data set can contain access paths (or pointers) to the records of another disjoint data set. This access path is called a manual subset.

In order to better understand an embedded data set, consider the data set EMPLOY containing records for each employee. This data set may contain an embedded data set named WRKHIS for each employee's work history. For every employee record in EMPLOY, 0, 1, or more work history records may be stored in the WRKHIS data set. Later, when employee records are being read, DMS delivers all the requested WRKHIS records which have been previously stored. If there are no WRKHIS records, DMS informs the RPG program of this condition.

To better understand how a manual subset is used, consider two disjoint data sets, DEPART and EMPLOY. To gain access to all of the employees within each department ordered by the employees last name, a manual subset may be used. The manual subset provides paths from one disjoint data set to another disjoint data set. That is, within each DEPART record there is a manual subset called DEPEMP.

The following DASDL source statements pertain to the previous example:

```
     DEPART DATA SET
         (
          .
          .
          .
          DEPEMP SUBSET OF EMPLOY KEY IS (LASNAM)
          .
          .
           )
     EMPLOY DATA SET
         (
          .
          .
          .
          LASNAM
          .
          .
          .
           )
```

By manual, it is meant that the RPG program must insert the record in the manual subset DEPEMP after creating and storing a record in the data set EMPLOY. Similarly, the RPG program must remove the record in the manual subset DEPEMP after deleting the record in the data set EMPLOY.

A data set may be accessed by way of various paths or, in DMS terminology, by means of sets or subsets. Sets, automatic subsets, and manual subsets with keys are structures that organize the data set records into logical sequences. In RPG terms, this is known as indexing by way of a key. A set provides access to all of the records of a data set. An automatic subset provides access to a limited collection of records, that is, a condition for membership in the subset exists, and the condition is checked each time a record is to be added to the data set.

Sets and automatic subsets declared as indexed-sequential in DASDL allow for accessing successive records based on the ordering sequence of the key, or a given record may be accessed based on the value of the key. In DMS, several sets and/or automatic subsets may exist for the same data set. That is, the same data set may be accessed via several different keys.

An indexed-random set is a structure which provides access to a data set record based on the value of a key. Unlike other sets and automatic subsets, ordering is not implied, except in the isolated case where two or more records contain duplicate key values. In this case, access to the NEXT record makes available the next record which contains a duplicate of the key. All sets other than manual subsets are maintained by DMS. When a record is added, updated, or deleted, DMS adjusts all indexes affected by the various "key" values in the record. No special action is required in the RPG program.

DATA BASE SPECIFICATION

There must be only one Data Base Specification in an RPG program.

The Data Base Specification is optional but is required when any DMS files are declared in the program. If used, the Data Base Specification must immediately follow the Control Specification, if one exists.

FIELD DEFINITIONS

1-2 PAGE

Refer to Section 2 for a complete description.

3-5 LINE

Refer to Section 2 for a complete description.

6 FORM TYPE

This field must contain the letter D.

7-16 DATA BASE NAME

This field must contain the name of the physical data base being used and must be the same data base name referenced by DASDL.

17-26 LOGICAL DATA BASE

This field may contain the name of the logical data base as specified by DASDL. The logical data base is a portion of the physical data base which may be accessed by the program. This logical data base name must have been defined in DASDL. If no entry is made, the entire physical data base is available to the RPG program.

27 ACCESS MODE

This field must contain the letter I or U. If the data base is to be accessed input only, enter the letter I. If the data base is to be updated, enter the letter U.

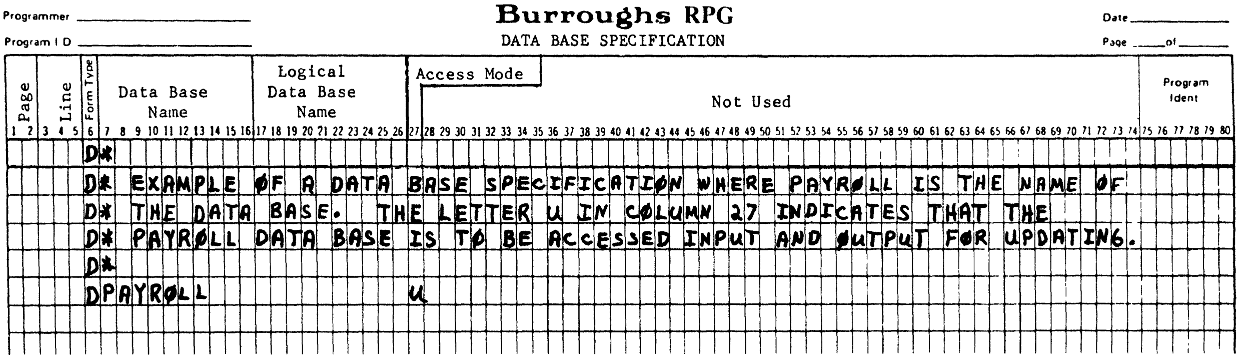
28-74 These columns must be left blank.

75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a complete description.

If the RPG library files created by DASDL reside on a disk other than the system disk, the Data Base Specification must be preceded by a $ PACKID or $ DISKID file identification option.

Figure 13-1 shows an example of coding a Data Base Specification.



Programmer ____________ **Burroughs RPG** Date ________

Program I D ____________ DATA BASE SPECIFICATION Page ____ of ____

Page	Line	Form Type	Data Base Name	Logical Data Base Name	Access Mode	Not Used	Program Ident
1 2	3 4 5	6	7 8 9 10 11 12 13 14 15 16	17 18 19 20 21 22 23 24 25 26	27	28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74	75 76 77 78 79 80

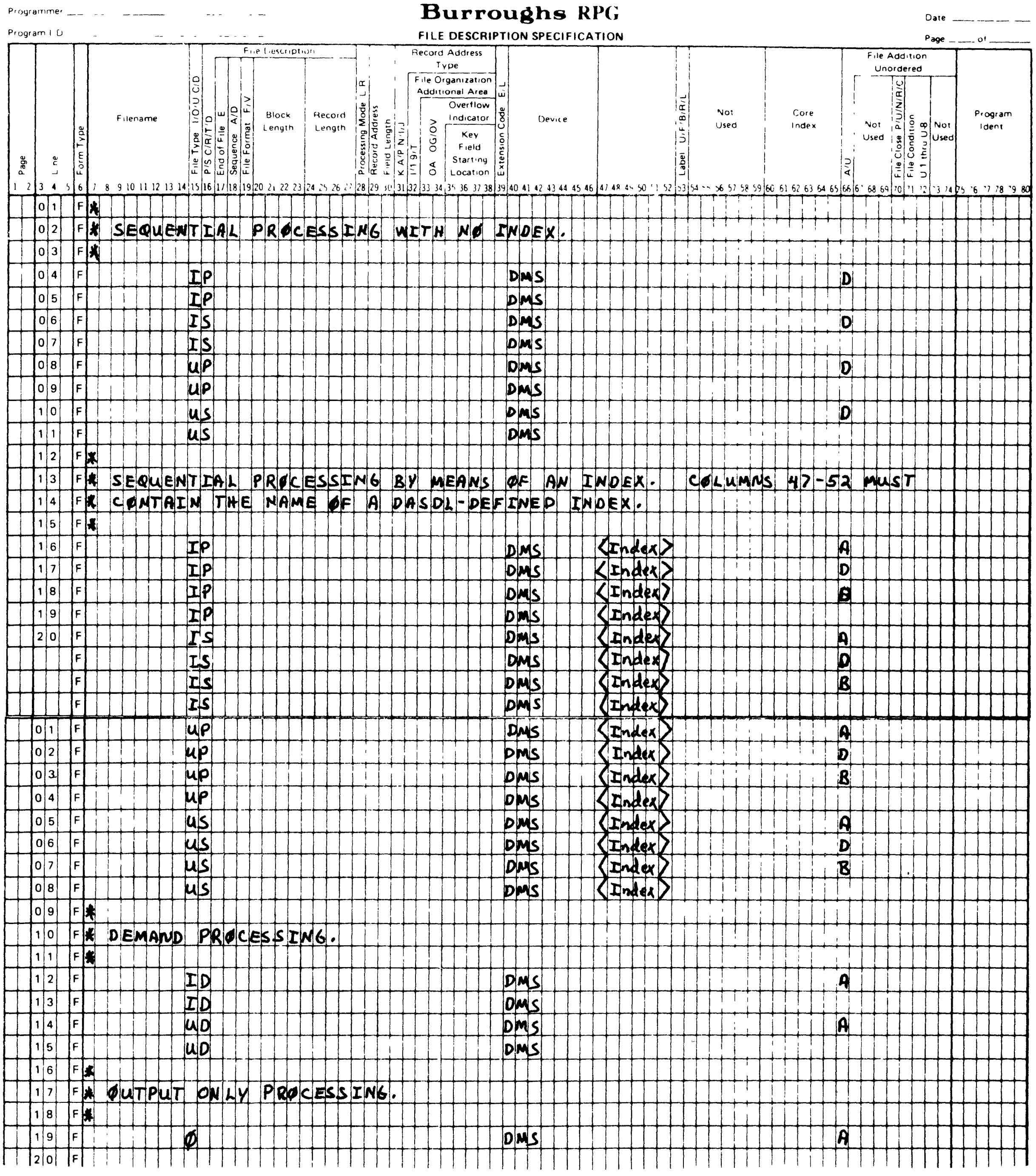
```
     D*
     D* EXAMPLE ØF A DATA BASE SPECIFICATIØN WHERE PAYRØLL IS THE NAME ØF
     D* THE DATA BASE.  THE LETTER U IN CØLUMN 27 INDICATES THAT THE
     D* PAYRØLL DATA BASE IS TØ BE ACCESSED INPUT AND ØUTPUT FØR UPDATING.
     D*
     DPAYRØLL            U
```

G12069

Figure 13-1. Data Base Specifications

DATA MANAGEMENT FILE DESCRIPTION SPECIFICATIONS

Every disjoint data set to be used by an RPG program must be described to the RPG compiler in the File Description Specifications. The disjoint data set must be specified in DASDL. Figure 13-2 shows the various processing methods for DMS files.



Programmer ____ Burroughs RPG Date ____

Program I D ____ FILE DESCRIPTION SPECIFICATION Page ___ of ___

```
  01 F*
  02 F* SEQUENTIAL PRØCESSING WITH NØ INDEX.
  03 F*
  04 F        IP                       DMS                       D
  05 F        IP                       DMS
  06 F        IS                       DMS                       D
  07 F        IS                       DMS
  08 F        UP                       DMS                       D
  09 F        UP                       DMS
  10 F        US                       DMS                       D
  11 F        US                       DMS
  12 F*
  13 F* SEQUENTIAL PRØCESSING BY MEANS ØF AN INDEX.  CØLUMNS 47-52 MUST
  14 F* CØNTAIN THE NAME ØF A DASDL-DEFINED INDEX.
  15 F*
  16 F        IP                       DMS    <Index>            A
  17 F        IP                       DMS    <Index>            D
  18 F        IP                       DMS    <Index>            B
  19 F        IP                       DMS    <Index>
  20 F        IS                       DMS    <Index>            A
     F        IS                       DMS    <Index>            D
     F        IS                       DMS    <Index>            B
     F        IS                       DMS    <Index>
  01 F        UP                       DMS    <Index>            A
  02 F        UP                       DMS    <Index>            D
  03 F        UP                       DMS    <Index>            B
  04 F        UP                       DMS    <Index>
  05 F        US                       DMS    <Index>            A
  06 F        US                       DMS    <Index>            D
  07 F        US                       DMS    <Index>            B
  08 F        US                       DMS    <Index>
  09 F*
  10 F* DEMAND PRØCESSING.
  11 F*
  12 F        ID                       DMS                       A
  13 F        ID                       DMS
  14 F        UD                       DMS                       A
  15 F        UD                       DMS
  16 F*
  17 F* ØUTPUT ØNLY PRØCESSING.
  18 F*
  19 F        Ø                        DMS                       A
  20 F
```

G12070

Figure 13-2. Processing Methods for DMS Files

FIELD DEFINITIONS

1-2 PAGE

Refer to Section 2 for a complete description.

3-5 LINE

Refer to Section 2 for a complete description.

6 FORM TYPE

This field must contain the letter F.

7-14 FILE NAME

The file name specified in the File Description Specifications must be the name of a disjoint data set which is defined in DASDL.

15 FILE TYPE

Valid entries for this field are:

Entry	Definition
I	The file specified is an input file.
U	The file specified is an update file.
O	The file specified is an output file.

16 FILE DESIGNATION

This field further describes the use of input and update files. Valid entries for this field are:

Entry	Definition
Blank	Output file.
P	Primary sequential file.
S	Secondary sequential file.
D	Demand file.

Chained, record address, and table files must not be specified as a DMS file.

17 END OF FILE

Refer to Section 4 for a complete description.

18 SEQUENCE

This field is used only by primary and secondary files to indicate whether or not the program is to check the sequence of the input records. Columns 61-62 of the Input Specifications must specify the match fields within the input file records, and this file must have an automatic subset or ordered set specified in columns 47-52 of the File Description Specifications. Valid entries for this field are:

Entry	Definition
Blank or A	Records with matching fields are to be sequence-checked in ascending order.
D	Records with matching fields are to be sequence-checked in descending order.

This column must be blank for unindexed primary and unindexed secondary DMS files.

Sequence checking is performed when matching fields have been specified for the records in a DMS file. If a record from a matching input DMS file is found to be out of sequence, the program halts. If the program halts, the operator may make one of the following entries:

Console Message	Definition
<program job number>AXGO	Ignore the record out of sequence and read the next record from the same DMS file.
<program job number>AXSTOP	Ignore the record out of sequence, turn on the LR indicator, and perform all end of job totals.
<program job number>DS	Discontinue the program.

All sequence checking is performed according to the EBCDIC collating sequence (see Appendix B). If any matching DMS file specifies descending (D) sequence, all matching files must specify descending sequence.

19 FILE FORMAT

The only valid entries for this field for DMS files are blank or the letter F.

20-39 These columns must be blank for DMS files.

40-46 DEVICE

The only valid entry for this field is DMS.

Specifying DMS in the DEVICE field indicates that this file is a DMS file. For all files declared as DMS files, the compiler obtains RPG-DMS library files from disk. These library files are created by the DASDL compiler (refer to $ RPGLIB in the DASDL Information and Suggestions subsection) for the data base declared in the Data Base

Specifications. The compiler uses these library files to determine which field names in the RPG source program are from the DMS file.

47-52 INDEX NAME

Enter the name of the automatic subset or ordered set which is the index into the specified DMS file. If this field is blank, then the specified data set is processed in its physical order rather than being processed by index order.

This entry must be blank for demand files, files specified as output, and primary or secondary DMS files which have no index specified in DASDL.

53-65 These columns must be left blank for DMS files.

66 FILE ADDITIONS/UNORDERED/DELETION

The letter U in this column for a DMS file is invalid.

Valid entries for this field are:

Entry	Definition
A	Records may be added to an existing file.
D	Records may be deleted from an existing file.
B	Records may be added to and/or deleted from an existing file.
Blank	Records are not to be added or deleted.

If ADD is specified in columns 16-18 of the Output-Format Specifications for this file, then enter the letter A or B in this field.

NOTE

Adding records to an indexed cycle-driven DMS file when the file is not the file which has just been read in the current RPG cycle can produce unexpected results. For instance, if a record is added to a secondary file following a read of a record from the primary file, the secondary file's record work area contains the added record information. At the time the record information for the secondary file is made available, the information reflects the added record. This situation has implications affecting which record the program receives when the file is processed by means of matching records or control level breaks. To avoid this situation, add records at detail time and when the record identification indicator is set ON for this file.

1-72 Since only disjoint data sets can be specified on File specs, any embedded data sets associated w/ a disjoint dataset conditioned by a U-indicator are treated as having been conditioned by same U-indicator. All calc specs and Output specs affecting embedded data set should be conditioned by the same U-indicator as the parents data set and are ignored if U-indicator is off.

If DEL is specified in columns 16-18 of the Output-Format Specifications for this file, enter the letter D or B in this field.

If the letter O is entered in column 15 of this File Description Specification, only the letter A is allowed in this field.

For primary or secondary DMS files with no index specified, the only valid entry in this field is blank or the letter D.

The only valid entry for demand DMS files is blank or the letter A.

Deletion of records from a demand DMS file is accomplished with the DELET operation code.

67-70 These columns must be left blank for DMS files.

73-74 → must be blank for DMS files

75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a complete description.

Since the location of all DMS files is maintained by DMS, the File Identification options $ PACKID, $ FAMILY, $ FILEID, and $ DISKID must not be specified for any DMS files defined in the File Description Specifications.

Figure 13-2 shows the various processing methods for DMS files.

71-72 → This field may be used on I, UPD, O, and restart DMS files and indicates whether the file is conditioned by an external indicator. If a DMS file is conditioned by an external indicator, the file is used only when that indicator is on. Indicator off, the file treated on I as if it is at EOF. In Calc Specs, if any DMS oper. are done to the conditional file, the oper. should be conditioned on the same U-indicator specified for the DMS file. If the DMS oper. aren't conditioned by the same U-indicator & 1 of the indicators is off, the oper. are ignored and the D1 resulting indicator, if used, is not affected. Likewise, any Output done to the DMS file should be conditioned by the same U-indicator used on the DMS file. If the same U-indicator isn't used, the O record is built (and any blank after oper done) but the record isn't written to the file.

U-indicators being off may still require a recompile of the DASDL pgm to prevent a VERSIONERROR (of dictionary) at Data Base open time.

It is impt. to note that DMS files are accessed by opening the database and that even though all DMS files in the pgm may be conditioned by U-indicators which are off, the data base is opened and any exceptions that might normally occur at open time can still happen.

* Refer to File Condition (cols. 71-71) on NON-DMS files page 4-34 for more info on external ind

ENTRY	
U1-U8	The specified external indicator is used to condition DMS file.
Blank	DMS file is not conditioned by external indicator.

DATA MANAGEMENT EXTENSION SPECIFICATIONS

Extension Specifications are used to describe all tables and arrays used by the RPG program.

FIELD DEFINITION

1-2 PAGE

Refer to Section 2 for a complete description.

3-5 LINE

Refer to Section 2 for a complete description.

6 FORM TYPE

This field must contain the letter E.

11-26 FROM AND TO FILENAMES

A DMS file must not be specified for these fields.

27-74 Refer to Section 5 for a complete description.

75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a complete description.

The rules for declaring vectors on Extension Specifications are unchanged. An OCCURS item defined in DASDL must be specified in the Extension Specifications as a vector.

DATA MANAGEMENT INPUT SPECIFICATIONS

Input Specifications describe the records within each disjoint data set or embedded data set to be used as input data for the RPG program.

The entries defined in the Input Specifications for DMS files are divided into the following two sections:

a. Field Definitions for Record Type Descriptions.

b. Field Definitions for Field Descriptions.

FIELD DEFINITIONS FOR RECORD TYPE DESCRIPTIONS

1-2 PAGE

Refer to Section 2 for a complete description.

3-5 LINE

Refer to Section 2 for a complete description.

6 FORM TYPE

This field must contain the letter I.

7-14 FILE NAME

Every DMS file which is described in the File Description Specifications as input or update must be described in the Input Specifications. An Input Specification is required for an embedded data set which is used with a FIND or LOCK operation.

Embedded data sets may only be accessed on a demand basis (for example, with a FIND operation). All embedded data sets described in the Input Specifications must be embedded in a disjoint data set which is described in the File Description Specifications.

15-18 SEQUENCE

Sequencing may be specified for a DMS file that is described in the File Description Specifications as primary or secondary and is accessed by means of an index. The File Description Specification entry for that file, columns 47-52, must be non-blank.

Refer to Section 8 for a complete description of sequencing.

19-20 RECORD IDENTIFYING INDICATOR

A DMS file must not have look-ahead fields or spread records specified in this field.

The DMS exception indicators D1, DA-DH, and DJ-DS must not be entered in this field.

Valid entries for this field are:

Entry	Definition
01-99	Record identifying indicator.
L1-L9	Control level indicator.
LR	Last record indicator.
H0-H9	Halt indicator.

21-41 RECORD IDENTIFYING CODES

Entries in these columns may be made if columns 44-51 of the record line contain a DASDL-defined, non-vector field name.

21-24, 28-31, AND 35-38 POSITION

An entry in these fields for a DMS file specifies the position (in digits for numeric items, in bytes for alphanumeric items) within the item specified in columns 44-51 of the record line, not a position in a record.

The item positions are numbered from left to right, with the left-most digit or byte considered to be position one.

These fields must be blank if the letter S is specified in columns 26, 33, or 40 respectively.

26, 33, AND 40 C/Z/D/S

In addition to the character (C), zone (Z), and digit (D) record identification, these fields may contain the letter S. The letter S specifies signed-numeric record identification. If the letter S is specified, columns 21-24, 28-31, and 35-38 must be blank, columns 27, 34, and 41 must contain a positive sign (+) or negative sign (-), and the DASDL source program must define the field specified in columns 44-51 as a signed-numeric field.

For DMS fields, the character, zone, digit, and sign record identification specify a value within the item specified in columns 44-51 of the record line, not a position in the record.

27, 34, AND 41 CHARACTER

If an entry in columns 26, 33, or 40 is the letter S, then the only valid entries for columns 27, 34, and 41 are:

(+) Declares that a positive sign be used for comparison with the sign position of the numeric field specified in columns 44-51.

(-) Declares that a negative sign be used for comparison with the sign position of the numeric field specified in columns 44-51.

42 STACKER SELECT

This field must be blank for DMS files.

43 PACKED

This field must be blank for DMS files.

44-51 FIELD LOCATION

When an entry is made in this field on a record description line, it specifies which item in the record is used for comparison in setting the record identifying codes.

When AND or OR lines occur for the record description line using record identification codes, columns 44-51 can contain any one of the following entries:

a. A blank, which defaults to the previous entry in columns 44-51.

b. The same item as previously defined.

c. A different DASDL-defined item.

When using record identifying codes, the entry in this field in the record description line must not contain a vector name. The only valid entry is a DASDL-defined item.

52-74 These columns must be blank.

75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a complete description.

FIELD DEFINITIONS FOR FIELD DESCRIPTIONS

1-2 PAGE

Refer to Section 2 for a complete description.

3-5 LINE

Refer to Section 2 for a complete description.

6 FORM TYPE

This field must contain the letter I.

7-43 These fields must be blank for field descriptions.

44-51 FIELD LOCATION

This field specifies the location and data characteristics of the FIELD NAME entered in columns 53-58. A DASDL-defined item must be specified here which effectively "locates" and "types" the entry in the FIELD NAME columns.

If this field is left blank and the FIELD NAME entry contains a DASDL-defined name, the FIELD LOCATION defaults to the same entry.

An entry in this field on a field description line must not be a subscripted vector.

52 DECIMAL POSITIONS

This field must be blank.

53-58 FIELD NAME

If the FIELD LOCATION entry is blank, the FIELD NAME entry must be a DASDL-defined name.

Only those fields which are used in the program need to be specified on the Input Specifications.

Table 13-1 lists the allowable combinations of the FIELD LOCATION and FIELD NAME entries, and the action taken.

59-60 CONTROL LEVEL

These fields must be blank for non-indexed, cycle-driven DMS files.

For all other primary and secondary files the L1-L9 control level indicators are valid.

61-62 MATCHING FIELDS

These fields must be blank for non-indexed, cycle-driven DMS files.

For all other primary and secondary files the M1-M9 matching field indicators are valid.

63-64 FIELD RECORD RELATIONS

The DMS exception indicators, D1, DA-DH, and DJ-DS, must not be entered in this field.

For all other files, the following are valid entries:

Entry	Definition
01-99	Record identifying indicator assigned to a record type.
L1-L9	Control level indicator defined elsewhere.
MR	Matching record indicator.
U1-U8	External indicator defined elsewhere.
H0-H9	Halt indicator defined elsewhere.

65-70 FIELD INDICATORS

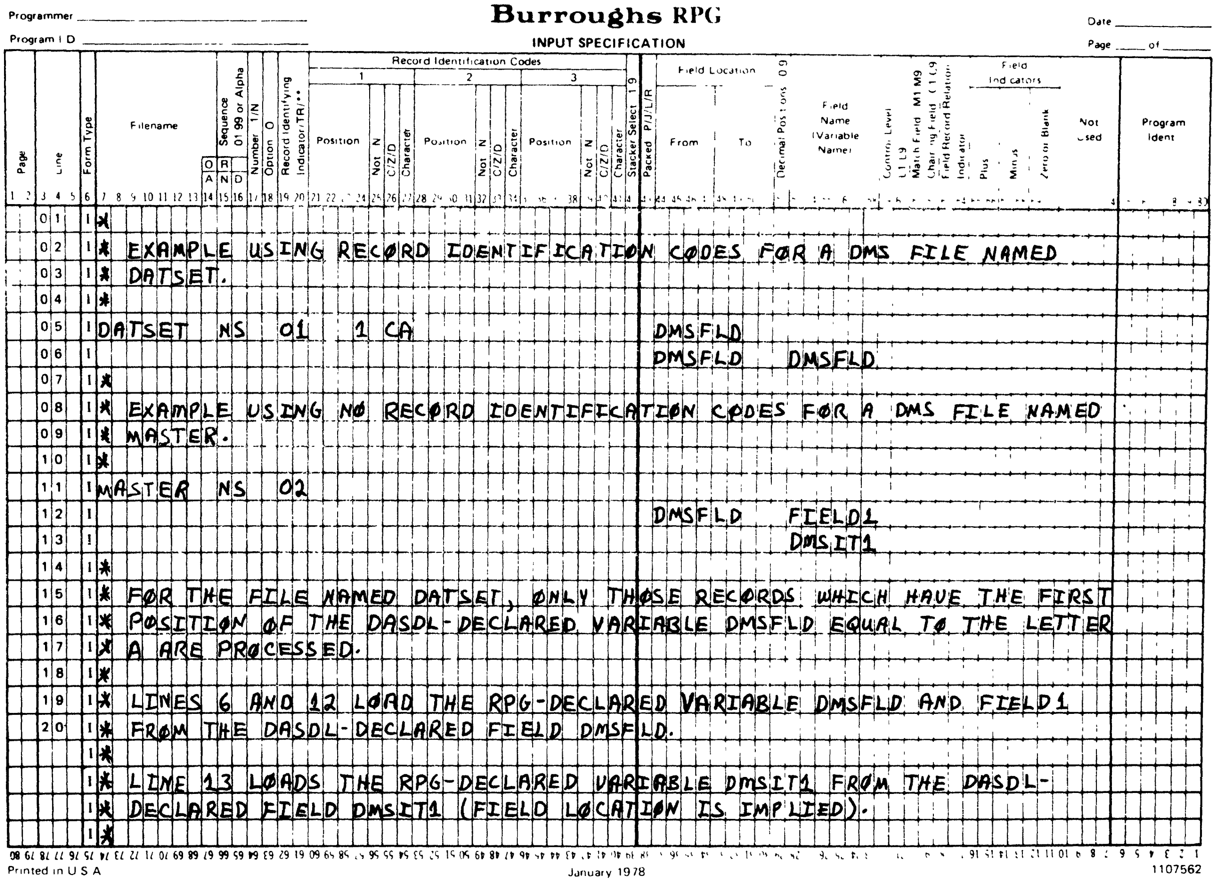
The DMS exception indicators, D1, DA-DH, and DJ-DS must not be entered in this field.

71-74 These columns must be blank.

75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a complete description.

Figure 13-3 illustrates two coding examples for DMS Input Specifications.



Programmer

Program I D

Burroughs RPG

INPUT SPECIFICATION

Date

Page ___ of ___

Page
Line
Form Type
Filename
Sequence
O R
A N D
01 99 or Alpha
Number 1/N
Option O
Record Identifying Indicator/TR/**
Record Identification Codes
1
2
3
Position
Not N
C/Z/D
Character
Stacker Select 1 9
Packed P/J/L/R
Field Location
From
To
Decimal Positions 0 9
Field Name (Variable Name)
Control Level L1 L9
Match Field M1 M9
Chaining Field C1 C9
Field Record Relation
Field Indicators
Plus
Minus
Zero or Blank
Not Used
Program Ident

```
  01 I*
  02 I* EXAMPLE USING RECØRD IDENTIFICATIØN CØDES FØR A DMS FILE NAMED
  03 I* DATSET.
  04 I*
  05 IDATSET  NS  01   1 CA                DMSFLD
  06 I                                     DMSFLD   DMSFLD
  07 I*
  08 I* EXAMPLE USING NØ RECØRD IDENTIFICATIØN CØDES FØR A DMS FILE NAMED
  09 I* MASTER.
  10 I*
  11 IMASTER  NS  02
  12 I                                     DMSFLD   FIELD1
  13 I                                              DMSIT1
  14 I*
  15 I* FØR THE FILE NAMED DATSET, ØNLY THØSE RECØRDS WHICH HAVE THE FIRST
  16 I* PØSITIØN ØF THE DASDL-DECLARED VARIABLE DMSFLD EQUAL TØ THE LETTER
  17 I* A ARE PRØCESSED.
  18 I*
  19 I* LINES 6 AND 12 LØAD THE RPG-DECLARED VARIABLE DMSFLD AND FIELD1
  20 I* FRØM THE DASDL-DECLARED FIELD DMSFLD.
     I*
     I* LINE 13 LØADS THE RPG-DECLARED VARIABLE DMSIT1 FRØM THE DASDL-
     I* DECLARED FIELD DMSIT1 (FIELD LØCATIØN IS IMPLIED).
     I*
```

Printed in U S A

January 1978

1107562

G12071

Figure 13-3. Input Specifications for DMS Files

Table 13-1. Combinations of Various DMS FIELD LOCATION and FIELD NAME Fields in the Input Specifications

FIELD LOCATION Columns 44 - 51	FIELD NAME Columns 53 - 58	Program Action
Vector	Array	Load the whole Array with the contents of the location specified by the whole DASDL-defined vector.
Vector	Array,J	Load Array,J with the contents of the location specified by the DASDL-defined Vector,J.
Vector	Table,J	Load Table,J with the contents of the location specified by the DASDL-defined Vector,J.
Vector	Array,3	Load Array,3 with the contents of the location specified by the DASDL-defined Vector,3.
Vector	Table,3	Load Table,3 with the contents of the location specified by the DASDL-defined Vector,3.
Vector	Table	Not allowed.
Field Name	Array	Not allowed.
Field Name	Array,J	Load Array,J with the contents of the location specified by the DASDL-defined Field Name.
Field Name	Array,3	Load Array,3 with the contents of the location specified by the DASDL-defined Field Name.
Field Name	Table,J	Load Table,J with the contents of the location specified by the DASDL-defined Field Name.
Field Name	Table,3	Load Table,3 with the contents of the location specified by the DASDL-defined Field Name.
Field Name	Field Name	Load to the contents of Field Name specified in columns 53-58 from the location of the DASDL-defined Field Name in columns 44-49.

Table 13-1. Combinations of Various DMS FIELD LOCATION and FIELD NAME Fields in the Input Specifications (Cont)

FIELD LOCATION Columns 44 - 51	FIELD NAME Columns 53 - 58	Program Action
Field Name	Table	Not allowed.
Blank	Array	Load the whole Array with the contents from the location of the DASDL-defined Array.
Blank	Array,J	Load to Array,J with the contents from the location of the DASDL-defined Vector,J.
Blank	Array,3	Load to Array,3 with the contents from the location of the DASDL-defined Vector,3.
Blank	Table	Not allowed.
Blank	Table,J	Load to Table,J with the contents from the location of the DASDL-defined Vector,J.
Blank	Table,3	Load to Table,3 with the contents from the location of the DASDL-defined Vector,3.
Blank	Field Name	Load to Field Name with the contents from the location of the DASDL-defined Field Name.

DATA MANAGEMENT CALCULATION SPECIFICATIONS

The entries defined on the Calculation Specifications for DMS files are described in the following paragraphs.

FIELD DEFINITIONS

1-2 PAGE

Refer to Section 2 for a complete description.

3-5 LINE

Refer to Section 2 for a complete description.

6 FORM TYPE

This field must contain the letter C.

7-8 CONTROL LEVEL

AND/OR lines (AN, OR). When several DMKEY lines must be specified to establish the required relationship of the DMKEY operations for the immediately preceding FIND or LOCK using random access, an AN or OR in columns 7-8 must be used on all DMKEY operations after the first DMKEY of a group. Columns 7-8 must be blank for the first DMKEY operation, except for the optional entry SR when the DMKEY operation is in a subroutine. See figure 13-5 for two examples of coding the DMKEY operation.

9-17 INDICATORS

Enter the conditioning indicators in these fields.

In addition to those indicators specified in Section 9, the following DMS exception indicators may also be used:

Entry	Definition
D1	ON EXCEPTION indicator
DA	NOTFOUND indicator
DB	DUPLICATES indicator
DC	DEADLOCK indicator
DD	DATAERROR indicator
DE	NOTLOCKED indicator
DF	KEYCHANGED indicator

Entry	Definition
DG	SYSTEMERROR indicator
DH	READONLY indicator
DJ	IOERROR indicator
DK	LIMITERROR indicator
DL	OPENERROR indicator
DM	CLOSEERROR indicator
DN	NORECORD indicator
DO	INUSE indicator
DP	AUDITERROR indicator
DQ	ABORT indicator
DR	SECURITYERROR indicator
DS	VERSIONERROR indicator

Refer to the B 1800/B 1700 Systems Data Management System (DMSII) Reference Manual, Form No. 1089794, for a complete description of the DMS exception conditions.

All conditioning indicators on a FIND or LOCK operation with random access are applied to the DMKEY operations immediately following the FIND or LOCK operation. Columns 9-17 must be blank for the DMKEY operation(s) which immediately follow the FIND or LOCK statements with random access.

The D1 indicator is set ON for all DMS exception conditions. In addition, one of the indicators DA-DH, DJ-DS is set ON to indicate the type of DMS exception condition.

18-27 FACTOR 1

When the operation code specified in columns 28-32 is a FIND, LOCK, INSRT, or REMOV operation, the allowable entries for FACTOR 1 are:

a. An index name (retrieval set, automatic subset, ordered set, or manual subset which is defined by DASDL) associated with the data set name specified in columns 33-42, except the REMOV operation.

b. For the FIND or LOCK operations, FACTOR 1 may be left blank when the access desired is not by means of an index.

For all other DMS operations, FACTOR 1 must be blank.

28-32 OPERATION FIELD

Enter the desired operation code in this field. Refer to Section 9 for a description of the available non-DMS operation codes, and refer to the subsection titled Data Management Operation Codes in this section for a description of the DMS operation codes.

The non-DMS operation codes which are described in Section 9 and cannot be specified for DMS files are as follows:

a. CHAIN

b. DEBUG

c. DSPLY

[redacted]

e. READ

f. RECV

g. SEND

h. SETLL

33-42 FACTOR 2

When a DMS operation code other than DMKEY or REMOV is specified in the OPERATION FIELD (columns 28-32), the only allowable entry in FACTOR 2 is a data set name. This data set name must be one of the following:

a. A disjoint data set previously defined on the File Description Specifications.

b. An embedded data set previously defined in DASDL and embedded in a disjoint data set previously defined in the File Description Specifications.

When the DMKEY operation code is specified in the OPERATION FIELD, the allowable entries in FACTOR 2 are a literal or a variable name.

If the operation code is REMOV, FACTOR 2 must be blank.

43-48 RESULT FIELD

The RESULT FIELD is used to specify the selection expression for FIND and LOCK, and specifies the DASDL-defined key name for the DMKEY operation. Valid entries for this field are:

Entry	Definition
Blank	Get the current record.
FIRST	Get the first record of the data set. Used with FIND or LOCK operation codes.
LAST	Get the last record of the data set. Usèd with FIND or LOCK operation codes.
NEXT	Get the next record of the data set. Used with FIND or LOCK operation codes. For index-sequential sets, if there is no current record, get the first record of the data set.
PRIOR	Get the previous record of the data set. Used with FIND or LOCK operation codes.
Key Name	Key name as defined in DASDL. Used with DMKEY statement.

For all other DMS operations, the RESULT FIELD must be blank.

Refer to the B 1800/B 1700 Systems Data Management System (DMSII) Reference Manual, Form No. 1089794, for a functional description of selection expression.

49-51 FIELD LENGTH

This field must be blank when a DMS operation is specified in the OPERATION FIELD.

52 DECIMAL POSITIONS

This field must be blank when a DMS operation is specified in the OPERATION FIELD.

53 HALF ADJUST/ACCESS METHOD

This field is used to specify the access method (random or sequential) for the FIND and LOCK operations.

When FIND or LOCK is specified in the OPERATION FIELD, the following are the only valid entries:

Entry	Definition
R	Random Access
S	Sequential Access

54-59 RESULTING INDICATORS

When a DMS operation other than DMKEY is specified in the OPERATION FIELD, the only valid entries for these columns are the entries D1 or blank. The D1 indicator is the general DMS exception indicator.

Columns 56-59 must be blank for all DMS operations other than DMKEY.

If the DMKEY operation code is specified in the OPERATION FIELD, then columns 54-56 must contain the DMKEY relation. Columns 57-59 must be blank.

If a DMS exception occurs, then:

a. If the D1 indicator is specified in columns 54-55, D1 is set ON (and one of the appropriate exception indicators DA-DH or DJ-DS).

b. If blanks are specified in columns 54-55, the program branches to the exception handling routine for one of the following:

    1. The data set specified in FACTOR 2 for the FIND, LOCK, FREE, TRBEG, TREND, STORE, and DELET operation codes.

    2. The manual subset specified in FACTOR 1 for the INSRT and REMOV operation codes.

c. The program terminates if blanks are specified in columns 54-55 and no exception handling routine exists for either of the following:

    1. The data set specified in FACTOR 2 for the FIND, LOCK, FREE, TRBEG, TREND, STORE, and DELET operation codes.

    2. The manual subset specified in FACTOR 1 for the INSRT and REMOV operation codes.

60-74 These columns must be blank.

75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a complete description.

Setting Indicators

The D1, DA-DH, and DJ-DS exception condition indicators are set OFF immediately prior to a DMS operation and are set ON as a result of a DMS exception condition. The D1 indicator may be used as a resulting indicator of a DMS operation. The DA-DH and DJ-DS indicators must not be used as resulting indicators for a DMS operation.

DATA MANAGEMENT OPERATION CODES

The RPG language provides explicit operation codes to interact with DMS. DMS structures are manipulated by these operations.

A valid data set name is defined as one of the following:

a. A disjoint data set which has been defined in the File Description Specifications as an input or update demand file.

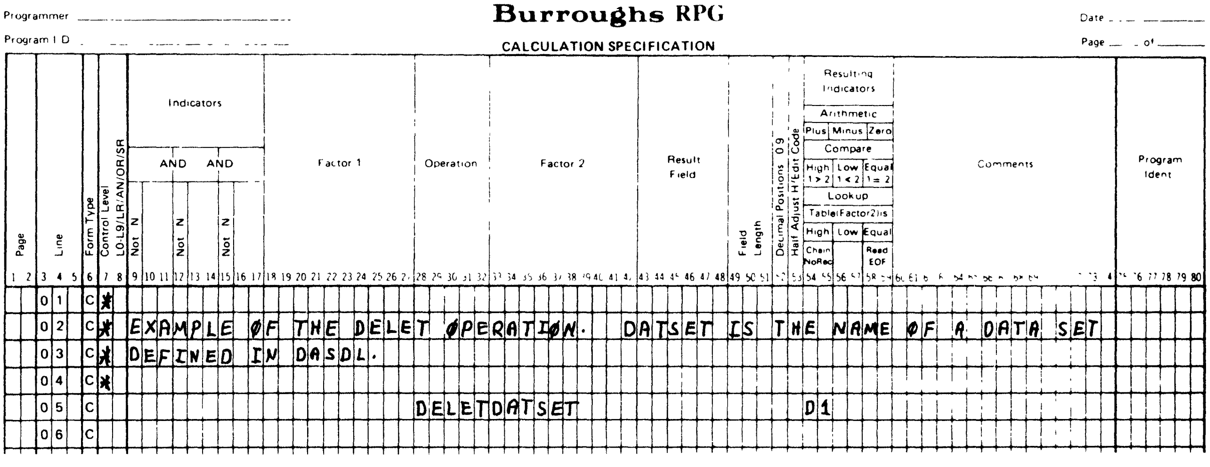
b. An embedded data set which is embedded in a disjoint data set defined in the File Description Specifications.

PROGRAMMED CONTROL OF DATA MANAGEMENT INPUT AND OUTPUT

Within the normal B 1800/B 1700 RPG cycle, a record is read, calculations are performed (using the data from the input record), and an output record is written. The DELET, FIND, FREE, INSRT, LOCK, REMOV, and STORE operations allow greater control over DMS input and output providing the capability to read and write records other than those normally available as part of the RPG program cycle.

DELET

The DELET operation code provides a method of deleting a record from the specified data set. Figure 13-4 shows one method of coding the DELET operation.



Programmer

Program I D

Burroughs RPG

CALCULATION SPECIFICATION

Date

Page of

```
  01C*
  02C* EXAMPLE ØF THE DELET ØPERATIØN. DATSET IS THE NAME ØF A DATA SET
  03C* DEFINED IN DASDL.
  04C*
  05C                  DELETDATSET                 D1
  06C
```

G12072

Figure 13-4. One Method of Coding the DELET Operation

The DELET operation requires entries in the following fields:

a. The DELET operation code in columns 28-32.

b. A valid data set name in FACTOR 2.

c. The D1 indicator or blanks in columns 54-55.

DELET prohibits entries in the following fields:

a. FACTOR 1

b. RESULT FIELD

c. RESULT FIELD LENGTH

d. DECIMAL POSITIONS FIELD

e. HALF ADJUST/ACCESS MODE FIELD

f. LOW and EQUAL RESULTING INDICATORS

Refer to the DELETE operation in the B 1800/B 1700 Systems Data Management System (DMSII) Reference Manual, Form No. 1089794, for a functional description of the DELET operation code.

DMKEY

This operation code provides a method of specifying the conditions which must be satisfied by the record accessed in the immediately previous FIND or LOCK operation using random access. Figure 13-5 shows two methods of coding the DMKEY operation.

The DMKEY operation requires entries in the following fields:

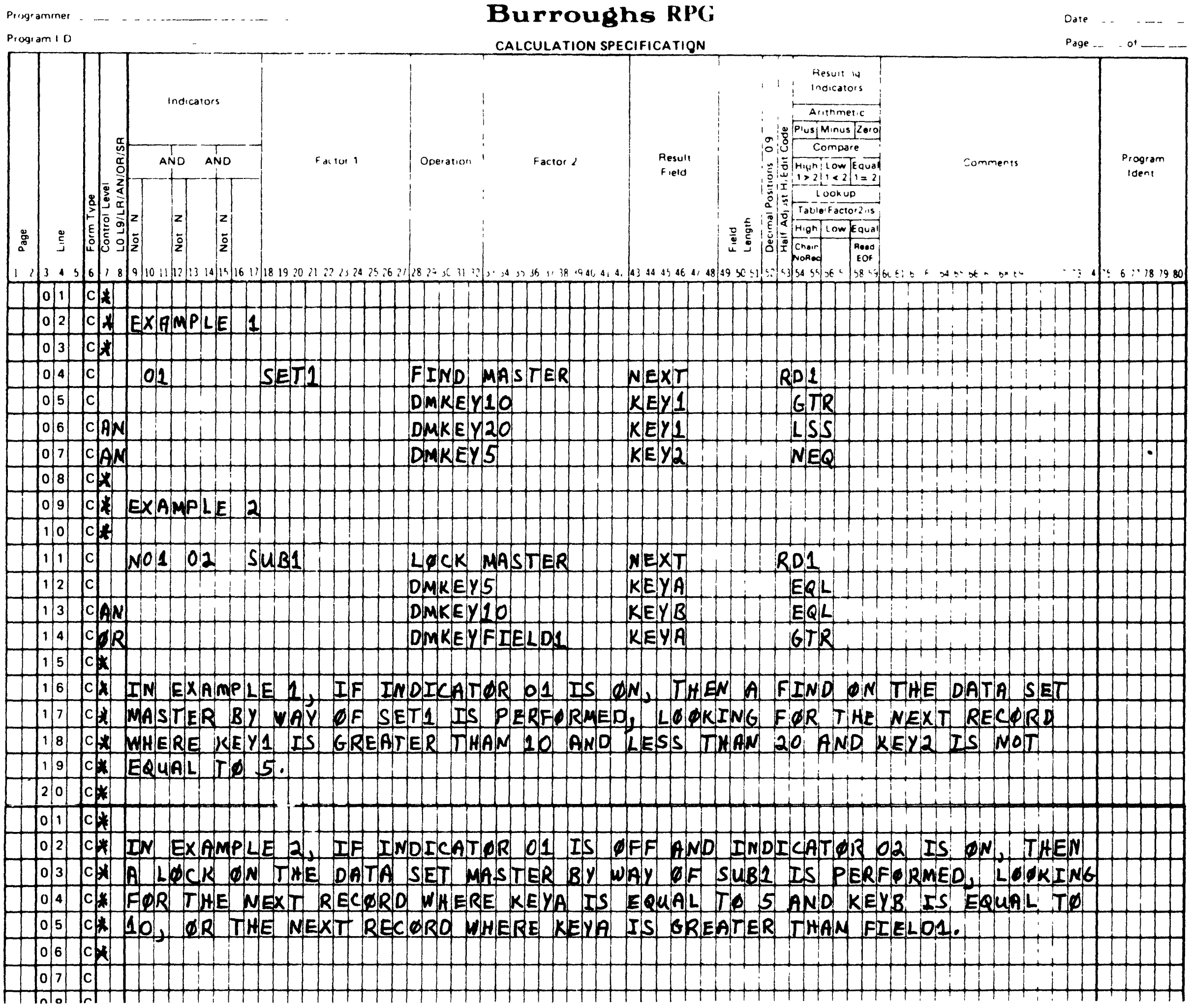
a. DMKEY operation code in columns 28-32.

b. A literal or variable name in FACTOR 2. The literal or variable provides a value for the key condition.

c. The RESULT FIELD contains a key item name. This item name, defined in DASDL, must be a key item of the set or subset name specified in FACTOR 1 of the immediately previous FIND or LOCK operation.

d. The key relationship desired in columns 54-56. Valid entries for this field are:

Entry	Definition
GTR	The item in the RESULT FIELD is greater than the value of the field or literal specified in FACTOR 2.
LSS	The item in the RESULT FIELD is less than the value of the field or literal specified in FACTOR 2.
EQL	The item in the RESULT FIELD is equal to the value of the field or literal specified in FACTOR 2.
GEQ	The item in the RESULT FIELD is greater than or equal to the value of the field or literal specified in FACTOR 2.
LEQ	The item in the RESULT FIELD is less than or equal to the value of the field or literal specified in FACTOR 2.
NEQ	The item in the RESULT FIELD is not equal to the value of the field or literal specified in FACTOR 2.



Programmer
Program I D
Burroughs RPG
CALCULATION SPECIFICATION
Date
Page of
Page
Line
Form Type
Control Level L0 L9/LR/AN/OR/SR
Indicators
AND AND
Not N
Not N
Not N
Factor 1
Operation
Factor 2
Result Field
Field Length
Decimal Positions 0-9
Half Adjust H/Edit Code
Result Indicators
Arithmetic
Plus Minus Zero
Compare
High 1>2 Low 1<2 Equal 1=2
Lookup
Table/Factor2 is
High Low Equal
Chain NoRec
Read EOF
Comments
Program Ident

```
  01 C*
  02 C* EXAMPLE 1
  03 C*
  04 C   01      SET1      FIND MASTER    NEXT      RD1
  05 C                     DMKEY10        KEY1       GTR
  06 CAN                   DMKEY20        KEY1       LSS
  07 CAN                   DMKEY5         KEY2       NEQ
  08 C*
  09 C* EXAMPLE 2
  10 C*
  11 C  NO1 02   SUB1      LØCK MASTER    NEXT      RD1
  12 C                     DMKEY5         KEYA       EQL
  13 CAN                   DMKEY10        KEYB       EQL
  14 CØR                   DMKEYFIELD1    KEYA       GTR
  15 C*
  16 C* IN EXAMPLE 1, IF INDICATØR 01 IS ØN, THEN A FIND ØN THE DATA SET
  17 C* MASTER BY WAY ØF SET1 IS PERFØRMED, LØØKING FØR THE NEXT RECØRD
  18 C* WHERE KEY1 IS GREATER THAN 10 AND LESS THAN 20 AND KEY2 IS NOT
  19 C* EQUAL TØ 5.
  20 C*
  01 C*
  02 C* IN EXAMPLE 2, IF INDICATØR 01 IS ØFF AND INDICATØR 02 IS ØN, THEN
  03 C* A LØCK ØN THE DATA SET MASTER BY WAY ØF SUB1 IS PERFØRMED, LØØKING
  04 C* FØR THE NEXT RECØRD WHERE KEYA IS EQUAL TØ 5 AND KEYB IS EQUAL TØ
  05 C* 10, ØR THE NEXT RECØRD WHERE KEYA IS GREATER THAN FIELD1.
  06 C*
  07 C
  08 C
```

G12073

Figure 13-5. Two Coding Examples of the DMKEY Operation

DMKEY prohibits entries in the following fields:

a. Any conditioning indicator in columns 9-17

b. FACTOR 1

c. RESULT FIELD LENGTH

d. DECIMAL POSITIONS FIELD

e. HALF ADJUST/ACCESS MODE FIELD

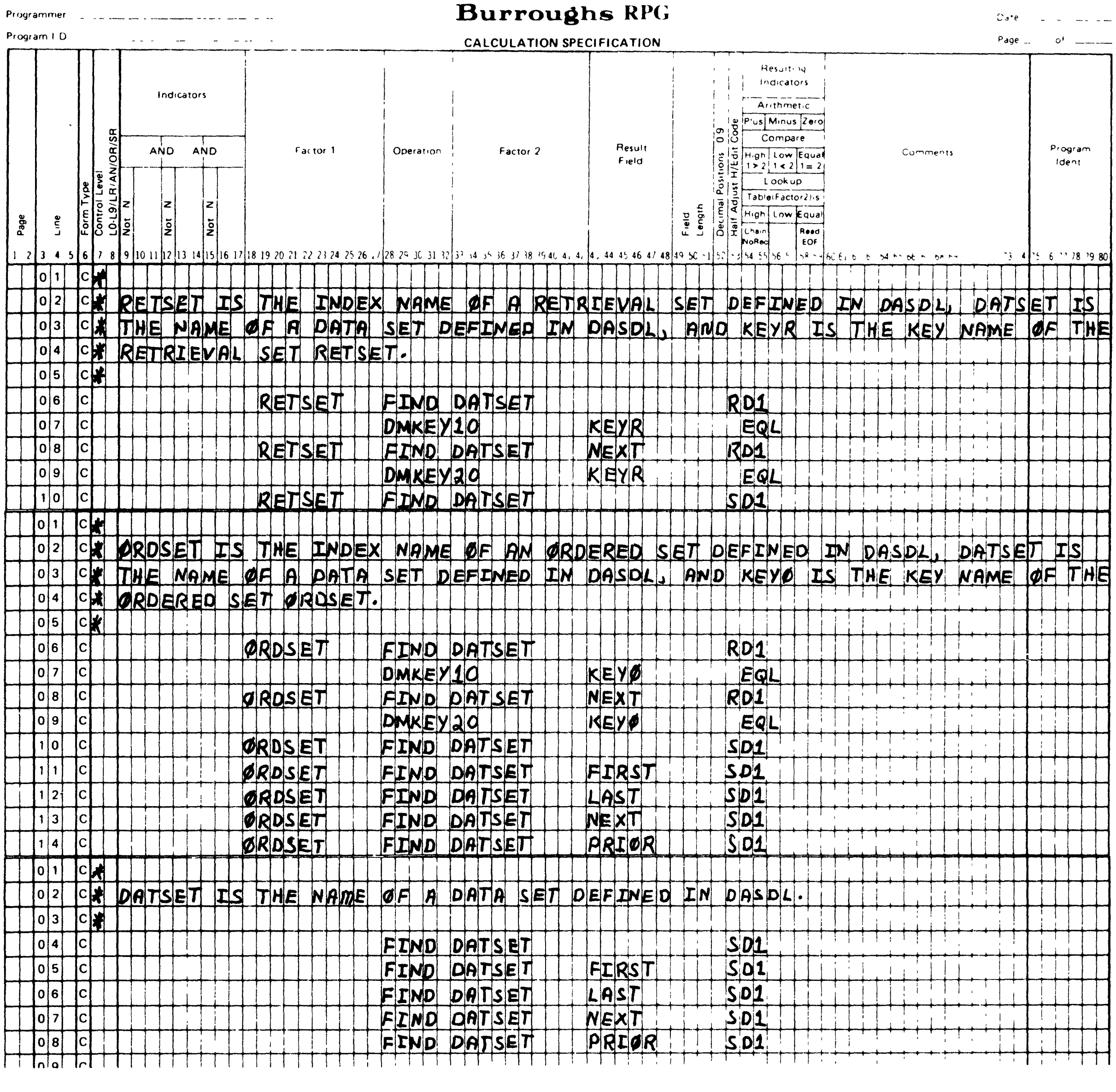
f. Columns 57-59

The DMKEY operation describes the key conditions for the immediately previous FIND or LOCK operation. Random access must be specified for the FIND or LOCK operation by entering the letter R in column 53 of the Calculation Specifications.

There must be at least one key of a set or subset defined for the DMKEY operation with a FIND or LOCK operation when random access is specified. If only some of the keys of a set or subset are defined, then any key relationship for the undefined key(s) is considered acceptable.

FIND

The FIND operation code provides a method of locating a record in a data set and transferring the record to the record area as defined in the Input Specifications. Figure 13-6 shows methods of coding the FIND operation.



Programmer

Program I D

Burroughs RPG

CALCULATION SPECIFICATION

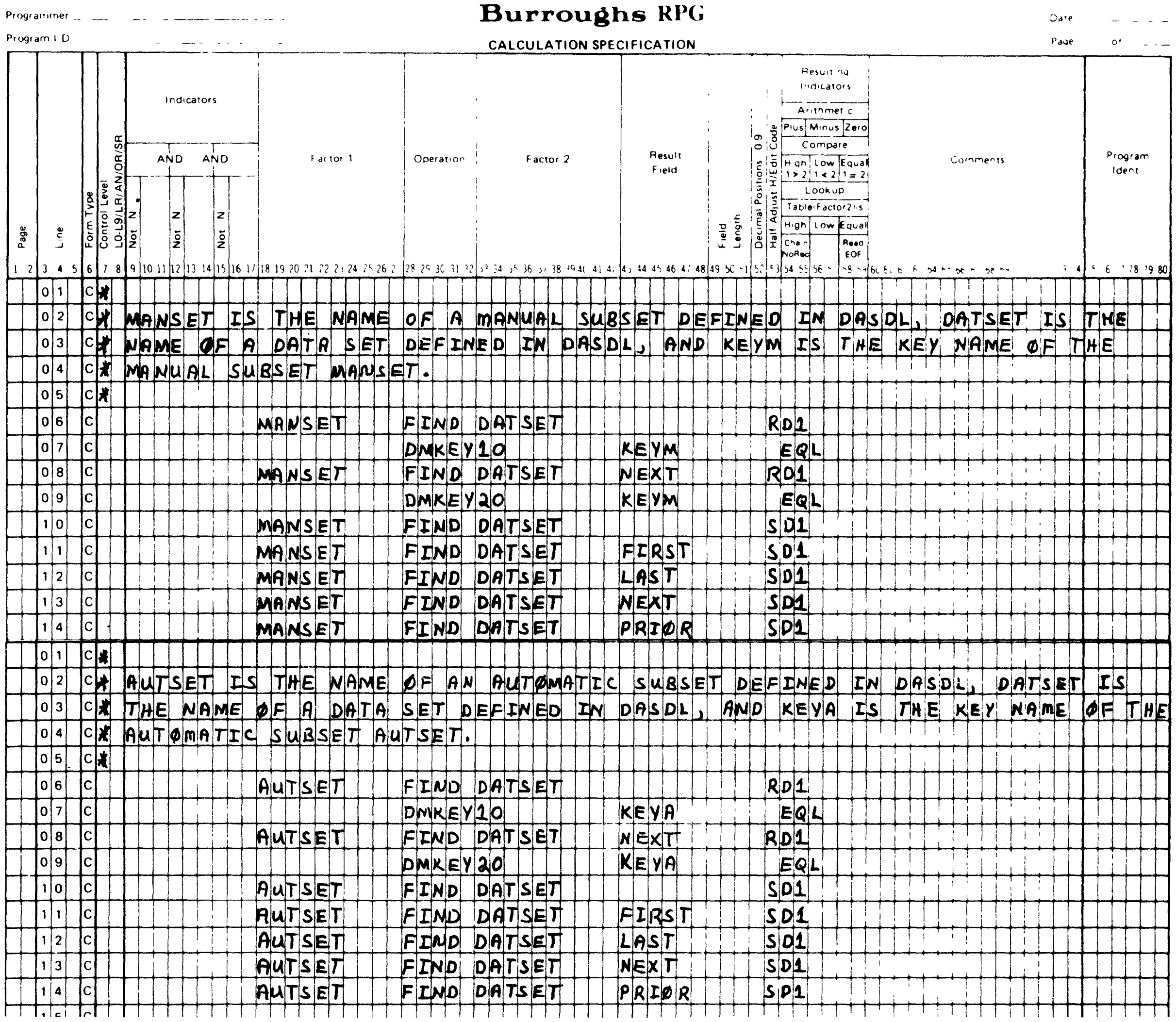
Date

Page of

```
  01 C*
  02 C* RETSET IS THE INDEX NAME ØF A RETRIEVAL SET DEFINED IN DASDL, DATSET IS
  03 C* THE NAME ØF A DATA SET DEFINED IN DASDL, AND KEYR IS THE KEY NAME ØF THE
  04 C* RETRIEVAL SET RETSET.
  05 C*
  06 C           RETSET    FIND DATSET         RD1
  07 C                     DMKEY10     KEYR     EQL
  08 C           RETSET    FIND DATSET NEXT    RD1
  09 C                     DMKEY20     KEYR     EQL
  10 C           RETSET    FIND DATSET         SD1
  01 C*
  02 C* ØRDSET IS THE INDEX NAME ØF AN ØRDERED SET DEFINED IN DASDL, DATSET IS
  03 C* THE NAME ØF A DATA SET DEFINED IN DASDL, AND KEYØ IS THE KEY NAME ØF THE
  04 C* ØRDERED SET ØRDSET.
  05 C*
  06 C          ØRDSET     FIND DATSET         RD1
  07 C                     DMKEY10     KEYØ     EQL
  08 C          ØRDSET     FIND DATSET NEXT    RD1
  09 C                     DMKEY20     KEYØ     EQL
  10 C          ØRDSET     FIND DATSET         SD1
  11 C          ØRDSET     FIND DATSET FIRST   SD1
  12 C          ØRDSET     FIND DATSET LAST    SD1
  13 C          ØRDSET     FIND DATSET NEXT    SD1
  14 C          ØRDSET     FIND DATSET PRIØR   SD1
  01 C*
  02 C* DATSET IS THE NAME ØF A DATA SET DEFINED IN DASDL.
  03 C*
  04 C                     FIND DATSET         SD1
  05 C                     FIND DATSET FIRST   SD1
  06 C                     FIND DATSET LAST    SD1
  07 C                     FIND DATSET NEXT    SD1
  08 C                     FIND DATSET PRIØR   SD1
  09 C
```

G12074/SHEET 1 OF 2

Figure 13-6. Methods of Coding the FIND Operation (Sheet 1 of 2)



Programmer

Program I D

Burroughs RPG

CALCULATION SPECIFICATION

Date

Page of

```
  01 C*
  02 C* MANSET IS THE NAME OF A MANUAL SUBSET DEFINED IN DASDL, DATSET IS THE
  03 C* NAME ØF A DATA SET DEFINED IN DASDL, AND KEYM IS THE KEY NAME ØF THE
  04 C* MANUAL SUBSET MANSET.
  05 C*
  06 C           MANSET    FIND DATSET              RD1
  07 C                     DMKEY10   KEYM            EQL
  08 C           MANSET    FIND DATSET    NEXT      RD1
  09 C                     DMKEY20   KEYM            EQL
  10 C           MANSET    FIND DATSET              SD1
  11 C           MANSET    FIND DATSET    FIRST     SD1
  12 C           MANSET    FIND DATSET    LAST      SD1
  13 C           MANSET    FIND DATSET    NEXT      SD1
  14 C           MANSET    FIND DATSET    PRIØR     SD1
  01 C*
  02 C* AUTSET IS THE NAME ØF AN AUTØMATIC SUBSET DEFINED IN DASDL, DATSET IS
  03 C* THE NAME ØF A DATA SET DEFINED IN DASDL, AND KEYA IS THE KEY NAME ØF THE
  04 C* AUTØMATIC SUBSET AUTSET.
  05 C*
  06 C           AUTSET    FIND DATSET              RD1
  07 C                     DMKEY10   KEYA            EQL
  08 C           AUTSET    FIND DATSET    NEXT      RD1
  09 C                     DMKEY20   KEYA            EQL
  10 C           AUTSET    FIND DATSET              SD1
  11 C           AUTSET    FIND DATSET    FIRST     SD1
  12 C           AUTSET    FIND DATSET    LAST      SD1
  13 C           AUTSET    FIND DATSET    NEXT      SD1
  14 C           AUTSET    FIND DATSET    PRIØR     SP1
```

G12074/SHEET 2 OF 2

Figure 13-6. Methods of Coding the FIND Operation (Sheet 2 of 2)

The FIND operation requires entries in the following fields:

a. FIND operation code in columns 28-32.

b. A valid data set name in FACTOR 2.

c. The letter R or S in column 53. The letter R specifies random access of the data set. The letter S specifies sequential access of the data set.

d. The D1 indicator or blanks in columns 54-55.

The FIND operation may also use the following entries:

a. The contents of FACTOR 1 may specify the name of a retrieval set, ordered set, automatic subset, or manual subset. This index name must have been previously defined in DASDL.

b. The RESULT FIELD may contain a selection expression. This selection expression specifies the record of a data set which is to be processed, relative to the last record accessed.

A FIND operation code with random access (the letter R in column 53 of the Calculation Specifications) must be immediately followed by appropriate DMKEY operation(s) to define the key condition(s) which must be satisfied.

The FIND operation prohibits entries in the following fields:

a. RESULT FIELD LENGTH

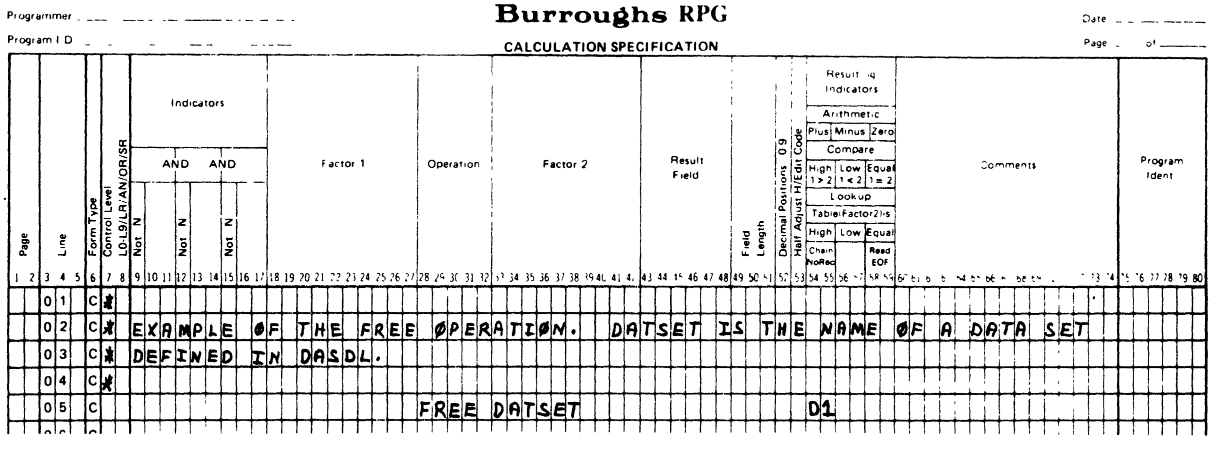
b. DECIMAL POSITIONS FIELD

c. LOW and EQUAL RESULTING INDICATORS

Refer to the FIND operation in the B 1800/B 1700 Systems Data Management System (DMSII) Reference Manual, Form No. 1089794, for a functional description of the FIND operation code.

FREE

The FREE operation code unlocks the current locked record for this program and this data set. Figure 13-7 shows one method of coding the FREE operation.



Burroughs RPG
CALCULATION SPECIFICATION

Programmer
Program I D
Date
Page of

Page
Line
Form Type
Control Level
L0-L9/LR/AN/OR/SR
Indicators
AND
AND
Not
Not
Not
Factor 1
Operation
Factor 2
Result Field
Field Length
Decimal Positions 0 9
Half Adjust H/Edit Code
Resulting Indicators
Arithmetic
Plus
Minus
Zero
Compare
High 1 > 2
Low 1 < 2
Equal 1 = 2
Lookup
Table(Factor2)is
High
Low
Equal
Chain NoRec
Read EOF
Comments
Program Ident

```
  01 C*
  02 C* EXAMPLE OF THE FREE OPERATION.  DATSET IS THE NAME OF A DATA SET
  03 C* DEFINED IN DASDL.
  04 C*
  05 C                    FREE DATSET                 D1
```

G12075

Figure 13-7. One Method of Coding the FREE Operation

The FREE operation requires entries in the following fields:

a. FREE entered in columns 28-32.

b. A valid data set name in FACTOR 2.

c. D1 exception indicator or blanks in columns 54-55.

The FREE operation prohibits entries in the following fields:

a. FACTOR 1

b. RESULT FIELD

c. RESULT FIELD LENGTH

d. DECIMAL POSITIONS FIELD

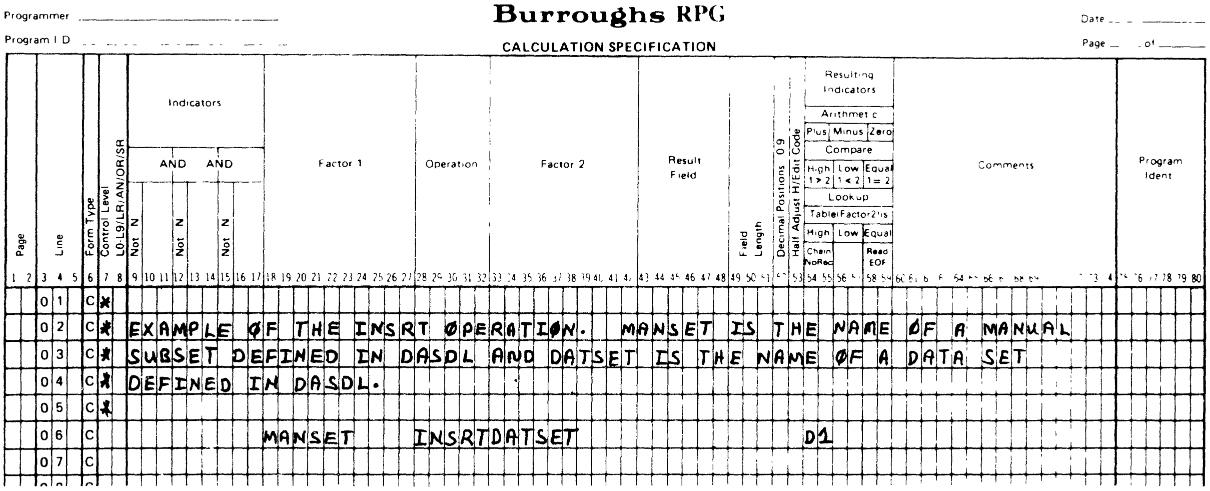
e. HALF ADJUST/ACCESS MODE FIELD

f. LOW and EQUAL RESULTING INDICATORS

Refer to the FREE operation in the B 1800/B 1700 Systems Data Management System (DMSII) Reference Manual, Form No. 1089794, for a functional description of the FREE operation code.

INSRT

The INSRT operation code specifies the insertion of a record from a data set into a manual subset. Figure 13-8 shows one method of coding the INSRT operation.



Programmer ____ Program I D ____

Burroughs RPG

CALCULATION SPECIFICATION

Date ____ Page ____ of ____

```
  01 C*
  02 C* EXAMPLE ØF THE INSRT ØPERATIØN.  MANSET IS THE NAME ØF A MANUAL
  03 C* SUBSET DEFINED IN DASDL AND DATSET IS THE NAME ØF A DATA SET
  04 C* DEFINED IN DASDL.
  05 C*
  06 C           MANSET    INSRTDATSET               01
  07 C
```

G12076

Figure 13-8. One Method of Coding the INSRT Operation

The INSRT operation requires entries in the following fields:

a. The manual subset name in FACTOR 1. This manual subset must have been previously defined by DASDL, and must be one of the following:

   1. An item of a disjoint data set defined on the File Description Specifications.

   2. An item of an embedded data set which is embedded in a disjoint data set defined on the File Description Specifications.

b. INSRT entered in columns 28-32.

c. A data set name in FACTOR 2. This data set name must be a disjoint data set which has been defined on the File Description Specifications.

d. The D1 exception indicator or blanks in columns 54-55.

The INSRT operation prohibits entries in the following fields:

a. RESULT FIELD

b. RESULT FIELD LENGTH

c. DECIMAL POSITIONS FIELD

d. HALF ADJUST/ACCESS MODE FIELD

e. LOW and EQUAL RESULTING INDICATORS

The data set name must be the declared source of records for a manual subset. For example, the manual subset name S1 must be a manual subset of the data set D, as the following example illustrates.

Example:

```
     DASDL:   S1 SUBSET OF D
     RPG:     S1         INSRT        D
         (FACTOR 1)  (OPERATION)  (FACTOR 2)
```

Refer to the INSERT operation in the B 1800/B 1700 Systems Data Management System (DMSII) Reference Manual, Form No. 1089794, for a functional description of the INSRT operation code.

LOCK

The LOCK operation code locates a record in the specified data set, transfers the data to the record area as described in the Input Specifications, and locks the record to prevent concurrent modifications by another user. If the record is not found, a NOTFOUND exception condition results. Figure 13-9 shows methods of coding the LOCK operation.

The LOCK operation requires entries in the following fields:

a. LOCK entered in columns 28-32.

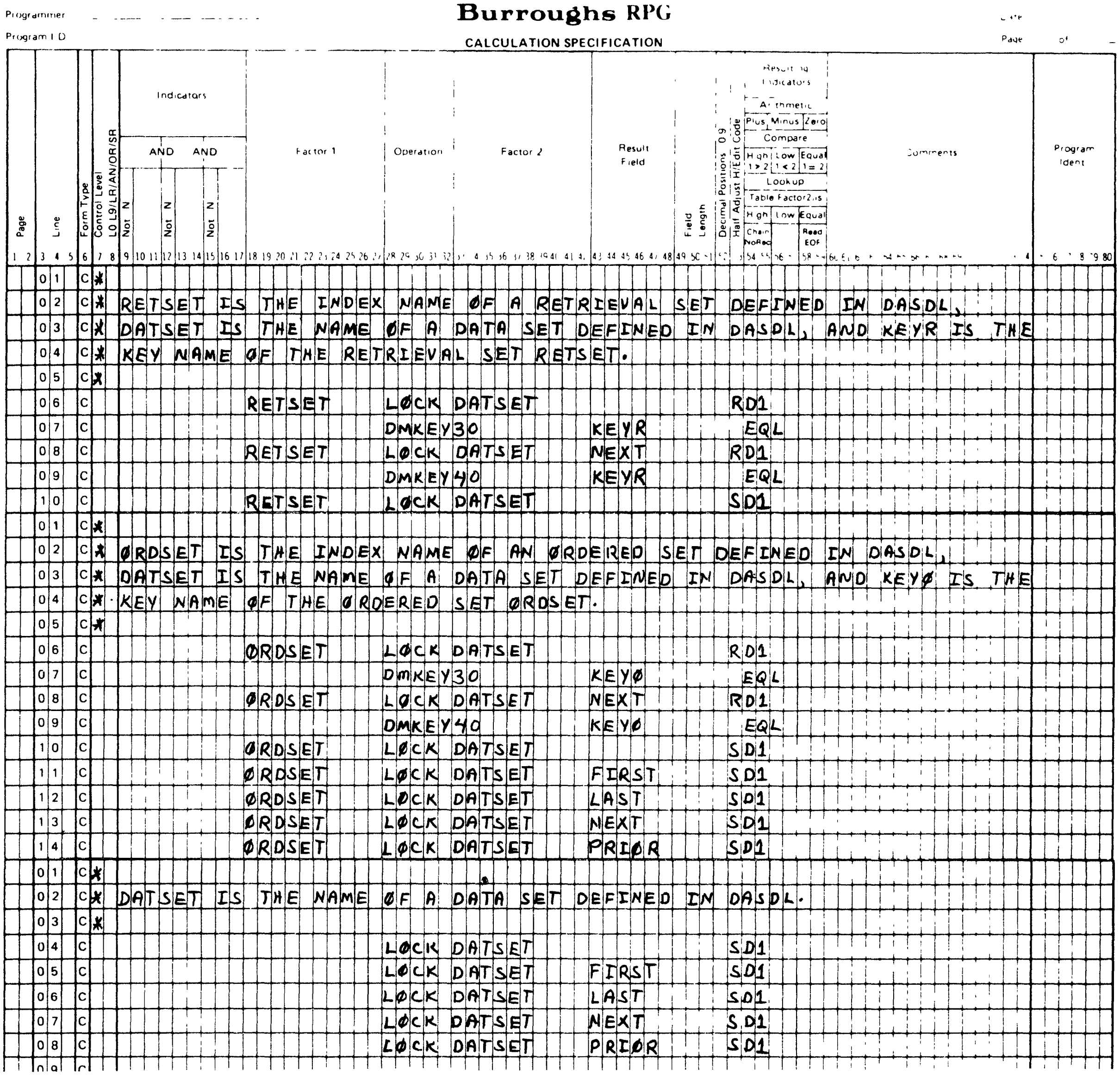
b. A valid data set name in FACTOR 2.

c. The letter R or S in column 53. If the letter R is specified, the data set is processed randomly. If the letter S is specified, the data set is processed sequentially.

d. The D1 exception indicator or blanks in columns 54-55.

In addition to the preceding required entries, the LOCK operation may use the following optional entries:

a. FACTOR 1 to specify the name of a retrieval set, ordered set, automatic subset, or manual subset. The index name must have been previously defined in DASDL.


Programmer
Program I D
Burroughs RPG
CALCULATION SPECIFICATION
Page
of
Page
Line
Form Type
Control Level L0 L9/LR/AN/OR/SR
Indicators
AND
AND
Not
Not
Not
Factor 1
Operation
Factor 2
Result Field
Field Length
Decimal Positions 0-9
Half Adjust H/Edit Code
Resulting Indicators
Arithmetic
Plus
Minus
Zero
Compare
High 1>2
Low 1<2
Equal 1=2
Lookup
Table Factor2 is
High
Low
Equal
Chain NoRec
Read EOF
Comments
Program Ident

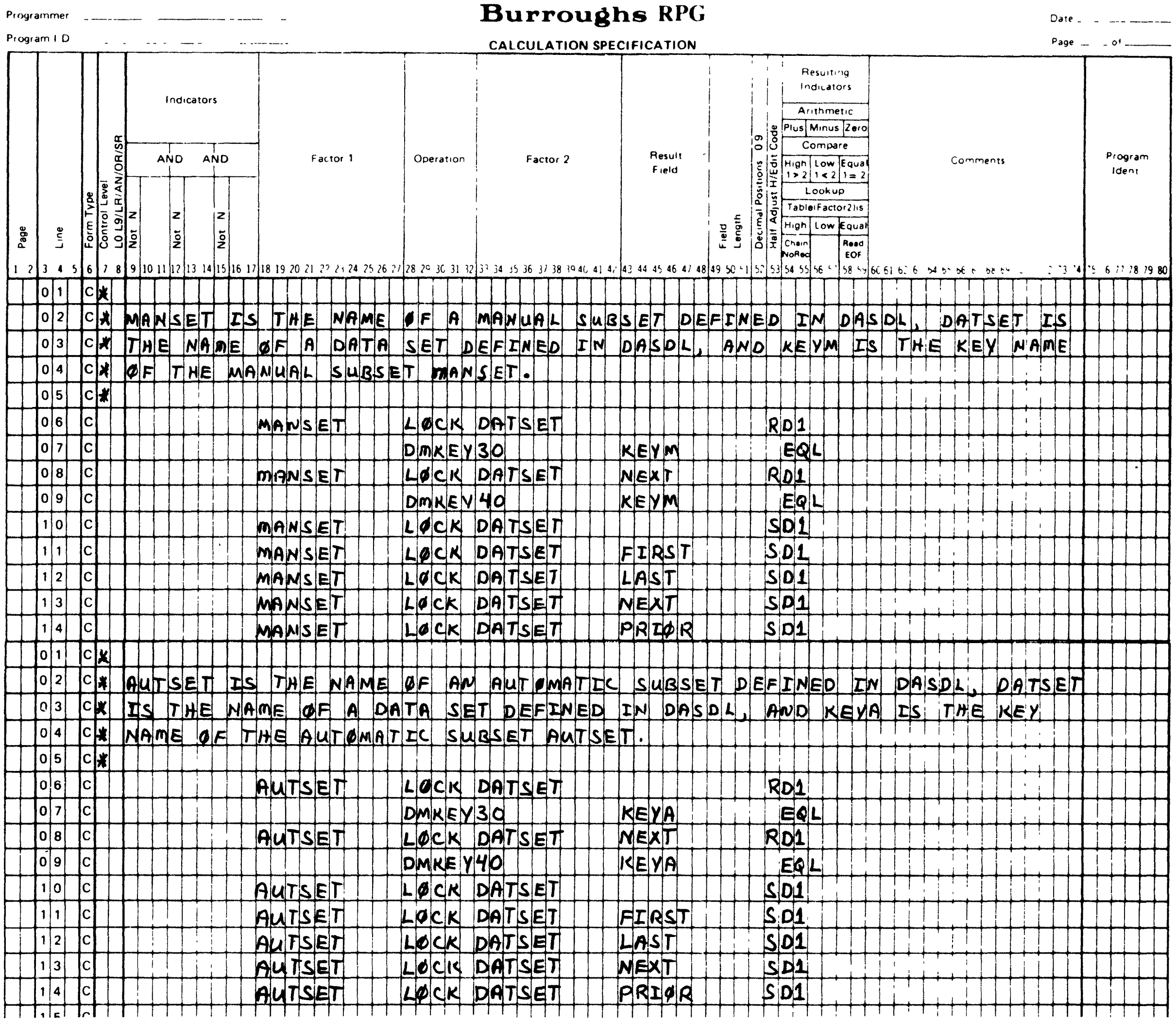
```
01 C*
02 C* RETSET IS THE INDEX NAME ØF A RETRIEVAL SET DEFINED IN DASDL,
03 C* DATSET IS THE NAME ØF A DATA SET DEFINED IN DASDL, AND KEYR IS THE
04 C* KEY NAME ØF THE RETRIEVAL SET RETSET.
05 C*
06 C           RETSET    LØCK DATSET               RD1
07 C                     DMKEY30        KEYR         EQL
08 C           RETSET    LØCK DATSET    NEXT       RD1
09 C                     DMKEY40        KEYR         EQL
10 C           RETSET    LØCK DATSET               SD1
01 C*
02 C* ØRDSET IS THE INDEX NAME ØF AN ØRDERED SET DEFINED IN DASDL,
03 C* DATSET IS THE NAME ØF A DATA SET DEFINED IN DASDL, AND KEYØ IS THE
04 C* KEY NAME ØF THE ØRDERED SET ØRDSET.
05 C*
06 C           ØRDSET    LØCK DATSET               RD1
07 C                     DMKEY30        KEYØ         EQL
08 C           ØRDSET    LØCK DATSET    NEXT       RD1
09 C                     DMKEY40        KEYØ         EQL
10 C           ØRDSET    LØCK DATSET               SD1
11 C           ØRDSET    LØCK DATSET    FIRST      SD1
12 C           ØRDSET    LØCK DATSET    LAST       SD1
13 C           ØRDSET    LØCK DATSET    NEXT       SD1
14 C           ØRDSET    LØCK DATSET    PRIØR      SD1
01 C*
02 C* DATSET IS THE NAME ØF A DATA SET DEFINED IN DASDL.
03 C*
04 C                     LØCK DATSET               SD1
05 C                     LØCK DATSET    FIRST      SD1
06 C                     LØCK DATSET    LAST       SD1
07 C                     LØCK DATSET    NEXT       SD1
08 C                     LØCK DATSET    PRIØR      SD1
09 C
```

G12077/SHEET 1 OF 2

Figure 13-9. Methods of Coding the LOCK Operation (Sheet 1 of 2)

b. The RESULT FIELD may contain a selection expression. This selection expression specifies the record of the data set which is to be processed, relative to the last record accessed.

A LOCK operation code with random access (the letter R in column 53 of the Calculation Specifications) must be immediately followed by appropriate DMKEY operation(s) to define the key condition(s) which must be satisfied.



Programmer

Program I D

Burroughs RPG

CALCULATION SPECIFICATION

Date

Page of

Line	Form Type	Control Level	Factor 1	Operation	Factor 2	Result Field	Resulting Indicators
01	C	*					
02	C	*	MANSET IS THE NAME ØF A MANUAL SUBSET DEFINED IN DASDL, DATSET IS				
03	C	*	THE NAME ØF A DATA SET DEFINED IN DASDL, AND KEYM IS THE KEY NAME				
04	C	*	ØF THE MANUAL SUBSET MANSET.				
05	C	*					
06	C		MANSET	LØCK	DATSET		RD1
07	C			DMKEY	30	KEYM	EQL
08	C		MANSET	LØCK	DATSET	NEXT	RD1
09	C			DMKEY	40	KEYM	EQL
10	C		MANSET	LØCK	DATSET		SD1
11	C		MANSET	LØCK	DATSET	FIRST	SD1
12	C		MANSET	LØCK	DATSET	LAST	SD1
13	C		MANSET	LØCK	DATSET	NEXT	SD1
14	C		MANSET	LØCK	DATSET	PRIØR	SD1
01	C	*					
02	C	*	AUTSET IS THE NAME ØF AN AUTØMATIC SUBSET DEFINED IN DASDL, DATSET				
03	C	*	IS THE NAME ØF A DATA SET DEFINED IN DASDL, AND KEYA IS THE KEY				
04	C	*	NAME ØF THE AUTØMATIC SUBSET AUTSET.				
05	C	*					
06	C		AUTSET	LØCK	DATSET		RD1
07	C			DMKEY	30	KEYA	EQL
08	C		AUTSET	LØCK	DATSET	NEXT	RD1
09	C			DMKEY	40	KEYA	EQL
10	C		AUTSET	LØCK	DATSET		SD1
11	C		AUTSET	LØCK	DATSET	FIRST	SD1
12	C		AUTSET	LØCK	DATSET	LAST	SD1
13	C		AUTSET	LØCK	DATSET	NEXT	SD1
14	C		AUTSET	LØCK	DATSET	PRIØR	SD1
15	C						

G12077/SHEET 2 OF 2

Figure 13-9. Methods of Coding the LOCK Operation (Sheet 2 of 2)

The LOCK operation prohibits entries in the following fields:

a. RESULT FIELD LENGTH

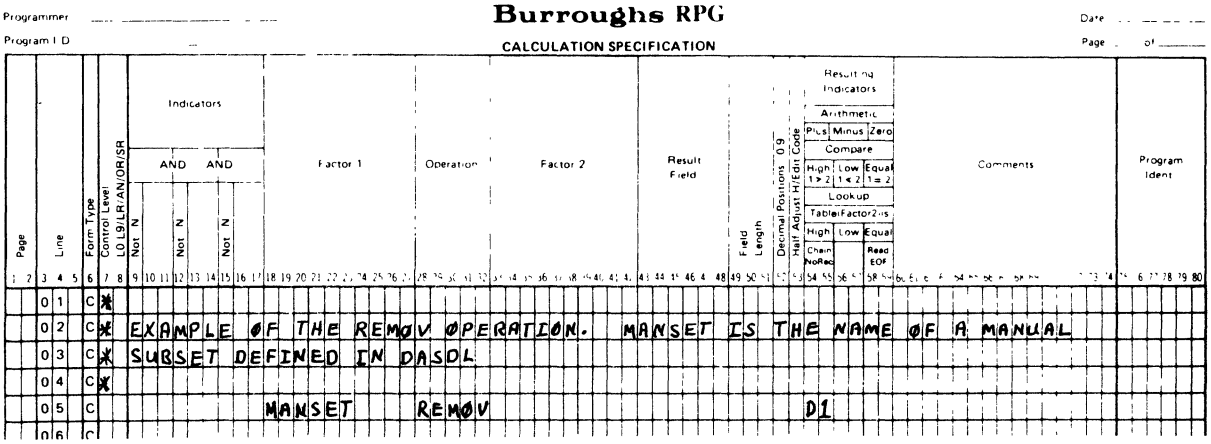
b. DECIMAL POSITIONS

c. LOW and EQUAL RESULTING INDICATORS

Refer to the MODIFY operation in the B 1800/B 1700 Systems Data Management System (DMSII) Reference Manual, Form No. 1089794, for a functional description of the LOCK operation code.

REMOV

The REMOV operation removes the current record from a manual subset. The record is not removed from the data set. Figure 13-10 shows one method of coding the REMOV operation.



Programmer

Program I D

Burroughs RPG

CALCULATION SPECIFICATION

Date

Page of

```
01 C*
02 C* EXAMPLE ØF THE REMØV ØPERATIØN.  MANSET IS THE NAME ØF A MANUAL
03 C* SUBSET DEFINED IN DASDL
04 C*
05 C          MANSET    REMØV                    D1
06 C
```

G12078

Figure 13-10. One Method of Coding the REMOV Operation

The REMOV operation requires entries in the following fields:

a. A manual subset name in FACTOR 1. The manual subset must be one of the following:

1. An item of a disjoint data set defined on the File Description Specifications.

2. An item of an embedded data set which is embedded in a disjoint data set defined on the File Description Specifications.

b. REMOV entered in columns 28-32.

c. The D1 indicator or blanks in columns 54-55.

The REMOV operation prohibits entries in the following fields:

a. FACTOR 2

b. RESULT FIELD

c. RESULT FIELD LENGTH

d. DECIMAL POSITIONS

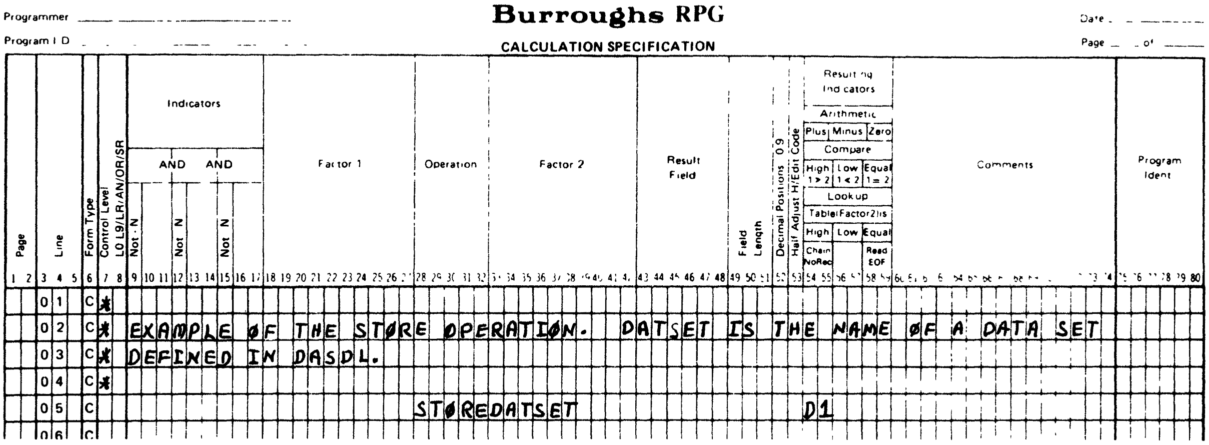
e. HALF ADJUST/ACCESS MODE FIELD

f. LOW and EQUAL RESULTING INDICATORS

Refer to the REMOVE operation in the B 1800/B 1700 Systems Data Management System (DMSII) Reference Manual, Form No. 1089794, for a functional description of the REMOV operation code.

STORE

The STORE operation provides a method of updating a record or adding a new record to a data set. Figure 13-11 shows one method of coding the STORE operation code.



Burroughs RPG

CALCULATION SPECIFICATION

```
  01 C*
  02 C* EXAMPLE ØF THE STØRE ØPERATIØN. DATSET IS THE NAME ØF A DATA SET
  03 C* DEFINED IN DASDL.
  04 C*
  05 C                     STØREDATSET               D1
  06 C
```

G12079

Figure 13-11. One Method of Coding the STORE Operation Code

The STORE operation requires entries in the following fields:

a. STORE entered in columns 28-32.

b. A valid data set name in FACTOR 2.

c. The D1 exception indicator or blanks in columns 54-55.

The STORE operation prohibits entries in the following fields:

a. FACTOR 1

b. RESULT FIELD

c. RESULT FIELD LENGTH

d. DECIMAL POSITIONS

e. HALF ADJUST/ACCESS MODE FIELD

f. LOW and EQUAL RESULTING INDICATORS

The STORE operation uses Output-Format Specifications which contain:

a. The data set name in columns 7-14.

b. The letter S in column 15.

Refer to the STORE operation in the B 1800/B 1700 Systems Data Management System (DMSII) Reference Manual, Form No. 1089794, for a functional description of the STORE operation code.

DATA MANAGEMENT EXCEPTION HANDLING

An RPG program which uses DMS operations may encounter any one of many DMS exception conditions which prevent the operation from being performed as specified. If an exception condition occurs, the RPG program terminates unless one of the following conditions exists:

a. Columns 54-55 of the Calculation Specifications for a DMS operation other than DMKEY contain the D1 indicator for the DMS operation which caused the exception condition.

b. An exception handling routine exists in the Calculation Specifications for the cycle-driven file, demand file, or manual subset for which the exception condition occurred.

INPUT EXCEPTIONS FOR CYCLE-DRIVEN DATA MANAGEMENT FILES

If a cycle-driven DMS file is used and one of the following exception conditions occurs, the operator must respond appropriately. The exception conditions listed below result in the following run-time diagnostics for cycle-driven DMS files:

a. DEADLOCK exception gives the operator the option to enter:

    1. <program job number>AXRETRY. Attempt the same operation again.

    2. <program job number>AXSTOP. Program performs an orderly termination.

b. IOERROR exception gives the operator the option to enter:

    1. <program job number>AXRETRY. Attempt the same operation again.

    2. <program job number>AXSTOP. Program performs an orderly termination.

c. SECURITYERROR exception only allows the operator to enter:

    1. <program job number>AXSTOP. Program performs an orderly termination.

d. VERSIONERROR exception only allows the operator to enter:

    1. <program job number>AXSTOP. Program performs an orderly termination.

A NOTFOUND exception condition is treated as an end of file for cycle-driven DMS files.

ALL OTHER EXCEPTION CONDITIONS

An exception handling routine may be coded for one of the following:

a. Those exceptions which occur as a result of an explicit data management operation other than DMKEY which does not have the indicator D1 entered in columns 54-55 for that operation.

b. Those exceptions which may occur for primary or secondary DMS files exclusive of the exceptions previously listed under Input Exceptions for Cycle-Driven Data Management Files.

The exception handling routine operation codes are used only to delimit the beginning and end of an exception handling routine. Calculation Specification lines within an exception handling routine must contain the entries UR, OR, AN, or blank in columns 7-8, and all exception handling routines must be specified immediately before all Output-Format Specifications. Exception handling routines must not be nested in the RPG program.

EXCEPTION HANDLING OPERATION CODES

The exception handling operation codes are described in the following paragraphs.

BEGUR

This operation code indicates the start of an exception handling routine. FACTOR 2 must contain the name of the disjoint data set, embedded data set, or manual subset for which the exception handling routine is applicable. For the BEGUR operation, columns 9-27 and 43-59 must be blank.

ENDUR

This operation code indicates the end of an exception handling routine. FACTOR 1 may contain a label. This label is used as a tag, thus allowing exits from different points within the exception handling routine. Columns 9-27 and 33-59 must be blank for this exception handling operation code.

The following rules must be followed when coding an exception handling routine:

a. Exactly one exception handling routine may be defined for:

1. Each disjoint data set defined in the File Description Specifications.

2. Each embedded data set which is embedded within a disjoint data set defined in the File Description Specifications.

3. Each manual subset which is an item of a disjoint data set or an item of an embedded data set.

b. The following operation codes must not appear within an exception handling routine:

   1. CHAIN
   2. DELET
   3. EXCPT
   4. EXSR
   5. FIND
   6. FORCE
   7. FREE
   8. INSRT
   9. LOCK
   10. READ
   11. RECV
   12. REMOV
   13. SEND
   14. STORE
   15. TRBEG
   16. TREND

c. Branching into or out of a given exception handling routine is not allowed (for example, a branch such as a GOTO or EXSR operation).

Additionally, the following must be noted:

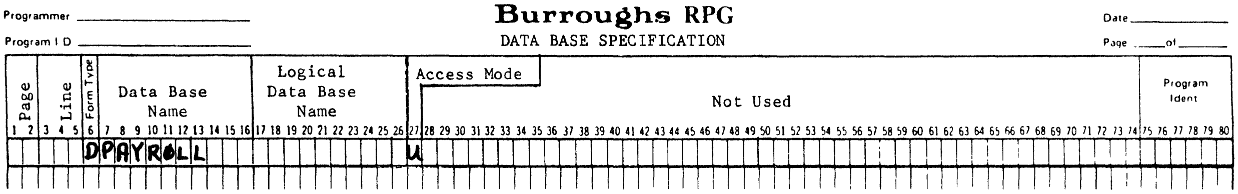
a. The program aborts if an exception condition occurs and no exception handling routine is defined for at least one of the following:

   1. A cycle-driven DMS file.
   2. A demand DMS file which does not have the D1 indicator coded in columns 54-55 of the Calculation Specifications for a DMS operation code other than DMKEY.
   3. A manual subset which does not have the D1 indicator coded in columns 54-55 of the Calculation Specifications for the DMS operations INSRT or REMOV.

b. If the exception condition occurs during detail or total output, the operation continues with the next detail or total output line after exiting the exception handling routine.

If an exception condition occurs as a result of an EXCPT or STORE operation code, the operation continues with the next output line which contains the letter E (if the operation was an EXCEPT) or the letter S (if the operation was a STORE) in column 15 of the Output-Format Specification or the next calculation operation in the Calculation Specifications after exiting the exception handling routine.

Figures 13-12 and 13-13 illustrate a method of retrying an output operation when an exception condition is reported for a DMS operation. Figure 13-12 shows a method of retrying when cycle-driven DMS files are specified. Figure 13-13 shows a method of retrying the output operation following an exception condition when the EXCPT operation code is specified in the Calculation Specifications.

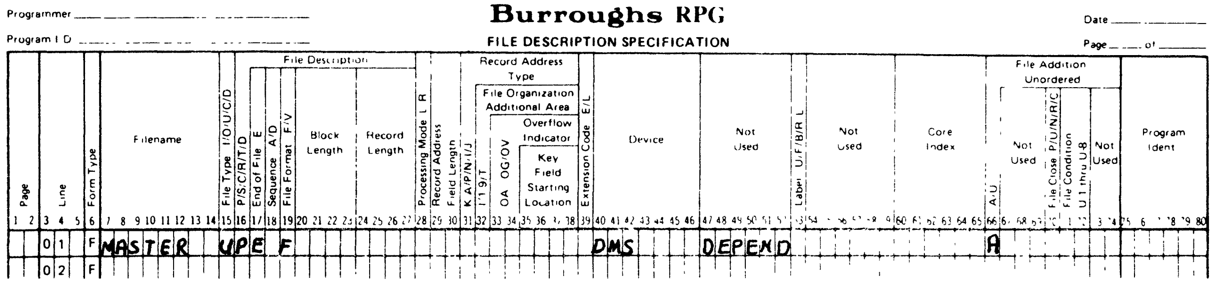


Programmer
Program I D

Burroughs RPG
DATA BASE SPECIFICATION

Date
Page of

Page (1-2)	Line (3-5)	Form Type (6)	Data Base Name (7-16)	Logical Data Base Name (17-26)	Access Mode (27)	Not Used (28-74)	Program Ident (75-80)
		D	PAYR0LL		U		

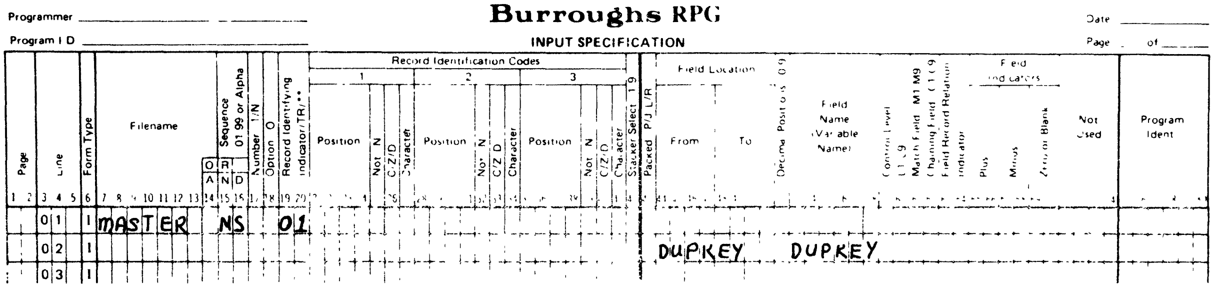


Programmer
Program I D

Burroughs RPG
FILE DESCRIPTION SPECIFICATION

Date
Page of

Page	Line	Form Type	Filename (7-14)	File Type I/O/U/C/D (15)	P/S/C/R/T/D (16)	End of File E (17)	Sequence A/D (18)	File Format F/V (19)	Block Length	Record Length	Processing Mode	Record Address	Key	OA OG/OV Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device (40-46)	Not Used (47-52)	Label U/F/B/R/L	Not Used	Core Index	File Addition Unordered A/U (66)	Not Used	File Close	File Condition	Not Used	Program Ident
	01	F	MASTER	U	P	E		F									DMS	DEPEND				A					
	02	F																									



Programmer
Program I D

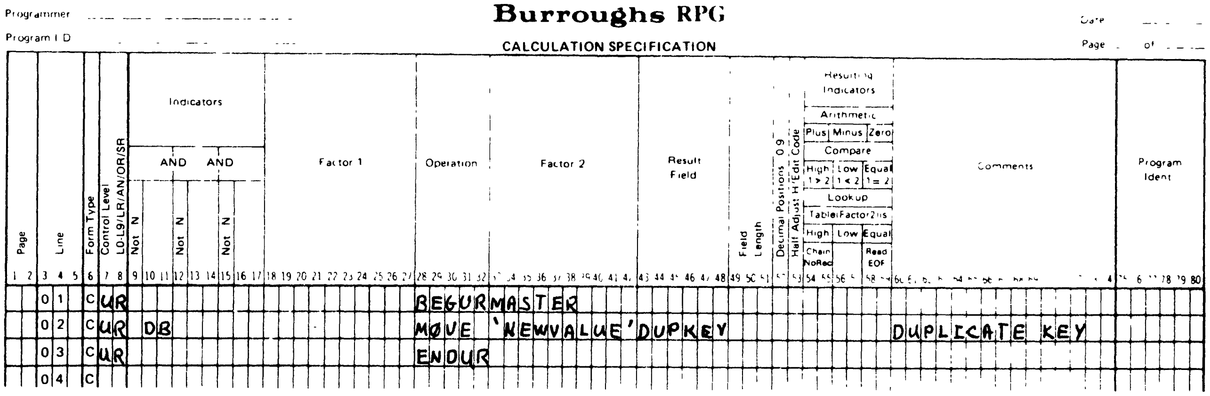
Burroughs RPG
INPUT SPECIFICATION

Date
Page of

Page	Line	Form Type	Filename (7-14)	Sequence (15-16)	Number 1/N	Option O	Record Identifying Indicator (19-20)	Record Identification Codes 1	2	3	Stacker Select	Packed P/L/R	Field Location From	To	Decimal Positions	Field Name (Variable Name)	Control Level	Matching Fields	Chaining Field / Field Record Relation	Field Indicators Plus	Minus	Zero or Blank	Not Used	Program Ident
	01	I	MASTER	NS			01																	
	02	I											DUPKEY			DUPKEY								
	03	I																						

G12080/SHEET 1 OF 2

Figure 13-12. Program Example of Retrying a Write after Exiting Exception Routine with Cycle-Driven Processing (Sheet 1 of 2)



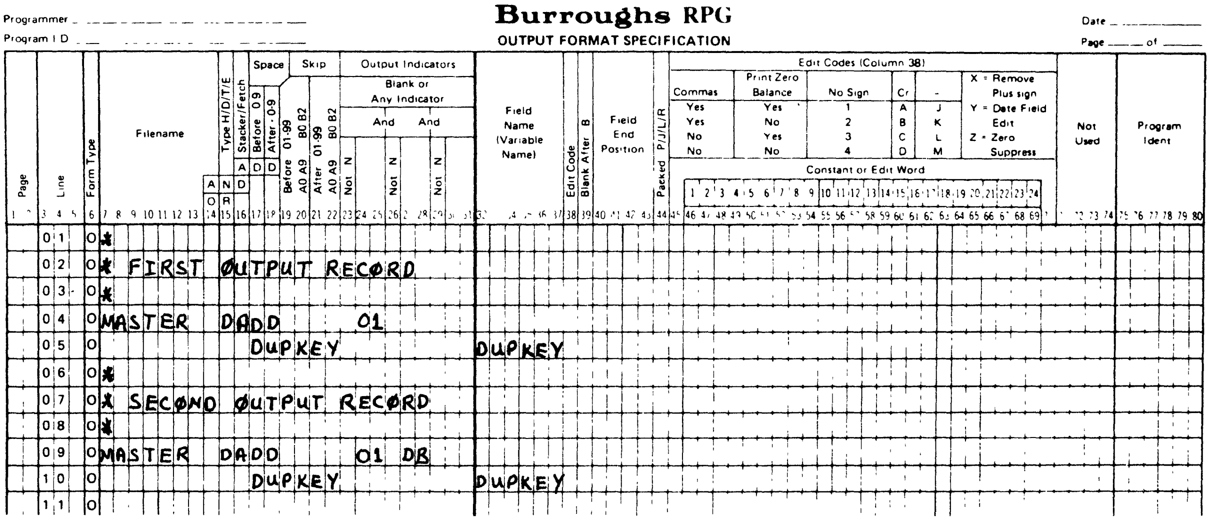
Programmer | Program I D

Burroughs RPG

CALCULATION SPECIFICATION

Date | Page of

Page	Line	Form Type	Control Level (L0-L9/LR/AN/OR/SR)	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust	Resulting Indicators	Comments	Program Ident
	01	C	UR			BEGUR	MASTER							
	02	C	UR	DB		MØVE	'NEWVALUE'	DUPKEY					DUPLICATE KEY	
	03	C	UR			ENDUR								
	04	C												



Programmer | Program I D

Burroughs RPG

OUTPUT FORMAT SPECIFICATION

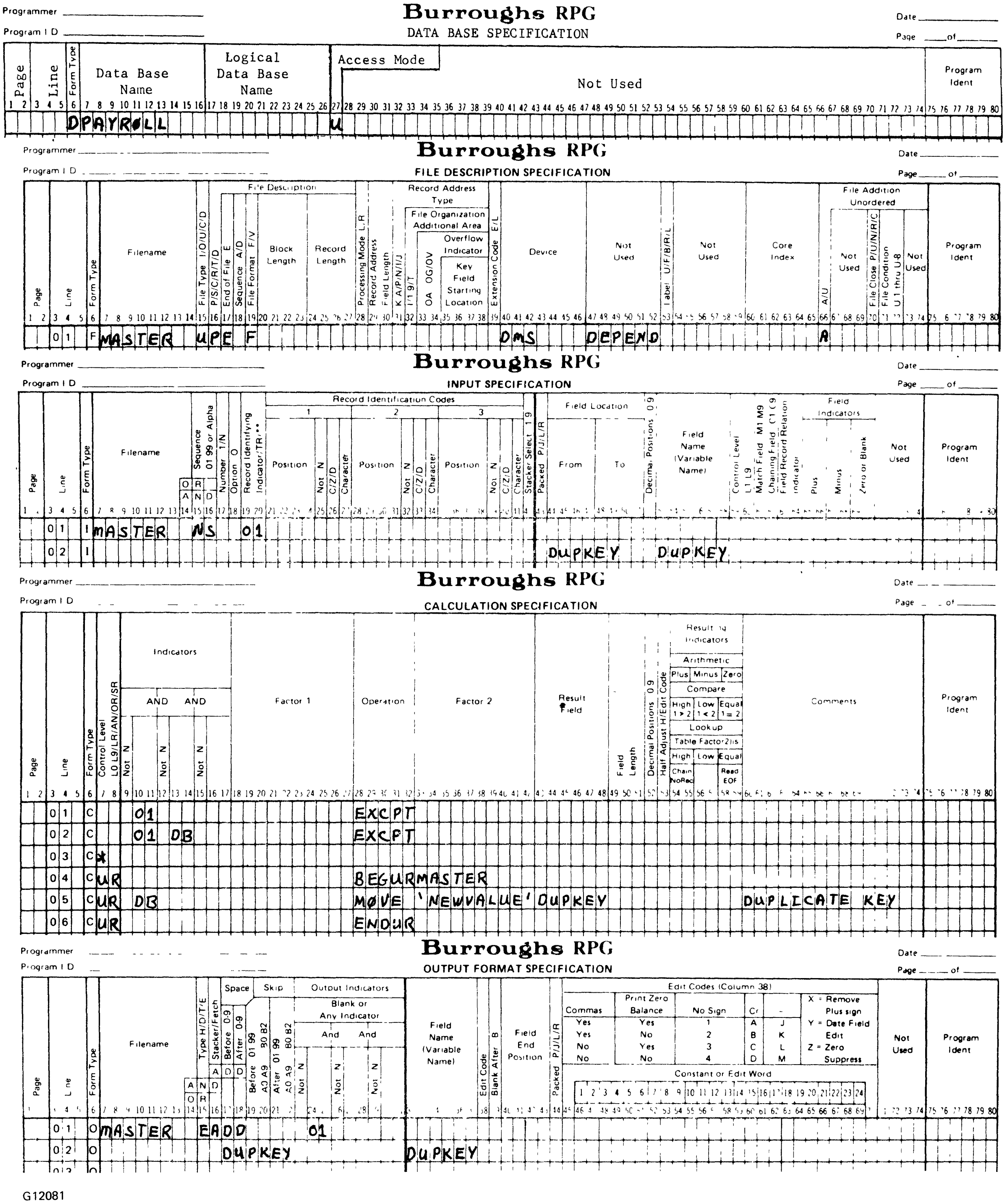
Date | Page of

Page	Line	Form Type	Filename	Type H/D/T/E	Stacker/Fetch / Space / Skip	Output Indicators	Field Name (Variable Name)	Edit Code	Blank After	Field End Position	Packed P/J/L/R	Constant or Edit Word	Not Used	Program Ident
	01	O	*											
	02	O	* FIRST ØUTPUT RECØRD											
	03	O	*											
	04	O	MASTER	D	ADD	01								
	05	O			DUPKEY		DUPKEY							
	06	O	*											
	07	O	* SECØND ØUTPUT RECØRD											
	08	O	*											
	09	O	MASTER	D	ADD	01 DB								
	10	O			DUPKEY		DUPKEY							
	11	O												

G12080/SHEET 2 OF 2

Figure 13-12. Program Example of Retrying a Write after Exiting Exception Routine with Cycle-Driven Processing (Sheet 2 of 2)

In both figures, the data base PAYROLL is opened update. The file MASTER, defined in DASDL as a disjoint data set, is an update primary DMS file with additions specified (an A in column 66 of the File Description Specifications), and is accessed by way of the index DEPEND. When a record from MASTER is read, the record identifying indicator 01 is turned on, and the value from the DASDL-defined location DUPKEY is loaded to the program-defined variable DUPKEY. Also, both examples use the same exception handling routine, delimited by the BEGUR and ENDUR operations in the Calculation Specifications.



Burroughs RPG
DATA BASE SPECIFICATION

```
     DPAYRØLL             U
```

Burroughs RPG
FILE DESCRIPTION SPECIFICATION

```
   01FMASTER  UPE F                    DMS    DEPEND             A
```

Burroughs RPG
INPUT SPECIFICATION

```
   01IMASTER  NS  01
   02I                                        DUPKEY    DUPKEY
```

Burroughs RPG
CALCULATION SPECIFICATION

```
   01C    01                 EXCPT
   02C    01 DB              EXCPT
   03C*
   04CUR                     BEGURMASTER
   05CUR  DB                 MØVE 'NEWVALUE'DUPKEY           DUPLICATE KEY
   06CUR                     ENDUR
```

Burroughs RPG
OUTPUT FORMAT SPECIFICATION

```
   01OMASTER  EADD     01
   02O                DUPKEY           DUPKEY
   03O
```

G12081

Figure 13-13. Program Example of Retrying a Write after Exiting Exception Routine with Exception Output Handling

In figure 13-12, a record is read and an attempt is made to add a record with the same key as was read by means of the RPG cycle. In figure 13-13, the first EXCPT operation in the Calculation Specifications attempts to add a record which has the same key as an existing record. In both examples, a DUPLICATES exception condition is reported and the exception condition indicators D1 and DB are set ON. Each program then branches to the exception handling routine, denoted by the BEGUR operation in the Calculation Specifications. Here, the literal NEWVALUE is moved into the key field DUPKEY and processing continues. In figure 13-12, the processing continues at the second output record specified on the Output-Format Specifications, and in figure 13-13, processing continues with the second EXCPT in the Calculation Specifications. Both methods achieve the same results.

It is recommended that writing to DMS files be done using the EXCPT and STORE operation codes. This insures timely identification and handling of exception conditions.

DATA MANAGEMENT OUTPUT-FORMAT SPECIFICATIONS

The entries defined in the Output-Format Specifications for DMS files are divided into the following two sections:

a. Record description section.

b. Field description section.

RECORD DESCRIPTION SECTION

1-2 PAGE

Refer to Section 2 for a complete description.

3-5 LINE

Refer to Section 2 for a complete description.

6 FORM TYPE

This field must contain the letter O.

7-14 FILE NAME

The file specified must be one of two types:

a. A disjoint data set previously described on the File Description Specifications as:

1. Input with ADD
2. Input with DELETE
3. Input with ADD/DELETE
4. Update
5. Update with ADD
6. Update with DELETE
7. Update with ADD/DELETE
8. Output with ADD

b. An embedded data set which is embedded in a disjoint data set defined on the File Description Specifications.

15 TYPE

The letter entries H, D, E, and T are all valid. Also valid is the DMS letter entry S.

Enter the letter S in this field when this record is to be written as a result of a STORE operation in the Calculation Specifications.

16-18 RECORD ADDITION/DELETION

A record can be added to an existing DMS file if the following conditions are met.

a. The file is defined in a File Description Specification as an input or update primary or secondary file with an index specified, an input or update demand file, or an output file.

b. Column 66 of the File Description Specification has the letter A or B.

c. DMS is entered in the DEVICE FIELD on the File Description Specifications.

d. ADD is entered in the Output-Format Specification in columns 16-18. ADD is implied for files declared as Input with ADD or Output with ADD.

e. ADD must not be specified on AND or OR lines of the Output-Format Specifications.

Also, a record can be added to a DMS file if the following conditions are met.

a. The file is an embedded data set.

b. ADD is entered in the Output-Format Specifications in columns 16-18.

c. ADD must not be specified in AND or OR lines of the Output-Format Specifications.

A record can be deleted from an existing DMS file only if:

a. The file is defined in the File Description Specification as an input or update primary or secondary file.

b. The letter D or B is entered in column 66 on the File Description Specifications.

c. DMS is entered in the DEVICE FIELD on the File Description Specifications.

d. DEL is entered in columns 16-18 in the Output-Format Specifications.

e. DEL must not be specified in AND or OR lines of the Output-Format Specifications.

f. No field description lines may appear with a DEL record description line.

Deletion of records in embedded data sets or demand disjoint data sets is accomplished with the DELET operation code. This does not reference Output-Format Specifications.

23-31 OUTPUT INDICATORS

Besides the indicators normally allowed for these fields, any of the DMS exception condition indicators (D1, DA-DH, DJ-DS) may also be used, but only if DMS files are specified in the File Description Specifications.

32-74 These fields must be blank for record description lines except for variable format data sets.

VARIABLE FORMAT RECORDS

When Variable Format Records are specified for a data set in the DASDL specifications, the RPG compiler requires certain record-type specifications in the Output-Format Specifications for added records to that data set.

The following example shows a DASDL source where Variable Format Records are specified:

Example:

```
DEPT  DATA SET (
    DEPNUM        NUMBER (5);
    DEPNAM        ALPHA  (10);
    RECKEY  RECORD TYPE NUMBER (S2);),

                  -1: (  A  ALPHA  (3) ;
                         B  ALPHA  (4)  ),

                   2: (  C  ALPHA  (7) ;
                         D  NUMBER (6).  ),

       MAXRECORDS = 100;
```

The following should be noted about the records in the DEPT data set:

a. Every record has a fixed portion and a variable portion which depends on the value of RECKEY.

b. The fixed portion of the record consists of the items:

DEPNUM
DEPNAM
RECKEY

c. When records are added to the DEPT data set, DMS uses RECKEY to determine which of two possible formats to assign to the variable portion of the record.

   1. If RECKEY = -1, the variable portion consists of items A and B.

   2. If RECKEY = 2, the variable portion consists of items C and D.

   3. If RECKEY = 0 or the value defined in DASDL as the FIXEDFORMATVALUE, this record has only the fixed portion.

   4. If RECKEY = any other value, a DATAERROR exception condition occurs when adding the record.

d. When variable format records are updated, the control item for the record type must not be changed. If a change is attempted, a DATAERROR exception condition results.

If a record is added to a data set with variable formatting, the first record line(s) of the Output-Format Specifications for the data set must specify the value to be assigned to the record-type. The following entries are required in the Output-Format Specifications:

a. Columns 32-37 must specify a field name or columns 45-70 must specify a constant. If a constant is specified, the constant must be formatted according to the rules for output of constants to data sets on the Output-Format Specifications. Specifying a field name in columns 32-37 and a constant in columns 45-70 is not allowed. Decimal alignment is performed when the record is written. Data truncation is not allowed.

b. If columns 14-16 contain the entry AND, columns 32-37 and columns 45-70 must be blank.

c. If columns 15-16 contain the entry OR, columns 32-37 and columns 45-70 may be blank. If columns 32-37 and columns 45-70 are blank, the field name or constant from the last specified OR line or the first record line is used. If the field name or constant position is not blank, the specified value as given applies. Figure 13-14 illustrates this possibility.

If indicators 01 and 02 are ON, FIELD contains the value for RECKEY.

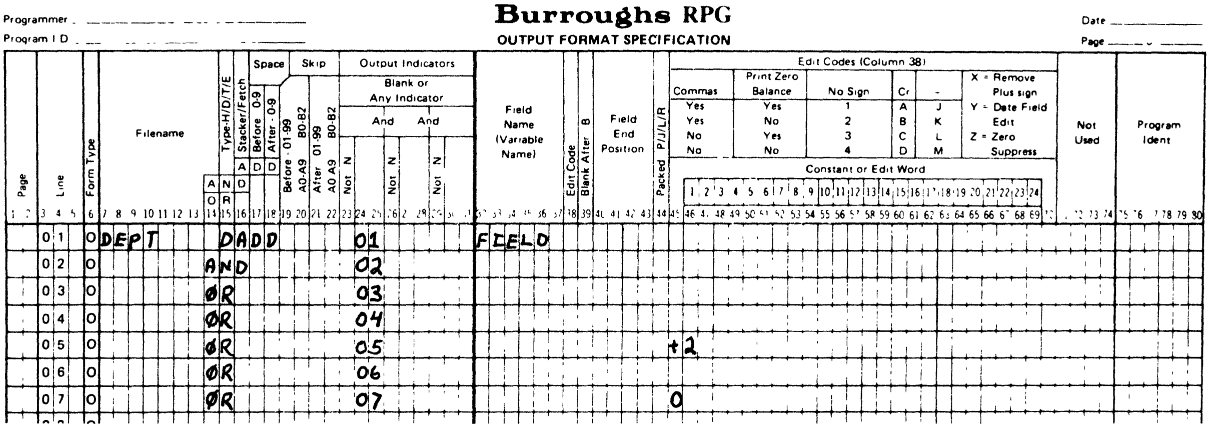
If indicator 03 is ON, FIELD contains the value for RECKEY.

If indicator 04 is ON, FIELD contains the value for RECKEY.

If indicator 05 is ON, +2 is the value for RECKEY.

If indicator 06 is ON, +2 is the value for RECKEY.

If indicator 07 is ON, 0 is the value for RECKEY.



Burroughs RPG

OUTPUT FORMAT SPECIFICATION

Line	Form Type	Filename	Type-H/D/T/E / Stacker/Fetch / Space / Skip	Output Indicators	Field Name (Variable Name)	Constant or Edit Word
01	O	DEPT	DADD	01	FIELD	
02	O		AND	02		
03	O		ØR	03		
04	O		ØR	04		
05	O		ØR	05		+1
06	O		ØR	06		
07	O		ØR	07		0

G12082

Figure 13-14. Example of Variable Format Records Coding

75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a complete description.

FIELD DEFINITIONS FOR FIELD DESCRIPTION LINES

1-2 PAGE

Refer to Section 2 for a complete description.

3-5 LINE

Refer to Section 2 for a complete description.

6 FORM TYPE

This field must contain the letter O.

7-16 These fields must be left blank for field description lines.

17-22 DMS OUTPUT LOCATION

This field is used in the Field Description line to indicate the output location and format of the variable name entered in columns 32-37 or the constant entered in column 45. The entry in this field must be a field name which has been previously defined in the specified data set in DASDL or may be blank if the variable name in the VARIABLE NAME field is a DASDL-defined name. An entry in this field must not be a subscripted vector. This is the only case of a field description line having entries in columns 17-22.

If the entry in this field is defined in DASDL as unsigned numeric, all signs are stripped off before the variable or constant is written.

23-31 OUTPUT INDICATORS

Besides the indicators normally allowed for these fields, any of the DMS exception condition indicators (D1, DA-DH, DJ-DS) can also be used, but only if DMS files are specified in the File Description Specifications.

32-37 VARIABLE NAME

This field is used to identify an output data field. The identifier used must have been previously defined in the Input Specifications, Extension Specifications, or Calculation Specifications. For DMS files, the entry may be the same as the entry made in the DMS output location (columns 17-22 on the Output-Format Specifications). The following conditions must be satisfied.

a. If the variable is numeric, the DASDL definition must be numeric. Decimal point alignment is automatically performed.

b. If the variable is alphanumeric, the DASDL definition must be alphanumeric. The data is stored left-justified with blank fill if required.

Truncation of data is not allowed.

Table 13-2 illustrates the various combinations of DMS OUTPUT LOCATION and VARIABLE NAME fields and the action taken.

38 EDIT CODES

This field must be blank for DMS files.

40-43 END POSITION

This field must be blank for DMS files.

44 PACKED

This field must be blank for DMS files.

45-70 CONSTANT

Edit words must not be entered in this field for DMS files.

The following rules must be observed when forming constants with DMS files:

a. The VARIABLE NAME field entry must be blank.

b. The value of any constant must be numeric if the DMS OUTPUT LOCATION entry is numeric and alphanumeric if the DMS OUTPUT LOCATION entry (columns 17-22) is alphanumeric.

c. If the DMS OUTPUT LOCATION entry (columns 17-22) is numeric, column 45 must contain:

1. + for positive values.

2. - for negative values.

3. A digit (0-9) to imply positive.

d. Decimal point alignment is performed on the entry.

e. Sign characters and decimal points are not included in the size of the literal.

Table 13-2. Combinations of DMS OUTPUT LOCATION and VARIABLE NAME Fields in the Output-Format Specifications

OUTPUT LOCATION Columns 17 - 22	VARIABLE NAME Columns 32 - 37	Program Action
Vector	Array	Write the contents of the whole Array to the location of the DASDL-defined Vector.
Vector	Array,J	Write the contents of Array,J to the location of the DASDL-defined Vector,J.
Vector	Table,J	Write the contents of Table,J to the location of the DASDL-defined Vector,J.
Vector	Array,3	Write the contents of Array,3 to the location of the DASDL-defined Vector,3.
Vector	Table,3	Write the contents of Table,3 to the location of the DASDL-defined Vector,3.
Vector	Table	Not allowed.
Vector	Field Name	Not allowed.
Field Name	Array	Not allowed.
Field Name	Array,J	Write the contents of Array,J to the location of the DASDL-defined Field Name.
Field Name	Table,J	Write the contents of Table,J to the location of the DASDL-defined Field Name.
Field Name	Array,3	Write the contents of Array,3 to the location of the DASDL-defined Field Name.
Field Name	Table,3	Write the contents of Table,3 to the location of the DASDL-defined Field Name.
Field Name	Field Name	Write the contents of the Field Name in columns 32-37 to the location of the DASDL-defined Field Name in columns 17-22.
Field Name	Table	Not allowed.
Blank	Array	Write the contents of the whole Array to the location of the DASDL-defined Vector. The definition used for columns 17-22 defaults to the DASDL-defined name in columns 32-37.
Blank	Array,J	Write the contents of Array,J to the location of the DASDL-defined Vector,J. The definition used for columns 17-22 defaults to the DASDL-defined name in columns 32-37.

Table 13-2. Combinations of DMS OUTPUT LOCATION and VARIABLE NAME Fields in the Output-Format Specifications (cont)

OUTPUT LOCATION Columns 17 - 22	VARIABLE NAME Columns 32 - 37	Program Action
Blank	Array,3	Write the contents of Array,3 to the location of the DASDL-defined Vector,3. The definition used for columns 17-22 defaults to the DASDL-defined name in columns 32-37.
Blank	Table	Not Allowed.
Blank	Table,J	Write the contents of Table,J to the location of the DASDL-defined Vector,J. The definition used for columns 17-22 defaults to the DASDL-defined name in columns 32-37.
Blank	Table,3	Write the contents of Table,3 to the location of the DASDL-defined Vector,3. The definition used for columns 17-22 defaults to the DASDL-defined name in columns 32-37.
Blank	Field Name	Write the contents of Field Name to the location of the DASDL-defined Field Name. The definition used for columns 17-22 defaults to the DASDL-defined name in columns 32-37.

f. The DASDL definition must be at least as long as the constant after decimal point alignment has been performed.

If the DMS OUTPUT LOCATION entry is numeric, embedded blanks must not appear in the constant field.

g. If the DMS OUTPUT LOCATION field (columns 17-22) is alphanumeric, then:

1. The DASDL definition must be at least as long as the constant.

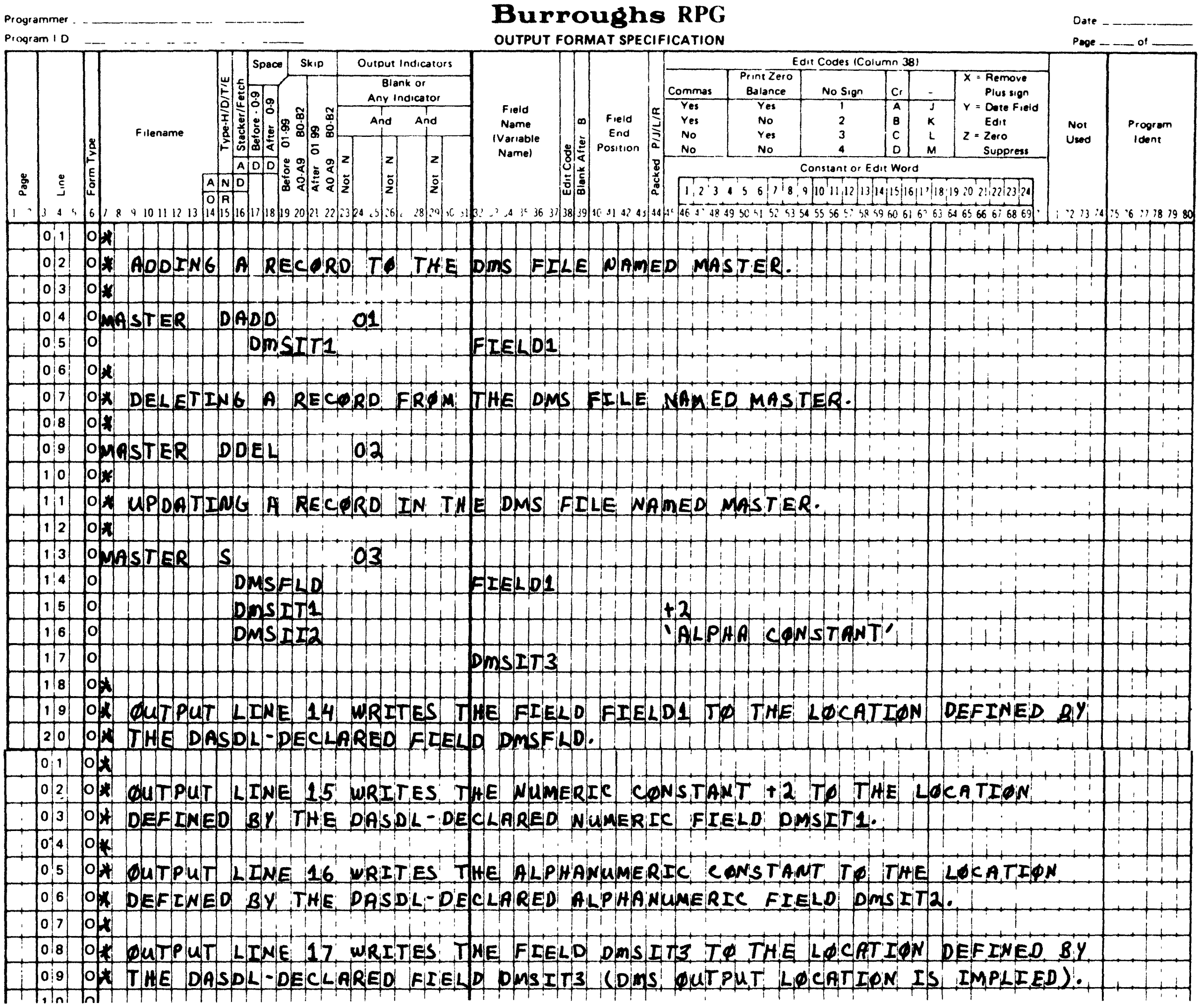
2. Data is left-justified with blank fill, if necessary.

71-74 These columns must be blank.

75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a complete description.

Figure 13-15 shows coding examples for data management files in the Output-Format Specifications.



Programmer

Program ID

Burroughs RPG

OUTPUT FORMAT SPECIFICATION

Date

Page ___ of ___

```
  01 O*
  02 O* ADDING A RECORD TØ THE DMS FILE NAMED MASTER.
  03 O*
  04 OMASTER  DADD     01
  05 O          DMSIT1         FIELD1
  06 O*
  07 O* DELETING A RECØRD FRØM THE DMS FILE NAMED MASTER.
  08 O*
  09 OMASTER  DDEL     02
  10 O*
  11 O* UPDATING A RECØRD IN THE DMS FILE NAMED MASTER.
  12 O*
  13 OMASTER  S        03
  14 O         DMSFLD          FIELD1
  15 O         DMSIT1                       +2
  16 O         DMSII2                       'ALPHA CØNSTANT'
  17 O                         DMSIT3
  18 O*
  19 O* ØUTPUT LINE 14 WRITES THE FIELD FIELD1 TØ THE LØCATIØN DEFINED BY
  20 O* THE DASDL-DECLARED FIELD DMSFLD.
  01 O*
  02 O* ØUTPUT LINE 15 WRITES THE NUMERIC CØNSTANT +2 TØ THE LØCATIØN
  03 O* DEFINED BY THE DASDL-DECLARED NUMERIC FIELD DMSIT1.
  04 O*
  05 O* ØUTPUT LINE 16 WRITES THE ALPHANUMERIC CØNSTANT TØ THE LØCATIØN
  06 O* DEFINED BY THE DASDL-DECLARED ALPHANUMERIC FIELD DMSIT2.
  07 O*
  08 O* ØUTPUT LINE 17 WRITES THE FIELD DMSIT3 TØ THE LØCATIØN DEFINED BY
  09 O* THE DASDL-DECLARED FIELD DMSIT3 (DMS ØUTPUT LØCATIØN IS IMPLIED).
  10 O
```

G12083

Figure 13-15. Output-Format Specifications for DMS Files

DATA MANAGEMENT AUDIT/RECOVERY

A processing failure can interrupt a logical update to a data base. For example, a data record may have been inserted into a data set but not into one of its automatic sets. To recover data base integrity after such a failure, it is necessary to preserve the "before" images so that partial operations can be removed from the data base. Storage failures can occur requiring "after" images to restore data base integrity. In order to recover the data base, an audit trail must be maintained by DMSII.

If the AUDIT option is set in DASDL, an audit trail is maintained by DMS. This audit trail can be stored on tape or disk. If disk is specified, use a disk pack other than the disk pack which contains the data base. If the audit trail and data base are maintained on the same disk pack, and an irrecoverable disk failure occurs, the data base can only be recovered to the point where the data base was dumped to tape or another disk pack. Maintaining separate disk packs for the audit trail and data base makes recovery of the data base faster.

When a data base is audited (AUDIT option set in DASDL), all changes to the data base are written sequentially to the audit file. If a processing failure occurs, this audit file contains each change so that they can be reapplied to the data base by the DMS recovery routines. All logically incomplete transactions are removed from the data base.

The DASDL source for an audited data base includes a RESTART DATA SET declaration. This restart data set holds the necessary information so that a program can realign itself with the last good transaction prior to the processing failure. When DMS recovers the data base following a processing failure, the DMS recovery routines store each program's restart information (based on the contents of the audit trail) into the restart data set. Since the restart data set is part of the data base, RPG programs can conveniently retrieve the restart information from this data set to reinitialize themselves after completion of the recovery process.

All updating to an audited data base must be done within transaction state. There are two operation codes in RPG which delimit transaction state for an RPG program. They are TRBEG and TREND. TRBEG specifies the beginning of transaction state and TREND specifies the end of transaction state. These are explicit operations and are not required for RPG cycle-driven file processing.

Cycle-driven DMS files generate TRBEG and TREND operation codes implicitly. This occurs when:

a. A restart data set is specified in the File Description Specifications.

b. One or more cycle-driven DMS files are specified in the File Description Specifications.

c. The data base is opened update.

A TRBEG operation is generated by the RPG compiler after the record identifying indicator is set ON for the cycle-driven DMS file. One other TRBEG operation is generated when end of file (the LR indicator ON) is reached for cycle-driven DMS files just prior to end-of-job totals. Figure H-1 in Appendix H shows the implicit TRBEG and TREND operations performed on cycle-driven files in relation to the RPG cycle.

A TREND operation with audit and no syncpoint is generated by the RPG compiler following detail output except when the 1P indicator is ON. One other TREND operation with no audit and syncpoint is generated after end-of-job totals and before all files are closed.

The DMS recovery routines are transaction-oriented and always recover to a time when no partial transactions are reflected in the data base.

A transaction consists of a sequence of DMS operations grouped together in an RPG program. These DMS operations constitute a logical change to the data base. The DMS operations which actually change the data base, (STORE, DELET, INSRT, REMOV) must be performed within a transaction state; otherwise, an AUDITERROR exception condition occurs. Other operations such as the FIND operation can be performed either in or out of a transaction state since the operation does not change any item in the data base.

Following the recovery of a data base, the RPG program must reprocess any transactions which were interrupted due to the processing failure. To assist the RPG program in restarting, restart information must be stored in the restart data set. If a processing failure occurs, the recovery routines insure that the information corresponding to the last good transaction is in the restart data set. Using this information, the program can restart at the point where the data base was recovered.

If a data base is opened update (the letter U in column 27 of the Data Base Specification) and the data base is being audited, the restart data set defined in DASDL must be defined in the File Description Specifications in the RPG program. If a data base is opened input only (the letter I in column 27 of the Data Base Specifications), the RPG program must not specify a File Description Specification for a restart data set.

RESTART DATA SET FILE DESCRIPTION SPECIFICATION

Every restart data set to be used by an RPG program must be described to the RPG compiler in the File Description Specifications. The restart data set must be specified and the AUDIT option must be set in the DASDL source program.

FIELD DEFINITIONS

1-2 PAGE

Refer to Section 2 for a complete description.

3-5 LINE

Refer to Section 2 for a complete description.

6 FORM TYPE

This field must contain the letter F.

7-14 DATA SET NAME

This field must contain the name of the restart data set which is specified in DASDL.

15 FILE TYPE

For a restart data set, this field must contain the letter R.

16-39 These fields must be blank for a restart data set.

40-46 DEVICE

For a restart data set, this field must contain the entry DMS.

47-65 For a restart data set, these fields must be blank.

66 FILE ADDITION/UNORDERED/DELETION

For a restart data set, all entries for this field which are valid for demand files with device type DMS are also valid for a restart data set. Valid entries for this field are:

Entry	Definition
A	Records may be added to this file.
Blank	Records may not be added to this file.

67-70 For a restart data set, these fields must be blank.

75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a complete description.

RESTART DATA SET INPUT SPECIFICATIONS

The Input Specifications for a restart data set are identical to those specifications for any demand-driven file whose device type is DMS. Input Specifications for a restart data set are required only if the program has FIND or LOCK operations applied to it.

RESTART DATA SET CALCULATION SPECIFICATIONS

All DMS operations which are valid for demand DMS data sets are also valid for the restart data set. Specifically, restart data sets are essentially update, demand DMS files. However, no operations which alter the restart data set, for example STORE, are allowed outside of a transaction state. If such an operation is attempted, an AUDITERROR exception condition occurs.

for 71-74 - see page 13-9.

Two additional operation codes are needed to explicitly define a transaction to DMS for an audited data base. These are the TRBEG and TREND operations. The use of these operation codes is defined as follows.

TRBEG

Initiates transaction state for this program. TRBEG must be entered in columns 28-32 of the Calculation Specifications. The D1 exception condition indicator may be entered in columns 54-55. All other fields must be blank. No record is written to the restart data set as a result of execution of the TRBEG operation.

TREND

Ends transaction state for this program. The FACTOR 2 field may contain the entry NOAUDIT, and the RESULT FIELD may contain the entry SYNCPT. The D1 exception condition indicator may be entered in columns 54-55.

If the FACTOR 2 field contains the entry NOAUDIT, no information is written to the restart data set. If the FACTOR 2 field is left blank, a STORE operation is performed on the restart record defined with the letter R in column 15 of the Output-Format Specifications.

If the RESULT FIELD contains the entry SYNCPT, a syncpoint is forced by DMS prior to returning control back to this program. The SYNCPT option ensures that all transactions performed prior to this TREND operation with syncpoint are reflected in the data base. If the RESULT FIELD is left blank, no syncpoint is forced.

It is recommended that the D1 exception condition indicator be entered in columns 54-55 of the Calculation Specifications for the TRBEG and TREND operations. If columns 54-55 are left blank for a TRBEG or TREND operation and an exception condition occurs, the exception handling routine for the restart data set is entered. Refer to the subsection titled Data Management Exception Handling in this section for more information. If no exception handling routine exists, the program aborts.

The ABORT exception condition can occur on a TRBEG operation or on a TREND with syncpoint operation. Any exception condition which can occur on a STORE operation can also occur on a TREND with audit operation.

Exception conditions can occur on the implicitly generated transaction operations, just as they can occur when TRBEG and TREND operations are specified in the Calculation Specifications. However, the D1 indicator cannot be specified for use on the implicitly generated transactions operations.

If an exception condition occurs on an implicitly generated TRBEG or TREND operation, the exception handling routine for the restart data set is entered. If no such routine exists and an exception condition occurs, the program aborts.

The implicit TREND operation generates an end transaction with audit and no syncpoint, except when the LR indicator is ON in which case the implicit TREND operation is generated with no audit, syncpoint. This guarantees that if this operation is successful, no data base operations for this program are backed out on a program abort or CLEAR/START.

Table 13-3 lists the RPG coding options available for the TRBEG and TREND operations.

Table 13-3. TRBEG and TREND Operation Coding Options

Operation Code	Factor 2	Result Field	Columns 54 - 55	Description
TRBEG	Blank	Blank	D1	Beginning of a transaction state.
TREND	Blank	Blank	D1	A STORE is performed by the RPG program to the restart data set. An Output-Format Specifications record is required which contains an R in column 15.
TREND	NOAUDIT	Blank	D1	A STORE operation is not performed to the restart data set.
TREND	Blank	SYNCPT	D1	Syncpoint is forced for auditing prior to returning control back to the program.
TREND	NOAUDIT	SYNCPT	D1	A STORE operation is not performed by the RPG program to the restart data set. Syncpoint is forced for auditing prior to returning control back to the program.

RESTART DATA SET OUTPUT-FORMAT SPECIFICATIONS

All allowable Output-Format Specifications entries for DMS files are also applicable to the restart data set. Also required are special Output-Format Specifications for the implicit or explicit TREND operations. These are defined as follows.

7-14 DATA SET NAME

This field must contain the name of the restart data set which is defined in the File Description Specifications.

15 TYPE

This field must contain the letter R whenever a TREND with audit is generated implicitly or explicitly.

The following rules apply when coding type R Output-Format Specifications for the TREND operation:

a. Only type R Output-Format Specifications are used for a TREND with audit operation.

b. If more than one type R output record is eligible for output at the time of the TREND operation, an AUDITERROR exception condition occurs. This is due to the fact that all writing to a restart data set must be done within a transaction. Once one STORE operation has been performed against the restart data set, the transaction ends until the next TRBEG operation.

c. If no type R records are eligible for output at the time of the TREND operation, an AUDITERROR exception condition occurs when the next TRBEG operation is performed. The AUDITERROR exception condition results from consecutive TRBEG operations because the first transaction state did not end with an implicit or explicit TREND operation.

The recovery record(s) contained in the restart data set may contain a programmatically defined key. In the case of two or more copies of the same RPG program running concurrently in the mix, the restart data set can contain unique keys (for example a batch ticket).

The effective restart of a program implies that the RPG program must have saved enough information in the restart data set to be able to commence processing as though no interruption had occurred. This includes the ability to reposition all non-DMS files to a point where integrity is assured. This requires RPG source code in each RPG program which is auditing the data base to read from the restart data set and use this information to realign the RPG program relative to the last good transaction.

Figure 13-16 is an example of one technique illustrating audit and recovery. The RPG program's function is to read records from a disk file and then apply these records (additions, changes, and deletions) to a DMS master file. The recovery method utilized needs the following information:

a. A data item, STATUS, which contains a value of C or I. If STATUS is equal to C, this run is complete. If STATUS is equal to I, this run is incomplete.

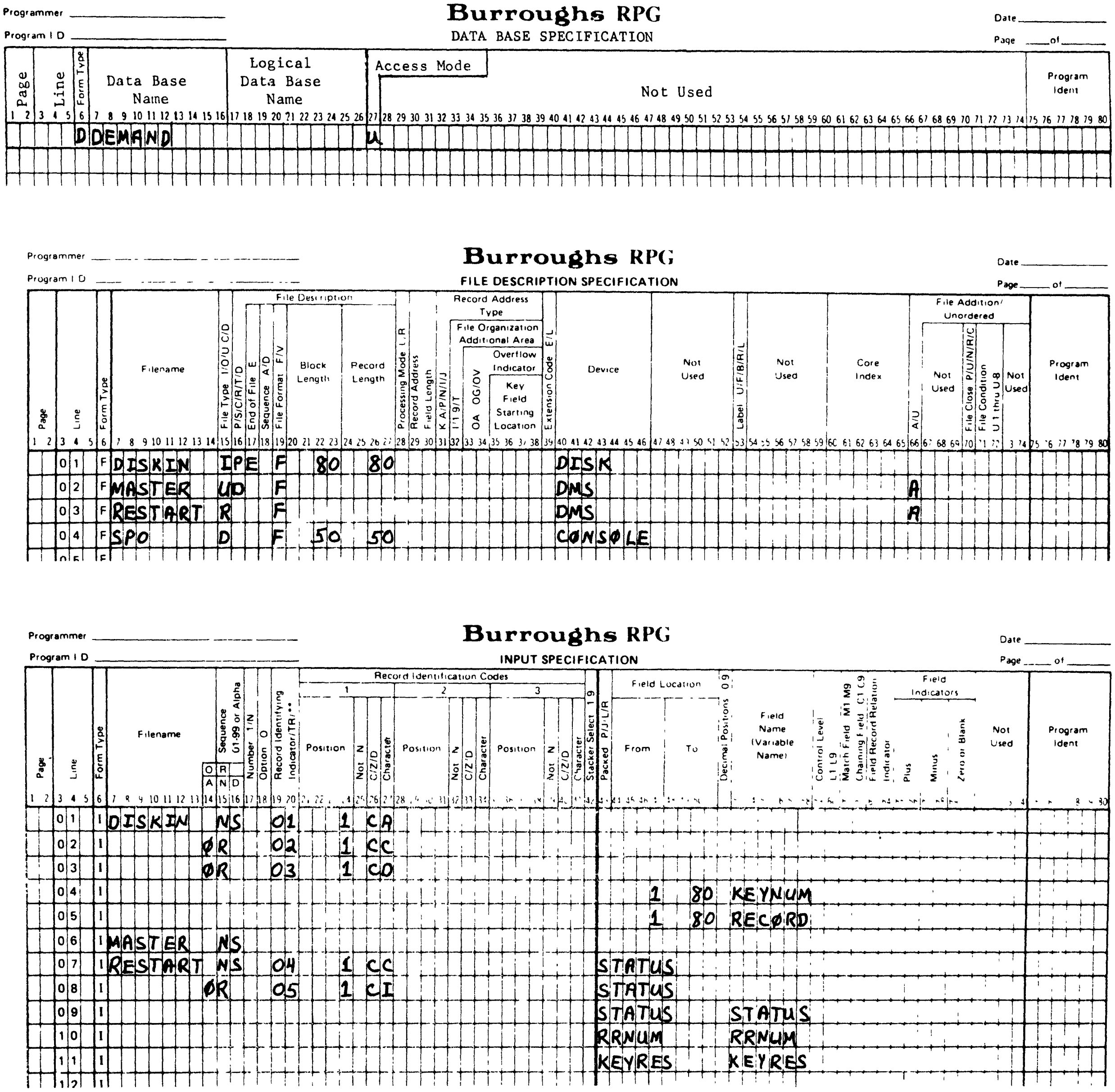
b. A data item, RRNUM, which contains the value of the last successfully processed relative record number in the input file.

c. Since there is only one restart data set for all programs using a data base, and each program needs its own restart record within the restart data set, there must be a "key" within each record of the restart data set which uniquely identifies that record as belonging to this program. In this example, the "key" is the program name, SAMPLE.

In a non-DMS environment, this program would read one record per cycle, increment a counter according to the record type, "chain" to the "master" file, and perform detail or exception output to the "master" file.

Figure 13-16 follows a similar pattern when a data set named MASTER and index named MSTKEY are substituted for the non-DMS indexed disk file in the situation described above.

The actual file maintenance occurs in one of three subroutines (ADD, CHANGE, or DELETE) dependent upon input record types. Prior to each STORE operation, transaction state must be entered using the TRBEG operation. The STORE operation is then performed followed by the TREND operation to end the transaction state. When the TREND operation is used, the program looks for a type R output record which is properly conditioned by indicators. In this case, it is the second output record for the data set RESTART of the Output-Format Specifications. The restart record is updated with the current relative record number and status.



Programmer ____ Program I D ____

Burroughs RPG

DATA BASE SPECIFICATION

Date ____ Page ____ of ____

Page	Line	Form Type	Data Base Name	Logical Data Base Name	Access Mode	Not Used	Program Ident
		D	DEMAND		U		

Programmer ____ Program I D ____

Burroughs RPG

FILE DESCRIPTION SPECIFICATION

Date ____ Page ____ of ____

Page	Line	Form Type	Filename	File Type I/O/U C/D	P/S/C/R/T/D	End of File E	Sequence A/D	File Format F/V	Block Length	Record Length	Device	A/U
	01	F	DISKIN	I	P	E		F	80	80	DISK	
	02	F	MASTER	U	D			F			DMS	A
	03	F	RESTART	R				F			DMS	A
	04	F	SPO	D				F	50	50	CØNSØLE	
	05	F										

Programmer ____ Program I D ____

Burroughs RPG

INPUT SPECIFICATION

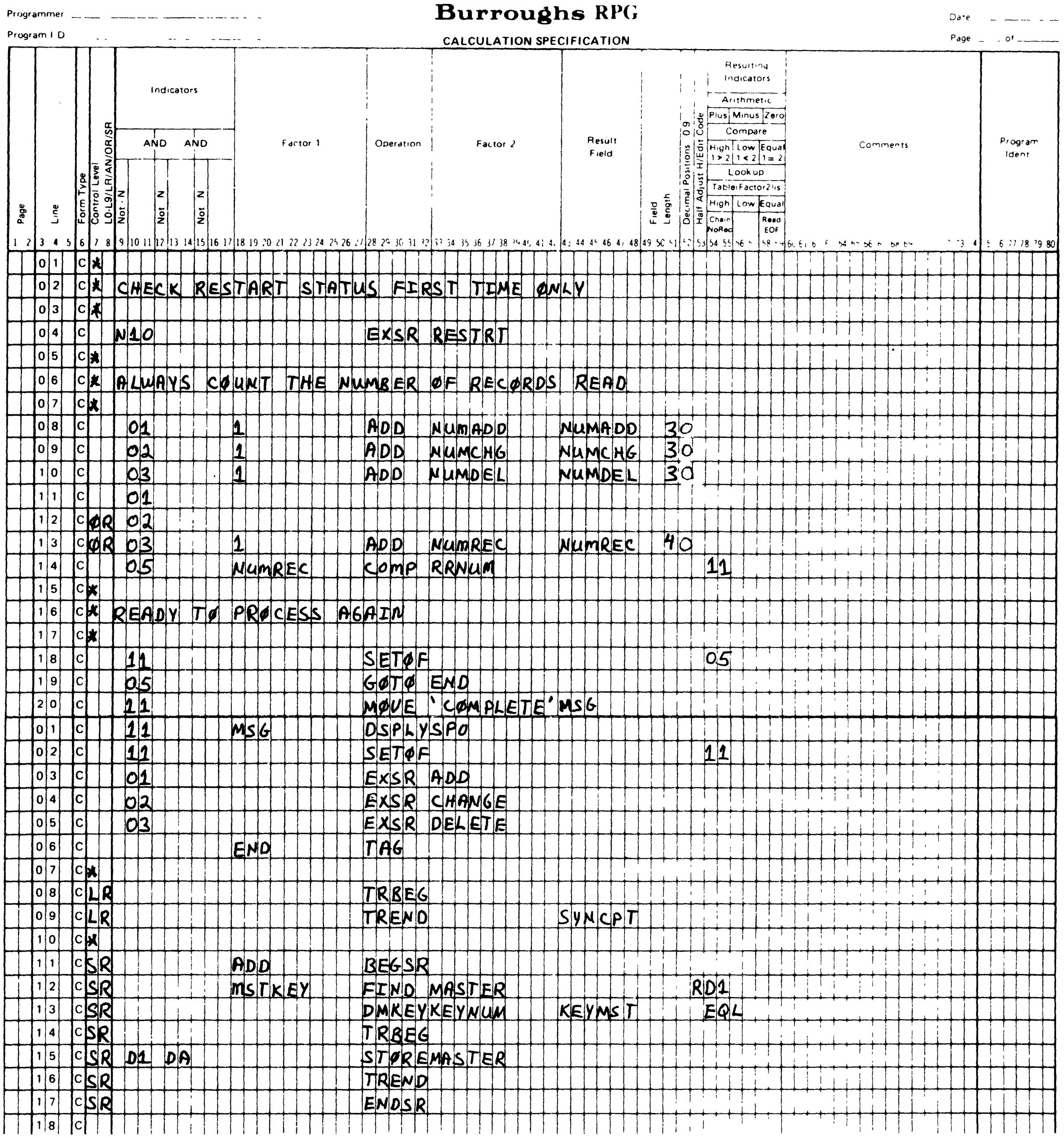
Date ____ Page ____ of ____

Page	Line	Form Type	Filename	OR/AND	Sequence	Record Identifying Indicator	Position	Not	C/Z/D	Character	Field Location From	To	Field Name
	01	I	DISKIN		NS	01	1		C	A			
	02	I		ØR		02	1		C	C			
	03	I		ØR		03	1		C	O			
	04	I									1	80	KEYNUM
	05	I									1	80	RECØRD
	06	I	MASTER		NS								
	07	I	RESTART		NS	04	1		C	C	STATUS		
	08	I		ØR		05	1		C	I	STATUS		
	09	I									STATUS		STATUS
	10	I									RRNUM		RRNUM
	11	I									KEYRES		KEYRES
	12	I											

G12084/SHEET 1 OF 4

Figure 13-16. Program Example of Restart with a Demand DMS File (Sheet 1 of 4)

This process continues until the End of File is reached on the primary file. When the LR indicator is on, the restart status field named STATUS must be reset to C to indicate completion of the run. By entering and exiting transaction state (by performing the TRBEG and TREND operations respectively), an update is made to the restart record to reflect STATUS = C (the last output record for the data set RESTART of Output-Format Specifications in figure 13-16). By including SYNCPT on the TREND operation, a syncpoint is forced on DMS. Thus, if the TREND operation is successful, the program is assured that all of its previous activity is protected by the recovery mechanism of DMS. That is, if the system fails after the TREND and before end of job, all of the program's updates are reflected in the data base.



Programmer

Program I D

Burroughs RPG

CALCULATION SPECIFICATION

Date

Page of

Page	Line	Form Type	Control Level (L0-L9/LR/AN/OR/SR)	Indicators	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Resulting Indicators	Comments	Program Ident
	01	C	*										
	02	C	*	CHECK RESTART STATUS FIRST TIME ØNLY									
	03	C	*										
	04	C		N10		EXSR	RESTRT						
	05	C	*										
	06	C	*	ALWAYS CØUNT THE NUMBER ØF RECØRDS READ									
	07	C	*										
	08	C		01	1	ADD	NUMADD	NUMADD	3	0			
	09	C		02	1	ADD	NUMCHG	NUMCHG	3	0			
	10	C		03	1	ADD	NUMDEL	NUMDEL	3	0			
	11	C		01									
	12	C	ØR	02									
	13	C	ØR	03	1	ADD	NUMREC	NUMREC	4	0			
	14	C		05	NUMREC	COMP	RRNUM				11		
	15	C	*										
	16	C	*	READY TØ PRØCESS AGAIN									
	17	C	*										
	18	C		11		SETØF					05		
	19	C		05		GØTØ	END						
	20	C		11		MØVE	'CØMPLETE'MSG						
	01	C		11	MSG	DSPLY	SPO						
	02	C		11		SETØF					11		
	03	C		01		EXSR	ADD						
	04	C		02		EXSR	CHANGE						
	05	C		03		EXSR	DELETE						
	06	C			END	TAG							
	07	C	*										
	08	C	LR			TRBEG							
	09	C	LR			TREND		SYNCPT					
	10	C	*										
	11	C	SR		ADD	BEGSR							
	12	C	SR		MSTKEY	FIND	MASTER				RD1		
	13	C	SR			DMKEY	KEYNUM	KEYMST			EQL		
	14	C	SR			TRBEG							
	15	C	SR	D1 DA		STØRE	MASTER						
	16	C	SR			TREND							
	17	C	SR			ENDSR							
	18	C											

G12084/SHEET 2 OF 4

Figure 13-16. Program Example of Restart with a Demand DMS File (Sheet 2 of 4)

In the event of a system failure, the operator must recover the data base. After the data base is recovered, the operator may re-execute any programs which were running at the time of the failure.

Programmer

Program I.D

Burroughs RPG

CALCULATION SPECIFICATION

Date

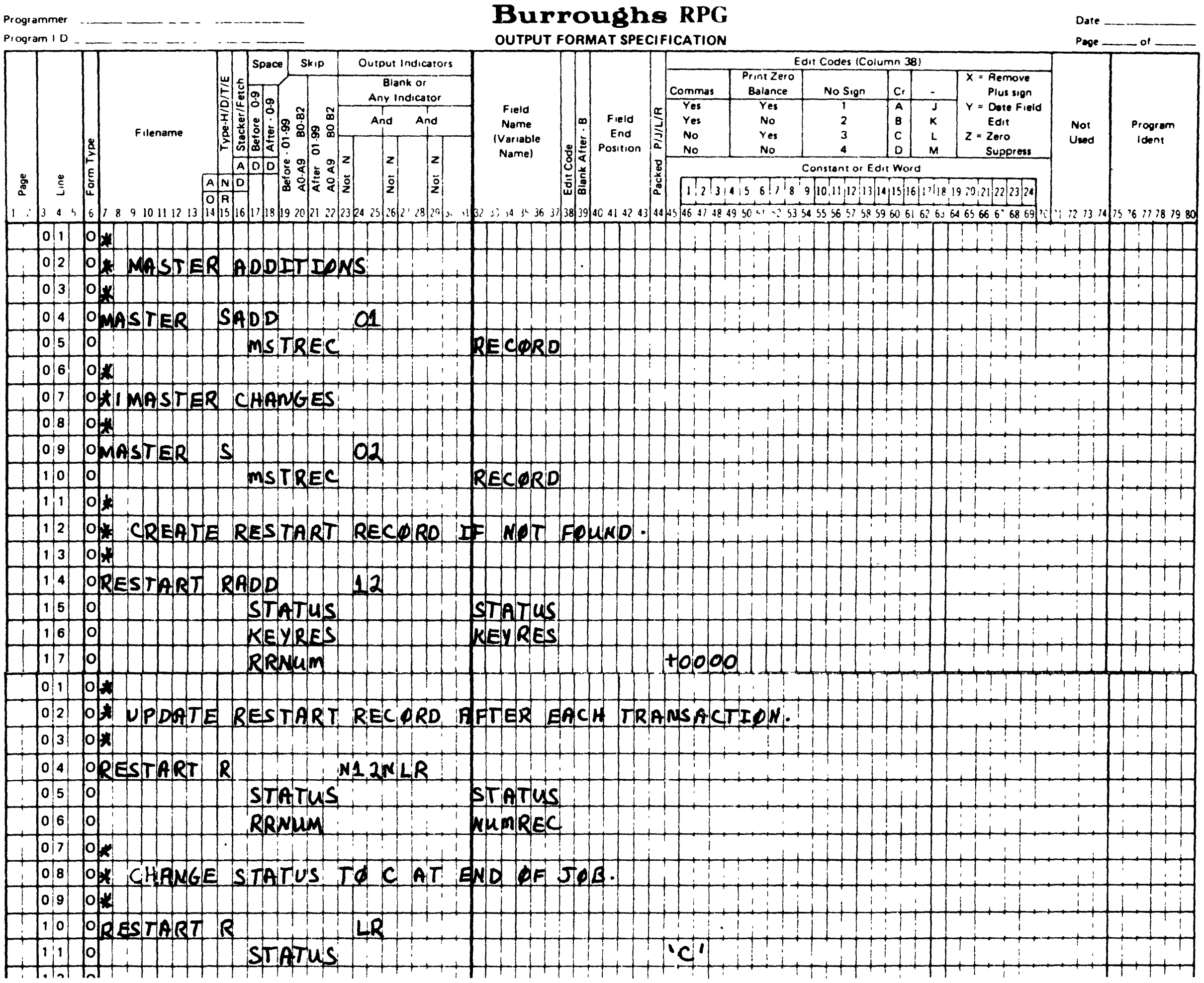
Page of

Page | Line | Form Type | Control Level L0-L9/LR/AN/OR/SR | Indicators: Not, AND Not, AND Not | Factor 1 | Operation | Factor 2 | Result Field | Field Length | Decimal Positions 0-9 | Half Adjust H/Edit Code | Resulting Indicators: Arithmetic Plus Minus Zero; Compare High 1>2 Low 1<2 Equal 1=2; Lookup Table/Factor2 is High Low Equal; Chain NoRec, Read EOF | Comments | Program Ident

```
  01 C*
  02 CSR         CHANGE    BEGSR
  03 CSR         MSTKEY    LØCK MASTER                RD1
  04 CSR                   DMKEYKEYNUM    KEYMST       EQL
  05 CSRND1                TRBEG
  06 CSRND1                STØREMASTER
  07 CSRND1                FREE MASTER
  08 CSRND1                TREND
  09 CSR                   ENDSR
  10 C*
  11 CSR         DELETE    BEGSR
  12 CSR         MSTKEY    FIND MASTER                RD1
  13 CSR                   DMKEYKEYNUM    KEVMST       EQL
  14 CSRND1                TRREG
  15 CSRND1                DELETMASTER
  16 CSRND1                TREND
  17 CSR                   ENDSR
  01 C*
  02 CSR         RESTRT    BEGSR
  03 CSR                   SETØN                       10
  04 CSR                   MØVE 'SAMPLE'  KEYRES
  05 CSR         RESKEY    LØCK RESTART               RD1
  06 CSR                   DMKEY'SAMPLE'  KEYRES       EQL
  07 C*
  08 C* IF RESTART RECØRD THERE AND LAST RUN CØMPLETE, THEN RESET AND
  09 C* GET ØUT.
  10 C*
  11 CSR 04                MØVE 'I'       STATUS
  12 CSR 04                GØTØ ENDRES
  13 C*
  14 C* IF RESTART RECØRD NØT THERE, THEN CREATE IT AND GET ØUT.
  15 C*
  16 CSR DA                MOVE 'I'       STATUS
  17 CSR DA                SETØN                       12
  18 CSR DA                TRBEG
  19 CSR DA                TREND
  20 CSR DA                SETØF                       12
     CSR DA                GØTØ ENDRES
  01 C*
  02 C* HAVE TØ DØ A RESTART; INDICATØR 05 ØN UNTIL RECØVERED.
  03 C*
  04 CSR 05                MØVEL'RECØVERY'MSG    16
  05 CSR 05                MOVE ' STARTED'MSG
  06 CSR 05      MSG       DSPLYSPØ
  07 CSR         ENDRES    ENDSR
  08 C
```

G12084/SHEET 3 OF 4

Figure 13-16. Program Example of Restart with a Demand DMS File (Sheet 3 of 4)



Programmer

Program I.D.

Burroughs RPG

OUTPUT FORMAT SPECIFICATION

Date

Page ___ of ___

```
  01 O*
  02 O* MASTER ADDITIONS
  03 O*
  04 OMASTER  SADD     01
  05 O          MSTREC         RECØRD
  06 O*
  07 O*IMASTER CHANGES
  08 O*
  09 OMASTER  S        02
  10 O          MSTREC         RECØRD
  11 O*
  12 O* CREATE RESTART RECØRD IF NØT FØUND.
  13 O*
  14 ORESTART RADD     12
  15 O          STATUS         STATUS
  16 O          KEYRES         KEYRES
  17 O          RRNUM                       +0000
  01 O*
  02 O* UPDATE RESTART RECØRD AFTER EACH TRANSACTIØN.
  03 O*
  04 ORESTART R       N12NLR
  05 O          STATUS         STATUS
  06 O          RRNUM          NUMREC
  07 O*
  08 O* CHANGE STATUS TØ C AT END ØF JØB.
  09 O*
  10 ORESTART R        LR
  11 O          STATUS                      'C'
  12 O
```

G12084/SHEET 4 OF 4

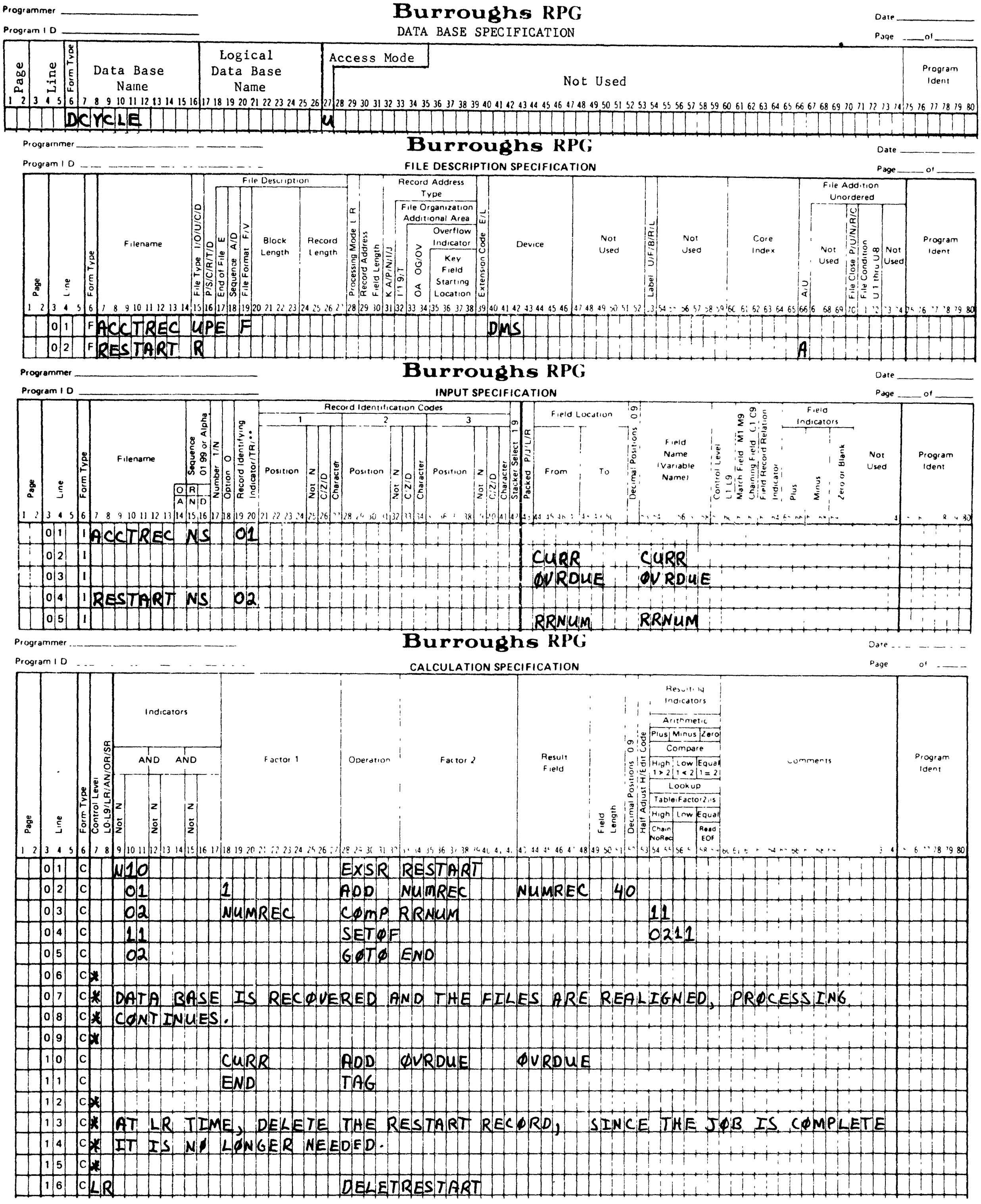
Figure 13-16. Program Example of Restart with a Demand DMS File (Sheet 4 of 4)

The first thing that the RPG program in figure 13-16 does is branch to the subroutine RESTART. This subroutine attempts to locate its restart record in the restart data set using its key SAMPLE. If the record is found and the STATUS = C designating complete, the STATUS field is updated with I, designating incomplete, for the current run. Processing continues with the next instruction following EXSR RESTART in the Calculation Specifications.

If the restart record is not found, the indicators D1 and DA are set ON indicating that this is the first time this program has been run. In this case, the restart record must be initialized and added to the restart data set. The addition is accomplished by the TRBEG and TREND operations. The record is added by the first RESTART output record of the Output-Format Specifications in figure 13-16. The STATUS then equals I, incomplete, for the current run, and processing begins.

If the restart record is found and STATUS equals I, the program aborted during its last execution and special action must be taken by the program. The field RRNUM contains the number of input records which were successfully processed during the previous run. This means that records must be skipped for this run. As long as indicator 05 is on and STATUS equals I, all updating activity is bypassed. When the number of records read during this execution exceeds the number of successfully processed records during the previous run, indicator 05 is set OFF and normal processing resumes.

Figure 13-17 is an example of a cycle-driven DMS file and a different method than illustrated in figure 13-16 for audit and recovery. The function of this RPG program is to sequentially read an entire data set and update one of its fields. The field CURR is added to the field OVRDUE and the new value of OVRDUE is stored in the data set record (first ACCTREC output record of the Output-Format Specifications, figure 13-17). In the event of a failure, the program must know which records have been updated and which have not, so that the updating process can continue at the correct point. Updating the same record twice results in an incorrect value for the field OVRDUE.



Burroughs RPG DATA BASE SPECIFICATION

```
DCYCLE               U
```

Burroughs RPG FILE DESCRIPTION SPECIFICATION

```
01FACCTREC UPE F                       DMS
02FRESTART R                                                   A
```

Burroughs RPG INPUT SPECIFICATION

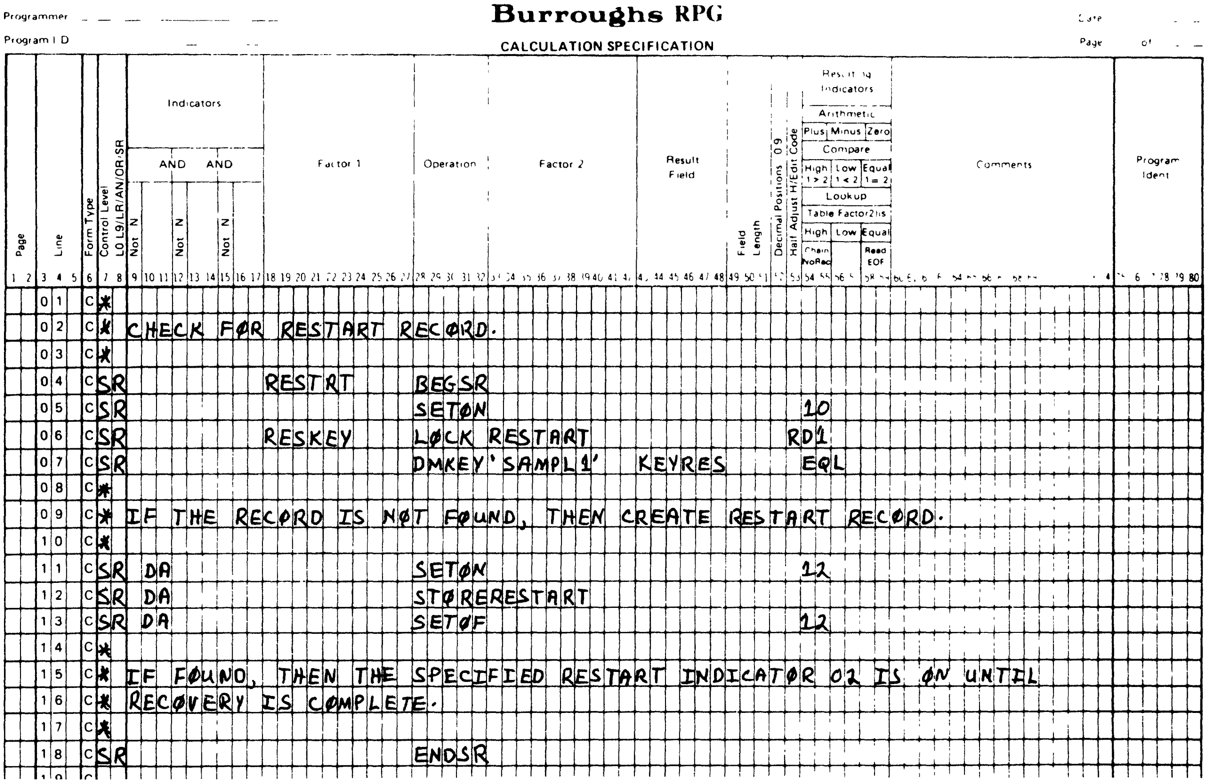
```
01IACCTREC NS 01
02I                                        CURR      CURR
03I                                        ØVRDUE    ØVRDUE
04IRESTART NS 02
05I                                        RRNUM     RRNUM
```

Burroughs RPG CALCULATION SPECIFICATION

```
01C  N10               EXSR RESTART
02C   01     1         ADD  NUMREC    NUMREC  40
03C   02     NUMREC    CØMP RRNUM                 11
04C   11               SETØF                      0211
05C   02               GØTØ END
06C*
07C* DATA BASE IS RECØVERED AND THE FILES ARE REALIGNED, PRØCESSING
08C* CØNTINUES.
09C*
10C          CURR      ADD  ØVRDUE    ØVRDUE
11C          END       TAG
12C*
13C* AT LR TIME, DELETE THE RESTART RECØRD, SINCE THE JØB IS CØMPLETE
14C* IT IS NØ LØNGER NEEDED.
15C*
16CLR                  DELETRESTART
```

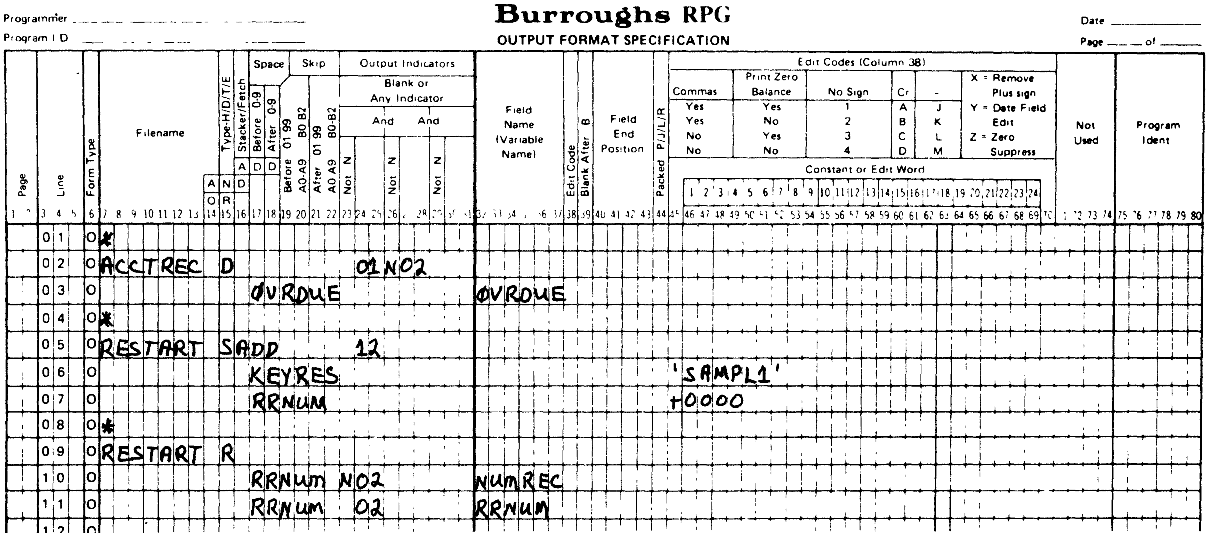
G12085/SHEET 1 OF 2

Figure 13-17. Program Example of Restart with a Primary DMS File (Sheet 1 of 2)



Programmer
Program I D
Burroughs RPG
CALCULATION SPECIFICATION
Date
Page of
Page
Line
Form Type
Control Level L0 L9/LR/AN/OR/SR
Indicators
AND AND
Not N
Factor 1
Operation
Factor 2
Result Field
Field Length
Decimal Positions 0-9
Half Adjust H/Edit Code
Resulting Indicators
Arithmetic
Plus Minus Zero
Compare
High 1 > 2 Low 1 < 2 Equal 1 = 2
Lookup
Table Factor2 is
High Low Equal
Chain NoRec
Read EOF
Comments
Program Ident

```
  01 C*
  02 C* CHECK FØR RESTART RECØRD.
  03 C*
  04 CSR         RESTRT    BEGSR
  05 CSR                   SETØN                     10
  06 CSR         RESKEY    LØCK RESTART             RD1
  07 CSR                   DMKEY'SAMPL1'  KEYRES     EQL
  08 C*
  09 C* IF THE RECØRD IS NØT FØUND, THEN CREATE RESTART RECØRD.
  10 C*
  11 CSR DA                SETØN                     12
  12 CSR DA                STØRERESTART
  13 CSR DA                SETØF                     12
  14 C*
  15 C* IF FØUND, THEN THE SPECIFIED RESTART INDICATØR 02 IS ØN UNTIL
  16 C* RECØVERY IS CØMPLETE.
  17 C*
  18 CSR                   ENDSR
  19 C
```



Programmer
Program I D
Burroughs RPG
OUTPUT FORMAT SPECIFICATION
Date
Page of
Page
Line
Form Type
Filename
Type-H/D/T/E
Stacker/Fetch
Space Before 0-9 After 0-9
Skip Before 01 99 A0-A9 B0 B2 After 01 99 A0 A9 B0-B2
Output Indicators
Blank or Any Indicator
And And
Not N
Field Name (Variable Name)
Edit Code
Blank After B
Field End Position
Packed P/J/L/R
Edit Codes (Column 38)
Commas Print Zero Balance 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
Not Used
Program Ident

```
  01 O*
  02 OACCTREC D        01N02
  03 O          ØVRDUE         ØVRDUE
  04 O*
  05 ORESTART SADD     12
  06 O          KEYRES                      'SAMPL1'
  07 O          RRNUM                       +0000
  08 O*
  09 ORESTART R
  10 O          RRNUM    N02   NUMREC
  11 O          RRNUM    02    RRNUM
  12 O
```

G12085/SHEET 2 OF 2

Figure 13-17. Program Example of Restart with a Primary DMS File (Sheet 2 of 2)

Instead of a STATUS field, as in figure 13-16, the restart data set named RESTART in this example uses a different method for determining whether or not a failure occurred. If the restart record is present at the beginning of job, then the previous run aborted. If the restart record is not present at the beginning of job, then the previous run was successful. The program adds the restart record to the restart data set named RESTART which has an initial RRNUM value of zero (first output record of the Output-Format Specifications for the restart data set named RESTART, figure 13-17).

As each record is updated, the value of NUMREC is incremented by 1 and stored in the RRNUM field of the restart record (third output record of the Output-Format Specifications, for the restart data name RESTART, figure 13-17). At the End of Job, the entire restart record is deleted.

At beginning-of-job, the RPG program checks for a restart record. This is performed by the subroutine RESTRT. If the restart record is not found, indicators D1 and DA are set ON and the restart record is initialized and added. If the restart record is found, the previous run has aborted and the restart record is required to realign the files. The record identifying indicator 02 is set ON when the restart record is read. As long as indicator 02 remains ON, the only action taken is to add 1 to the counter for each input record, NUMREC, and compare it to RRNUM in the restart record. If NUMREC exceeds RRNUM, indicator 02 is set OFF and normal processing continues.

The RPG compiler generates an implied TRBEG operation at the beginning of each cycle and an implied TREND operation at the end of each cycle. The implied TREND operation uses the Output-Format Specification line with the letter R in column 15 for the data set (third output record of the Output-Format Specifications, for the restart data set named RESTART, figure 13-17).

DASDL INFORMATION AND SUGGESTIONS

The following information and suggestions can be used when compiling the DASDL source program with the DASDL compiler.

a. Include a $ RPGLIB DASDL compiler dollar sign option when compiling the DASDL source statements. This option informs the DASDL compiler to create RPG library files to be used by the RPG compiler. These library files reside on disk with the following naming convention:

   &<data base name>/<filename>

   The ampersand (&) appears before the data base name portion of the two-name filename. The ampersand designates the file as an RPG library file. There is one library file created on disk for each disjoint data set declared in the DASDL source statements.

b. Include a $ RPG DASDL compiler dollar sign option when compiling the DASDL source statements. This option informs the DASDL compiler to check for RPG syntax. For example, all data names declared cannot exceed six characters in length.

c. [redacted]

d. If a restart data set is specified in the RPG program, the DASDL source program must specify the same restart data set, and the AUDIT option must be specified.

GLOSSARY OF COMMONLY USED DATA MANAGEMENT TERMS

Refer to the B 1800/B 1700 Systems Data Management System (DMSII) Reference Manual, Form No. 1089794, for a complete description of the following data management terms.

ABORT Exception

An ABORT exception is returned to a program when another program aborts or goes to end of job while in transaction state. At this point, the DMS recovery routines may have backed out some of the program's transactions. The program must be restarted in order to recover the backed out transactions.

AUDITERROR Exception

An AUDITERROR exception occurs when any of the following occur:

a. A program attempts an invalid sequence of begin or end transaction operations. That is, two begin transactions or two end transactions in a row.

b. A program attempts a begin or end transaction when the AUDIT option is not set in the data base.

c. A program attempts to update a data base outside of transaction state.

d. A program attempts to close the data base while in transaction state.

CLOSEERROR Exception

A CLOSEERROR exception occurs when a program attempts to close a data base which is not open.

Contention Limit

Contention limit is the amount of time that a program attempting to lock a record waits on another program which has that record locked. If that amount of time is exceeded, a DEADLOCK condition results.

DASDL

DASDL is an acronym for Data And Structure Definition Language. Compilation of a DASDL source deck with the DASDL compiler defines the structure and type of data allowed in a data base.

DATAERROR Exception

A DATAERROR exception occurs when any of the following occur:

a. A program attempts to store a record with a null key or null required item.

b. A program attempts to store a null record.

c. A program attempts to store a record and a DASDL verify condition is not met.

Data Set

A data set is a collection of related records. Only data sets and embedded data sets have records.

DEADLOCK Exception

A DEADLOCK exception is returned to a program following resolution of a deadly embrace. DMS automatically performs a FREE operation on all locked records for the program.

Deadly Embrace

A deadly embrace occurs when all of the following occur:

a. Two or more programs are accessing the same structure(s).

b. Each program owns a set of locked records.

c. Each program requests access to a record locked by one of the other programs.

When a deadly embrace occurs, a DEADLOCK exception is returned to the lowest priority program involved in the embrace.

Defined State

A pointer is in a defined state when it refers to a valid record or path.

Disjoint Data Set

A disjoint data set is a data set which is not an item within another data set. Disjoint sets can only refer to disjoint data sets.

DUPLICATES Exception

A DUPLICATES exception occurs when duplicate records are not allowed in a set or manual subset and one of the following occurs:

a. A program attempts to store a duplicate record in a set.

b. A program attempts to insert a duplicate record in a manual subset.

Embedded Data Set

An embedded data set is a data set which is an item within a data set. An embedded data set can only be referenced by an embedded set on the same level.

Index

An index is a table of pointers to a data set which provides a specified access to a data set.

INUSE Exception

An INUSE exception occurs when a program attempts to delete a record with a non-null embedded structure.

IOERROR Exception

An IOERROR exception occurs when a parity error (I/O error) occurred while trying to read from the data base.

Key

A key is a value used to locate specific records in a data set spanned by a set or referenced by a subset.

KEYCHANGED Exception

A KEYCHANGED exception occurs when a program attempts to store a record when the value of the item used as a key in the set is illegally changed (duplicates not allowed or embedded set).

LIMITERROR Exception

A LIMITERROR exception occurs when a program attempts to store a record in a structure, and the amount of data in the structure exceeds its physical size.

Master Record

A master record is a data set record which has dependent data sets. A master can itself be a record in an embedded data set. An embedded data set cannot be accessed without accessing the master.

NORECORD Exception

A NORECORD exception occurs when any of the following occur:

a. A current record pointer is not valid for an INSRT operation.

b. A current record pointer is not valid for a FIND operation on a manual subset.

c. A current record of master record is not valid.

NOTFOUND Exception

A NOTFOUND exception occurs when any of the following occur:

a. There is no record which satisfies the selection expression for a LOCK or FIND operation.

b. The key value in the record does not match the key of a manual subset.

c. The requested record does not exist in the data set (previously deleted).

d. The current record pointer is undefined.

e. The embedded structure is empty.

NOTLOCKED Exception

A NOTLOCKED exception occurs when a STORE operation is not preceded by a LOCK or another STORE operation.

OPENERROR Exception

An OPENERROR exception occurs when any of the following occur:

a. A program attempts to open a data base which is not initialized.

b. A program attempts to open a data base which is already open.

c. A program attempts to open a data base, and the data base is not the proper level.

d. The data base is not open prior to the first DMS operation.

Path

A path is an access to a data set record. A set is an index of paths. The current path pointer associated with every set and subset (but not data set) refers to the last record accessed by way of that set or subset. Each current path pointer retains its reference until explicitly changed or until the record referenced by the current path pointer is deleted from the data base.

READONLY Exception

A READONLY exception occurs when a program opens a data base as input only, and either a STORE, DELET, INSRT, or REMOV operation is attempted.

Record

A record contains all the information that pertains to an entity. A record is considered to be the current record of a data set if the appropriate current record pointer refers to an existing record in the data base. The current record pointer for a data set is changed by any operation which causes a new record to be written to the data base or read from the data base. Changing the current record pointer automatically unlocks any previously locked record and, if required, locks the new one.

SECURITYERROR Exception

A SECURITYERROR exception occurs when a usercode does not satisfy the usercode requirements of the data base.

Set

A set is an index of paths to a data set with a pointer to each record of that data set.

Subset

A subset is an index to records of a data set. If a manual subset exists in a data base, the specified records of the data set to be referenced must be programmatically inserted into the subset. If an automatic subset exists in a data base, the condition is evaluated at each STORE operation, and if satisfied, DMS does the insertion.

SYSTEMERROR Exception

A SYSTEMERROR exception occurs when too many data bases are open at the same time. The maximum number of data bases which can open at the same time is six.

VERSIONERROR Exception

A VERSIONERROR exception occurs when a program attempts to open a data base, and the program was compiled against a different version of the data base other than the version currently maintained in the DASDL dictionary.

APPENDIX A

B 1800/B 1700 CARD CODES

Figure A-1 shows the zone and digit portions of an 80-column card. The zone for character A is 12, and the digit is 1. The zone for character R is 11, and the digit 9.

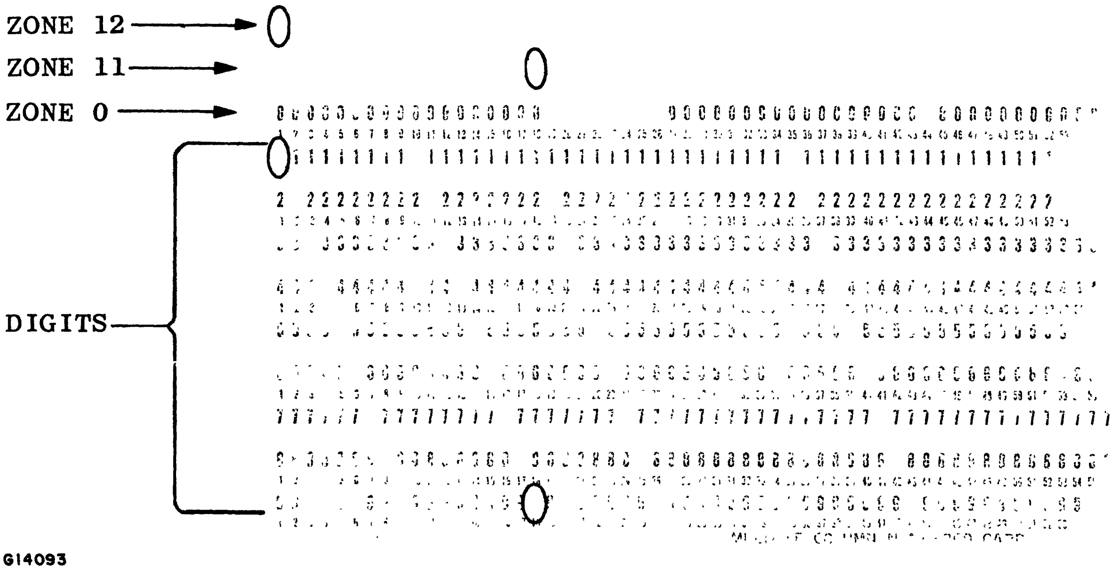
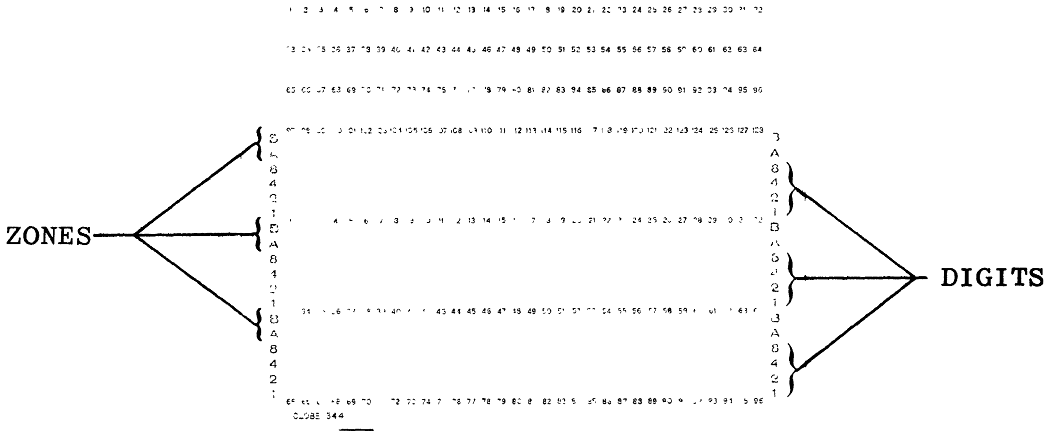


Figure A-1. Zone and Digit Portions of an 80-Column Card

Figure A-2 shows the zone and digit portions of a 96-column card. The zones 12, 11, and 0 on a 96-column card are BA, B, and A, respectively.

Table A-1 lists the 80-column and 96-column card codes for the complete B 1800/ B 1700 character set.



G14094

Figure A-2. Zone and Digit Portions of a 96-Column Card

Table A-1. Punch Card Codes for the B 1800/B 1700 Character Set

Character	EBCDIC 80-Column Card Code	BCL 96-Column Card Code
blank	no punches	no punches
[	12-8-2	B-A-8-2
.	12-8-3	B-A-8-2-1
<	12-8-4	B-A-8-4
(	12-8-5	B-A-8-4-1
+	12-8-6	B-A-8-4-2
|	12-8-7	B-A-8-4-2-1
&	12	B-A
]	11-8-2	B-8-2
$	11-8-3	B-8-2-1
*	11-8-4	B-8-4
)	11-8-5	B-8-4-1
;	11-8-6	B-8-4-2
¬	11-8-7	B-8-4-2-1
-	11	B
/	0-1	A-1
,	0-8-3	A-8-2-1
%	0-8-4	A-8-4
_	0-8-5	A-8-4-1
>	0-8-6	A-8-4-2
?	0-8-7	A-8-4-2-1
:	8-2	8-2
#	8-3	8-2-1
@	8-4	8-4
'	8-5	8-4-1
=	8-6	8-4-2
"	8-7	8-4-2-1
A	12-1	B-A-1
B	12-2	B-A-2
C	12-3	B-A-2-1
D	12-4	B-A-4
E	12-5	B-A-4-1
F	12-6	B-A-4-2
G	12-7	B-A-4-2-1
H	12-8	B-A-8
I	12-9	B-A-8-1
J	11-1	B-1
K	11-2	B-2
L	11-3	B-2-1
M	11-4	B-4
N	11-5	B-4-1
O	11-6	B-4-2
P	11-7	B-4-2-1
Q	11-8	B-8
R	11-9	B-8-1
S	0-2	A-2
T	0-3	A-2-1
U	0-4	A-4
V	0-5	A-4-1
W	0-6	A-4-2
X	0-7	A-4-2-1
Y	0-8	A-8
Z	0-9	A-8-1
1	1	1
2	2	2
3	3	2-1
4	4	4
5	5	4-1
6	6	4-2
7	7	4-2-1
8	8	8
9	9	8-1
0	0	A

APPENDIX B

HEXADECIMAL VALUES FOR THE B 1800/B 1700 CHARACTER SET

Table B-1 presents the RPG collating sequence, and table B-2 may be used in converting hexadecimal digits to binary code. In table B-1, the zone portion of a character is represented by the first hex digit, and the digit portion of the character is represented by the second hex digit.

Table B-1. B 1800/B 1700 RPG Collating Sequence

Character	Hexadecimal Equivalent
blank	40
[	4A
.	4B
<	4C
(	4D
+	4E
|	4F
&	50
]	5A
$	5B
*	5C
)	5D
;	5E
¬	5F
-	60
/	6A
,	6B
%	6C
_	6D
>	6E
?	6F
:	7A
#	7B
@	7C
'	7D
=	7E
'	7F
A	C1
B	C2
C	C3
D	C4
E	C5
F	C6
G	C7
H	C8
I	C9
!	D0
J	D1
K	D2
L	D3
M	D4
N	D5
O	D6
P	D7
Q	D8
R	D9
S	E2
T	E3

Ascending Order

Ascending Order

Table B-1. B 1800/B 1700 RPG Collating Sequence (Cont)

Character	Hexadecimal Equivalent
U	E4
V	E5
W	E6
X	E7
Y	E8
Z	E9
0	F0
1	F1
2	F2
3	F3
4	F4
5	F5
6	F6
7	F7
8	F8
9	F9

Ascending Order ↓

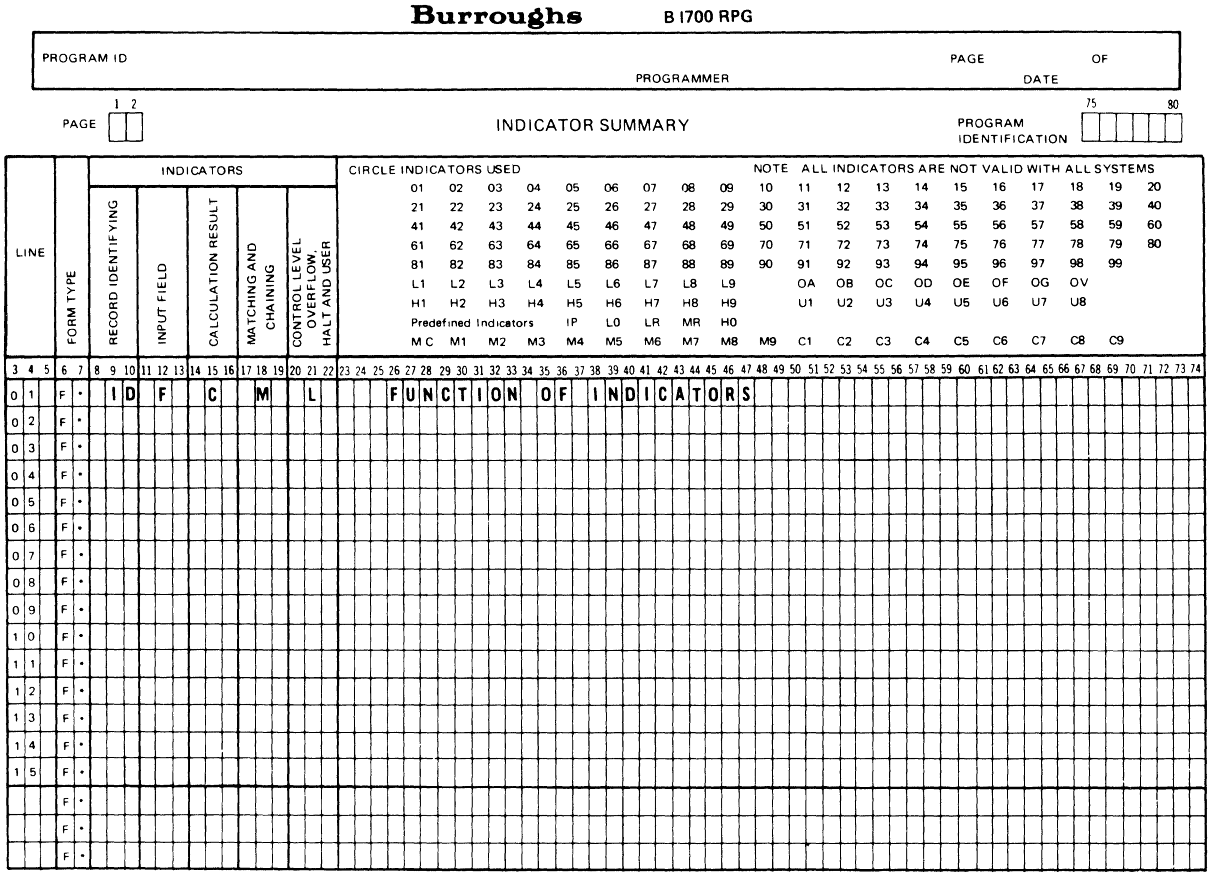
Table B-2. Hex to Binary Conversion Table

Hex Digit	Binary Equivalent
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

APPENDIX C

BURROUGHS INDICATOR SUMMARY FORM

The RPG Indicator Summary Form, illustrated in figure C-1 is used strictly for documentational purposes. Its function is to provide an accurate record of the indicators that are used and the function of those indicators in the RPG Program.



Burroughs B 1700 RPG

PROGRAM ID PAGE OF
PROGRAMMER DATE

PAGE 1 2 □□ INDICATOR SUMMARY PROGRAM IDENTIFICATION 75 □□□□□□ 80

CIRCLE INDICATORS USED NOTE ALL INDICATORS ARE NOT VALID WITH ALL SYSTEMS

01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	
L1	L2	L3	L4	L5	L6	L7	L8	L9		OA	OB	OC	OD	OE	OF	OG	OV		
H1	H2	H3	H4	H5	H6	H7	H8	H9		U1	U2	U3	U4	U5	U6	U7	U8		
Predefined Indicators				IP	LO	LR	MR	HO											
MC	M1	M2	M3	M4	M5	M6	M7	M8	M9	C1	C2	C3	C4	C5	C6	C7	C8	C9	

		INDICATORS					
LINE	FORM TYPE	RECORD IDENTIFYING	INPUT FIELD	CALCULATION RESULT	MATCHING AND CHAINING	CONTROL LEVEL OVERFLOW, HALT AND USER	
3 4 5	6 7	8 9 10	11 12 13	14 15 16	17 18 19	20 21 22	23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
0 1	F •	ID	F	C	M	L	FUNCTION OF INDICATORS
0 2	F •						
0 3	F •						
0 4	F •						
0 5	F •						
0 6	F •						
0 7	F •						
0 8	F •						
0 9	F •						
1 0	F •						
1 1	F •						
1 2	F •						
1 3	F •						
1 4	F •						
1 5	F •						
	F •						
	F •						
	F •						

G14095

Figure C-1. Burroughs Indicator Summary Form

The indicator Summary Form fields and their content are:

Column	Content
6	Form type (optional predefined F).
7	Asterisk required - causes complete card to be commented.

Column	Content
8-10	Record identifying indicator.
11-13	Input field indicators.
14-16	Calculation resulting indicators.
17-19	Matching and chaining indicators.
20-22	Control level, Overflow, Halt, and User indicators.
23-74	Function of the indicators described in columns 8-22.
75-80	Program ID.

APPENDIX D

COMPILE-TIME ERROR AND DIAGNOSTIC MESSAGES

GENERAL

G085 - FILE NAME NEVER DEFINED
A file name has been referenced that has never been defined in File Specification.

G095 - INVALID AND/OR LINE SPECIFICATION

G304 - U1 - U8 MAY BE SET EXTERNALLY ONLY
These external indicators may not be set programmatically and should only be used to indicate at execution time whether or not the file is to be used by the program.

G306 - INDICATOR NEVER DEFINED
This message is caused by an indicator being tested that has never been set.

G315 - FIELD NAME NEVER DEFINED
This message is caused by an undefined variable name being referenced.

G524 - CHAINING KEY LENGTH NOT EQUAL CHAINED FILE KEY LENGTH
This message indicates that the key length specified on the File Specification does not agree with the variable used to initiate chaining.

G999 - DATA NOT NEEDED/EXPECTED IN THIS CARD

FILE DESCRIPTION SPECIFICATIONS

Refer to Section 4 for further information.

F023 - INVALID FILE NAME
The message is caused by an invalid file name in columns 7-14. A file name must be unique in the first seven characters.

F024 - REDEFINED FILE NAME
The file name specified in columns 7-14 does not contain a unique name in the first seven characters.

F026 - INVALID FILE TYPE
This message is caused by an illegal entry in column 15 of the File Specification form. Valid entries are I, O, U, C, and D.

F028 - INVALID FILE DESIGNATION
This message is caused by an illegal entry in column 16 of the File Specification form. Blank must be used for output files; otherwise one of the following entries are required: P, S, C, T, or D.

F033 - NO SEQUENTIAL INPUT FILE FOUND
A primary or secondary file is required.

F036 - INVALID END OF FILE ENTRY
This message is caused by an invalid entry in column 17. Valid entries are blank or E.

F039 - INVALID SEQUENCE ENTRY
This message is caused by an invalid entry in column 18. Valid entries are A, D, or blank.

F042 - INVALID BLOCK-RECORD LENGTH
When the Block and Record length contain blanks, a default of 132 is assumed for print files, and 80 for all other devices. When the block length is blank, it will assume the record length. When the block and record length contain valid numbers, the block length must be an integral part of the record length.

F044 - MULTI BUFFERS AND STACKER SELECTION ON SAME FILE
Stacker selection may not be used when multiple buffers are specified, owing to the processing of the records from the buffer areas.

F044 - INVALID ENTRY TYPE OF ORGANIZATION
This message is caused by an invalid entry in column 32. Valid entries must contain I, 1-9, or blank.

F046 - INVALID ENTRY ADDITION-UNORDERED
This message is caused by an invalid entry in column 66. Valid entries are U, A, or blank.

F049 - INVALID/MISSING EXTENSION CODE
This message is caused by an invalid entry in column 39. Valid entries are blank or E for table files; L may be specified for print files.

F052 - INVALID DEVICE
This message is caused by an invalid device specified in columns 40-46. Valid entries are READER, MFCU1, PUNCH, PRINTER, PRINTR2, TAPE, DISK, SPO, or CONSOLE.

F056 - INVALID FILE FORMAT/FILE CLOSE
This message is caused by an invalid entry in column 19. Valid entries are F, V, or blank.

F057 - INVALID INDICATOR MUST BE U1-U8
This message is caused by an invalid entry in columns 71-72. Valid entries are U1-U8 or blank.

F400 - INVALID ENTRY MODE
This message is caused by an invalid entry in column 28. This field must contain R or blank.

F404 - INVALID ENTRY ADDRESS TYPE
This message is caused by an invalid entry in column 31. Valid entries are A, K, P, N, or blank. On indexed files, this field specifies the format of the key; it must be left blank if the file is not indexed.

F407 - INVALID ENTRY ADDITION-UNORDERED

F543 - INVALID KEY LENGTH ENTRY
The key length is required when using indexed processing. When used, this field (columns 29-30) must contain a valid number from 1-99, and the key length plus the key start location must not exceed the record length.

F549 - INVALID KEY START LOCATION
The key start location field is required when using indexed processing. This field (columns 35-38) must contain a valid number, and the key start location plus the key length must not exceed the record length.

F553 - INVALID KEYS IN CORE ENTRY
The keys in core entry (columns 60-65) must contain a valid number not exceeding 9,999.

EXTENSION SPECIFICATIONS

Refer to Section 5 for further information.

E061 - CHAINING FIELD FOR THIS LEVEL NOT DEFINED ON INPUT
This message indicates that automatic chaining has been specified, and the corresponding level has not been specified on input.

E062 - INVALID FROM FILE NAME ENTRY
This message is caused by an invalid entry in columns 11-18. It is used as the file name of every execution-time vector, but must be left blank if the vector is to be loaded at compile time or via Input or Calculation Specifications.

E063 - INVALID TO FILE NAME ENTRY
This message is caused by an invalid entry in columns 19-26. It is used as the file name of a vector file that is to be written or punched. A valid file name must be used as well as being previously defined on the File Description Specification form.

E067 - INVALID VECTOR NAME
This message is caused by an invalid table or array name in column 27-32. Valid vector names cannot exceed six characters and must be unique. Table names must begin with the letters TAB.

E068 - INVALID ENTRY NUMBER OF ENTRIES PER RECORD
This message is caused by an invalid entry in columns 33-35. This field must contain a valid number. This entry times the length of Vector A plus the length of Vector B (if applicable) must not be greater than the From File length or (if applicable) 96.

E070 - INVALID ENTRY NUMBER OF ENTRIES PER VECTOR
This message is caused by an invalid entry in columns 36-39. This entry must be a valid number limited only to the size of the field.

E072 - INVALID ENTRY LENGTH OF ENTRY
This message is caused by an invalid entry in columns 40-42/52-54. This entry must be a valid number specifying the length of the entry of the vector. Alpha entries must not exceed 511 and numeric entries 31.

E074 - INVALID ENTRY PACKED
This message is caused by an invalid entry in column 43/55. Valid entries are blank and P. P is only allowed on numeric fields.

E076 - INVALID DECIMAL POSITIONS
This message is caused by an invalid entry in column 44/56. Valid entries are 0-9 or blank.

E077 - INVALID SEQUENCE ENTRY
This message is caused by an invalid entry in column 45/57. Valid entries are A, D, or blank.

E079 - INVALID ALTERNATE VECTOR NAME
This message is caused by an invalid table or array name in columns 46-51. Valid vector names cannot exceed six characters and must be unique. Table names must begin with the letters TAB.

E122 - REDEFINED VECTOR NAME
This message is caused by a vector name that has been redefined. Vector names must be redefined using the same length, type, and decimal positions as used on the first definition.

E228 - INVALID/OUT OF BOUNDS LITERAL -- VECTOR, LITERAL --
Literals must contain digits 0-9 and cannot exceed the size of the vector.

E315 - INVALID/UNDEFINED FIELD NAME -- VECTOR, FIELD NAME --
This message is caused by an illegal field name or a field name that has not been defined.

LINE COUNTER SPECIFICATIONS

Refer to Section 6 for further information.

L085 - INVALID FILE NAME OR FILE TYPE
This message indicates that a file name in columns 7-14 is not valid or has not been assigned to a print file. The file name must also have been previously defined on the File Specification.

L088 - WHEN DEFINED -FL- MUST BE IN COL 18-19

L090 - INVALID CHANNEL ENTRY
This message is caused by an illegal channel entry. Valid entries are 01-12, OL, or FL.

L091 - INVALID LINE ENTRY EXCEEDS FORM LENGTH OR 112
The line position entry must not exceed the form length or 112.

INPUT SPECIFICATIONS

Refer to Section 8 for further information.

I084 - RECORD LINE ASSUMED 43-70 TREATED AS BLANK

I092 - INVALID FILE NAME
This message indicates that the file name in columns 7-14 is not valid or has not been assigned to an input file. The file name must also have been previously defined on the File Specifications.

I092 - FILE NOT DEFINED AS INPUT
The file type in column 15 of the File Specification form has not been specified as I, U, C, or D.

I093 - FIELD LINE ASSUMED 7-42 TREATED AS BLANK

I094 - RECORD LINE WITH FILE MUST BE FIRST INPUT SPEC
The first Input Specification must contain a file name.

I097 - NO FIELDS DEFINED IN LAST RECORD

I101 - SEQUENCE FIELD SPECIFICATION
This message is caused by an invalid entry in columns 15-16. It must contain a valid sequence entry, either 2 alpha characters other than ND or Rb, or a numeric entry from 01-99 in ascending order within the file.

I102 - INVALID ENTRY -NUMBER-
The number position entry in column 17 must contain 1 or N if the sequence entry is numeric, or blank if alphabetic.

I103 - INVALID ENTRY -OPTION-
This message is caused by an invalid entry in column 18. Valid entries are 0 and blank. The 0 may only be used on numerical entries, and one numerical entry should be non-optional.

I107 - INVALID RECORD ID CODE
This message is caused by an invalid entry between 21-41. The position entry must fall within the record length.

I109 - INVALID ENTRY -STACKER-
This message is caused by an invalid entry in column 42. Valid entries are blank and 1-6 on card files.

I111 - INVALID PACKED ENTRY
This message is caused by an invalid entry in column 43. Valid entries are P or blank. P may be specified only on numeric field lines, and if specified, column 52 (decimal positions) must not be blank.

I113 - INVALID FROM TO ENTRY
The FROM and TO entries must be valid numbers and cannot exceed the record length. Numeric fields must not exceed 31. Alpha fields must not exceed 511. Vector load elements must be the same length as the element defined on the Extension Specifications, and the entire array load must not be greater than the array length and modulo the element length.

I118 - INVALID FIELD NAME
This message is caused by an invalid field name in columns 53-58.

I119 - INVALID CONTROL LEVEL INDICATOR
This message is caused by an invalid entry in columns 59-60. Valid entries are blank and level indicators L1-L9.

I120 - INVALID ENTRY MATCHING-CHAINING FIELD
This message is caused by an invalid entry in columns 61-62. Valid entries are blank, M1-M9, and C1-C9.

I122 - REDEFINED FIELD NAME - LOOKAHEAD
A look-ahead field name must be unique.

I122 - REDEFINED FIELD NAME
When used, a duplicate field name must contain the same length and decimal positions entry as the original.

I159 - INVALID RECORD INDICATOR
This message occurs when columns 19-20 do not contain a valid indicator or look-ahead.

I166 - INVALID FIELD RESULTING INDICATOR
This message occurs when columns 65-70 do not contain blanks or a valid indicator.

I170 - INVALID ENTRY FOR CHAIN OR DEMAND FILE
Chain or demand files may not contain control levels or matching fields.

I172 - INPUT FILE ORDER IS DIFFERENT FROM FILE SPEC
The input files must be in the same order as defined on the File Specifications.

I181 - REDEFINED LENGTH OF MATCHING-CHAINING FIELD
REDEFINED LENGTH OF CONTROL FIELD.
This message indicates that the total length of a given level on the current record is not identical to the total length of the same level of a previous record.

I188 - INVALID RECORD RELATION INDICATOR
This message is caused by an invalid entry in columns 63-64. This field must contain a valid indicator or blank.

I226 - INVALID ENTRY DECIMAL POSITIONS
The decimal positions entry in column 52 must contain a blank or 0-9. On vector loads, the decimal positions entry must be the same as the defined element.

I575 - UNREFERENCED OR MISGROUPED REFERENÇE OF FILE NAME
This message indicates that an Input Specification has been referenced that has not been declared in the File Specification, or that the file names appear more than once and not consecutively, or that it does not appear in the same order as the File Specification.

CALCULATION SPECIFICATIONS

Refer to Section 9 for further information.

C122 - REDEFINED FIELD NAME
This message indicates that the name defined in the result field has been previously defined with different length and decimal positions.

C122 - REDEFINED FIELD NAME - LOOKAHEAD
This message indicates that the name defined in the result field has been previously defined as a look-ahead field with different length and decimal positions.

C123 - INVALID CONTROL LEVEL INDICATOR
This message is caused by an invalid entry in columns 7-8. Valid entries are L0-L9, LR, AN, OR, and, when in a subroutine, SR.

C125 - INVALID FACTOR 1
The field in columns 18-27 containing Factor 1 must be present if required, or absent if not required as specified by the Operation Code. When present, it must be a legal variable.

C128 - INVALID OPERATION CODE
The operation code specified in columns 28-32 is not legal.

C131 - INVALID FACTOR 2
The field in columns 33-42 containing Factor 2 must be present if required, or absent if not required as specified by the operation code. When present, it must be a legal variable.

C135 - INVALID RESULT FIELD
The field in columns 43-48 containing the result field must be present if required, or absent if not required as specified by the operation code. When present, it must be a legal variable but may not contain a literal.

C135 - INVALID RESULT FIELD NAME

C137 - INVALID RESULT FIELD LENGTH
This message is caused by an invalid entry in columns 49-51. The result field length must contain a valid number between 1-31 for numeric fields or between 1-511 for alpha fields.

C138 - INVALID DECIMAL POSITIONS
This message is caused by an invalid entry in column 52. Valid entries are blank and 0-9.

C140 - INVALID ENTRY HALF ADJUST
This message is caused by an invalid entry in column 53. Valid entries are blank and H. H may only be used with arithmetic operators.

C190 - INVALID BEGSR/ENDSR RELATIONSHIP
This message is used in conjunction with subroutines. Each subroutine must begin with a BEGSR and end with an ENDSR, and may not contain another subroutine.

C200 - INVALID RESULTING INDICATOR
This message is caused by an invalid indicator or one that should not have been specified.

C207 - INVALID ALPHA LITERAL
Alpha literals must be contained in apostrophes and must follow the rules for forming literals.

C207 - INVALID NUMERIC LITERAL
This message is caused by an invalid numeric literal. Numeric literals start with +, -, ., or 0-9 and may contain only one comma or one decimal point, with no embedded blanks, and only digits 0-9.

C214 - INVALID GOTO - TAG OR EXSR - BEGSR RELATIONSHIP
GOTO operations are valid only in conjunction with TAG and ENDSR labels. EXSR operators must be associated with a BEGSR.

C215 - FACTOR 1 AND FACTOR 2 BOTH LITERALS
It is illegal to have literals in both Factor 1 and Factor 2.

C221 - RESULT FIELD MAY NOT BE LARGE ENOUGH
This warning occurs when an overflow condition is likely to occur, which would cause high order digits to be lost.

C232 - INVALID OR DUPLICATE TAG, BEGSR, OR ENDSR NAME

C304 - INVALID INDICATOR
A conditioning indicator must be specified on AND/OR lines and must be a valid indicator.

C519 - INVALID AN - OR ENTRY
This message is caused by an illegal entry. The last of a series of AND/OR lines must contain a calculation operation.

OUTPUT-FORMAT SPECIFICATIONS

Refer to Section 10 for further information.

0097 - NO FIELDS OR LITERALS DEFINED IN LAST RECORD

0142 - RECORD LINE ASSUMED 32-70 TREATED AS BLANK

0143 - INVALID OUTPUT RECORD TYPE
This message is caused by an invalid entry in column 15. Valid entries are H, D, T, or E. This entry may also contain an N or R on AND/OR lines.

0146 - INVALID FILE NAME
This message indicates that the file in columns 7-14 is not valid or that it has not been previously defined on the File Description Specifications as an output file.

0148 - INVALID FIELD NAME
This message is caused by an invalid entry in columns 32-37. A field name previously defined on Input, Extension, or Calculations Specification may be used, or one of the special reserved words may be used.

0150 - INVALID BLANK AFTER ENTRY
The blank after entry in column 39 must contain a blank or B.

0151 - INVALID ENDING POSITION
This message is caused by an invalid entry in columns 40-43. This entry must contain a valid number that must not exceed the record length of the file, and the length of the field must not underflow the record. This must also include the size of the editing symbols when used.

0152 - INVALID PACKED ENTRY
This message is caused by an invalid entry in column 44. Valid entries are blank or P. When P is specified, a numeric variable must be described in columns 32-37.

0154 - RECORD LINE WITH FILE MUST BE FIRST OUTPUT SPEC
The first Output Specification must be a record line containing a file specification.

0154 - FIELD LINE ASSUMED 7-22 TREATED AS BLANK

0212 - EXCPT CALC WITHOUT EXCEPTION OUTPUT RECORD TYPE
The EXCPT operation code has been used when no Exception output records have been defined.

0256 - INVALID STACKER/FETCH ENTRY
This message is caused by an invalid entry in column 16. Valid entries on punch files are 1-6, and on Print files, are F or blank.

0258 - INVALID SPACE/SKIP ENTRY
This message is caused by an invalid entry in columns 17-18 or 19-22. Valid entries for Spacing are blank, 0, or 1-9; for Skipping, 0-99, AO-A9, or BO-B2; and must only be used on Print files.

0273 - INVALID OUTPUT INDICATOR
The output indicators in columns 23-31 must contain a blank or a valid indicator. Valid indicators are 01-99, LO-L9, LR, MR, HO, H9, U1-U8, OA-OG, OV, or 1P.

0276 - INVALID EDIT CODE
Column 38 must contain a blank if the field name field contains an alpha variable or literal; when numeric, it may contain a blank, 1-4, A-B-C-D, J-K-L-M, or X-Y-Z.

0277 - INVALID EDIT WORD
The number of replaceable characters in the edit word (columns 45-70) must be equal to or greater in length than the length of the field to be edited.

0279 - INVALID CONSTANT SIZE

0283 - INVALID FILE TYPE FOR OUTPUT RECORD
The file type has not been defined as O, U, C, or an input/add file.

0289 - *PLACE OR *PRINT PRECEDES ALL FIELDS AND CONSTANTS

0548 - INVALID FILE ADDITION
The file referenced has not been declared as an add file but ADD was specified. ADD will be assumed.

0554 - ADD NOT SPECIFIED ASSUME -ADD-
All files, except update files, using "A" in column 66 in the File Specifications should have "ADD" in columns 16-18 of each record in the output of the corresponding file.

0998 - EDIT TO BE PERFORMED ON ALPHANUMERIC FIELD
Only numeric items may be edited.

0999 - BLANK AFTER INVALID FOR OUTPUT LITERAL
Blank after may not be used after an output literal.

DOLLAR CARD SPECIFICATIONS

Refer to Section 11 for further information.

DOL1 - INVALID ENTRY -NEGATE-
This message is caused by an invalid or illegal entry in column 8. Valid entries are blank or N; however, on some Dollar Card Specifications, it is illegal to use the Not option.

DOL2 - INVALID ENTRY -KEYWORD-
This message is caused by the keyword entry not containing a valid option.

DOL3 - INVALID ENTRY -VALUE-

This message is caused by an illegal entry in the value field of the $ option. Alpha entries must contain a character from A to Z in column 15, and numeric entries must contain blanks or zeroes in columns 15 and 16.

APPENDIX E

EXECUTION-TIME ERROR MESSAGES

Certain conditions may arise during program execution which require operator notification and, in most cases, acknowledgment. Some program errors are recoverable, and some are not. For more information, refer to B 1800/B 1700 Systems, System Software Operational Guide, Form No. 1068731.

INPUT ERROR MESSAGES

The following messages all denote error conditions arising during input, and all are recoverable. For each error condition, the program requests a reply from the operator. A GO response, <program job number>AXGO, causes the program to ignore the erroneous record and read the next record from the same file. A STOP response, <program job number>AXSTOP, causes the program to ignore the erroneous record, set the LR indicator on, and perform all final detail output, total calculations, and total output.

IDENT HALT

The occurrence of the following message results from reading an unidentifiable input record (that is, none of the designated Record Identification Codes for the file could be found in the record read):

```
<program-name> = <program job number> :  IDENT
```

MSEQ HALT

The occurrence of the following message results from reading a record that is out of sequence when matching records are specified (the indicators M1-M9 in columns 61-62 of the Input Specifications). All records in matching files must be in either ascending or descending sequence.

```
<program-name> = <program job number> :  MSEQ
```

SEQ HALT

The occurrence of the following message results from reading a record which is out of sequence, as specified by the entries in columns 15-18 of the Input Specifications:

```
<program-name> = <program job number> :  SEQ
```

REOF HALT

The occurrence of the following message results from reading a demand file which is at End-of-File and no indicator is specified in columns 58-59 of the Calculation Specifications:

```
<program-name> = <program job number> :  REOF
```

READ ERROR HALT

The occurrence of the following message results from reading a file that contains an error (for example, a parity error):

```
<program-name> = <program job number> :  READ ERROR
```

PROGRAMMED HALTS

The following message is displayed at the end of a program cycle if any of the halt indicators (H0-H9) are set. Each n is either blank (indicator not set) or a number 0-9 (indicator set).

```
<program-name> = <program job number> :  HALT nnnnnnnnnn
```

If the H0, H3, H4, H7, and H9 indicators are set, the following message appears:

```
<program-name> = <program job number> :  HALT 0..34..7.9
```

Following this message, the program requests a reply from the operator. A GO response, <program job number>AXGO, turns off all halt indicators and the program continues processing. A STOP response, <program job number>AXSTOP, sets the LR indicator on and the program performs all total calculations and output. If the LR indicator is on when a halt occurs, the program displays the halt message and continues without waiting for an operator response.

FORMS POSITIONING

The following message is displayed after all output lines conditioned by the 1P indicator are printed and FORMS POSITIONING is specified (the number 1 in column 41 of the Control Specification):

```
<program-mix> = <program job number> :  AGAIN?
```

After the message is displayed, the program requests a reply from the operator. A YES response, <program job number>AXYES, causes all output lines conditioned by the 1P indicator to be printed again, and the message is repeated.

A NO response, <program job number>AXNO, causes normal processing to continue.

Printing of all output lines conditioned by the 1P indicator can be requested as many times as necessary in order to align the forms properly. Requests for forms positioning occurs only when the file writes directly to the line printer.

ARITHMETIC ERRORS

DIVIDE BY ZERO HALT

The occurrence of the following message results from an attempt to perform a DIV (divide) operation using a divisor of zero. The MCP automatically discontinues (DS-ED) the program.

```
<program-name> = <program job number> :  DIVIDE BY ZERO
```

SQRT HALT

The occurrence of the following message results from an attempt to perform a SQRT (square root) operation on a negative argument:

<program-name> = <program job number> : SQRT

Following the message, the program requests a reply from the operator. A GO response, <program job number>AXGO, sets the value contained in the RESULT FIELD to zero, and the program continues processing. A STOP response, <program job number>AXSTOP, sets the LR indicator on, and the program performs all final detail, total calculation, and total output.

VECTOR (ARRAY, TABLE) ELEMENT ERRORS - INVALID SUBSCRIPT HALT

The occurrence of the following message results from an attempt to reference a vector element that is out of bounds, that is, an index value less than or equal to zero, or greater than the maximum size of the array as specified in columns 36-39 of the Extension Specifications. The MCP automatically discontinues (DS-ED) the program.

<program-name> = <program job number> : INVALID SUBSCRIPT

PRE-EXECUTION TIME VECTOR - VSEQ HALT

The occurrence of the following message results from reading a pre-execution time vector record which is out of sequence as specified by the contents of the SEQUENCE fields in the Extension Specifications. Any response causes the program to terminate.

<program-name> = <program job number> : VSEQ

SEQUENCE ERROR - KEYSEQ HALT

The occurrence of the following message indicates that the indexed data file has a key which is out of sequence. Any response forces end-of-job. The program's data file requires resorting or the program loading the file or updating the file requires reworking.

<program-name> = <program job number> : KEYSEQ FOR FILE = <filename>

LIMITS FILE - LIMITS PAIR ERROR

When processing a file within limits by means of a limits file and the low record key is greater in value than the high record key, the following message occurs:

<program name> = <program job number> : LIMITS PAIR

MUST BE IGNORED FOR FILE = <filename>

LOW KEY GREATER THAN HIGH KEY

If the limits pairs are entered through the console, then the program waits for the operator to re-enter the correct limits pair before processing continues. If the limits pairs are not entered through the console, the program ignores the erroneous limits pair and reads the limits file for the next limits pair.

INVALID TAG FILE/DATA FILE MESSAGES

The following message occurs when a tag file contains a record which is out-of-bounds:

```
<program name> = <program job number> FILE = <filename>, TAGFILE

POINTS TO INVALID DATA RECORD (KEY = 0 OR KEY GTR EOF)
```

The program terminates with the following response:

```
<program job number>AX
```

APPENDIX F

RPG/BTF PROGRAM

RPG/BTF (Build Tag File) is a utility program which creates a tag file from an existing indexed sequential data file on disk.

RPG/BTF RULES

The following rules apply to RPG/BTF:

a. The data file is not required to be in key sequence order. The tag file is built in the proper sequence.

b. When a duplicate key is encountered in a data file, an error message on the line printer specifies the relative record numbers of both records. In addition, the tag file is not entered in the disk directory.

c. The RPG/BTF program can handle key fields up to 29 bytes in length for indexed files and 99 bytes for B-indexed files. If column 31, RECORD ADDRESS TYPE, of the File Description Specification is left blank, then alphanumeric keys are assumed.

RPG/BTF INPUT

Input to RPG/BTF can consist of the entire RPG source program; however, RPG/BTF ignores all but the following cards:

a. The Control Card Specifications with an "R" or "L" (blank) in column 17. The default is "L" when the Control Card Specifications are not included in the input to RPG/BTF.

b. The $ RSIGN dollar option card. This dollar option must appear prior to each File Description Specification when the indexed data file is right-signed and "R" is not specified in column 17 of the Control Card Specifications. $ RSIGN/$NRSIGN can be used when selected files are right-signed and others are left-signed.

c. The $ PACKID, $ FAMILY, and $ FILEID dollar option card. These dollar options must be correctly related to each indexed file's File Description Specification when the external file name is different from the internal file name.

d. File Description Specifications for each tag file to be built. Tag files are created for File Description Specifications that are declared as input, update, or output with additions. Each File Description Specification describes the indexed data file to the RPG/BTF program. This gives the RPG/BTF program the following information:

   1. Key length in bytes.

2. Key type (alphanumeric, numeric, or packed).

3. Key field starting location of each record in the data file.

All other source cards are ignored.

RPG/BTF FUNCTIONS

The following functions are performed by the RPG/BTF program:

a. First, the RPG/BTF program reads from a card file called RPG/CARD. This card file may contain the entire RPG source program. The information required to build the tag file is obtained from each File Description Specification in the RPG source program that defines an indexed file. Files must be declared input, update, or output with additions.

b. Next, the RPG/BTF program locates each indexed data file on disk in the order in which they appear in the File Description Specifications section of the RPG program. If a data file is not on disk, the RPG/BTF program continues with the next indexed data file described in the RPG source program; the message "FILE NOT ON DISK" appears in the printer error file called TAGFILE/INFO.

c. A tag file is then created for each indexed data file located on disk. Each tag file contains the keys and relative record numbers of each record in its corresponding indexed data file. The keys in the new tag file remain unchanged from the keys in the data file. The relative record number, which is the position number of the record in the data file, also remains unchanged.

d. After the tag file is created, RPG/BTF invokes the SORT/VSORT utility which sorts the tag file records by key in ascending order.

e. After sorting, the RPG/BTF program locks the tag file in the Disk Directory and returns to step a. When the input card file, RPG/CARD, is exhausted, the RPG/BTF program goes to End-of-Job.

Table F-1 illustrates the internal and external file-names of the tag file.

Table F-1. Internal and External File Names of the Tag File and Data File

Data File Internal File Name	Data File External File Name	Tag File Internal File Name	Tag File External File Name
DISK	A/ABCDEFGHIJ	TAGDISK	A/TAGABCDEFG
DISKIN		TAGDISKIN	DISKIN/TAG
DISK	CCC/A/B	TAGDISK	CCC/A/TAGB
DISK	A/ABCDEFGHIJ	TAGDISK	A/TAGABCDEFG
DISK	A/B/	TAGDISK	A/B/TAG

Table F-2 shows the internal and external file names, associated devices, and functions of the files used by the RPG/BTF program.

Table F-2. Internal and External File Names of RPG/BTF

Internal File Name	Device	External File Name	Function or Description
LINE	Printer	TAGFILE/INFO	Error listing for RPG/BTF.
CARDS	Card	RPG/CARD	Input file for RPG/BTF.
DATA.FILE	Disk	BOBB/DATA.FILE	Workfile used to input data file.
TAG.FILE	Disk	BOBB/TAG.FILE	Workfile used to build and sort tag file.

The following example illustrates the cards necessary to build tag files for indexed files INDEX and USER/RPG/TEST.

Example:

```
? EXECUTE RPG/BTF
? DATA RPG/CARD

     H (optional)

     FINDEX    IPE F 180  60  5PI     10 DISK

      $ PACKIDUSER
      $ FAMILYRPG
      $ FILEIDTEST

     FTEST      IS F 180 180 30AI    125 DISK

        (Remainder of RPG source deck is not required)

?END
```

APPENDIX G

TAGSORT WITH RPG

TAGSORT is a method of sorting records in an input file by creating a tag file by which the input file is sorted. The tag file is placed on disk and may be referenced by any RPG program which contains the original input file for which the tag file was created.

The method for creating a tag file is discussed in the B 1800/B 1700 System Software Operational Guide, Form No. 1068731.

Basically, the tag file is created by a TAGSORT program which goes through the input file, extracting the key field(s) from the record, and adding the position of the record, relative to the first record in the file. The tag file is then sorted, using the regular sort concept, and leaving a file of relative record numbers (4 bytes in length) sorted in the specified sequence. The original input file remains unchanged.

The record address file (TAGFILE) created by the TAGSORT program can be used in an RPG program to perform record address (ADDROUT) processing. For detailed information, refer to the subsection titled Random Processing - Addrout Files in Section 4.

In the example shown in table G-1, the input file DISKFI is used to build a tag file called TAGFILE. Refer to the B 1800/B 1700 System Software Operational Guide, Form No. 1068731.

Table G-1. Building a Tag File

Contents of DISKFI	TAGSORT Program	Contents of TAGFILE
RECD NO		
00001 456 2222	FILE IN DISKFI (DISK (100) 180 1) OUT TAGFILE (DISK (270) 4 45) KEY (1 3 A A) TAGSORT	00000002
00002 123 0000		00000005
00003 879 1111		00000008
00004 653 3333		00000001
00005 258 4444		00000007
00006 785 5555		00000004
00007 551 6666		00000006
00008 327 7777		00000003
00009 965 8888		00000009

The contents of DISKFI consists of nine records. The first three digits comprise the key field, the last four comprise the data field.

Since DISKFI was sorted by key in ascending order and record number 2 has the lowest key value, record 2 is placed in the first position of the tag file. Record 5 has the second smallest key (258), so 5 is placed in the second position of the tag file, etc.

APPENDIX H

RPG PROGRAM CYCLE

For each record from a primary or secondary file that is processed, the RPG object program goes through the same general cycle of operations. After a record is read, there are two different instances in time when calculation operations are performed and records written out. First, all total calculation operations (those conditioned by control level indicators in columns 7-8) and all total output operations are done. Second, all detail calculation operations and all detail output operations are done.

Total calculations are performed before the information on the record selected for processing is made available. Detail calculations are performed after the information on the selected record is made available. The following discussion describes this concept in more detail.

Whenever a record is read, a check is made to determine if information in a control field (when one has been specified) is different from the control field information on the previous card. A change in the control field information indicates that all records from a particular control group have been read and a new group is starting. When all records from a group have been read (indicated by control level indicators being ~~set ON~~), operations may be done using information accumulated from all records in that group. It is at this time that all calculations conditioned by control level indicators in columns 7-8 are done. Total output operations are also performed immediately after all total calculation operations are completed. Information on the record read at the beginning of the program cycle is not used in these operations; only information from records in the previous control group is used.

Detail calculations occur after the information on the selected record has been made available. Detail calculations are used to calculate values needed each time a record is processed. They are also used to calculate totals for the current control group (if control fields are specified). Immediately after detail calculation operations are completed, detail output operations are performed.

The specific steps taken in one program cycle are shown in figure H-1. The item numbers in the following description refer to the number in the figure. A program cycle begins with step 3 and continues through step 39.

1. Initialization:

   a. Data fields and indicators may have been preset to their initial values by the compiler; if not they must be initialized now:

      1) All data fields and vectors are initialized to zero (if numeric) or blank (if alphanumeric).

      2) All match fields are initialized to high (if descending sequence) or low (ascending) collating values.

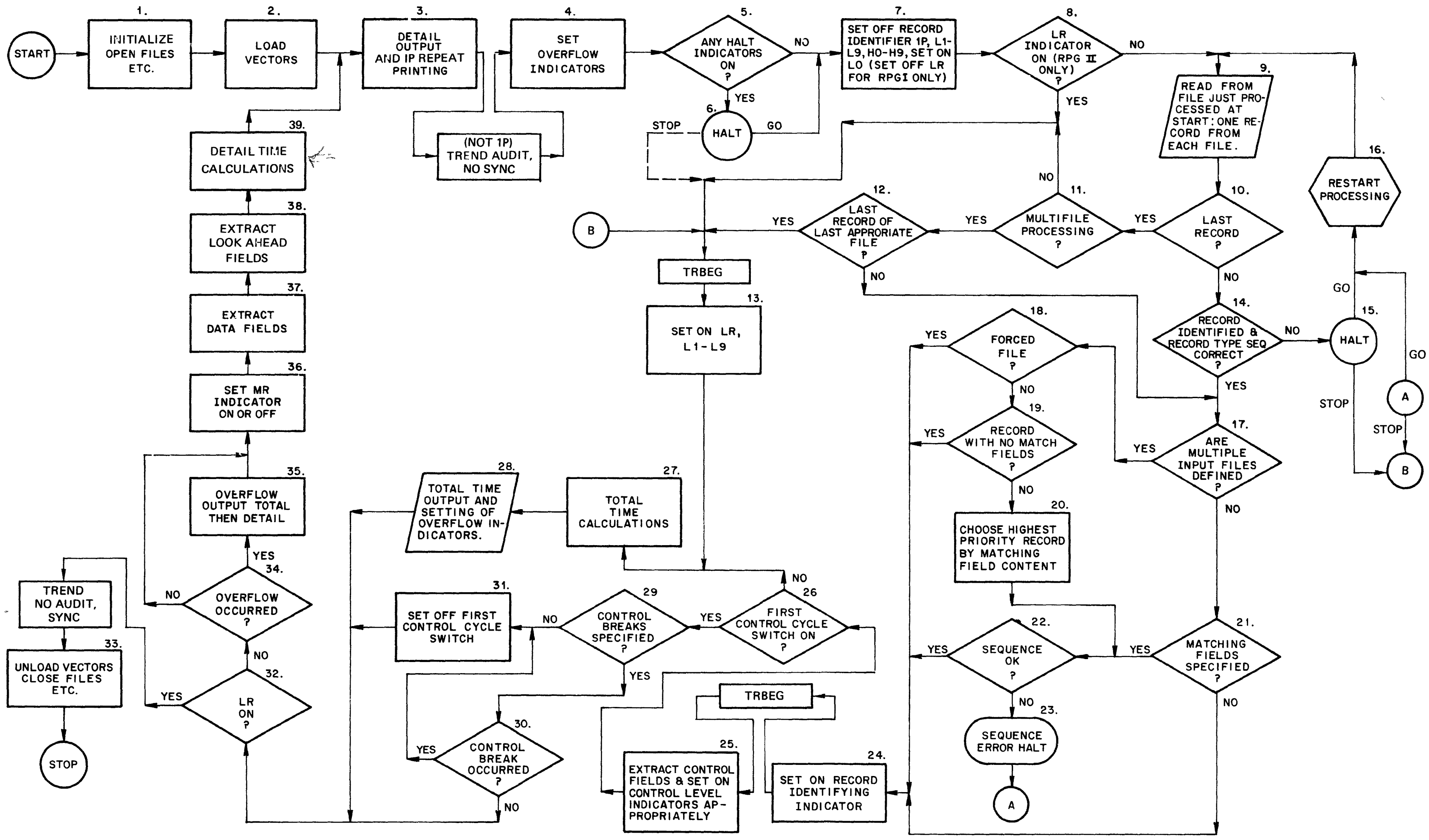

START
1. INITIALIZE OPEN FILES ETC.
2. LOAD VECTORS
3. DETAIL OUTPUT AND IP REPEAT PRINTING
(NOT 1P) TREND AUDIT, NO SYNC
4. SET OVERFLOW INDICATORS
5. ANY HALT INDICATORS ON ?
NO
YES
6. HALT
STOP
GO
7. SET OFF RECORD IDENTIFIER 1P, L1-L9, H0-H9, SET ON LO (SET OFF LR FOR RPGI ONLY)
8. LR INDICATOR ON (RPG II ONLY) ?
NO
YES
9. READ FROM FILE JUST PROCESSED AT START: ONE RECORD FROM EACH FILE.
10. LAST RECORD ?
YES
NO
11. MULTIFILE PROCESSING ?
YES
NO
12. LAST RECORD OF LAST APPRORIATE FILE ?
YES
NO
B
TRBEG
13. SET ON LR, L1-L9
14. RECORD IDENTIFIED & RECORD TYPE SEQ CORRECT ?
NO
YES
15. HALT
GO
STOP
16. RESTART PROCESSING
GO
A
STOP
B
17. ARE MULTIPLE INPUT FILES DEFINED ?
YES
NO
18. FORCED FILE ?
YES
NO
19. RECORD WITH NO MATCH FIELDS ?
YES
NO
20. CHOOSE HIGHEST PRIORITY RECORD BY MATCHING FIELD CONTENT
21. MATCHING FIELDS SPECIFIED ?
YES
NO
22. SEQUENCE OK ?
YES
NO
23. SEQUENCE ERROR HALT
A
24. SET ON RECORD IDENTIFYING INDICATOR
TRBEG
25. EXTRACT CONTROL FIELDS & SET ON CONTROL LEVEL INDICATORS APPROPRIATELY
26 FIRST CONTROL CYCLE SWITCH ON ?
NO
YES
27. TOTAL TIME CALCULATIONS
28. TOTAL TIME OUTPUT AND SETTING OF OVERFLOW INDICATORS.
29 CONTROL BREAKS SPECIFIED ?
NO
YES
30. CONTROL BREAK OCCURRED ?
YES
NO
31. SET OFF FIRST CONTROL CYCLE SWITCH
32. LR ON ?
YES
NO
33. UNLOAD VECTORS CLOSE FILES ETC.
TREND NO AUDIT, SYNC
STOP
34. OVERFLOW OCCURRED ?
YES
NO
35. OVERFLOW OUTPUT TOTAL THEN DETAIL
36. SET MR INDICATOR ON OR OFF
37. EXTRACT DATA FIELDS
38. EXTRACT LOOK AHEAD FIELDS
39. DETAIL TIME CALCULATIONS
G14096

Figure H-1. RPG Program Cycle

3) All control field storage areas are initialized to hexadecimal zeroes.

4) Indicators LO and lP are set ON, all other indicators are set OFF. Under the RPGl dialect any indicators from 01-99 used as zero/blank indicators will be set ON, unless also used as record identifying indicators.

b. UDATE, UDAY, UMONTH, UYEAR fields are set.

c. External indicators are set.

d. Files are opened (unless files are to be opened automatically when first accessed).

e. The first-control-cycle switch is set on. This is an internal switch used to determine when to suppress total time.

2. Pre-execution time vectors are loaded (if specified).

3. Detail Output. All heading and detail output operations whose indicator conditions are met are performed (this will not include output conditioned on overflow indicators). On the very first cycle, the option for forms alignment may have been specified. If so, step 3 (detail output) is repeated as many times as the operator requires. Generally, the programmer will control the heading output required on the first cycle by conditioning it on the indicator lP (which will be off for all cycles except the first).

4. All overflow indicators are set OFF. If any printer file is on the overflow line or between the overflow line and the end of the page, any overflow indicator associated with that file is set on.

5. If any of the halt indicators HO - H9 are on, the program branches to step 6; otherwise, to step 7.

6. A message is displayed indicating which halt indicators are on. The operator has 2 options:

   a. STOP. Program branches to step 13. After performing "last record," total calculations, and total output, the program will terminate.

   b. GO. The program continues at step 7.

7. All record identifying indicators are set OFF. lP, Ll-L9 and HO-H9 are set OFF. LO is set ON. Under the RPGl dialect LR is also set OFF.

8. Under the RPGl dialect, the program will always continue at step 9; otherwise, LR is tested. If LR is ON, the program branches to step 11; otherwise, to step 9.

9. The next input record is read. On the first cycle, one record is read from each primary and secondary file. On all subsequent cycles, the record is read from the file that was processed last. For an input (not update or combined) file with look-ahead fields, the record will already have been read at step 38 of the previous cycle.

10. A test is performed to determine whether the file just read is at End-of-File or is conditioned by an external indicator which is OFF (on the first cycle, the test is whether any of the files read satisfy this condition). If yes, the program branches to step 11; otherwise, to step 14.

11. If this program has a primary file and at least one secondary file (i.e., the program performs multifile processing), the program branches to step 12, otherwise, to step 13.

12. If all required files are at End-of-File (as specified in column 17 of the File Specifications), the program branches to step 13; otherwise, to step 17.

13. Indicators LR and L1-L9 are set ON.

14. The record (or in the case of the first cycle, all the records) read at step 9 is identified (but the record identifying indicator is not yet set ON). A test is performed to determine whether the input records are in the sequence specified on the Input Specifications. If the record type sequence is incorrect (or if sequenced records are specified but the current record cannot be identified), the program branches to step 15; otherwise, to step 17.

15. The record type sequence error causes the program to halt after displaying an appropriate message to the operator.

16. The operator may order the program to resume. In which case, the out of sequence record is ignored and the next record is read by branching to step 9.

17. If multiple input files are specified, the program branches to step 18 in order to select the next record to be processed. If no secondary files are specified, the program continues at step 21.

18. If the FORCE opcode was performed during the previous program cycle, the forced file is selected for processing and the program branches to step 24.

19. The current records from each primary and secondary file are inspected in the order that the files were specified on File Specifications. The first record that has no match fields is selected and the program branches to step 24. If all records have match fields, the program branches to step 20.

20. The record with the highest priority matching field value (highest collating value for descending matching sequence, lowest collating value for ascending sequence) is selected to be processed next. If two or more files have equal and highest priority matching field values, the one with highest priority is selected (priority corresponds to the order in which files were specified on File Specifications).

21. As there are no secondary files, record selection is unnecessary. If the current record from the primary file has match fields, the program branches to step 22 to perform matching sequence checking.

22. The match field value of the selected record is compared to the match field value of the previous record processed. If it is in sequence, the program continues at step 24. If the match field value is out of sequence, the program branches to step 23.

23. The program halts after displaying a message to the operator indicating that a match field sequence error has occurred.

24. The program sets ON the record identifying indicator specified for the selected record. However, data from this record is not available for processing until step 37. If look-ahead fields are specified for update or combined files, they are extracted at this point (all other look-ahead fields are not extracted until step 38).

25. If the selected record has control fields, a test is performed to see if the contents are equal to the contents of the control field storage area. If the contents are unequal, a control break has occurred; the appropriate control level indicators are set ON and the new control field contents are stored in the control field storage area. Note that until the first control break occurs, the control field storage area contains its initial value of hexadecimal zeros.

26. If the first-control-cycle switch is ON, the program branches to step 29; otherwise, to step 27.

27. Total calculations (calculations conditioned by control level indicators L0-L9 in columns 7-8) are performed.

28. Total output that is not conditioned by an overflow indicator is performed. When this output is complete, the program tests to see if the overflow condition has occurred. If any printer file is on the overflow line or between the overflow line and the end of the page, any overflow indicator associated with that file is set ON.

29. If control fields were specified in this program, the program branches to step 30; otherwise, to step 31.

30. If a control break occurred on this cycle (at step 25), the program branches to step 31; otherwise, to step 32.

31. The first-control-cycle switch is set OFF. Total Calculations and Total Output will be performed on all subsequent cycles (if no control break occurs, only L0 operations will be performed).

32. If indicator LR is ON, the program branches to step 33.

33. End-of-Job routine.

    a. Output any vectors specified with a TO FILENAME on the Extensions Specifications.

    b. Perform any file maintenance e.g., sorting an indexed file if unordered or if additions have been made.

    c. Close all files.

34. & 35. Overflow output is performed for each printer file on which overflow has occurred on this cycle. The test and the output performed take one of two forms. If an overflow indicator is assigned to the file on File Specifications or if overflow output is specified on Output Specifications for the file, the test is whether the overflow indicator for that file is ON (it may have been set ON by SETON; it may also have been set OFF if overflow output has been Fetched). If so, any Total and then any Detail output for that file which is conditioned on the overflow indicator is performed (if none is specified, no overflow output will be performed). The overflow indicator is then set OFF.

    If no overflow indicator is assigned and no overflow output is specified for the file, the test is whether the overflow condition was satisfied in steps 4 or 28. If overflow did occur and has not been Fetched, the object program automatically generates a skip to line 1 of a new page.

36. The matching record indicator MR is set ON or OFF. MR is set ON if a match has occurred (i.e., if selected record has a match field and the current match field value was originally selected from a primary record).

37. Data is extracted from the current record and made available for processing. Any field indicators specified are set ON or OFF to reflect the status of the data fields.

38. If the current record is from an input (not update or combined) file which has look-ahead fields, the next record from that file is read, and the look-ahead fields are extracted and made available to the program.

    Note that look-ahead fields from an update and combined file are made available at step 24.

39. Detail time calculations are performed. Processing continues with step 3.

INDEX