# Burroughs

# B 1700 SYSTEMS

## Report Program Generator

## REFERENCE MANUAL

$4.00

Burroughs

# B 1700 Systems
# Report Program Generator

## REFERENCE MANUAL

Burroughs Corporation
Detroit, Michigan 48232

$4.00

Printed in U. S. America 2-75 1057189

Burroughs believes that the information described in this manual is accurate and reliable, and much care has been taken in its preparation. However, no responsibility, financial or otherwise, is accepted for any consequences arising out of the use of this material. The information contained herein is subject to change. Revisions may be issued to advise of such changes and/or additions.

This revision incorporates the information released under the following:
PCN 1057189-001 (August 16, 1974)

Correspondence regarding this document should be forwarded using the Remarks Form at the back of the manual, or may be addressed directly to Systems Documentation, Technical Information Organization, TIC-Central, Burroughs Corporation, Burroughs Place, Detroit, Michigan 48232.

# TABLE OF CONTENTS

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

## TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

LIST OF ILLUSTRATIONS

## LIST OF ILLUSTRATIONS (Cont)

Figure Page No.

## LIST OF ILLUSTRATIONS (Cont)

## LIST OF TABLES

LIST OF TABLES (Cont)

## LIST OF APPLICABLE B 1700 PUBLICATIONS

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

| Publication Title | Title Page Date | Form No. |
|---|---|---|
| B 1700 System Software Operational Guide | 7-73 | 1068731 |
| B 1700 Systems COBOL Reference Manual | 10-72 | 1057197 |
| B 1700 Systems Network Definition Language (NDL) Reference Manual | 4-74 | 1073715 |

# INTRODUCTION

Burroughs RPG (Report Program Generator) is a machine-independent programming language suitable for implementation in a wide variety of data processing applications. 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 1700 to allow optional extensions to the above-mentioned implementations.

Burroughs 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 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.

f. Facilities for program conversion to COBOL (refer to appendix F).

This manual describes Burroughs 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.

This reference manual reflects the MARK IV.1 System Software Release version of the RPG compiler.

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 COBOL 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 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 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 and arrays 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.

TELECOMMUNICATIONS CARD SPECIFICATIONS

The Telecommunications card is used to further define a file specified on the File Description Specifications as a DATACOM or BSCA file. There is no special form for coding Telecommunications Specifications.

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-FQRMAT 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 Section 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. 1055860)

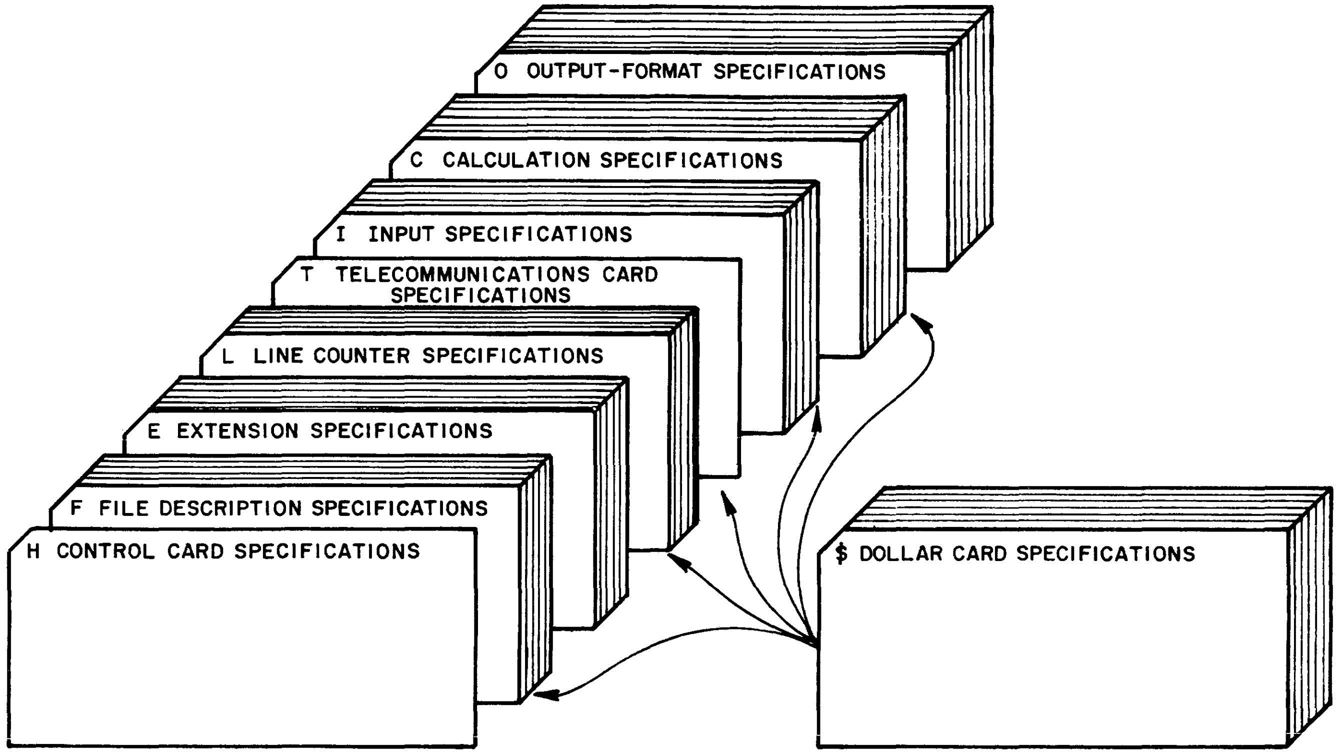
d. Calculation Specifications (Form No. 1055878)

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 Section 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 supported in Burroughs RPG:

    1. Binary data format.

    2. Sterling data format.

    3. Alternate collating sequence.

    4. Record address files.

    5. Inquiry programs.

    6. Spread card or TR format.

    7. Printer keyboard output files.

    8. External assembler subroutines.

    9. Blank after literals.

    10. Redefined field lengths.

    11. Factor 1 and Factor 2 both literals.

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

    13. File translation.

    14. Processing sequential files within limits.

# 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).

### 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:

PAGE
PAGE1
PAGE2
UDATE
UMONTH
UDAY
UYEAR
*PRINT
*PLACE

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

NOTE

The special words JDATE and UTIME have been reserved and, if used, will cause a syntax error.

### 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.

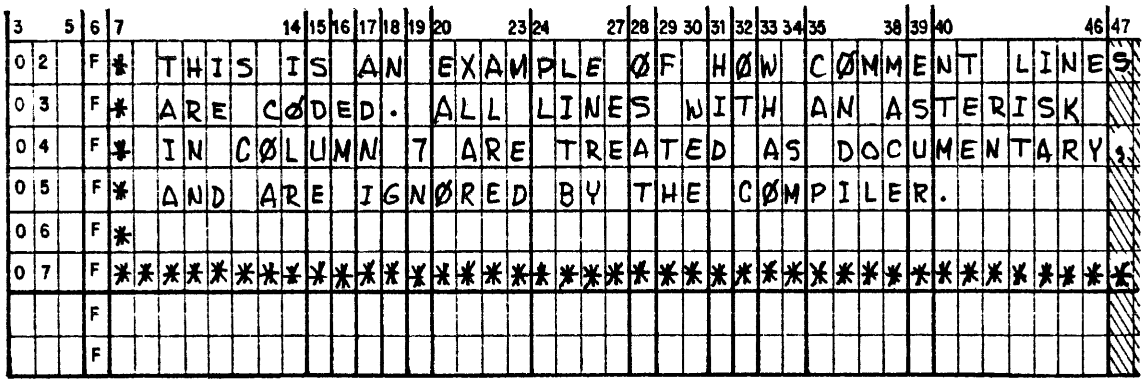
## COMMON FIELD DEFINITIONS

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.

| 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
0 2   F * THIS IS AN EXAMPLE ØF HØW CØMMENT LINES
0 3   F * ARE CØDED. ALL LINES WITH AN ASTERISK
0 4   F * IN CØLUMN 7 ARE TREATED AS DOCUMENTARY
0 5   F * AND ARE IGNØRED BY THE CØMPILER.
0 6   F *
0 7   F *****************************************
      F
      F
```

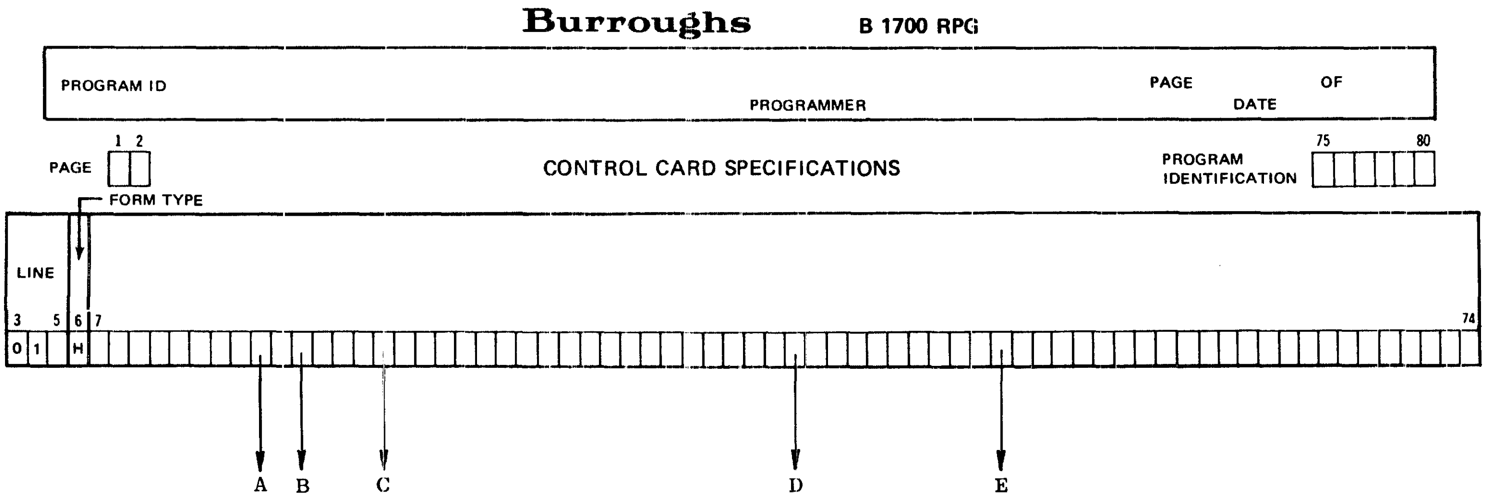
G14005

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 these 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.



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

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

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

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

E. 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

41 FORMS POSITIONING

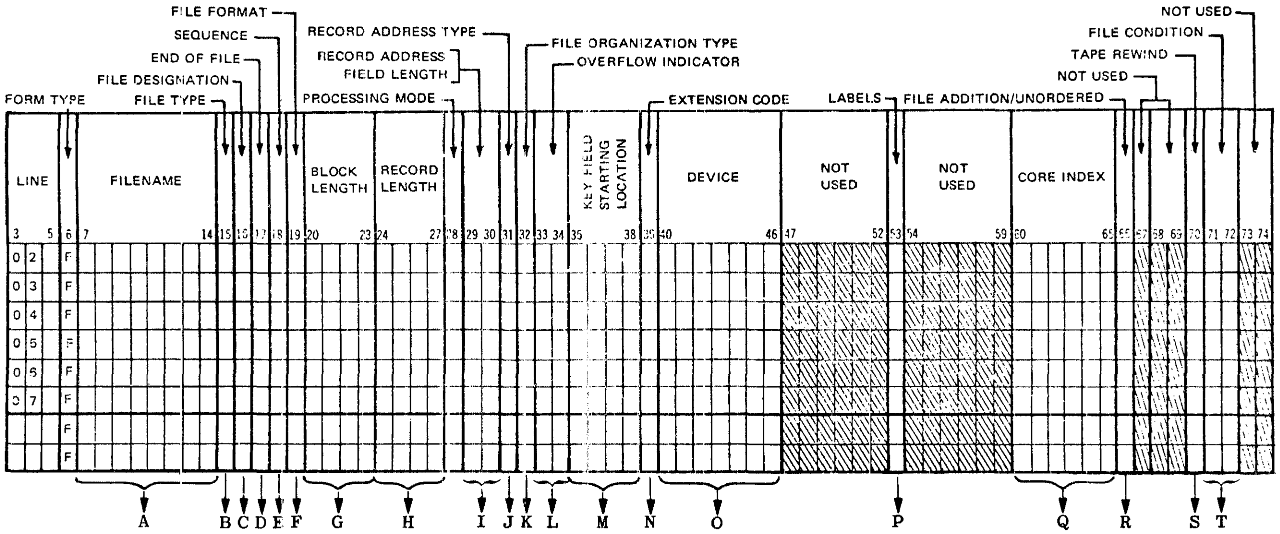
Since it sometimes becomes necessary to align special forms in the line printer before beginning to print a report, a 1 in column 41 will cause all output lines conditioned by the 1P indicator to be printed more than once when the program is executed. Each time the lines are printed, the program is temporarily 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.



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: I, 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, or V.

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

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

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

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

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

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

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

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

O. 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.

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

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

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

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

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

G14007

Figure 4-1. File Description Specifications Summary Sheet

## 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. |

P <u>Primary File</u>

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 <u>Secondary Files</u>

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 <u>Chained File</u>

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. Chained indexed files may only be input or update.

T <u>Table File</u>

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.

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 |
|---|---|
| <mix index> AXGO | Ignore the record out of sequence and read the next record from the same file. |
| <mix index> AXSTOP | Ignore the record out of sequence, turn on the LR indicator, and perform all final detail and total calculation procedures. |
| <mix index> 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 for files other than those assigned to tape. The acceptable entries are:

| Entry | Definition |
|---|---|
| Blank or F | Fixed record length. |
| V | Variable record length (tape files only). |

Table 4-3. Default Block and Record Lengths

| Device | Block and Record Length |
|---|---|
| All Printers | 132 |
| All Disk | 180 |
| DATA 96, MFCU1, MFCU2 | 96 |
| All other devices | 80 |

For a fixed-length file, the length of one block and one record must be entered in the BLOCK LENGTH and RECORD LENGTH fields respectively. 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.

Variable-length records (tape files only) will be defined later.

28 PROCESSING MODE

Any entry in this field is not used by the compiler and should be left blank. Any entry other than blank or R is treated as an error for a disk file. For non-disk files this field must be blank.

29-30 RECORD ADDRESS FIELD LENGTH

This field applies only to indexed files. It specifies the length of the key field within the records of the file. All key fields in the file must be the same length. The key length must not be greater than 28 characters. The numeric entry must be right-justified within the field. Leading zeros may be omitted. The length entered is always the number of bytes occupied by the key field.

31 RECORD ADDRESS TYPE

This field applies to indexed disk files and describes the format of the keys within the records of the indexed file. The acceptable entries are:

| Entry | Definition |
|---|---|
| Blank | Not indexed. |
| K, A | Alphanumeric. |
| P | Packed. |
| N | All numeric data in alphanumeric format. |

Direct

A direct file is a randomly organized file whose records have been created in a specific location within the file, not one after the other as in a sequential file.

When a record is written to a direct file, the MCP checks to see whether the area to which that record belongs already exists. For example, if a file has 1000 records per area and 10 areas (a maximum of 105 areas may be specified), a write of record number 1500 will write into area number 2, a write of record number 6005 would write into area number 7, etc. If the area exists, the record is written in the appropriate position. If not, space for the new area is sought. Therefore, a file containing few records could utilize many of its file areas on disk, if the records are scattered.

The advantage of blocking is partially lost on a direct file. As records are created in random sequence, it is highly unlikely that more than one record in a particular block would be created before a record from a completely different part of the file causes the buffer to be overwritten.

The position in which the particular record is to be created is determined by the value of a "key". This value specifies a record number, relative to the first record position in the file, of the record to be created. For example, if the value of the key is 6, then to determine where that record will be written, multiply the record length by 6, and add the result to the base address of the file (this is done automatically by the MCP), giving the address of the record number 6.

## File Accessing Methods

B 1700 RPG files may be accessed in three ways: sequential, direct, or indexed-sequential. The manner in which a file is accessed need not have any bearing on the way in which the file was created (organized). Once the file has been created the user need not be concerned as to the type of file organization.

File accessing methods for the B 1700 are discussed in the following paragraphs.

Sequential

A file created either sequential or direct may be accessed sequentially.

Sequentially accessing a sequentially organized file will simply access the records in the order in which they were created, that is, one after the other.

Sequentially accessing a randomly organized file will access the records one after the other, regardless of whether the record had ever been created or not, provided the disk space for that area has already been assigned. That is, records which have never been created will be accessed, and whatever information happens to be in that record area, from the previous time that space was used, will be read as the record contents.

An indexed file may be loaded in ascending key sequence or unordered. If unordered, a U must be placed in column 66 of the File Specifications. Files loaded unordered will be sorted in sequence according to the key field prior to End-of-Job.

Additional I/O Areas

The use of additional I/O areas (additional buffers) for a file increases the efficiency of the program when it is executed; however, it also increases the size of the program. A good balance between increased efficiency and increased size must be reached in order to achieve the greatest throughput from the system when programs are run in a multiprogramming environment.

Additional I/O areas cannot be used with table, combined, display, or demand files, nor with indexed accessing of disk files.

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 line is reached) 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 | Default to OF or OV according to entry in columns 40-46 (DEVICE). |

If this field is left blank for any printer file, except PRINTR2 and PRINTUF, OF will be assigned (by default) as the overflow indicator. If this field is left blank for a PRINTR2 or PRINTUF file, OV will be assigned (by default) as the overflow indicator.

If this field is left blank and the appropriate indicator (OF or OV) is not used to condition output, then no overflow indicator will be assigned to the file. Overflow will be handled automatically by the RPG object program.

If this field specifies an overflow indicator but no output is conditioned on that indicator, overflow will not be handled automatically: a continuous listing will be produced.

Specific lines on output will be printed when an overflow condition is reached if the overflow indicators, whether specified or assigned by default (OV or OF), are used to condition them on output.

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.

Table 4-4. Device Assignment for Files

| File | Type | Devices |
|---|---|---|
| Primary or Secondary Input Files | Card | MFCU1 or MFCU2<br>READER |
| | Disk | DISK |
| | Tape | TAPE |
| Demand Files | Card | MFCU1 or MFCU2<br>READER |
| | Disk | DISK |
| | Tape | TAPE |
| Table Files | Card | MFCU1 or MFCU2<br>READER |
| | Disk | DISK |
| | Tape | TAPE |
| Chained Input Files | Disk | DISK |
| Update Files | Disk | DISK |
| Combined Files | Card | MFCU1 or MFCU2 |
| Output Files | Card | MFCU1 or MFCU2<br>PUNCH |
| | Disk | DISK |
| | Listing | PRINTER, PRINTR2 |
| Display File | Listing | CONSOLE |

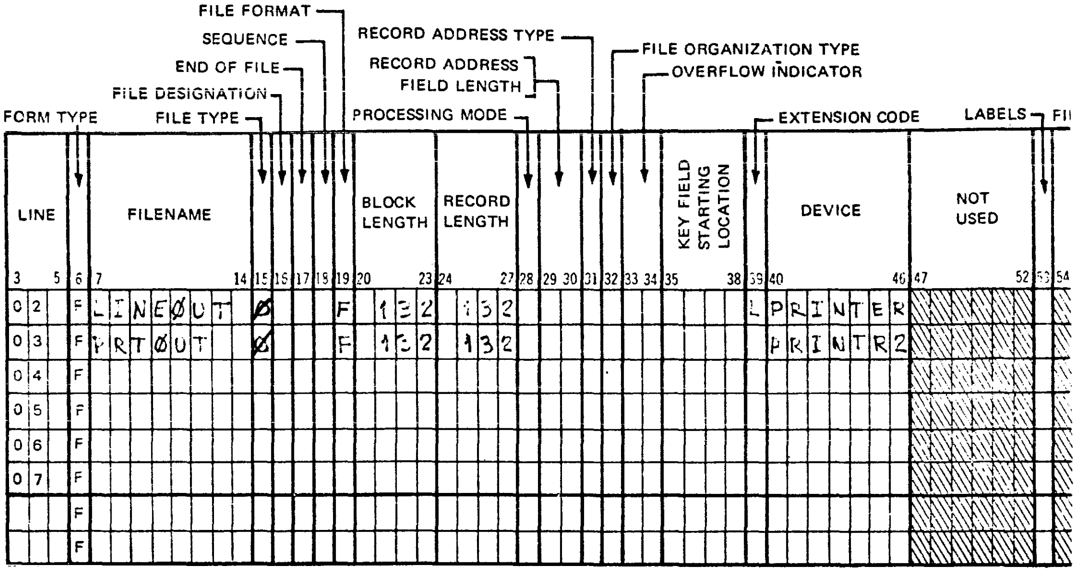
The B 1700 will also accept the following additional device names in columns 40-46:

READ01
MFCM1
MFCM2
READ20
READ40
READ42
CRP
CRP20
DATA96

PUNCH20
PUNCH42
PRINTUF
PRINTLF
DISK11
DISK11F
DISK45
SPO

Figures 4-4 through 4-7 are File Description Specifications coding examples for MFCU, console, printer, and tape files.

FILE DESCRIPTION SPECIFICATIONS



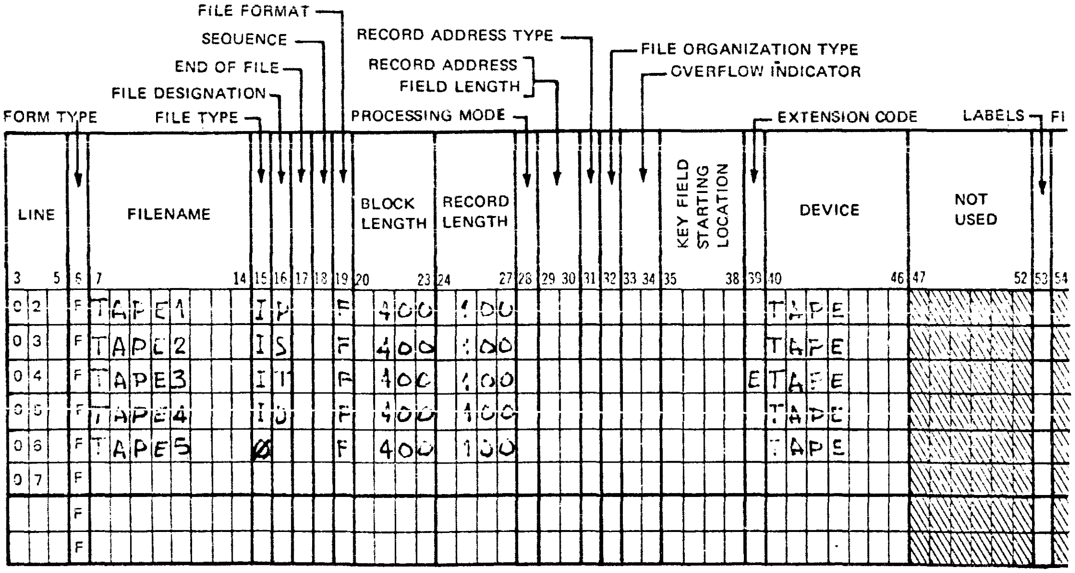
FILE FORMAT
SEQUENCE
END OF FILE
FILE DESIGNATION
FORM TYPE
FILE TYPE
RECORD ADDRESS TYPE
RECORD ADDRESS FIELD LENGTH
PROCESSING MODE
FILE ORGANIZATION TYPE
OVERFLOW INDICATOR
EXTENSION CODE
LABELS
FI

| LINE 3-5 | 6 | FILENAME 7-14 | 15 | 16 | 17 | 18 | 19 | BLOCK LENGTH 20-23 | RECORD LENGTH 24-27 | 28 | 29-30 | 31 | 32 | 33-34 | KEY FIELD STARTING LOCATION 35-38 | 39 | DEVICE 40-46 | NOT USED 47-52 | 53 | 54 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 02 | F | LINEØUT | Ø | | | | F | 132 | 132 | | | | | | | L | PRINTER | | | |
| 03 | F | PRTØUT | Ø | | | | F | 132 | 132 | | | | | | | | PRINTR2 | | | |
| 04 | F | | | | | | | | | | | | | | | | | | | |
| 05 | F | | | | | | | | | | | | | | | | | | | |
| 06 | F | | | | | | | | | | | | | | | | | | | |
| 07 | F | | | | | | | | | | | | | | | | | | | |
| | F | | | | | | | | | | | | | | | | | | | |
| | F | | | | | | | | | | | | | | | | | | | |

G14012

Figure 4-6. Device Coding Example - Printer Files

FILE DESCRIPTION SPECIFICATIONS



FILE FORMAT
SEQUENCE
END OF FILE
FILE DESIGNATION
FORM TYPE
FILE TYPE
RECORD ADDRESS TYPE
RECORD ADDRESS FIELD LENGTH
PROCESSING MODE
FILE ORGANIZATION TYPE
OVERFLOW INDICATOR
EXTENSION CODE
LABELS
FI

| LINE 3-5 | 6 | FILENAME 7-14 | 15 | 16 | 17 | 18 | 19 | BLOCK LENGTH 20-23 | RECORD LENGTH 24-27 | 28 | 29-30 | 31 | 32 | 33-34 | KEY FIELD STARTING LOCATION 35-38 | 39 | DEVICE 40-46 | NOT USED 47-52 | 53 | 54 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 02 | F | TAPE1 | I | P | | | F | 400 | 100 | | | | | | | | TAPE | | | |
| 03 | F | TAPE2 | I | S | | | F | 400 | 100 | | | | | | | | TAPE | | | |
| 04 | F | TAPE3 | I | T | | | F | 400 | 100 | | | | | | | E | TAPE | | | |
| 05 | F | TAPE4 | I | U | | | F | 400 | 100 | | | | | | | | TAPE | | | |
| 06 | F | TAPE5 | Ø | | | | F | 400 | 100 | | | | | | | | TAPE | | | |
| 07 | F | | | | | | | | | | | | | | | | | | | |
| | F | | | | | | | | | | | | | | | | | | | |
| | F | | | | | | | | | | | | | | | | | | | |

G14013

Figure 4-7. Device Coding Example - Tape Files

60-65 CORE INDEX

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 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 two techniques are possible. The program will always activate the disk SCAN operator first, then will revert to a binary search technique when necessary (see figure 4-8).

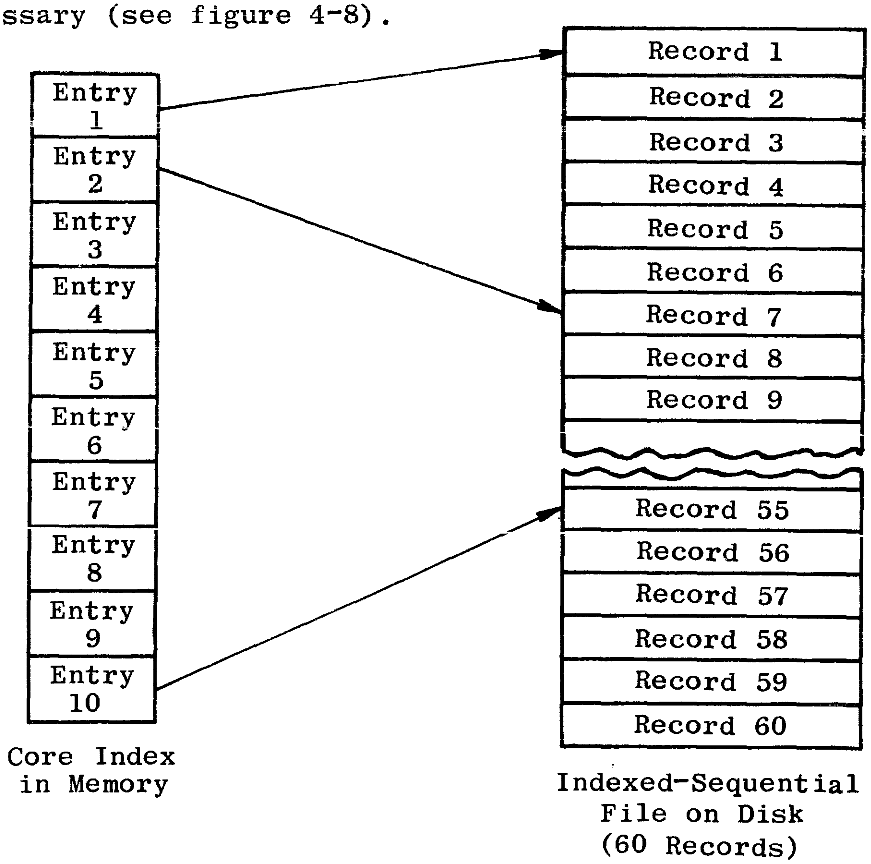


Figure 4-8. Core Index Selection of Data Keys

If column 15 contains an I and column 66 is blank, records of an indexed file are read without adding new records or updating records.

If column 15 contains an I and column 66 contains an A, records of an indexed file are read and new records are added to the file that are not presently there. No updating is performed. An ADD entry is required in columns 16-18 of the Output Specifications.

If column 15 contains a U and column 66 is blank, records of an indexed file are updated without adding new records.

If column 15 contains a U and Column 66 contains an A, records of an indexed file are updated and new records are added to the file. An ADD entry is required in columns 16-18 of the Output Specifications.

Figures 4-9 through 4-12 illustrate coding methods for disk files on the File Description Specifications.

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. |

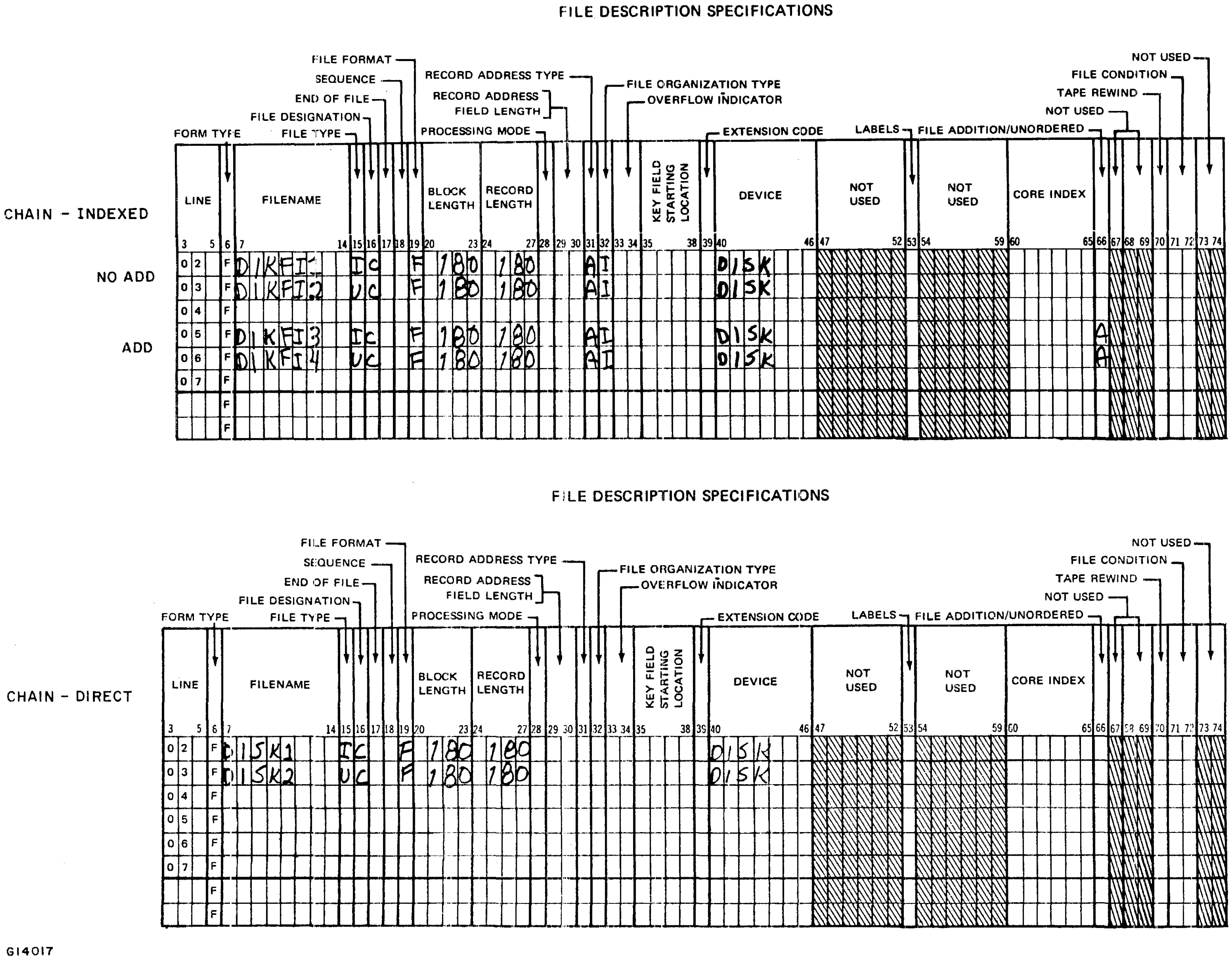
To show the effects of the various options, each type of file is discussed separately in the paragraphs that follow.

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.



CHAIN - INDEXED

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) | NOT USED (68-69) | TAPE REWIND (70) | FILE CONDITION (71-72) | NOT USED (73-74) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NO ADD | 02 | F | DIKFI1 | I | C | | | F | 180 | 180 | | | A | I | | | | DISK | | | | | | | | | | |
| | 03 | F | DIKFI2 | U | C | | | F | 180 | 180 | | | A | I | | | | DISK | | | | | | | | | | |
| | 04 | F | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ADD | 05 | F | DIKFI3 | I | C | | | F | 180 | 180 | | | A | I | | | | DISK | | | | | A | | | | | |
| | 06 | F | DIKFI4 | U | C | | | F | 180 | 180 | | | A | I | | | | DISK | | | | | A | | | | | |
| | 07 | F | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | F | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | F | | | | | | | | | | | | | | | | | | | | | | | | | | |

CHAIN - DIRECT

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) | NOT USED (68-69) | TAPE REWIND (70) | FILE CONDITION (71-72) | NOT USED (73-74) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 02 | F | DISK1 | I | C | | | F | 180 | 180 | | | | | | | | DISK | | | | | | | | | | |
| 03 | F | DISK2 | U | C | | | F | 180 | 180 | | | | | | | | DISK | | | | | | | | | | |
| 04 | F | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 05 | F | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 06 | F | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 07 | F | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | F | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | F | | | | | | | | | | | | | | | | | | | | | | | | | | |

G14017

Figure 4-11. Processing Methods of Disk Files - By CHAIN

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.

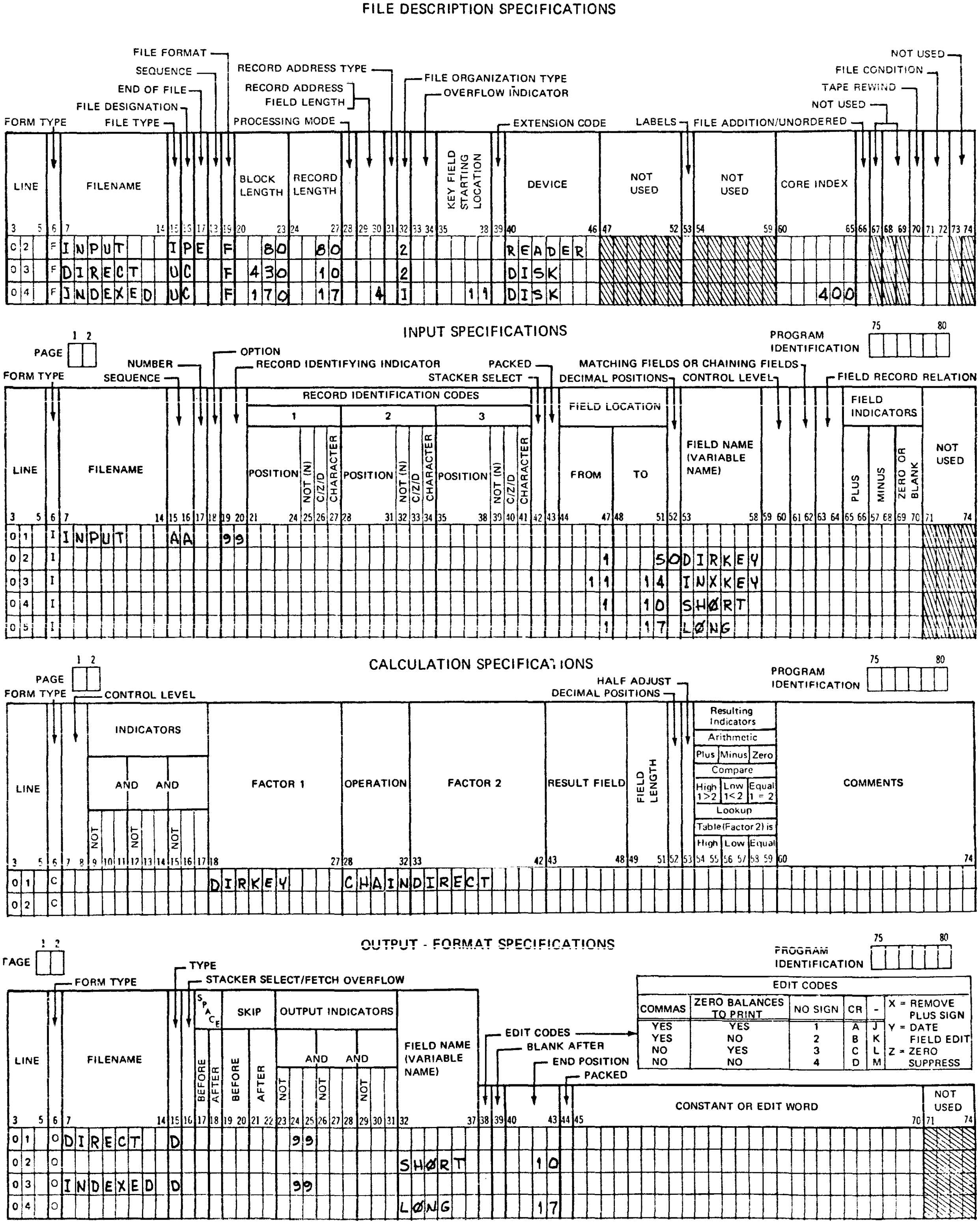
| Entry | Effect |
|---|---|
| P | An old file is removed from the disk and deleted from the Directory, or a new file is removed from disk. |
| U | For an old file, the file remains in the Directory and is made available. A new file is entered in the Directory (thereby making it an old file) and made 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. |

The following figures are RPG coding examples for indexed and direct files. Figure 4-13 illustrates the updating of an existing file; figure 4-14 illustrates the creation of a new file.


FILE DESCRIPTION SPECIFICATIONS
FORM TYPE
FILE TYPE
FILE DESIGNATION
END OF FILE
SEQUENCE
FILE FORMAT
RECORD ADDRESS TYPE
RECORD ADDRESS FIELD LENGTH
PROCESSING MODE
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
INPUT SPECIFICATIONS
PAGE
PROGRAM IDENTIFICATION
NUMBER
SEQUENCE
OPTION
RECORD IDENTIFYING INDICATOR
PACKED
STACKER SELECT
MATCHING FIELDS OR CHAINING FIELDS
DECIMAL POSITIONS
CONTROL LEVEL
FIELD RECORD RELATION
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
CALCULATION SPECIFICATIONS
CONTROL LEVEL
INDICATORS
AND
AND
NOT
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
OUTPUT - FORMAT SPECIFICATIONS
TYPE
STACKER SELECT/FETCH OVERFLOW
SPACE
BEFORE
AFTER
SKIP
OUTPUT INDICATORS
FIELD NAME (VARIABLE NAME)
EDIT CODES
BLANK AFTER
END POSITION
PACKED
CONSTANT OR EDIT WORD
EDIT CODES
COMMAS
ZERO BALANCES TO PRINT
NO SIGN
CR
-
X = REMOVE PLUS 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

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
```

Figure 4-13. Indexed and Direct Files - File Update

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.

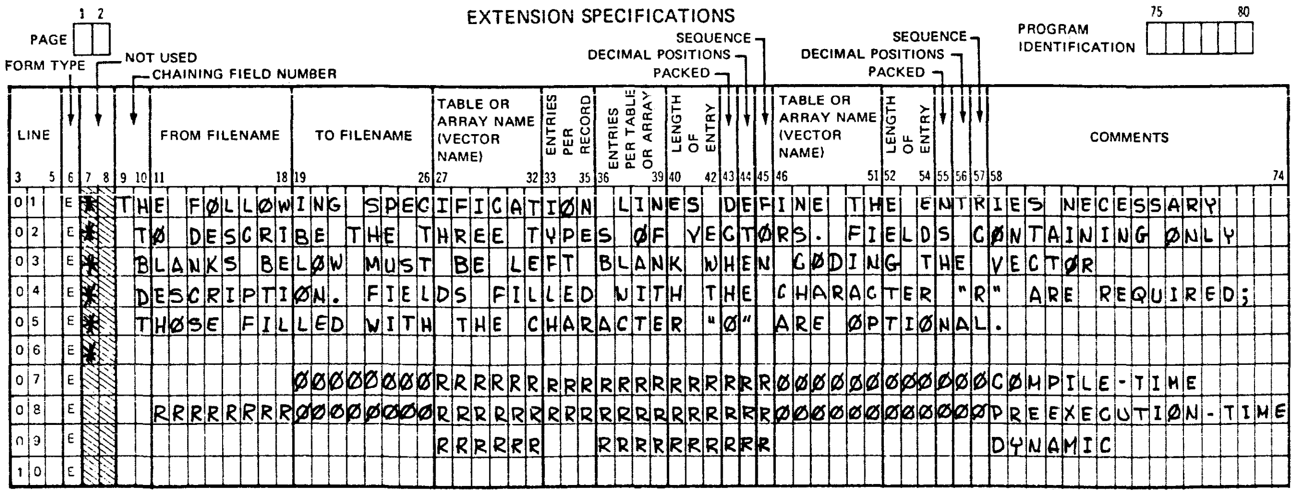
DYNAMIC/EXECUTION-TIME VECTORS

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 vectors, is completely under programmatic control.

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

## EXTENSION SPECIFICATIONS

All vectors in a program must be described on the Extension Specifications sheet. Certain entries are required for all types of vectors, regardless of the time at which they will be loaded (see figure 5-1). Columns 27-45 must be completed for each vector (columns 46-57 if the vector is loaded in alternating format with another vector). These columns 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 packed decimal format (PACKED), the number of decimal positions in each entry (DECIMAL POSITIONS), and the order in which the elements are sequenced (SEQUENCE).



EXTENSION SPECIFICATIONS

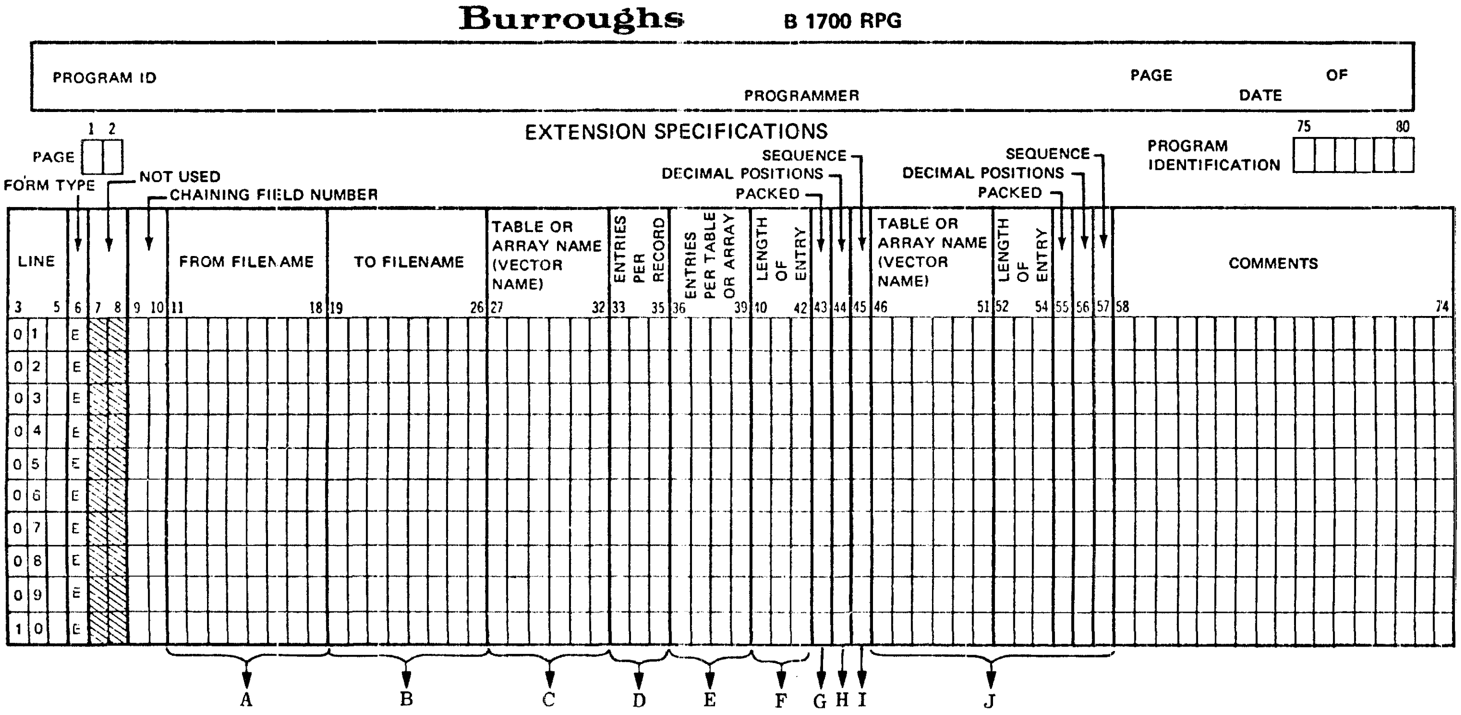
PAGE 1 2 □□ — FORM TYPE — NOT USED — CHAINING FIELD NUMBER

PROGRAM IDENTIFICATION 75 □□□□□□ 80

| LINE 3-5 | FORM TYPE 6 | 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 | * | | 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 | | | | | ØØØØØØØØ | RRRRRR | RRR | RRRR | RRR | R | R | R | ØØØØØØ | ØØØ | Ø | Ø | Ø | CØMPILE-TIME |
| 08 | E | | | | RRRRRRRR | ØØØØØØØØ | RRRRRR | RRR | RRRR | RRR | R | R | R | ØØØØØØ | ØØØ | Ø | Ø | Ø | PREEXECUTIØN-TIME |
| 09 | E | | | | | | RRRRRR | | RRRR | RRR | R | R | R | | | | | | DYNAMIC |
| 10 | E | | | | | | | | | | | | | | | | | | |

G14021

Figure 5-1. Entries Necessary to Describe a Vector



A. 11-18 Contains the name of a pre-execution time vector file.

B. 19-26 Contains 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 packed or unpacked decimal format. Entries: Blank 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 5-2. Extension Specifications Summary Sheet

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 31 characters; 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.

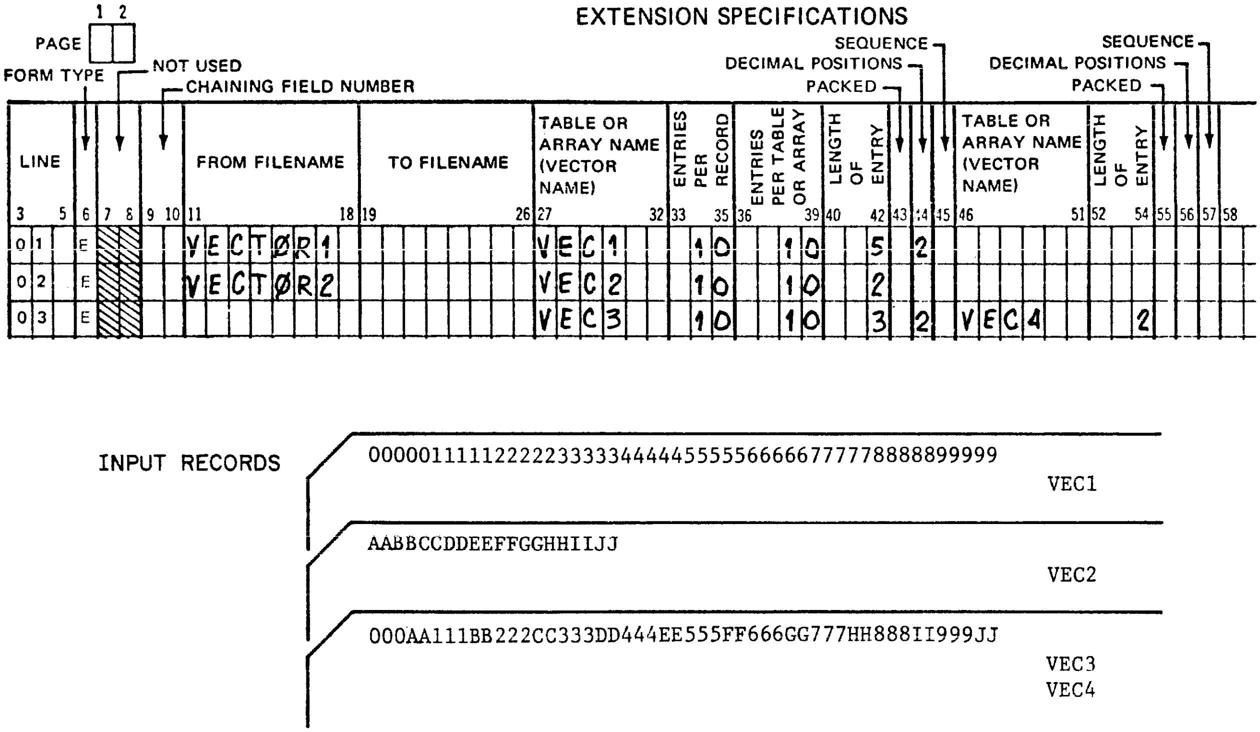
43 PACKED

This field specifies the external format of the vector elements. 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. |

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).

Vectors specified as occurring in alternating format have related elements contained in alternating format on each input record (see figure 5-3). 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

```
00000111112222233333444445555566666777778888899999
                                                      VEC1

AABBCCDDEEFFGGHHIIJJ
                                                      VEC2

000AA111BB222CC333DD444EE555FF666GG777HH888II999JJ
                                                      VEC3
                                                      VEC4
```

G14023

Figure 5-3. Vectors in Alternating Format

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

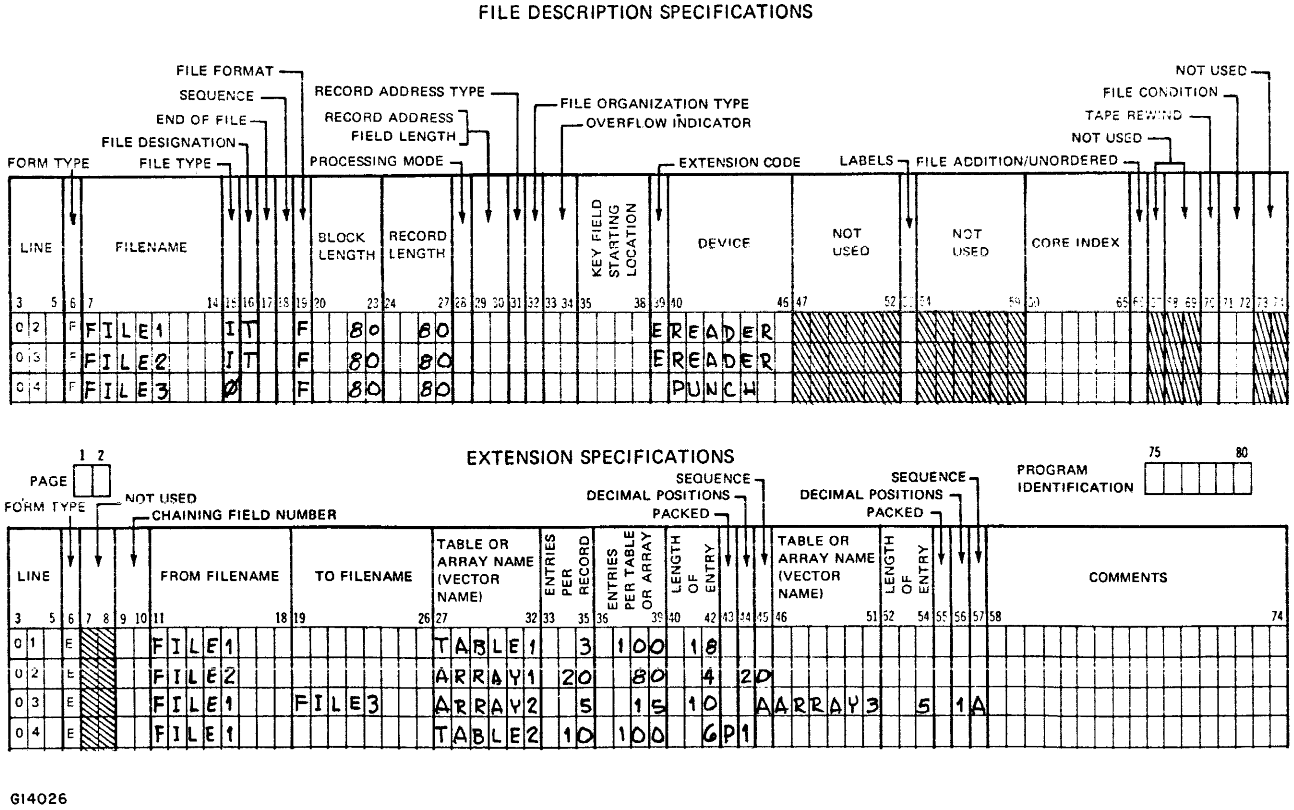
Refer to Section 2 for a complete description.

## VECTOR LOADING

How a vector will be loaded depends on the entries made on the Extension Specifications. The following paragraphs describe loading for compile time, pre-execution time and dynamic/execution time vectors.

## 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 5-6). 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
FORM TYPE
FILE DESIGNATION
END OF FILE
SEQUENCE
FILE FORMAT
FILE TYPE
RECORD ADDRESS TYPE
RECORD ADDRESS FIELD LENGTH
PROCESSING MODE
FILE ORGANIZATION TYPE
OVERFLOW INDICATOR
EXTENSION CODE
LABELS
FILE ADDITION/UNORDERED
NOT USED
FILE CONDITION
TAPE REWIND
NOT USED
KEY FIELD STARTING LOCATION
NOT USED
NOT USED
CORE INDEX
EXTENSION SPECIFICATIONS
PAGE
NOT USED
CHAINING FIELD NUMBER
SEQUENCE
DECIMAL POSITIONS
PACKED
PROGRAM IDENTIFICATION
COMMENTS
G14026

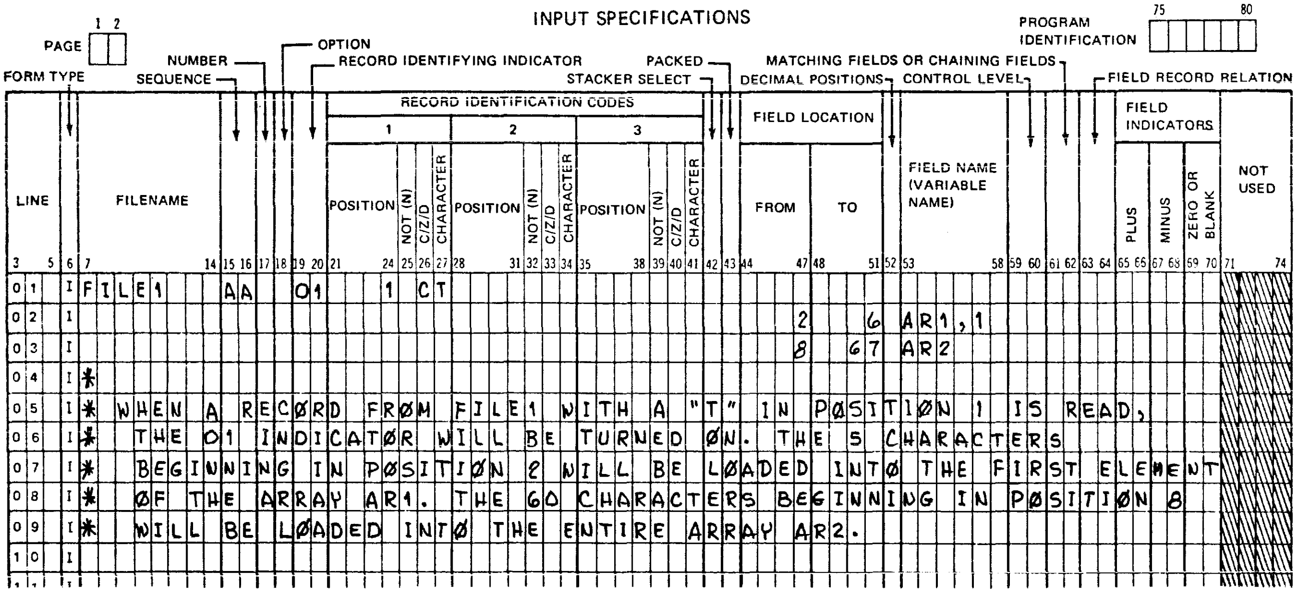
| LINE (3-5) | FORM TYPE (6) | FILENAME (7-14) | FILE TYPE (15) | FILE DESIGNATION (16) | FILE FORMAT (19) | BLOCK LENGTH (20-23) | RECORD LENGTH (24-27) | EXTENSION CODE (39) | DEVICE (40-46) |
|---|---|---|---|---|---|---|---|---|---|
| 02 | F | FILE1 | I | T | F | 80 | 80 | E | READER |
| 03 | F | FILE2 | I | T | F | 80 | 80 | E | READER |
| 04 | F | FILE3 | Ø |  | F | 80 | 80 |  | PUNCH |

| LINE (3-5) | FORM TYPE (6) | 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 |  |  |  |  |  |  |  |

Figure 5-6. Pre-Execution Time Vector Load

## 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.

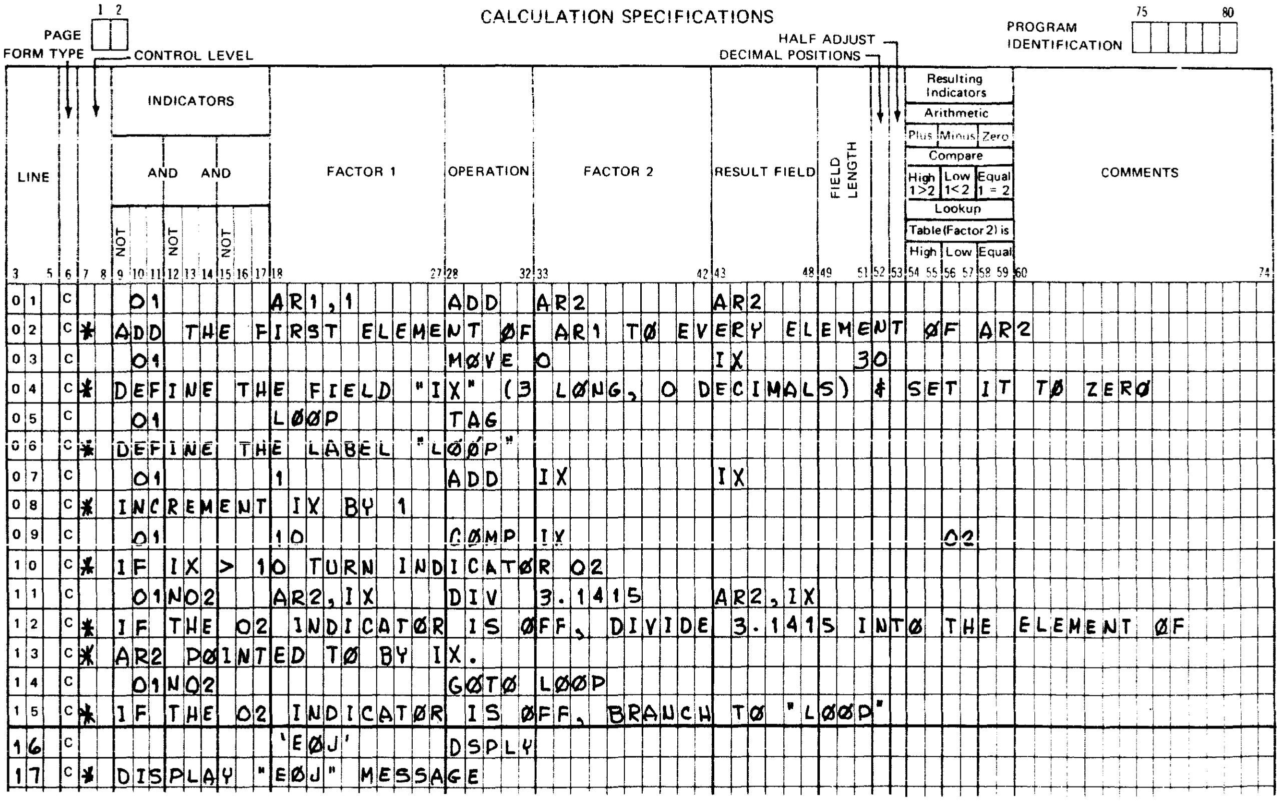


INPUT SPECIFICATIONS

PAGE 1 2 · PROGRAM IDENTIFICATION 75 80

FORM TYPE · NUMBER · SEQUENCE · OPTION · RECORD IDENTIFYING INDICATOR · RECORD IDENTIFICATION CODES 1 2 3 (POSITION, NOT (N), C/Z/D, CHARACTER) · PACKED · STACKER SELECT · DECIMAL POSITIONS · FIELD LOCATION (FROM, TO) · FIELD NAME (VARIABLE NAME) · CONTROL LEVEL · MATCHING FIELDS OR CHAINING FIELDS · FIELD RECORD RELATION · FIELD INDICATORS (PLUS, MINUS, ZERO OR BLANK) · NOT USED

```
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
```



CALCULATION SPECIFICATIONS

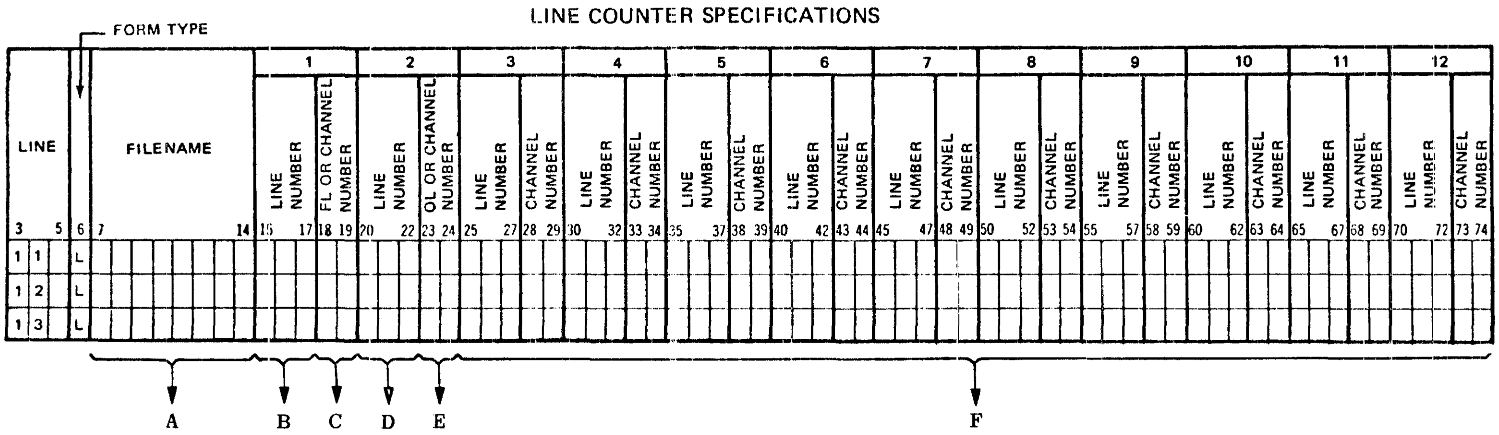
PAGE 1 2 · PROGRAM IDENTIFICATION 75 80

FORM TYPE · CONTROL LEVEL · INDICATORS (AND, AND, NOT) · 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

```
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
```

GI4028

Figure 5-7. Dynamic Vector Load (Sheet 2 of 2)



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. 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.

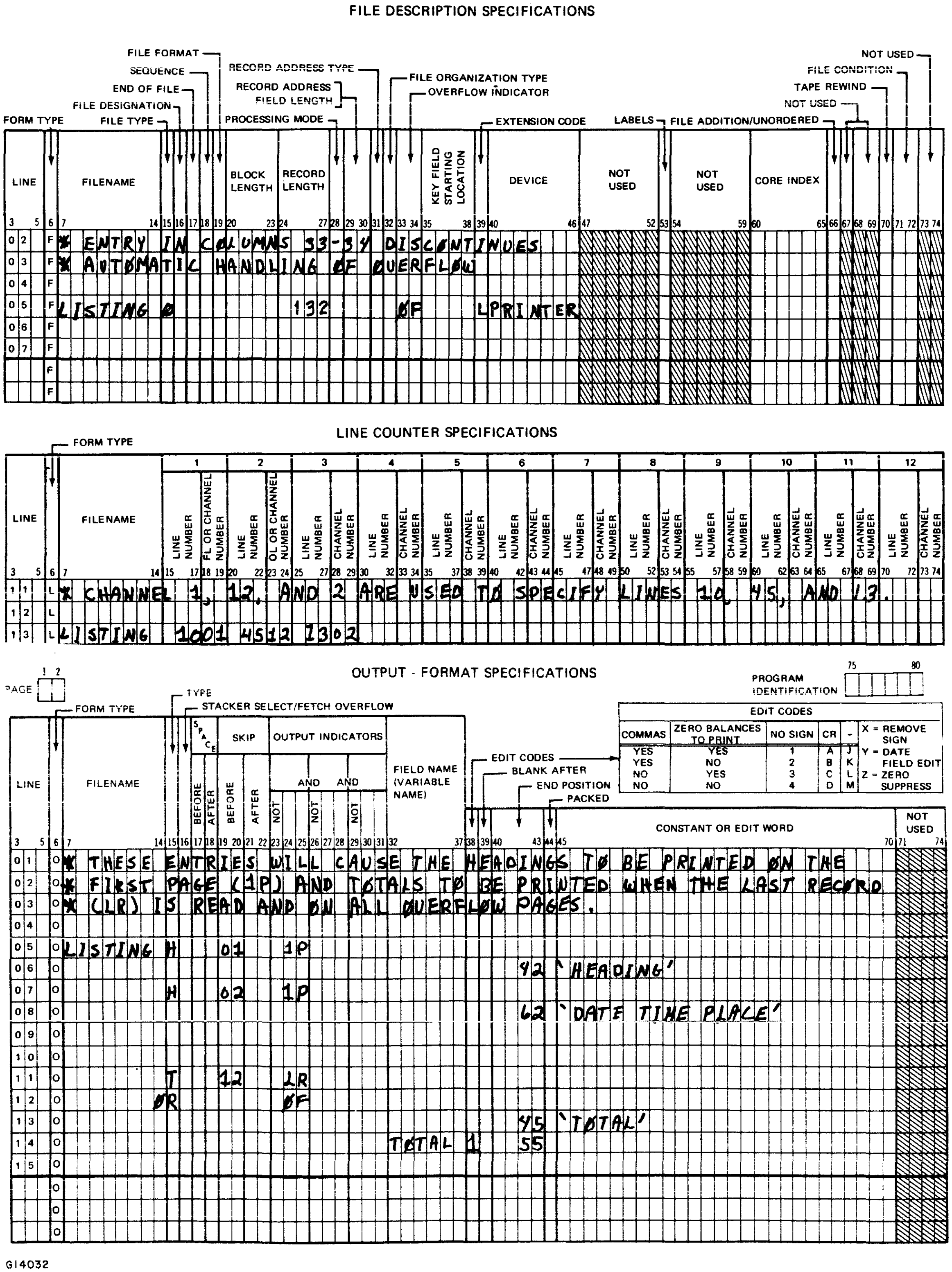
D. 20-22 If 23-24 contains OL, this entry specifies the line number of the overflow line. If 23-24 are numeric, this entry specifies the number of lines from the top of the page to associate with that channel number. Entries: 1-N, right-justified (N=entry in 15-17).

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-N, right-justified (N=entry in 15-17 or the default length 66).

G14029

Figure 6-1. Line Counter Specifications Summary Sheet



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
LINE COUNTER SPECIFICATIONS
FORM TYPE
LINE
FILENAME
LINE NUMBER
FL OR CHANNEL NUMBER
OL OR CHANNEL NUMBER
CHANNEL NUMBER
OUTPUT - FORMAT SPECIFICATIONS
PAGE
PROGRAM IDENTIFICATION
FORM TYPE
TYPE
STACKER SELECT/FETCH OVERFLOW
SPACE
BEFORE
AFTER
SKIP
BEFORE
AFTER
OUTPUT INDICATORS
AND
AND
NOT
FIELD NAME (VARIABLE NAME)
EDIT CODES
BLANK AFTER
END POSITION
PACKED
CONSTANT OR EDIT WORD
NOT USED

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
```

LINE COUNTER SPECIFICATIONS

```
11 L* CHANNEL 1, 12, AND 2 ARE USED TO SPECIFY LINES 10, 45, AND 13.
12 L
13 LLISTING  1001 4512 1302
```

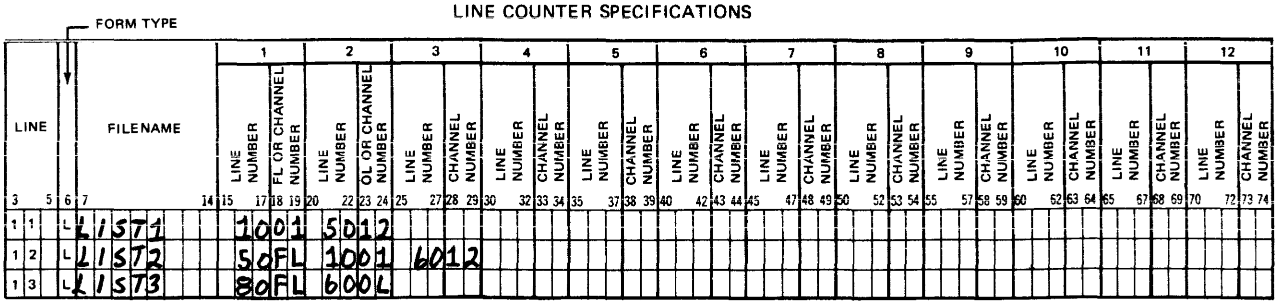
OUTPUT - FORMAT SPECIFICATIONS

| EDIT CODES | | | | | | |
|---|---|---|---|---|---|---|
| COMMAS | ZERO BALANCES TO PRINT | NO SIGN | CR | - | | X = REMOVE SIGN |
| YES | YES | 1 | A | J | | Y = DATE FIELD EDIT |
| YES | NO | 2 | B | K | | Z = ZERO SUPPRESS |
| 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
```

G14032

Figure 6-3. Overflow Coding Example 2



LINE COUNTER SPECIFICATIONS

| LINE | FORM TYPE | FILENAME | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | | 8 | | 9 | | 10 | | 11 | | 12 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 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 |
| 11 | L | LIST1 | 10 | 01 | 50 | 12 | | | | | | | | | | | | | | | | | | | | |
| 12 | L | LIST2 | 50 | FL | 10 | 01 | 60 | 12 | | | | | | | | | | | | | | | | | | |
| 13 | L | LIST3 | 80 | FL | 60 | OL | | | | | | | | | | | | | | | | | | | | |

G14030

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

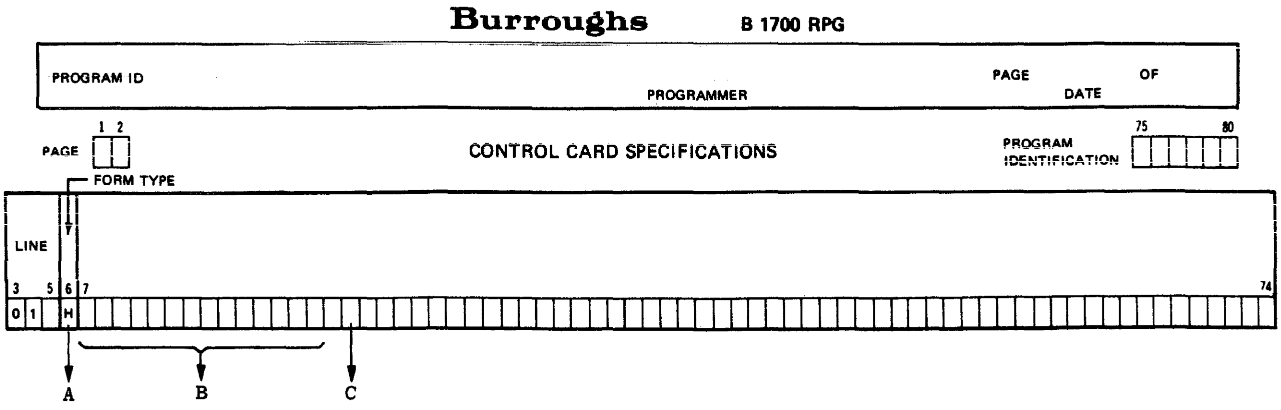
LIST2 specifies the form length (50 lines) and channels 1 and 12.

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

Figure 6-4. Line Counter Specifications Code Example

75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a complete description.



A. 6 Change entry to the letter T.

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

C. 16 Specifies if the file is to be transmitted or received. Entries: T or R.

G14033

Figure 7-1. Telecommunications Card Specifications Summary Sheet

16 TRANSMIT/RECEIVE

Acceptable entries for this field are:

| Entry | Definition |
|---|---|
| T | Transmit messages. |
| R | Receive messages. |

If T is specified, messages will be transmitted from the DATACOM file to the corresponding NDL file. The file must be an output or combined file and should appear on the Output-Format Specifications. Combined files must alternatingly transmit and receive.

If R is specified, messages will be received from the NDL file into the DATACOM file. The file must be an input or combined file and should appear on the Input Specifications. Combined files must alternatingly receive and transmit.

17 TYPE OF LINE CONTROL

This field is unused. Any entry other than Blank or T will be given a syntax error.

18 TRANSMISSION CODE

This field is unused. Any entry other than A, U, E, or blank will be given a syntax error.

60 LAST FILE

This field is unused. Any entry other than L or blank will be given a syntax error.

61-62 TRANSMIT ADDRESS

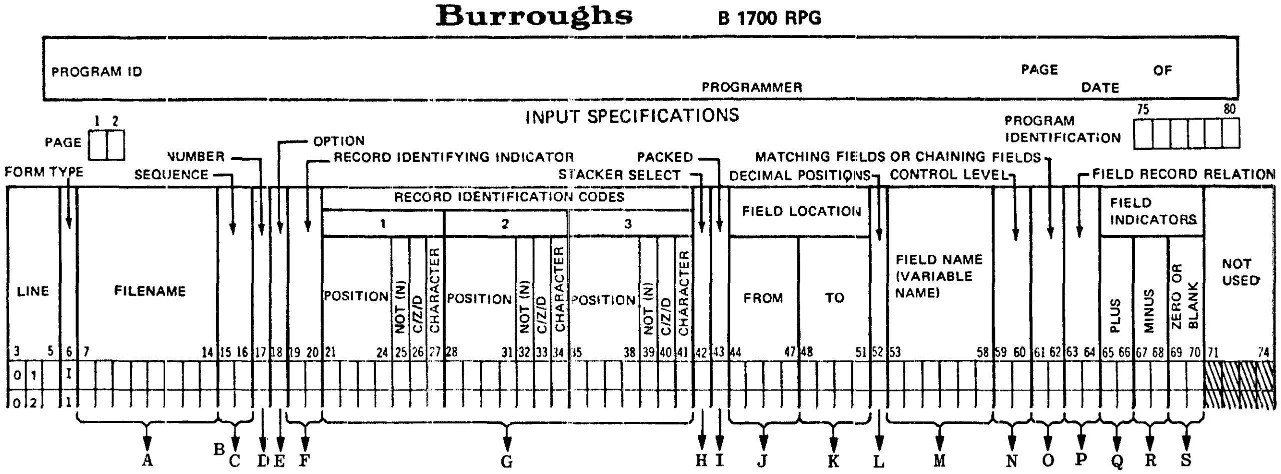
This field is unused and may contain any alphanumeric or blank entry. The entry should reflect the entry in the corresponding NDL file.

63-63 RECEIVE ADDRESS

This field is unused and may contain any alphanumeric or blank entry. The entry should reflect the entry in the corresponding NDL file.

75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a complete description.



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 0.

F. 19-20 Contains either a record identifying indicator (01-99), a control level indicator (L1-L9), a halt indicator (H1-H9), 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. Entries: Blank or P.

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.

G14034

Figure 8-1. Input Specifications Summary Sheet

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. All chained and demand files must have an alphabetic entry in this field.

When coding this field the programmer must not use the alphabetic entries ND or Rb (b equals blank), because the compiler may mistake them for the ND or R of an AND or OR 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 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.

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 8-3). Valid entries for this field are:

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.

19-20 RECORD IDENTIFYING INDICATOR

This field may be used for two purposes:

a. To assign an indicator to each record type.

b. To indicate look-ahead fields.

| Entry | Definition |
|---|---|
| 01-99 | Record identifying indicator. |
| L1-L9 | Control level indicator. |
| LR | Last record indicator. |
| H0-H9 | Halt indicator. |
| ** | 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.

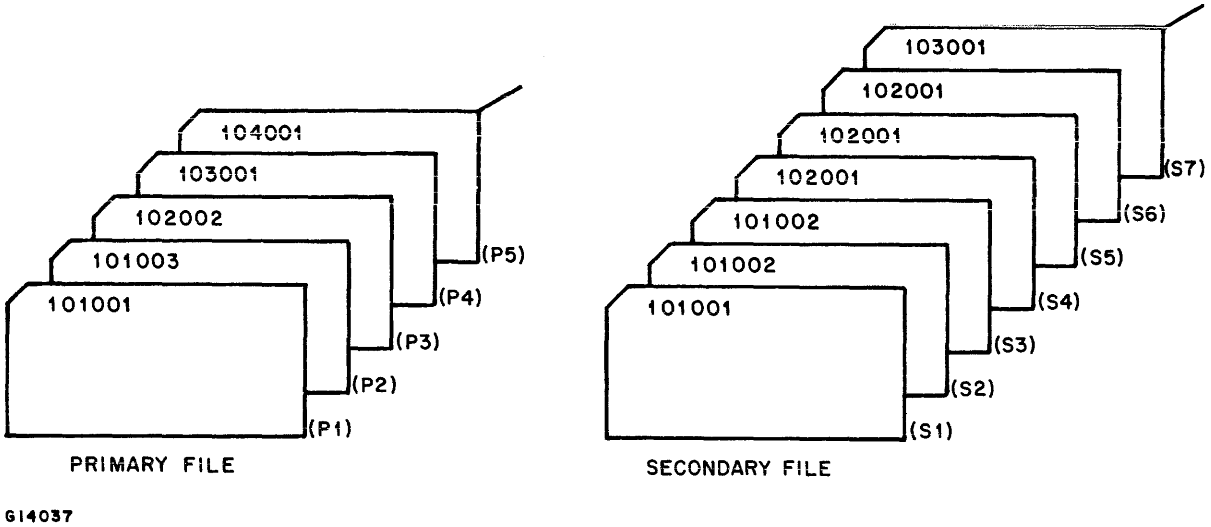


Figure 8-4. Records Available for Look-Ahead: Two Input Files

Look-ahead fields may be specified for primary or secondary files only. One set of look-ahead fields may be specified per file and the field descriptions apply to all records in that file.

For combined and update files, the look-ahead fields apply to the next record in the file only if the current record was read from some other file. Therefore, in a program which reads from one file only and that file is a combined or update file, look-ahead fields will apply to the current record.

Figure 8-5 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 |

At the time that the last record of a file is being processed, any look-ahead fields for that file will contain all nines, either signed numeric or alpha-numeric depending on the field type specified.

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 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).

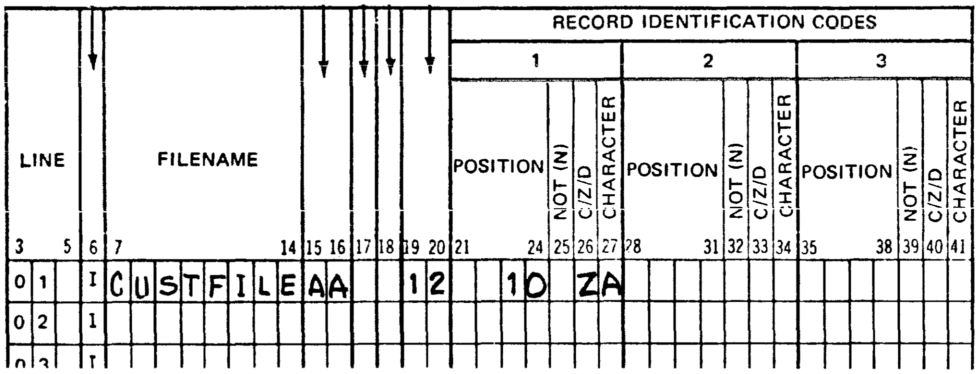
25,32,39 NOT

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. |

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 8-7, 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.

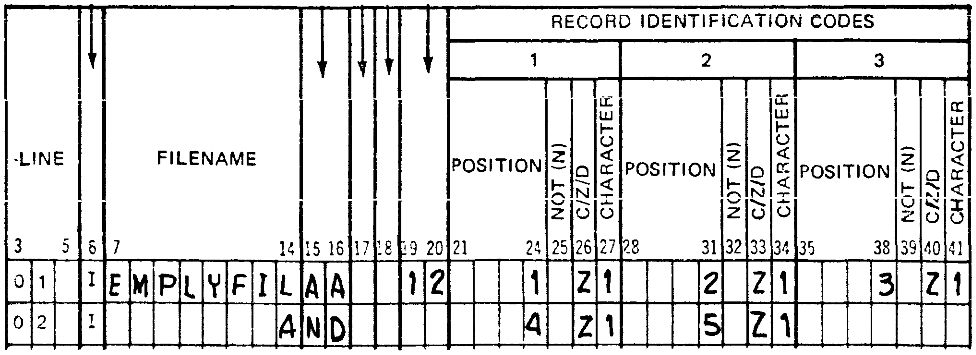


| | | | | | | | RECORD IDENTIFICATION CODES | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | 1 | | | | 2 | | | | 3 | | | |
| LINE | | FILENAME | | | | | POSITION | NOT (N) | C/Z/D | CHARACTER | POSITION | NOT (N) | C/Z/D | CHARACTER | 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 | CUSTFILE | AA | | | 12 | 10 | | Z | A | | | | | | | | |
| 02 | I | | | | | | | | | | | | | | | | | |
| 03 | I | | | | | | | | | | | | | | | | | |

G14040

Figure 8-7. C/Z/D Coding Example 1

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



| | | | | | | | 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 | EMPLYFIL | AA | | | 12 | 1 | | Z | 1 | 2 | | Z | 1 | 3 | | Z | 1 |
| 02 | I | A | ND | | | | 4 | | Z | 1 | 5 | | Z | 1 | | | | |

G14041

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

| 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

This field is used to specify that the external format of an input field is in packed decimal format. Valid entries are:

| Entry | Description |
|---|---|
| Blank | Field is in unpacked decimal format or is alphanumeric. |
| P | Field is in packed decimal format. |

NOTE

96-column devices will not handle packed decimal format.

Whether in packed 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 to packed decimal format with the sign at the most significant position.

Unpacked decimal format means that each byte of disk/tape 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.

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 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).

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 Section 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 8-11, 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:

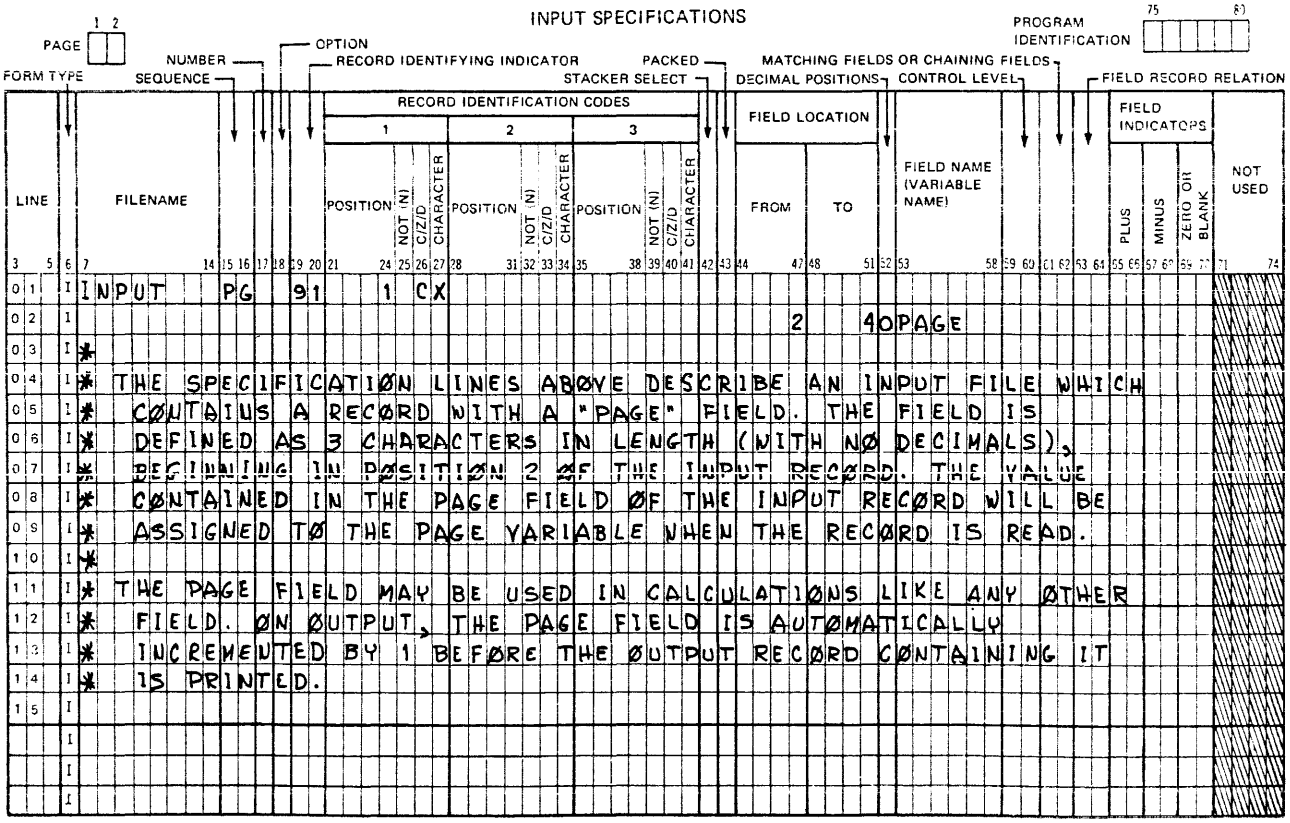
PAGE<br>
PAGE1<br>
PAGE2

If page numbering is to be done on output, the special word "PAGE" (or "PAGE1", or "PAGE2", for two more printer files) is used to indicate that page numbering is to be done. Page field coding is illustrated in figure 8-12.

This feature allows a page number to be entered through a field in an input record, the field called "PAGE" (or "PAGE1" or "PAGE2"). 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 8-12). The "PAGE" field may be used in calculations like any other field.

The same "PAGE" entry (PAGE", "PAGE1", or "PAGE2") may be used for two different output files, but this is not recommended.

Figure 8-12 is an example of PAGE field coding on Input Specifications.



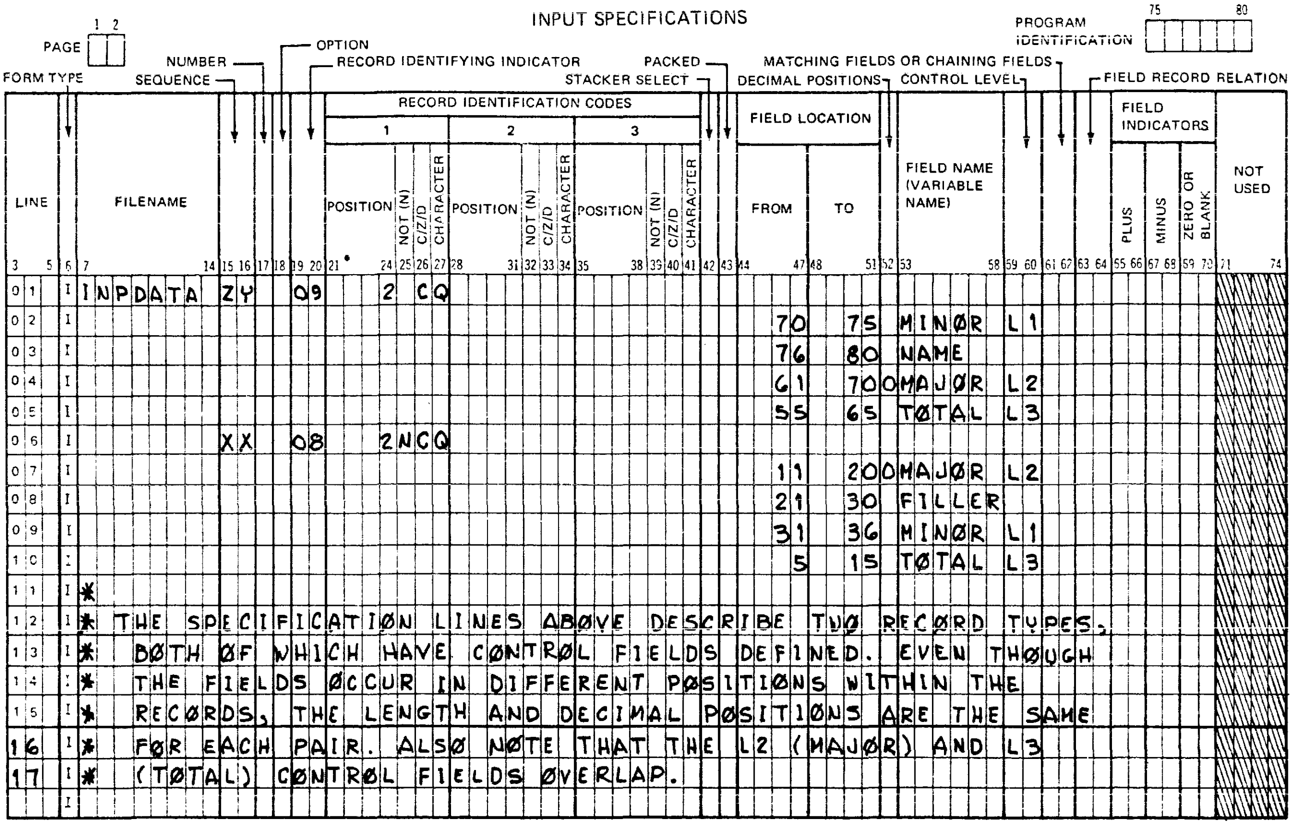
INPUT SPECIFICATIONS

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

FORM TYPE | SEQUENCE | NUMBER | OPTION | RECORD IDENTIFYING INDICATOR | RECORD IDENTIFICATION CODES 1, 2, 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

```
LINE FILENAME
3  56 7      14 15 17 19 21  24 25 27 28 ... 44  47 48  51 52 53    58
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
```

Figure 8-12. PAGE Field Coding



INPUT SPECIFICATIONS

```
LINE  FORM TYPE  FILENAME  SEQUENCE  RECORD IDENTIFYING INDICATOR  POSITION/NOT(N)/C/Z/D/CHARACTER  FROM  TO  DECIMAL POSITIONS  FIELD NAME  CONTROL LEVEL
3  56 7     14 15 17 19 21  24 26   44  48  5253    5859
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
```

Figure 8-13. Control Fields in Two Record Types

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 hexadecimal 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 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.

c. Any one portion of a numeric split control field may not exceed the maximum size for numeric fields as specified in Section 1; however, the total length of all fields assigned to one control level indicator (within each record type) may be as great as 255 characters.

d. A mixture of packed and unpacked control fields is allowed.

e. No other specification lines may come between lines describing split control fields.

61-62 MATCHING FIELDS

This field is used to indicate matching fields and sequence checking. Valid entries are:

| Entry | Definition |
|---|---|
| Blank | No matching is done. |
| M1-M9 | Matching to be performed when two or more input, update, or combined files specify matching fields.<br>Sequence-checking to be performed when only one input, update, or combined file specifies matching fields. |

Matching fields and sequence checking cannot be specified for chained, demand, or table files. The entry in this field designates:

a. Matching fields and sequence checking when two or more input, update, or combined files specify the same matching field value (M1-M9), or

b. Sequence checking only, when only one input, update, or combined file specifies a matching field (M1-M9).

Sequence Checking

Matching field values allow sequence checking of records within a file when one or more primary or secondary file(s) specifies matching fields. As many as nine (M1-M9) fields within the record may be selected for sequence checking. When a record is encountered which has a field or fields out of sequence, the program will halt. When the system operator resumes program operation, the record in error is ignored and the next record from the same file is read.

The following rules must be observed when assigning matching field values 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 values and are treated as one contiguous field.

Match Fields from a secondary file. The matching field values (M1-M9) are used to specify which fields in each record are to be matched, and cause MR to turn ON when a match occurs. M1-M9 are not indicators; MR is used to condition those operations that should be done only when records match.

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 should (not required) contain the same Match Fields (M1-M9); otherwise, a match may not be obtained.

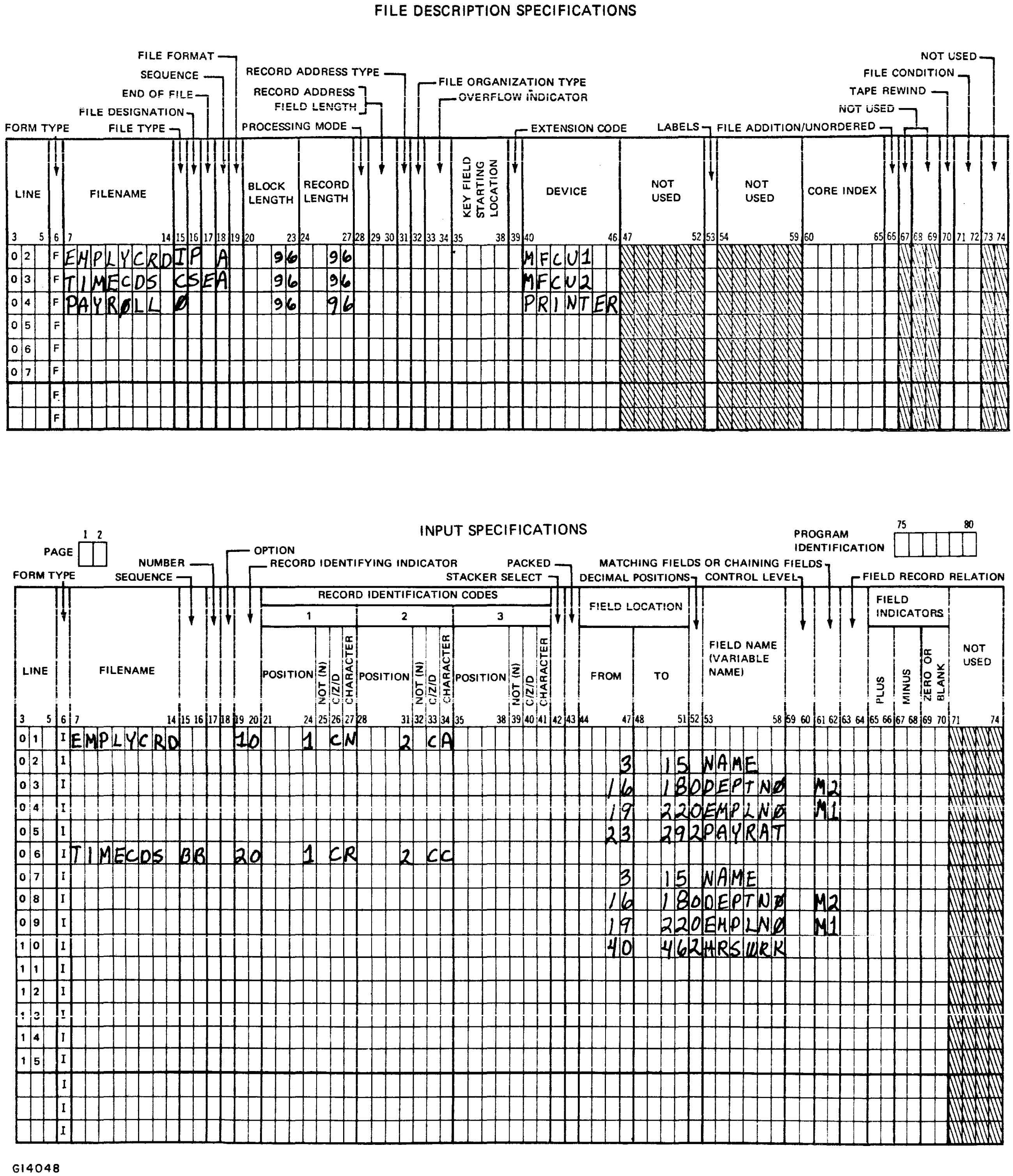
f. When more than one Match Field is designated for a record type, all fields specified are combined in order by ascending sequence of matching field values and are treated as one contiguous Match Field. M9 is high order.

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.



## 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

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) | 1 POSITION (21-24) | 1 NOT (N) (25) | 1 C/Z/D (26) | 1 CHARACTER (27) | 2 POSITION (28-31) | 2 NOT (N) (32) | 2 C/Z/D (33) | 2 CHARACTER (34) | FIELD LOCATION FROM (44-47) | FIELD LOCATION TO (48-51) | DECIMAL POSITIONS (52) | FIELD NAME (VARIABLE NAME) (53-58) | MATCHING FIELDS OR CHAINING 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 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | I |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | I |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | I |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

GI4048

Figure 8-16. Coding Matching Fields

The example shown in figure 8-18 illustrates the order in which records with three matching fields are processed.

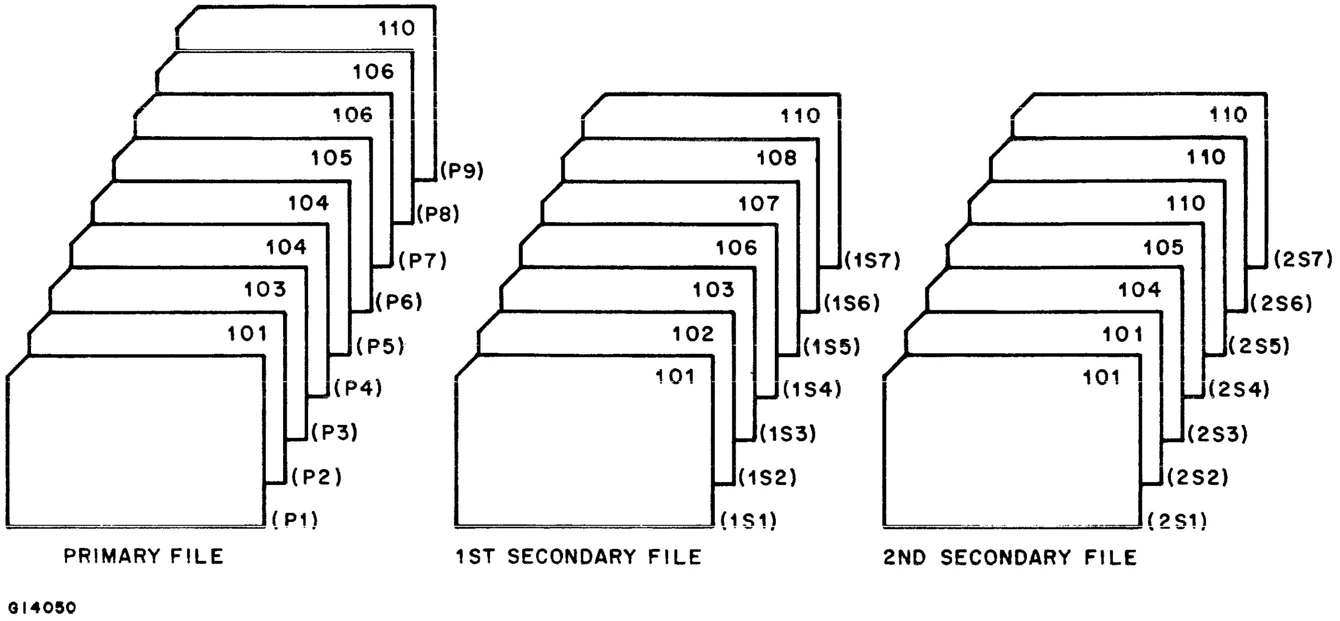


Figure 8-18. Record Selection From Three Matching Files

The files (figure 8-18) will be processed in the following order:

P1

P2<br>
1S1<br>
2S1<br>
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

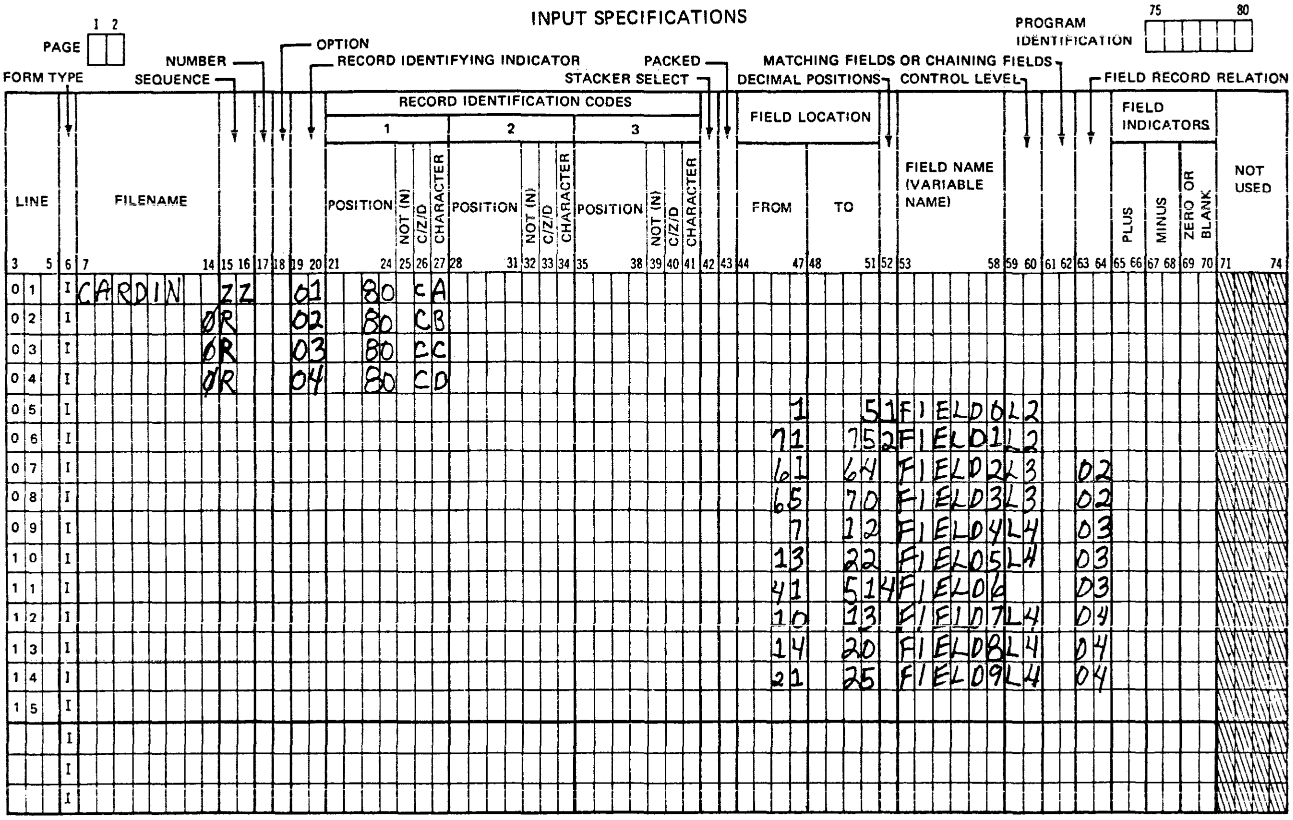
P8

1S4

1S5

1S6

P9<br>
1S7<br>
2S5<br>
2S6<br>
2S7 } The MR indicator is ON. 2S7 is the last record processed.



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 | 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 | 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 8-19. Field Record Relations - Using the OR Relationship

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**

Certain fields may be used only when a special condition, such as a control break or matching records, occurs. To indicate the condition under which data will be accepted from a particular field, the control level (L1-L9) or matching record (MR) indicator is used. Data from the associated field will be accepted only when the specified indicator is ON.

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. Note that if RPG 1 dialect is specified on the Control Card Specifications, all valid indicators used as zero or blank indicators will be initialized ON at the beginning of the object program execution, unless they are also used as record identifying indicators.

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.

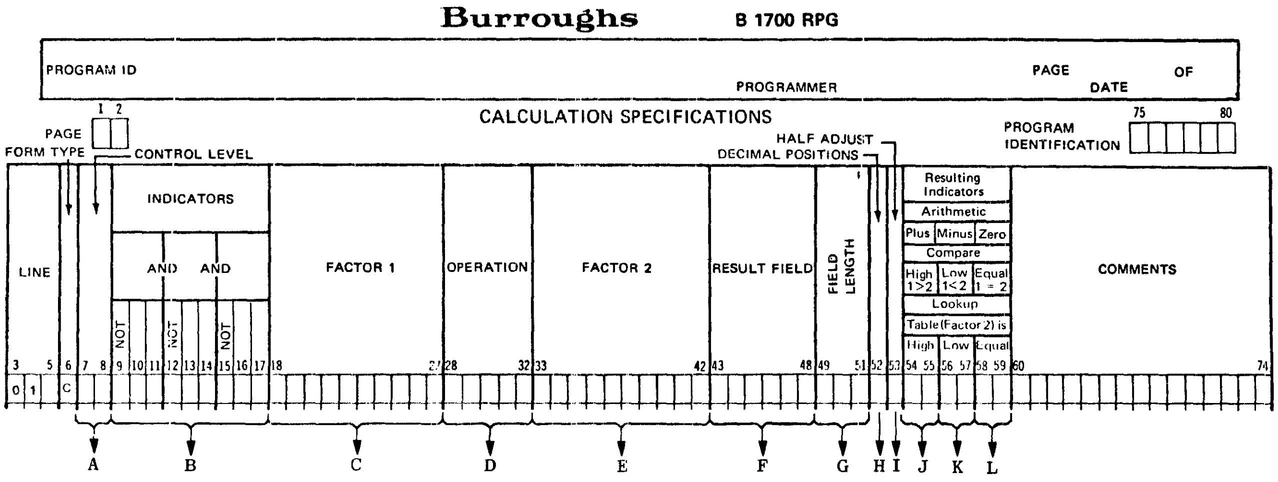
d. The state of a field indicator assigned to more than one field within one record type is determined by the last field to which it is assigned.

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.

75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a 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, MOVEL, MLLZO, MHHZO, MLHZO, MHLZO, COMP, TESTZ, BITON, BITOF, TESTB, SETON, SETOF, GOTO, TAG, ZIP, LOKUP, BEGSR, ENDSR, EXSR, FORCE, EXCPT, DSPLY, READ, CHAIN, 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.

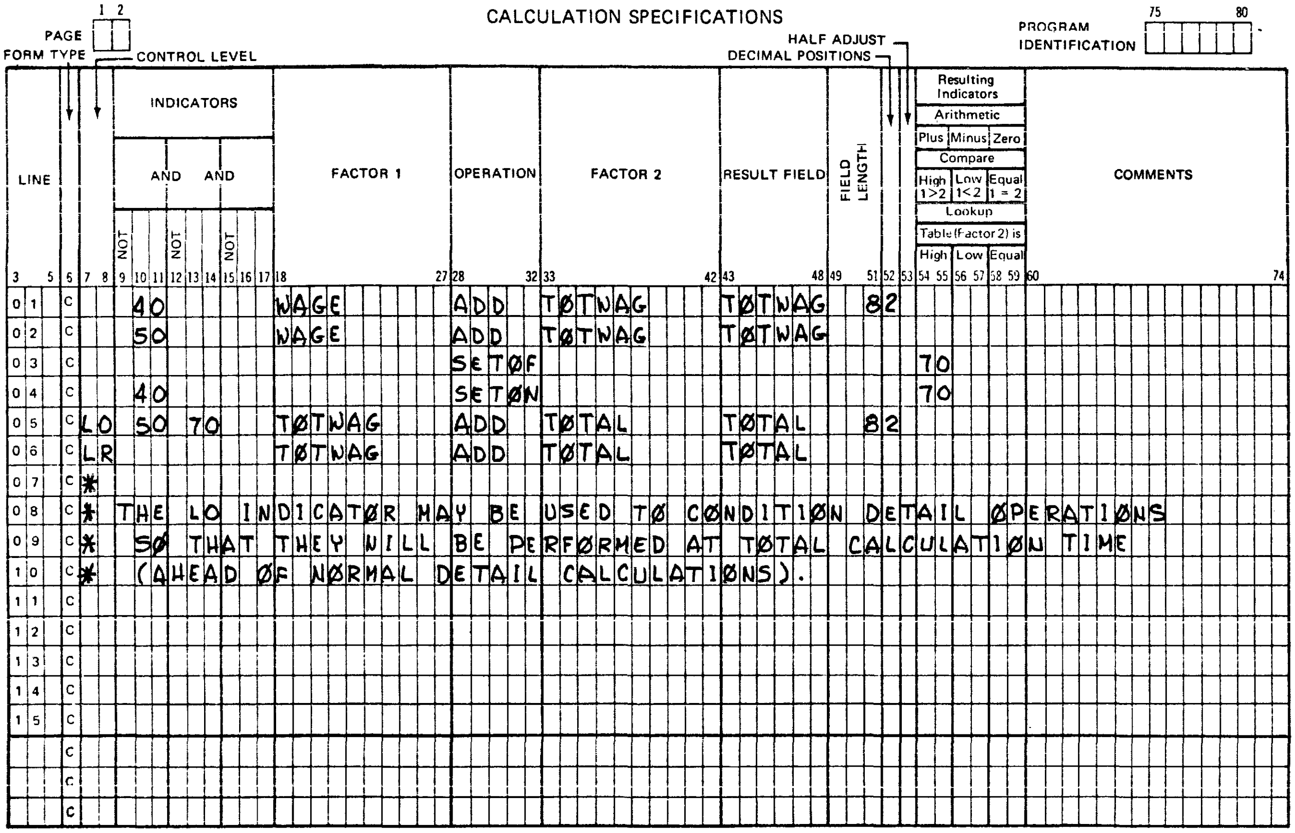
J. 54-55 Entry is turned on if the result field is positive or FACTOR 1 is the highest in a compare operation, or FACTOR 2 is the highest in a lookup operation or a tested zone (TESTZ) is a plus-zone. Entries: 01-99, L1-L9, LR, H0-H9, 0A-0G, OV or blank.

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



CALCULATION SPECIFICATIONS

PAGE □□ FORM TYPE — CONTROL LEVEL — HALF ADJUST — DECIMAL POSITIONS — PROGRAM IDENTIFICATION □□□□□□

| LINE | FORM TYPE | CONTROL LEVEL | INDICATORS | FACTOR 1 | OPERATION | FACTOR 2 | RESULT FIELD | FIELD LENGTH | DECIMAL POSITIONS | HALF ADJUST | RESULTING INDICATORS (High / Low / Equal) | COMMENTS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 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.

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.

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. AND/OR relationship is illustrated in figure 9-3. Each AND/OR line, and the previous line, must contain at least one indicator in columns 9-17.

| 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.

01-99 The indicators 01-99 may be used to condition calculation operations that are to be performed only for specific input record types (record identifying indicators), to condition calculation operations that are to be performed only when an input field meets certain conditions (field indicators), or to condition calculation operations according to the results of previous operations in the Calculation Specifications (resulting indicators).

L0-L9 The control level indicators, L1-L9, assigned in the CONTROL LEVEL field (Input Specifications) or the RESULTING INDICATORS field (Calculation Specifications), may be used to condition operations that are to be performed only on the first record of a new Control Group. L0 may be specified, but is always ON.

LR The LR indicator is used to condition operations that are to be performed at End-of-Job.

MR The MR indicator is used to condition operations that are to be performed only when matching input records are found.

d. A table or array name (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, PAGE1, and PAGE2.

f. A label (TAG or ENDSR operation only).

Entries, including numeric literals, in this field must be left-justified, and are dependent upon the particular operation being specified.

28-32 OPERATION

This field is used to specify the proper Operation Code for the type of 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 (except those which belong to subroutines) must appear after those operations not conditioned by control level indicators.

Each of the operation codes is discussed in detail in the second half of the section.

33-42 FACTOR 2

This field is used to supply the data to be operated upon by the Operation Code specified in columns 28-32. Allowable entires 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 table or array name (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, UMINTH, UDAY, UYEAR, PAGE, PAGE1, and PAGE2.

f. A label (GOTO operation only).

g. A filename (DEBUG, DSPLY, CHAIN, READ,or FORCE operation only).

Entries in this field must be left-justified, and are dependent upon the particular operation being specified.

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

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 is 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 conjunction with 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 Section 2 for a complete description.

## OPERATION CODES

Various categories of Operation Codes are provided to allow most types of 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 arithmetic or logical operation is to be performed on the associated operands. In the following paragraphs an indexed vector or a table name is valid anywhere that a field name is valid.

Table 9-1 summarizes each of the operation codes discussed in the following paragraphs.

### ARITHMETIC OPERATIONS

Arithmetic operations are only allowed on numeric data items. FACTOR 1 and FACTOR 2 must name numeric fields or contain numeric literals; the RESULT FIELD must be numeric. All results will be signed and decimal alignment will be performed. If the RESULT FIELD is not large enough to hold the result of an arithmetic operation, the result will be truncated at the most significant end.

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.

A whole array may be entered in FACTOR 1 or FACTOR 2, as long as the RESULT FIELD also specifies a whole array. In this case, the designated arithmetic operation will be performed on each element of the array or arrays designated as factors, 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 shortest array is reached. If the RESULT FIELD is an array, and FACTOR 1 and FACTOR 2 are not arrays, the result of the operation performed on FACTOR 1 and FACTOR 2 is placed in all elements of the RESULT FIELD 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 9-2 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.

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; if the RESULT FIELD is shorter, truncation may occur. 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:

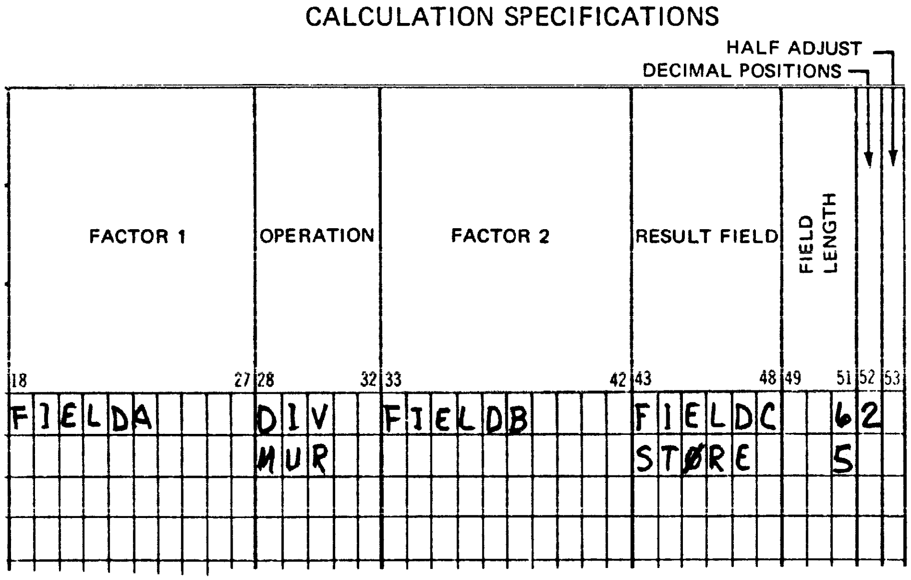
   1) The number of decimal positions in FACTOR 1 of the previous DIV operation.

   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.



CALCULATION SPECIFICATIONS

| FACTOR 1<br>18–27 | OPERATION<br>28–32 | FACTOR 2<br>33–42 | RESULT FIELD<br>43–48 | FIELD LENGTH<br>49–51 | DECIMAL POSITIONS<br>52 | HALF ADJUST<br>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.

## 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.

MOVE operations are shown in table 9-4.

Table 9-4. MOVE Operations

| Contents of FACTOR 2 | Contents of Numeric RESULT FIELD | | Contents of Alphanumeric RESULT FIELD | |
|---|---|---|---|---|
| | Before MOVE Operation | After MOVE Operation | Before MOVE Operation | After MOVE Operation |
| AB4SK | +123456789 | -123412422 | 123456789 | 1234AB4SK |
| +9876543 | +123456789 | +129876543 | ABCDEFGHI | AB987654C |
| GEBKLM4SK | +56789 | -34422 | ABCDE | LM4SK |
| -9876543 | +56789 | -76543 | ABCDE | 7654L |
| AB4SK | +56789 | -12422 | 56789 | AB4SK |
| -87654 | +12345 | -87654 | ABCDE | 8765M |

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.

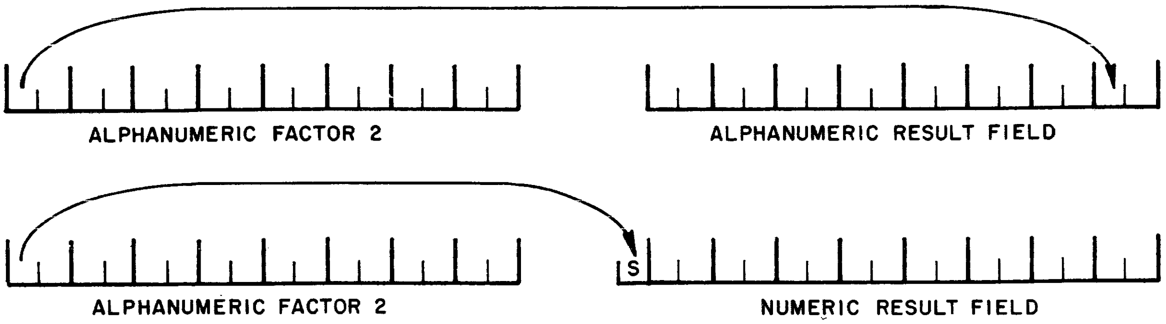
The sign is transferred only if the RESULT FIELD is numeric and not greater in length than FACTOR 2 or if the result is alphanumeric and equal in length to FACTOR 2. The sign of FACTOR 2 is the algebraic sign if it is a numeric data item or low order zone if FACTOR 2 is alphanumeric.

MOVEL operations are shown in table 9-5.

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.

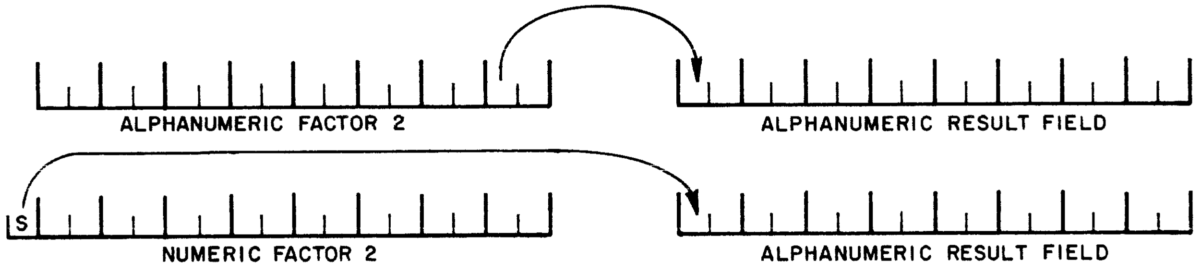
For the purpose of move zone operations, the "zone" of a variable is defined as (1) the algebraic sign, if the variable is numeric; or, (2) the low-order or high-order zone, if the variable is alphanumeric. For a numeric field only the low zone, which is defined as the sign position, may be referenced.



G14059

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).

Comparison of Alphanumeric Fields

The comparison of alphanumeric fields begins with the leftmost character of each and proceeds, character-by-character, to the right until a pair of unequal characters is encountered. These characters are compared according to the EBCDIC collating sequence, and the field that contains the character found to be higher in the collating sequence is considered to be the HIGH field (see Appendix A).

If the character-by-character comparison reaches the end of the fields, they are considered to be EQUAL.

Both fields are automatically aligned on their leftmost characters, and trailing spaces are supplied to the shorter field.

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:

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.

<u>Columns 54-55</u>

An indicator in these columns is turned ON if each bit specified in FACTOR 2 is OFF (0) in the Result Field.

<u>Columns 56-57</u>

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.

<u>Columns 58-59</u>

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.

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

The lookup operation is used to search a table or array (vector) for a special element.

LOKUP This operation is used to search a vector for a particular data item. The vector name is entered in FACTOR 2. FACTOR 1 is used to name a search word (data for which it is desired to find a match in the vector). The LOKUP operation causes the designated vector to be searched in an attempt to find an element that matches the search word.

b. An indicator assigned to HIGH (columns 54-55) specifies that the vector is to be searched for the item that is nearest to but higher in sequence than the search word. The first such entry found causes the indicator assigned to HIGH to turn ON.

c. An indicator assigned to LOW (columns 56-57) specifies that the vector is to be searched for the item that is nearest to but lower in sequence than the search word. The first such entry found causes the indicator assigned to LOW to turn ON.

The following rules must be observed when assigning RESULTING INDICATORS to a LOKUP operation:

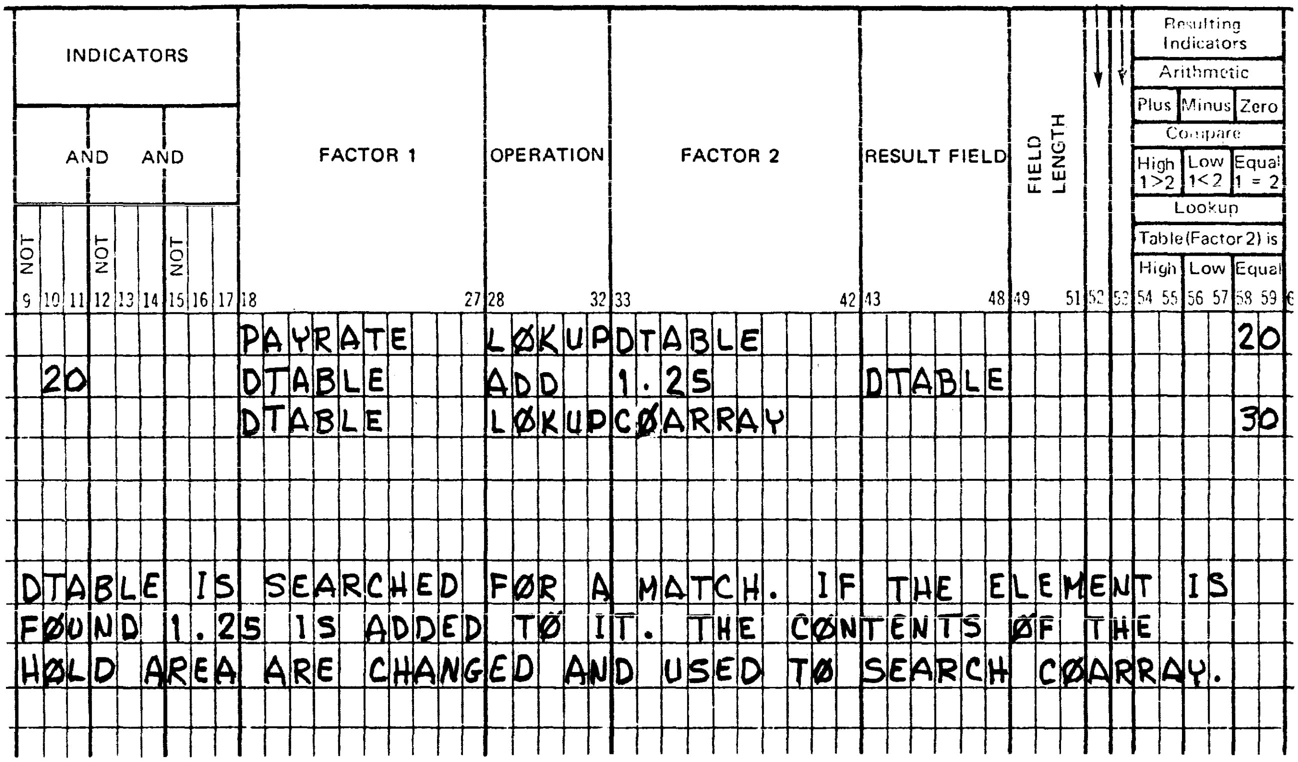
a. At least one RESULTING INDICATOR must be assigned, but more than two are not allowed.

b. If two RESULTING INDICATORS are assigned, one of them must be assigned to EQUAL. The program then searches for an item that satisfies either condition with EQUAL given precedence; that is, if no EQUAL entry is found, then the nearest low or higher entry satisfies the search.

c. Any search operation for other than an EQUAL condition (LOW, HIGH, LOW and EQUAL, 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; a vector out of sequence will cause undefined results.

d. If the search is successful, the RESULTING INDICATOR designating the type of search is turned ON. If two indicators are assigned, only the one indicating the comparison of the search word and the vector element will be turned ON.

e. If no element of the vector being searched satisfies the conditions, no RESULTING INDICATOR will be turned ON.



| INDICATORS (9–17) | FACTOR 1 (18–27) | OPERATION (28–32) | FACTOR 2 (33–42) | RESULT FIELD (43–48) | FIELD LENGTH (49–51) | Resulting Indicators: Plus / High 1>2 / High (54–55) | Minus / Low 1<2 / Low (56–57) | Zero / Equal 1 = 2 / Equal (58–59) |
|---|---|---|---|---|---|---|---|---|
| | PAYRATE | LØKUP | DTABLE | | | | | 20 |
| 20 | DTABLE | ADD | 1.25 | DTABLE | | | | |
| | DTABLE | LØKUP | CØARRAY | | | | | 30 |

DTABLE IS SEARCHED FØR A MATCH. IF THE ELEMENT IS FØUND 1.25 IS ADDED TØ IT. THE CØNTENTS ØF THE HØLD AREA ARE CHANGED AND USED TØ SEARCH CØARRAY.

G14066

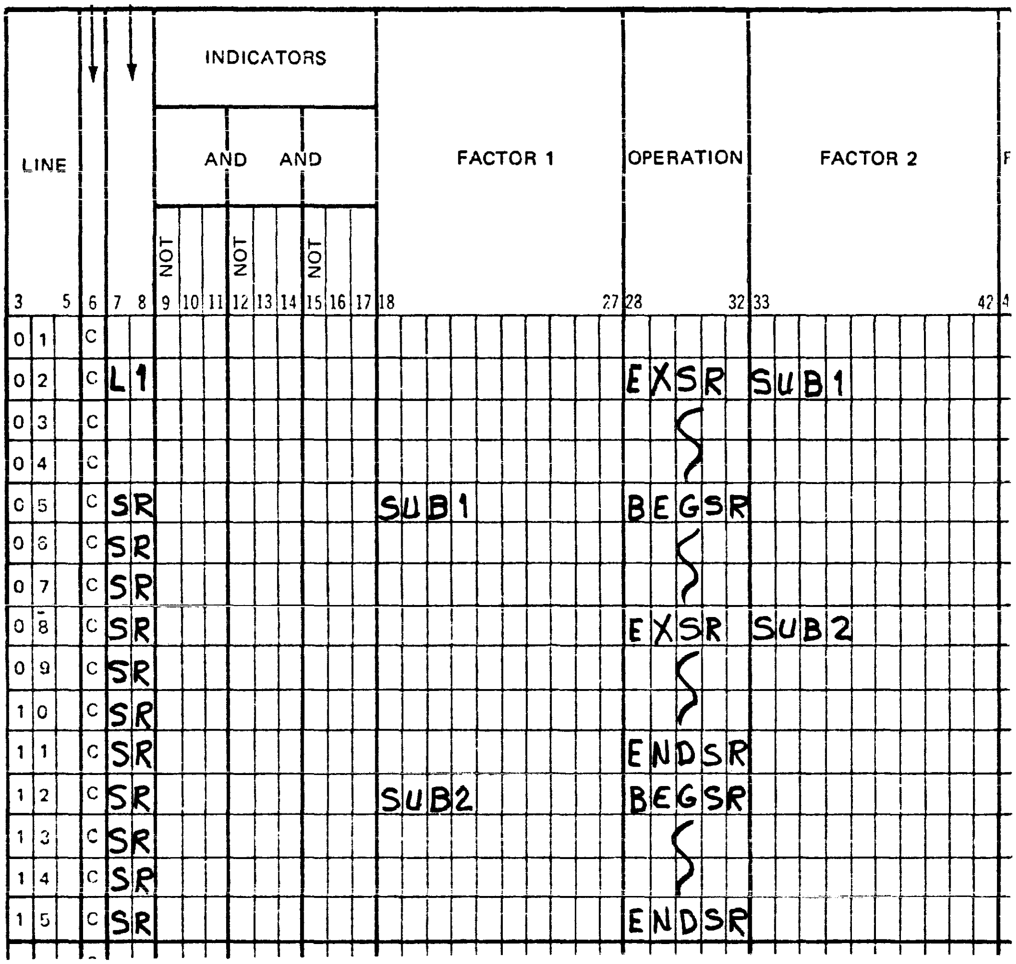
Figure 9-15. Using a Table in an Arithmetic Operation After LOKUP

If a data item is assigned to the vector as an index, it will contain the number of the element that satisfied the conditions. For an array, the element is not saved in a hold area; the index only points to the location of the element in the array.

If the search is not successful, no RESULTING INDICATORS are turned ON. If FACTOR 2 is a table name, the hold area is not affected, but the hold pointer is set to point to the first element of the table. If FACTOR 2 is a vector with a data item assigned as index, the index is set to 1.

## Two-Vector LOKUP

When searching two vectors, FACTOR 1 (the search word), FACTOR 2 (the vector to be searched), the RESULT FIELD (a corresponding or related vector), and at least one (but not more than two) RESULTING INDICATORS must be specified. CONTROL LEVEL and conditioning INDICATORS may also be used (see figure 9-16).



| LINE | 6 | 7 8 | INDICATORS 9-17 | FACTOR 1 | OPERATION | FACTOR 2 |
|---|---|---|---|---|---|---|
| 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 9-17. Subroutine Coding Examples

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 1. 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 allowing exits 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.

Direct Files

FACTOR 1 must contain the actual key (relative record number), either as a numeric literal (which must be integral and not less than 1) or as the name of a numeric field (which must be integral) that contains the key. FACTOR 2 must contain the filename for the file from which the record will be read. This file must be described as a direct chain file on the File Specification.

The actual key specified will be converted automatically to an absolute disk address, from which location the input record will be read. If the designated key points to a record outside the limits of the file, the indicator specified in columns 54-55 will turn ON and the record will not be read.

Indexed-Sequential Files

FACTOR 1 must contain the name of the data key, the field that will be compared against the index,and the KEY FIELDS in the data file, in order to locate the desired record. FACTOR 1 may be alphanumeric or numeric. FACTOR 2 must contain the filename for the file from which the record will be read. The filename must have been declared on the File Specifications as a chained indexed disk file.

The data key specified will be compared to the entries in the index, which will point to a portion of the file on disk. That portion of the file pointed to by the index will be searched until the data key matches the KEY FIELD of the data record. If the desired record cannot be located in the file, the indicator specified in columns 54-55 will turn ON and no record will be read. (If no indicator is specified then the program will be terminated.)

The field length of FACTOR 1 must be the same as the key length specified for the file on the File Description Specifications. Note that if FACTOR 1 is a literal, leading zeros or trailing blanks may be required to achieve this.

The following paragraphs discuss how the CHAIN operation functions for each of the file types (input, output, and update).

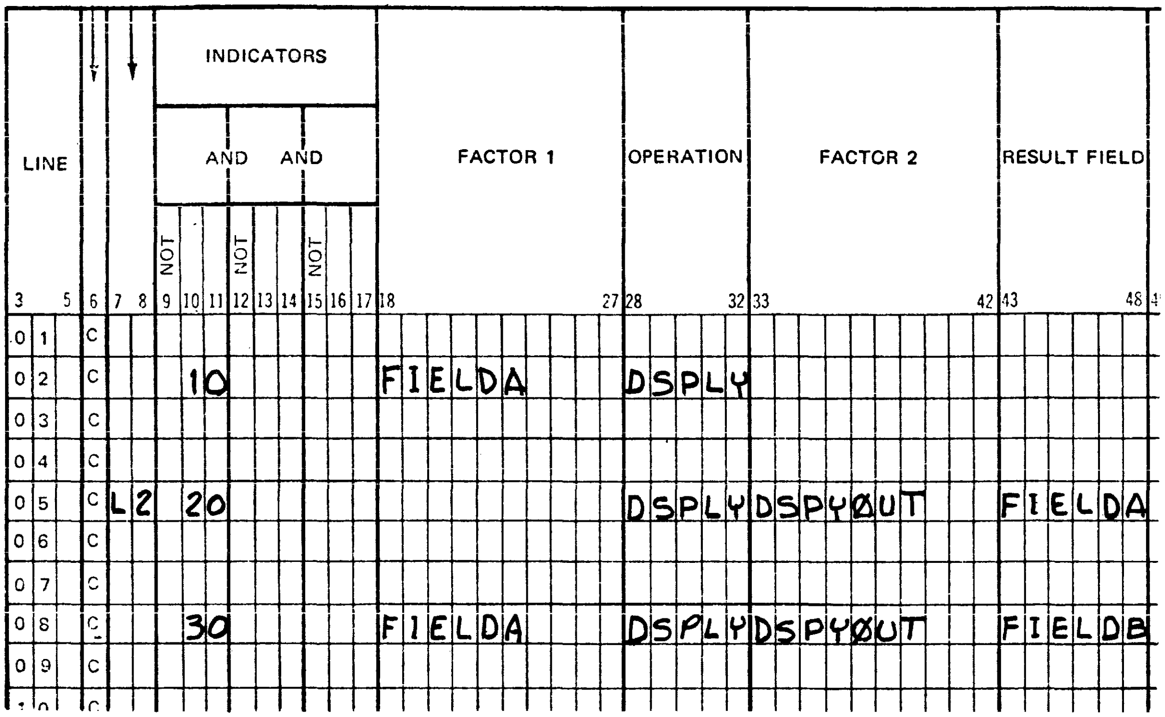
Chained Input Files

For chained files declared as input, the CHAIN operation causes a record to be read during calculations. The desired record must be identified to the program, so that the proper input data will be made available from disk. For direct files, an actual key (relative record number) is used; for indexed files, a data key is used.

Chained Output Files

The CHAIN operation is applicable to direct output files only; it is not allowed for indexed files.

For chained files declared as output, the CHAIN operation causes the proper disk address to be generated for writing the output record during normal output operations. The desired record location must be made available prior to output, so that the record will be written into the proper place in the file.



| LINE | | | INDICATORS | AND | AND | FACTOR 1 | OPERATION | FACTOR 2 | RESULT FIELD |
|---|---|---|---|---|---|---|---|---|---|
| 3 5 | 6 | 7 8 | NOT 9 10 11 | NOT 12 13 14 | NOT 15 16 17 | 18 27 | 28 32 | 33 42 | 43 48 |
| 01 | C | | | | | | | | |
| 02 | C | | 10 | | | FIELDA | DSPLY | | |
| 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 | | | | | | | | |

G14069

Figure 9-18. DSPLY Operation Coding Examples

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 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. (See figure 9-18, line 8.)

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, of course, 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.

## 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.

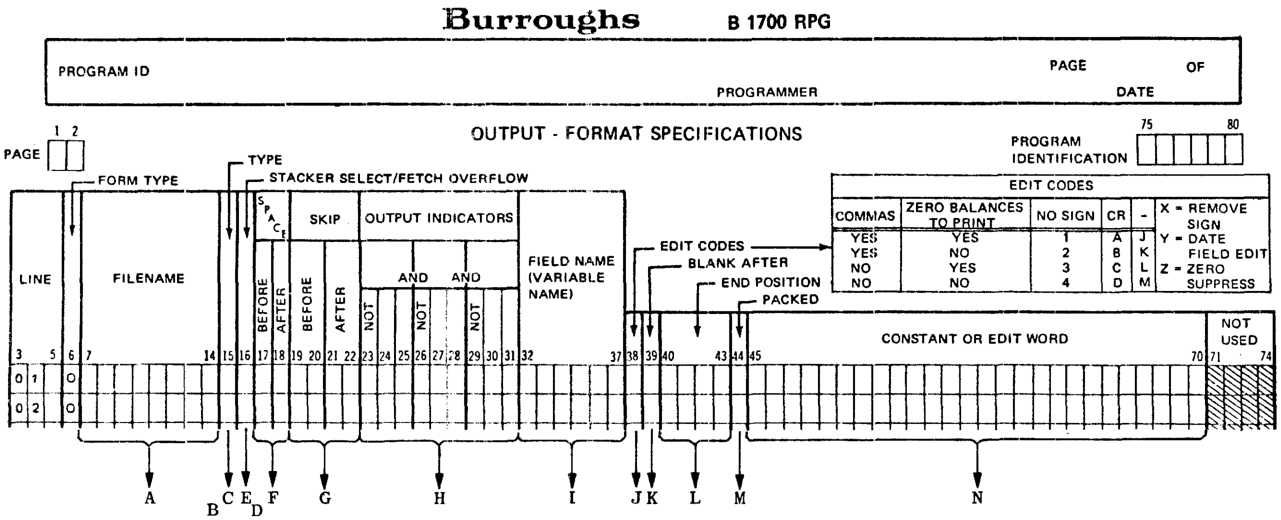
The DEBUG operation may be conditioned by indicators in columns 7-17. Columns 49-59 must be left blank. RESULT FIELD is optional, but if specified it must be a field, vector, or indexed vector whose contents are written on the second and subsequent records.

### 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 9-19 for an example of the output produced by the DEBUG operation.

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. |



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.

I. 32-37 Contains a previously defined field name or vector or one of the special field names PAGE, PAGE1, PAGE2, *PLACE, *PRINT, UDATE, UMONTH, UYEAR, UDAY.

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 packed decimal format. Entries: Blank or P.

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).

G14071

Figure 10-1. Output-Format Specifications Summary Sheet

## 16 STACKER SELECT/FETCH OVERFLOW

This field may be used to specify:

a. The stacker into which the output card is to be placed after being punched, or

b. That the overflow routine can be invoked at this point for a printer file.

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

Only card output or combined files may be stacker selected. Output files may be stacker selected only in the Output-Format 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.

Stacker selection on the basis of matching records should be specified only for detail output lines, because the MR indicator is on only at detail output time to signal the matching status. Record 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.

### 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 takes place. These are described later in this section in the discussion on overflow indicators. Briefly, remaining detail lines, total lines, and overflow lines (lines conditioned by the overflow indicator) are printed on the page after overflow has occurred.

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.

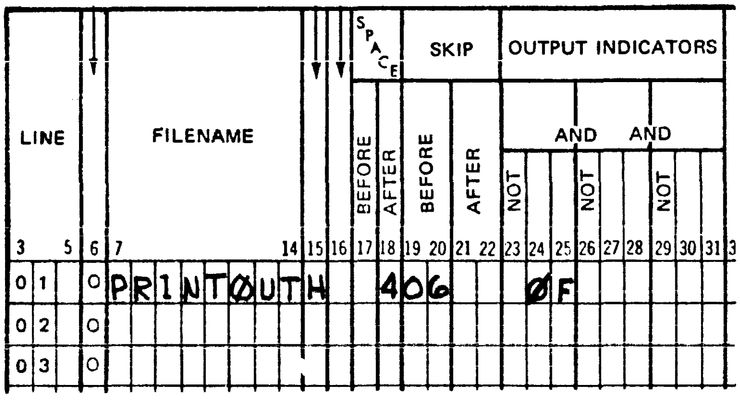
07 fetches overflow.

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 | | 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 |
| 01 | O | PRINTØUTH | | | | 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.

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.

If any of the four subfields contains an entry, all desired skipping and spacing should be specified.

Spacing or skipping to or beyond the overflow line causes the overflow indicator to turn ON. However, spacing or skipping beyond the overflow line to a line on the next page does not cause the overflow indicator to turn ON.

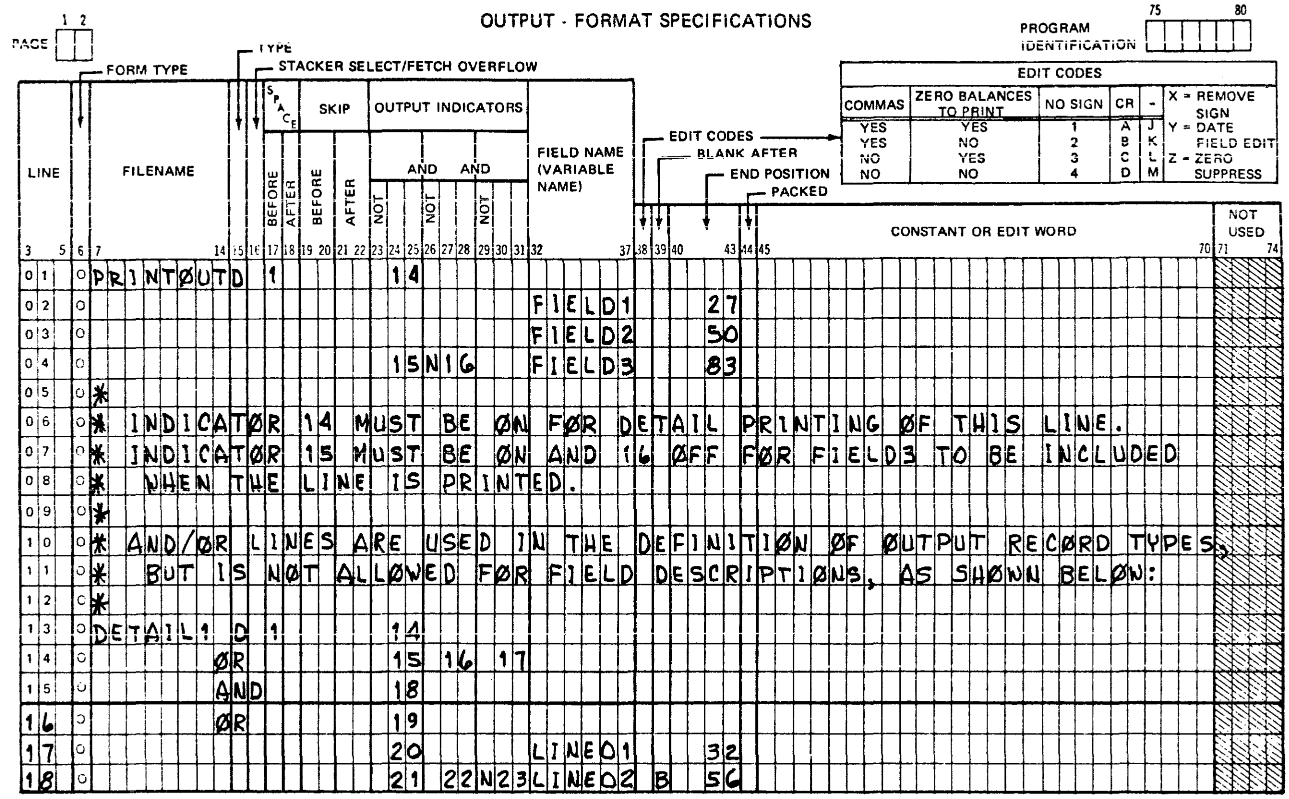
The user should exercise his option to ensure skipping and spacing AFTER printing since it is more efficient than skipping and spacing BEFORE printing.

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.

19-22 SKIP (RPG 1 DIALECT)

If RPG 1 dialect is specified on the Control Card Specifications, forms skipping is related directly to the operation of the printer carriage control tape. The SKIP field is divided into two subfields such that skipping both BEFORE and AFTER printing may be specified. Valid entries for this field are 1 thru 11, which designate skip to channel 1 thru 11. Skipping is done according to the format punched in the carriage control tape, and sensing a channel 12 punch will cause the proper overflow indicator to be turned ON.

Line Counter Specifications are not required. If, however, the printer output is going to be written to backup during execution, Line Counter Specifications must be included if overflow output is ever to occur. They are used to associate line numbers with the channel numbers referenced, so that the program will be able to determine when the overflow line has been reached.



OUTPUT - FORMAT SPECIFICATIONS

| EDIT CODES | | | | | | |
|---|---|---|---|---|---|---|
| COMMAS | ZERO BALANCES TO PRINT | NO SIGN | CR | - | X = REMOVE SIGN |
| YES | YES | 1 | A | J | Y = DATE FIELD EDIT |
| YES | NO | 2 | B | K | Z - ZERO SUPPRESS |
| NO | YES | 3 | C | L | |
| NO | NO | 4 | D | M | |

```
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
```

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.

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.

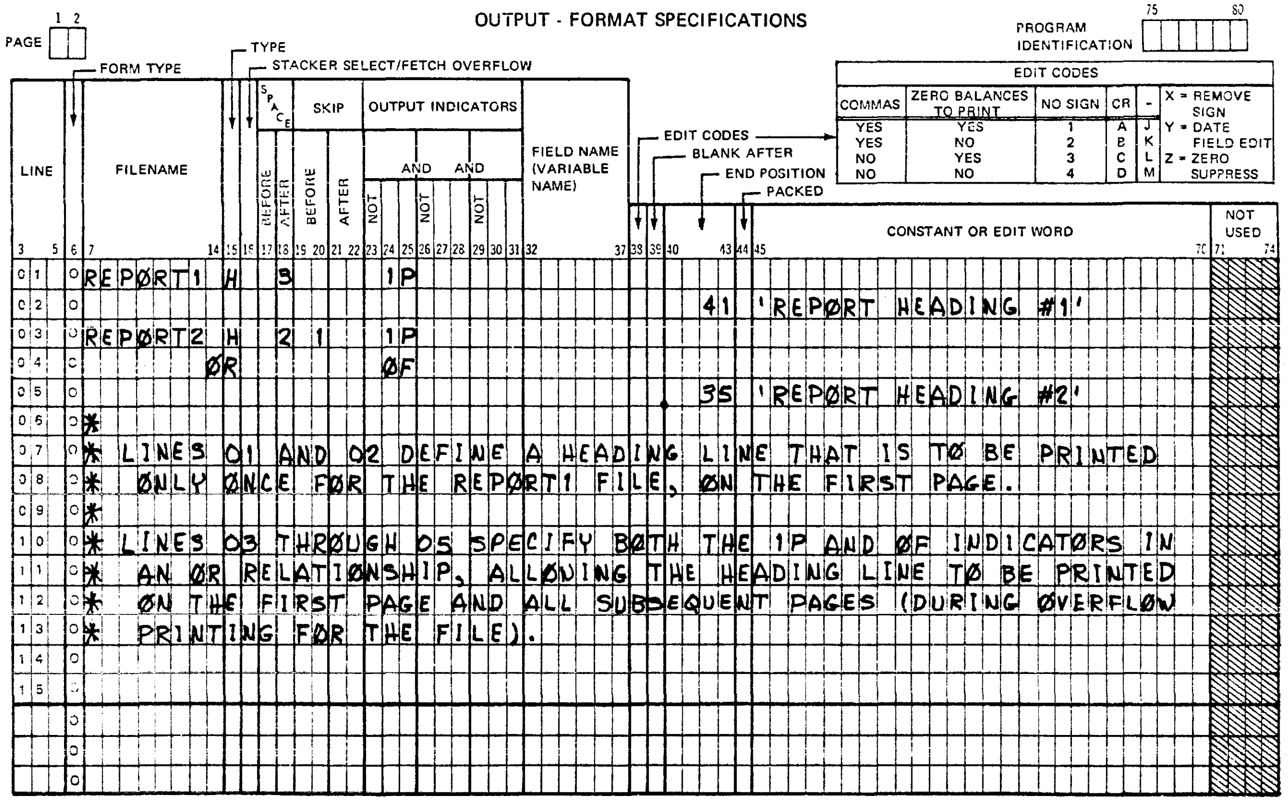
b. Spacing past the overflow line causes the overflow indicator to turn on.

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.

1P The first page indicator is used to allow printing of heading information on the first page of a printer output file. In conjunction with an overflow indicator in an OR relationship, it may be used to allow printing of the same heading information on all pages. The use of the 1P indicator is illustrated below (see figure 10-8).

The first page indicator is invalid for update and combined files.



OUTPUT - FORMAT SPECIFICATIONS

PAGE 1 2
PROGRAM IDENTIFICATION 75 80
FORM TYPE
TYPE
STACKER SELECT/FETCH OVERFLOW
LINE
FILENAME
SPACE
BEFORE
AFTER
SKIP
BEFORE
AFTER
OUTPUT INDICATORS
AND
AND
NOT
NOT
NOT
FIELD NAME (VARIABLE NAME)
EDIT CODES
BLANK AFTER
END POSITION
PACKED
CONSTANT OR EDIT WORD
NOT USED
3 5 6 7 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 37 38 39 40 43 44 45 70 71 74

| EDIT CODES | | | | | |
|---|---|---|---|---|---|
| COMMAS | ZERO BALANCES TO PRINT | NO SIGN | CR | - | X = REMOVE SIGN<br>Y = DATE FIELD EDIT<br>Z = ZERO SUPPRESS |
| YES | YES | 1 | A | J | |
| YES | NO | 2 | B | K | |
| NO | YES | 3 | C | L | |
| NO | NO | 4 | D | M | |

```
01 OREPORT1 H  3     1P
02 O                                    41 'REPORT HEADING #1'
03 OREPORT2 H  2 1   1P
04 O       ØR        ØF
05 O                                    35 'REPEPØRT 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
   O
   O
   O
```

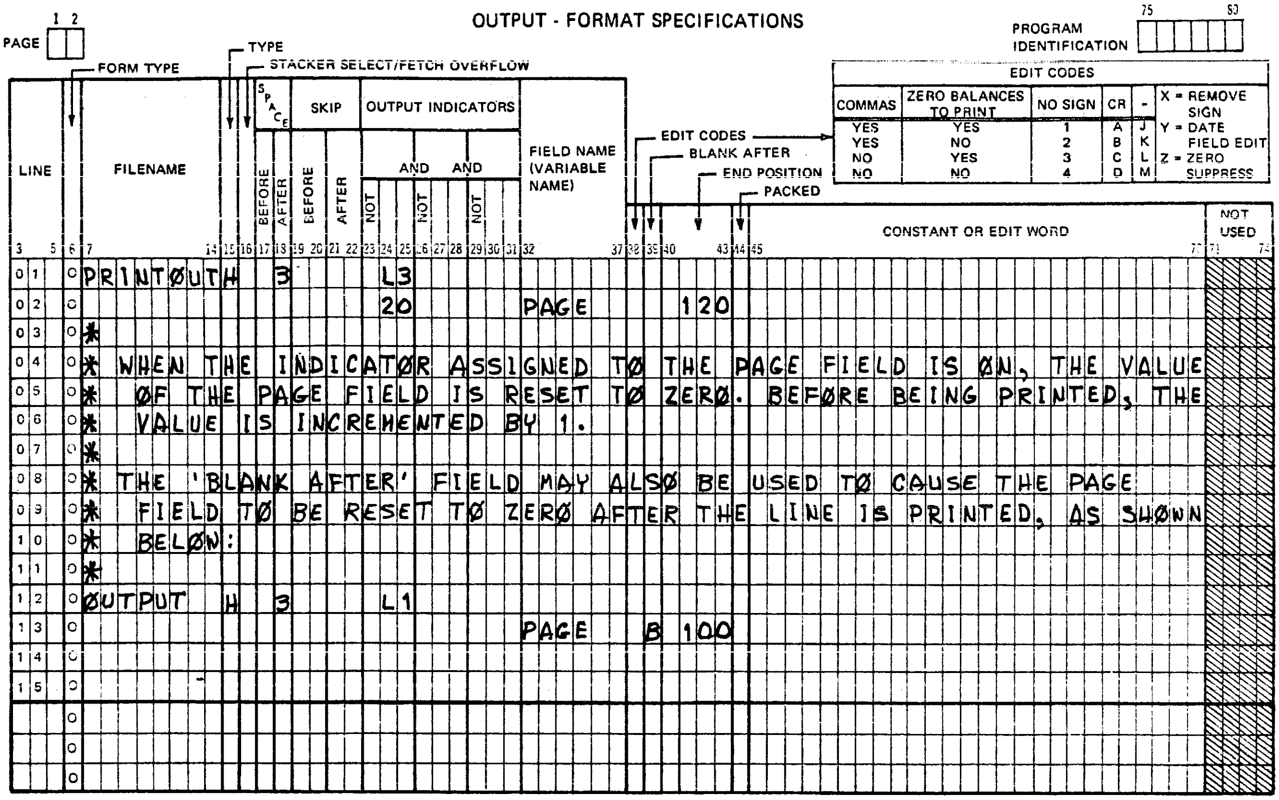
G14078

Figure 10-8. 1P Indicator Coding Example

The first page indicator can be used to condition heading or detail lines.

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 indi-



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
CONSTANT OR EDIT WORD
NOT USED
PROGRAM IDENTIFICATION

| EDIT CODES | | | | | |
|---|---|---|---|---|---|
| COMMAS | ZERO BALANCES TO PRINT | NO SIGN | CR | - | X = REMOVE SIGN<br>Y = DATE FIELD EDIT<br>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
```

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 entry (PAGE, PAGE1, or PAGE2) 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, and UYEAR)

Four special words allow the program to obtain the current value of the date as supplied through the 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.

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.

Table 10-1. Edit Codes Table

| | | | | | Printout on Zero Balance* | | |
|---|---|---|---|---|---|---|---|
| Edit Code | Commas | Decimal Point | Zero Suppress | Sign For Negative 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:

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.

## 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 record, in the position specified. The following rules must be observed when forming constants:

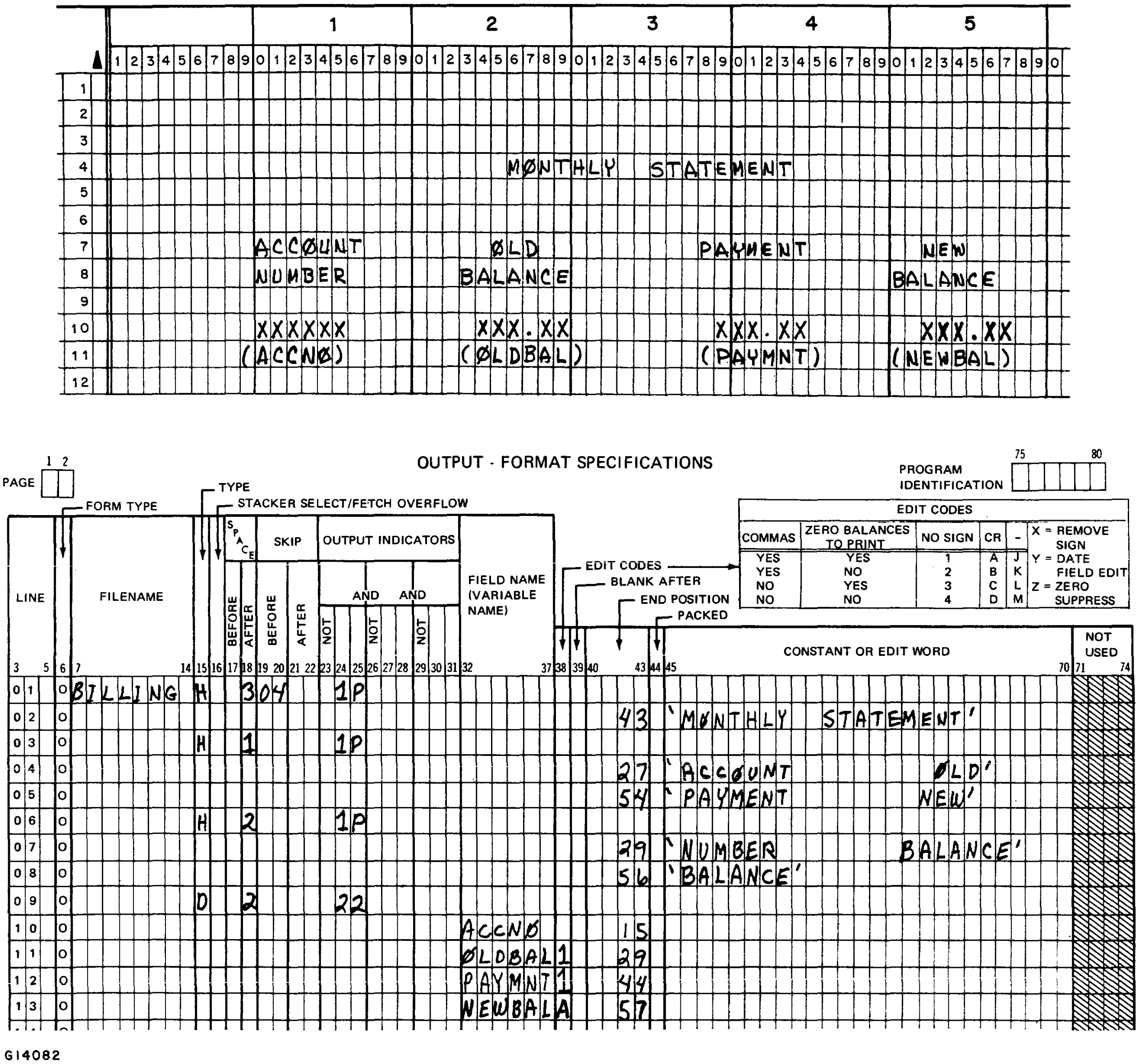
a. The VARIABLE NAME entry must be left blank.

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.

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.



```
                   1         2         3         4         5
    1234567890123456789012345678901234567890123456789012345678901
 1
 2
 3
 4                            MØNTHLY STATEMENT
 5
 6
 7           ACCØUNT        ØLD                  PAYMENT NEW
 8           NUMBER       BALANCE                     BALANCE
 9
10           XXXXXX        XXX.XX         XXX.XX       XXX.XX
11          (ACCNØ)       (ØLDBAL)       (PAYMNT)    (NEWBAL)
12
```

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 3-5 | FORM TYPE 6 | FILENAME 7-14 | TYPE 15 | STACKER SELECT/FETCH OVERFLOW 16 | SPACE BEFORE 17 | SPACE AFTER 18 | SKIP BEFORE 19-20 | SKIP AFTER 21-22 | OUTPUT INDICATORS NOT 23 | 24-25 | AND NOT 26 | 27-28 | AND NOT 29 | 30-31 | FIELD NAME (VARIABLE NAME) 32-37 | EDIT CODES 38 | BLANK AFTER 39 | END POSITION 40-43 | PACKED 44 | CONSTANT OR EDIT WORD 45-70 | NOT USED 71-74 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 | | 57 | | | |

G14082

Figure 10-12. Coding of Output Constants

## Edit Words

Edit words are used instead of an edit code when a special edit format is desired. There are two exceptions: '*' and '$' in columns 45-47, which are used in conjunction with edit codes as described under Edit Codes; otherwise column 38 must be blank.

The VARIABLE NAME entry must be a numeric variable. Edit words must be left-justified, enclosed in apostrophes, and no more than 24 characters in length. Columns 40-43 (END POSITION) must contain an entry.

An edit word is divided into three sections:

a. The body.

b. The status section.

c. The extension section.

Generally the body contains the special characters used in editing the numeric contents of the corresponding VARIABLE NAME entry.

The status section, if used, follows the body and must contain the editing characters, CR or - (minus). The status section is used to designate whether the numeric variable is positive or negative. Any character following the body section and preceding the CR or - (minus) editing symbols is considered part of the status section. The status section is printed only if the value of the numeric variable is negative.

The extension section, if used, follows the status section and contains information which is not changed by any operation and is always printed as coded.

Any valid RPG character may appear in any of the sections of the edit word, although certain characters have special uses and care should be taken in the placement of these characters. The ampersand always appears as a blank unless RPG 1 dialect is specified on the Control Card Specifications. In this case, the ampersand, if specified, will be printed as an ampersand in the extension section.

Special Characters Used In Edit Words

The character set used for special purposes within edit words consists of the following nine characters:

| | | | |
|---|---|---|---|
| | blank | , | comma |
| 0 | zero | CR | credit symbol |
| $ | dollar sign | * | asterisk |
| . | decimal point | & | ampersand |
| | | - | minus sign |

The characters 0 (zero), blank, and floating dollar sign are used as replacement characters. Their use is shown below.

| Character | Use |
|---|---|
| 0 | Zero suppression. |
| * | Asterisk fill. |
| | Blank. |
| $ | Floating dollar sign (if it appears immediately to the left of zero suppress). |

A fixed dollar sign, decimal points, commas, ampersands, negative signs (CR and -), and constant information are not replaceable characters.

All leading zeros are suppressed unless a zero or asterisk is specified in the edit word. The zero or asterisk indicates that the last leading zero in the edit word field is to be replaced by a blank or asterisk.

Any zeros or asterisks following the leftmost zero or asterisk are treated as constants (they are not replaceable characters).

Any constant to the left of the zero suppression stop character (except $) will be suppressed unless a significant digit precedes the constant.

The number of replacement characters (0, *, blank, $) in the edit word must not be less than the length of the field to be edited.

If the number of replacement characters is greater than the length of the field to be edited and the first character of the edit word is a zero, leading zeros will not be suppressed.

If the number of replacement characters equals the field length and there are leading zeros, the first is suppressed but subsequent zeros are not suppressed.

Example

| Source Data | Edit Word | Result |
|---|---|---|
| 0012345 | '0bbbbb.bb' | 00123.45 |
| 000000 | '0bbbbbb' | 0000000 |

There are two cases where an extra space should be provided in the edit word:

a. An extra space must be left in the edit word for the floating dollar sign, or no dollar sign will appear if the output field is full.

Example

| Source Data | Edit Word | Result |
|---|---|---|
| -724325 | 'bb,b$0.bb&-' | $7,243.25b- |

b. An extra space may be left in the edit word if the first character of the edit word is zero. In this case, the field will not be zero suppressed, but all other specified editing will be performed.

Example

| Source Data | Edit Word | Result |
|---|---|---|
| +007461 | '0bbb,bbb' | 007,461 |

The following paragraphs and corresponding examples describe how the various editing characters function. The symbol b is used to indicate where blank spaces will appear in the output record.

No Editing

If no edit word is used (figure 10-13), the data in the output record has the same format as the unedited data. Note that the low-order position of the output field is printed as an alphabetic character (J-R), if the source data field is negative.

If a blank edit word is used, all leading zeros are suppressed and any sign in the unedited field is removed. Negative values are not identified.

| SOURCE DATA | CONSTANT OR EDIT WORD<br>45 … 70 | OUTPUT RECORD |
|---|---|---|
| +0000123456 | | 0000123456 |
| -0000123455 | | 000012345N |
| 0000000000 | '          ' | bbbbbbbbbb |
| +0000123456 | '          ' | bbbb123456 |
| -0000123456 | '          ' | bbbb123456 |

G14083

Figure 10-13. No Editing Coding Examples

Zero Suppression

When the zero suppression zero is specified (figure 10-14), blanks replace leading zeros and constants until a significant digit is encountered or through the position of the zero suppression zero.

When the zero suppression zero is not specified, zero suppression occurs throughout the field when the value of the source data is zero. Therefore, if it is desired that the zero value be printed in the output field, the zero suppression zero must be specified.

Dollar Sign

If the dollar sign is placed just left of the zero suppression zero, it becomes a floating dollar sign. In an edited data field, the floating dollar sign always appears to the immediate left of the first significant digit. Note that an extra position should be left in the high-order portion of the edit word to accommodate the floating dollar sign.

If a dollar sign is placed at any other point in the body portion of the edit word, it becomes a fixed dollar sign. A fixed dollar sign should generally be placed in the extreme left position of the edit word (see figure 10-15).

| SOURCE DATA | CONSTANT OR EDIT WORD (45–70) | OUTPUT RECORD |
|---|---|---|
| 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 10-14. Zero Suppression Zero Coding Examples

| 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'` | $0000123456bbNET |
| -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

Asterisk Fill

Asterisk fill, generally used for printing checks, follows the same rules as zero suppression (see figure 10-16).

Asterisks replace all positions in the edit word to the left of the first significant digit.

Zero suppression and asterisk fill can be used together. If zero suppression is not used and the asterisk appears in the right most position of the body portion, the entire field will contain asterisks when the value of the source data is zero.

| 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 10-16. Asterisk Fill Coding Examples

Negative Signs

The credit symbol (CR) and the minus sign (-) have special meanings only when they appear in the status portion of an edit word, otherwise, they are treated as constants. CR, -, and any other constant appearing in the status portion of the edit word are printed only if the value of the source data is negative (figure 10-17).

Ampersand

The ampersand is used to ensure spacing in the edit word (figure 10-18). If RPG 1 dialect is specified (figure 10-19), ampersands will appear as ampersands in the extension section of the output record.

Constants in the Edit Word

Constants may appear anywhere in an edit word (figure 10-20). In the body portion, only constants to the right of the most significant digit and/or to the right of a dollar sign, asterisk, or zero in the output field will be printed. Constants in the status portion will be printed only when the value of the source data is negative. Constants in the extension portion are always printed as specified in the edit word.

| SOURCE DATA | CONSTANT OR EDIT WORD |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | OUTPUT RECORD |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  | 45 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 70 |  |
| +0000123456 | \ |  |  |  |  |  |  |  |  |  |  | & | C | R | & | N | E | T | ' |  |  |  |  |  |  |  | bbbb123456bbbbNET |
| +0000123456 | \ |  |  |  |  |  |  |  |  |  |  | & | C | R |  | N | E | T | ' |  |  |  |  |  |  |  | bbbb123456bbbbNET |
| -0000123456 | \ |  |  |  |  |  |  |  |  |  |  | & | C | R |  | N | E | T | ' |  |  |  |  |  |  |  | bbbb123456bCRbNET |
| -0000123456 | \ |  |  |  |  |  |  |  |  |  |  | & | - |  |  | N | E | T | ' |  |  |  |  |  |  |  | bbbb123456b-bbNET |
| 0000123456 | \ |  |  |  |  |  |  |  |  |  |  | & | N | E | T | & | C | R | ' |  |  |  |  |  |  |  | bbbb123456bbbbbbb |
| -0000123456 | \ |  |  |  |  |  |  |  |  |  |  | & | N | E | T | & | C | R | ' |  |  |  |  |  |  |  | bbbb123456bbNETbCR |
| -1234567890 | \ | $ | & |  |  | , |  |  |  | , |  |  | O | . |  |  | C | R | ' |  |  |  |  |  |  |  | $b12,345,678.90CR |
| +0000123456 | \ |  |  | , |  |  |  | , |  |  | * | . |  |  | * | C | R | * | * | ' |  |  |  |  |  |  | *****1,234,56bbb** |
| -0000123456 | \ |  |  | , |  |  |  | , |  |  | * | . |  |  | * | C | R | * | * | ' |  |  |  |  |  |  | *****1,234.56*CR** |

G14087

Figure 10-17. Negative Signs Coding Examples

| SOURCE DATA | CONSTANT OR EDIT WORD |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | OUTPUT RECORD |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  | 45 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 70 |  |
| -1234567890 | \ | $ | & |  |  | , |  |  |  | , |  | O |  | . |  |  | & | - |  | G | R | Ø | S | S | ' |  | $b12,345,678.90b-bGROSS |
| -1234567890 | \ | $ | & |  |  | , |  |  |  | , |  | O |  | . |  |  | & | - | N | E | T | & | P | A | Y | ' | $b12,345,678.90b-NET PAY |
| -0000123456 | \ | $ | & |  |  | , |  |  |  | , |  |  | * | . |  |  | C | R | ' |  |  |  |  |  |  |  | $******1,234.56CR |
| +000002 | \ |  |  | O |  | L | B | S | . | & |  |  | Ø | Z | . | T | A | R | E | & | - | ' |  |  |  |  | bbb0LBS.b02bbbbbbbbb |
| 031274 | \ |  |  | & |  |  | & |  |  | & | L | A | T | E | R | ' |  |  |  |  |  |  |  |  |  |  | b3b12b74bLATER |

G14088

Figure 10-18. Ampersand Coding Examples (RPG II Dialect)

| SOURCE DATA | CONSTANT OR EDIT WORD |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | OUTPUT RECORD |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|  | 45 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 70 |  |
| 0000123456 | \ |  |  |  |  |  |  |  |  |  |  | & | & | P | R | Ø | F | I | T | ' |  |  |  |  |  |  | bbbb123456&&PROFIT |
| 0000123456 | \ |  |  |  |  |  |  |  |  |  |  | & | C | R | & | N | E | T | ' |  |  |  |  |  |  |  | bbbb123456bbb&NET |
| 1234567890 | \ | $ | & |  |  | , |  |  |  | , |  | O |  | . |  |  | & | - | N | E | T | & | P | A | Y | ' | $b12,345,678.90b-NET&PAY |

G14089

Figure 10-19. Ampersand Coding Examples (RPG 1 Dialect)

| 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.

# 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.

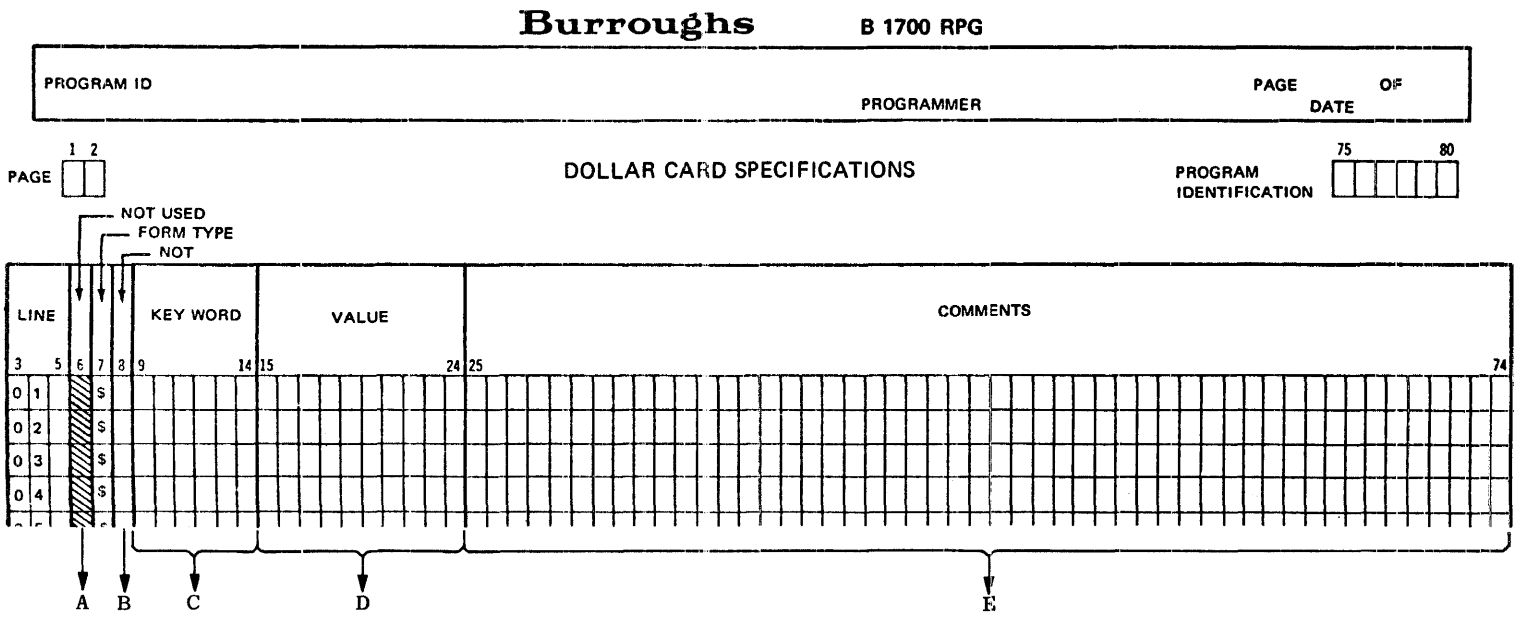
7 $ OPTION

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).

An external name may be in one of the following four forms:

family name<br>
family name/file ID<br>
disk pack ID/family name/file ID<br>
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
03 FMASTER  IP    180
```

MASTER/MASTERFILE

```
02 F$ FAMILYPAYRØLL
03 FMASTER  IP    180
```

PAYROLL

```
02 F$ PACKIDPAYMASTER
03 FMASTER  IP    180
```

PAYMASTER/MASTER/

```
02 F$ FAMILYPAYRØLL
03 F$ FILEIDMASTERFILE
04 FMASTER  IP    180
```

PAYROLL/MASTERFILE

```
04 F$ FAMILYPAYRØLL
05 F$ PACKIDPAYMASTER
06 FMASTER  IP    180
```

PAYMASTER/PAYROLL/

```
02 F$ FAMILYPAYRØLL
03 F$ FILEIDMASTERFILE
04 F$ PACKIDPAYMASTER
05 FMASTER  IP    180
```

PAYMASTER/PAYROLL/MASTERFILE

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 500 unless specified otherwise. If records per area is not an even multiple of blocks per area the compile will default to the nearest even multiple of blocks per area.

$ OPEN — Explicity open allows for all files to be opened at Beginning-of-Job. Default is an implicit open when the files are actually called for.

$ CLOSE — Explicity close allows all input serial files to remain opened until End-of-Job. Default is the implicit close of files at 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 block and record lengths declared on the file header unless the $ REFORM option is used. However, on input or update chained indexed file specifications "data keys in core" option, it may be desirable to also use $ REFORM to indicate to the compiler that it may juggle the blocking factor to optimize the speed of chaining. Under this condition, the block length and record length specified on the File Description Specifications must be the same as when the file was output. This combination will produce the fastest chaining possible.

$ 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 contains the external file identifier of the indexed file including the disk pack ID.

Note that REORG uses the record and block length as specified in the disk file header, unlike the normal sort option which creates the parameters as specified in the File Description Specifications. The record and block length of the file additions must be the same as specified in the disk file header; otherwise the file will not sort correctly.

$ RESIGN — 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.

$ CHAIN — Specifies that only the binary search code is to be emitted for chained indexed disk files.

$ DNAME — The value field contains an alpha mnemonic for the physical hardware device to which it refers.

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 (with the exception of SEG) 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 COBOL 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. Should a N (NOT) be specified in column 8, the indicator will be turned OFF, overriding the original RPG 1 or RPG II specifications.

$ ZBINIT — Specifies that all ZERO or BLANK indicators are initialized ON at Beginning-of-Job; or, if a N (NOT) is specified in column 8, they will be initialized OFF regardless of the specifications for RPG 1 or RPG II.

$ 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. At the completion of the compilation it is necessary to manually execute the RPG/XREF program in order to obtain 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.

$ SEG — Orders the compiler to begin placing code in an overlayable segment identified by the integer in the VALUE field (right-justified, between 0 and 7 inclusive). Segmentation is an automatic function of the RPG compiler. When the SEG option is used, automatic segmentation is not suppressed.

$ 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.

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 Section 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 SOURCE 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 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 BCL (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
```

### 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. The files may be read from a medium other than cards, through use of the FILE statement.

## COMPILER FILE NAMES

Table 12-1 lists the files that are used by the compiler for source input and compilation output.

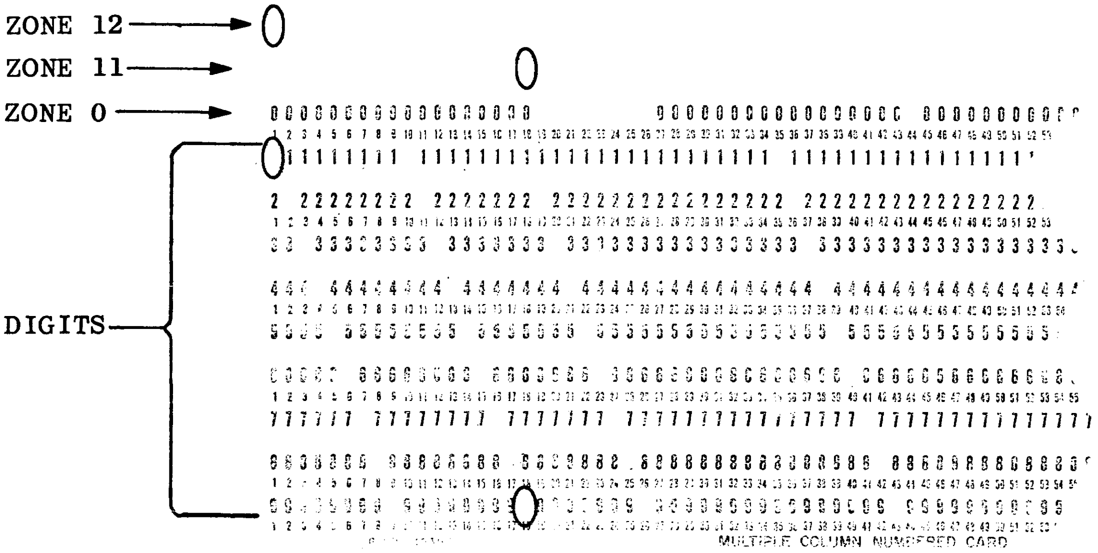
Table 12-1. Files Used for Source Input and Compilation Output

| File Type | Device | Internal File Name | External File Name |
|---|---|---|---|
| Source Input | Card Reader | SOURCE | RPG/CARD |
| Table Input | Card Reader | TABCRD | RPG/VECTOR |
| Output Listing | Line Printer | LINE | RPG/PRINT |

# APPENDIX A

# 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.



G14093

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 1700 character set.

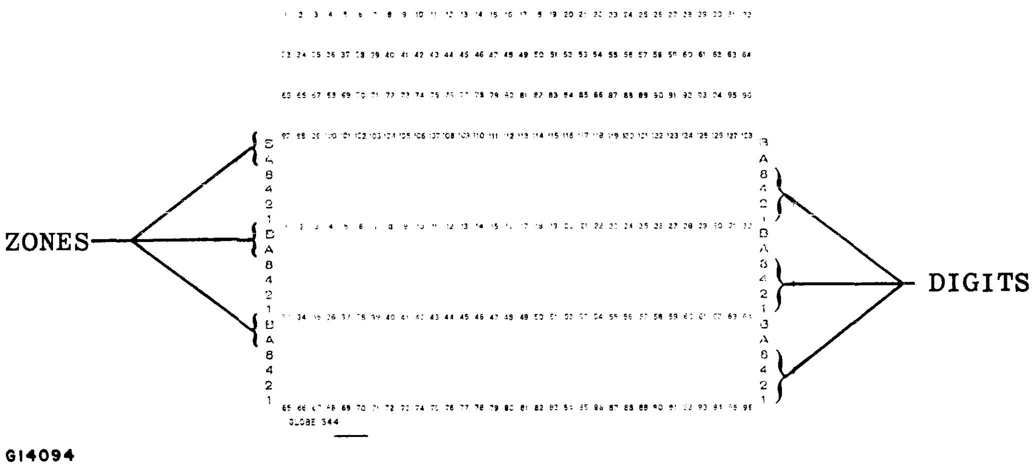


Figure A-2. Zone and Digit Portions of a 96-Column Card

Table A-1. Punch Card Codes for the 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 | 0 |

APPENDIX B

# HEXADECIMAL VALUES FOR THE 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 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 1700 RPG Collating Sequence (Cont)

| Character | Hexadecimal Equivalent |
|---|---|
| U | E4 |
| V | E5 |
| W | E6 |
| X | E7 |
| Y | E8 |
| Z | E9 |
| 0 | F0 |
| 1 | F1 |

Ascending Order

| Character | Hexadecimal Equivalent |
|---|---|
| 2 | F2 |
| 3 | F3 |
| 4 | F4 |
| 5 | F5 |
| 6 | F6 |
| 7 | F7 |
| 8 | F8 |
| 9 | F9 |

Ascending Order

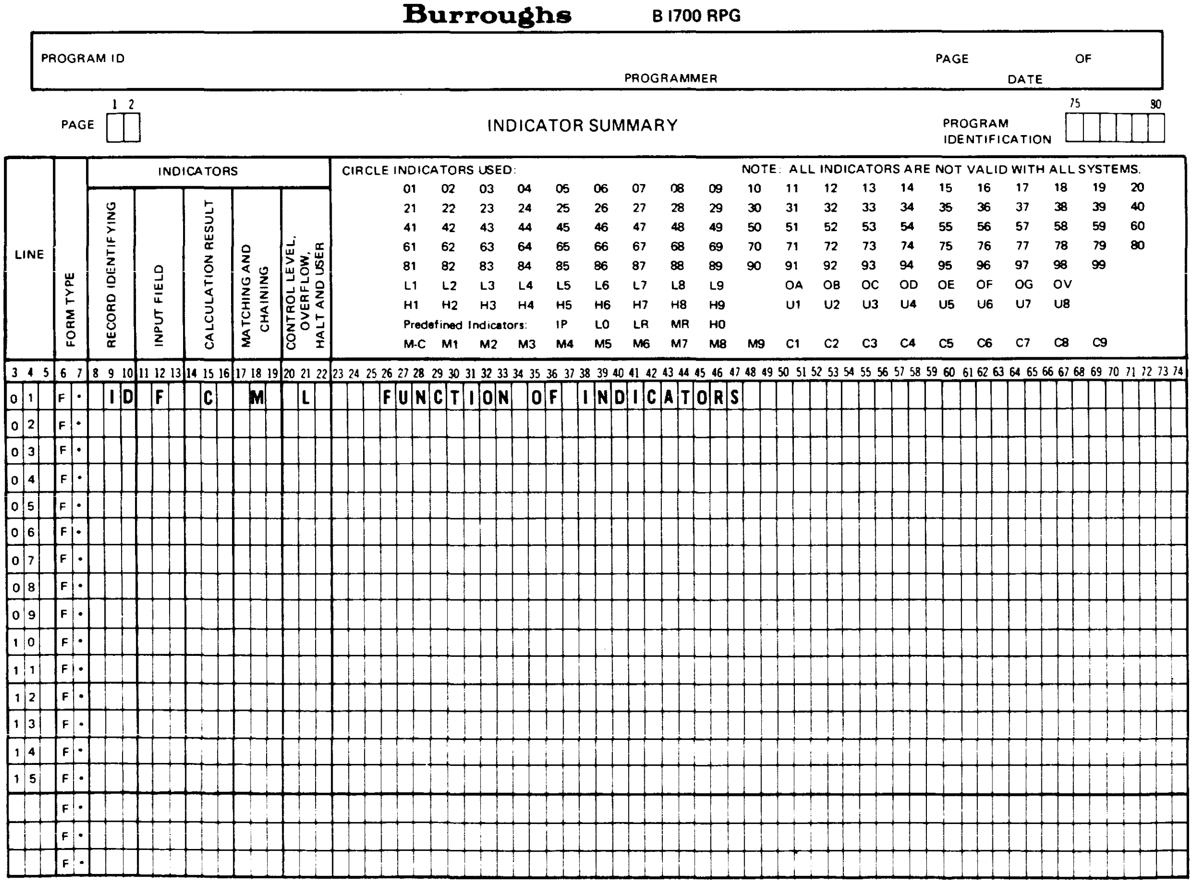
Table B-2. Hex to Binary Conversion Table

| 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

PROGRAMMER

PAGE OF

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 | | | | | | | | | | | |
| M-C | M1 | M2 | M3 | M4 | M5 | M6 | M7 | M8 | M9 | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | |

| LINE | FORM TYPE | | INDICATORS: 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.

<u>LINE COUNTER SPECIFICATIONS</u>

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 O and blank. The O 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 REFERENCE 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, A0-A9, or B0-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, L0-L9, LR, MR, H0, 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 the B 1700 System Software Operational Guide, Form No. 1068731.

## INPUT ERROR MESSAGES

The following messages all denote error conditions arising during input, and all are recoverable. The program will request a reply from the operator. He may respond with "GO" (<mix index>AXGO ), in which case the erroneous record will be ignored and the next record from the same file will be read; or he may respond with "STOP" (<mix index>AXSTOP ), in which case the erroneous record will be ignored, the LR indicator will be turned ON, and all final detail, total calculations, and output will be performed.

The occurrence of the following message results from reading an unidentifiable input record (i.e., none of the designated record Identification Codes for the file could be found in the input record).

```
<program-name> = <mix index> : IDENT
```

The occurrence of the following message results from reading an input record from a file with matching fields specified that is out of sequence. All records in matching files must be in sequence, either ascending or descending.

```
<program-name> = <mix index> : MSEQ
```

The occurrence of the following message results from reading an input record which is out of sequence, as specified by the entries in columns 15-18 of the Input Specifications.

```
<program-name> = <mix index> : SEQ
```

The occurrence of the following message results from reading an execution-time vector record which is out of sequence, as specified by the SEQUENCE fields in the Extension Specifications.

```
<program-name> = <mix index> : VSEQ
```

The occurrence of the following message results when trying to read a demand file which is at End-of-File and no End-of-File indicator has been specified.

```
<program-name> = <mix index> : REOF
```

The occurrence of the following message results when reading a file that contains an error.

```
<program-name> = <mix index> : READ ERROR
```

PROGRAMMED HALTS

The following message will be displayed at the end of a program cycle if any of the halt indicators (H0-H9) are set. Each n will either be blank (indicator not set) or a number 0-9 (indicator set).

```
<program-name> = <mix index> : HALT nnnnnnnnnn
```

If the H0, H3, H4, H7, and H9 indicators were set, the message would appear as:

```
<program-name> = <mix index> : HALT 0  . . 34 . . 7 . 9
```

The program will request a reply from the operator; he may respond with "GO" (<mix index> AXGO ), in which case all halt indicators will be turned OFF and the program will continue; or, he may respond with "STOP" (<mix index> AXSTOP), in which case the LR indicator will be turned ON, and all total calculations and output will be performed.

FORMS POSITIONING

The following message will be displayed after all output lines conditioned by the 1P indicator have been printed, if FORMS POSITIONING has been specified in the Control Card (column 41). The program will request a reply from the operator; if he responds with "YES" (<mix index>AXYES), all 1P lines will be printed again, and the above message will be repeated; if he responds with "NO (<mix index>AXNO), normal processing will begin.

```
<program-name> = <mix index> : AGAIN?
```

Printing of the 1P lines may be requested as many times as necessary in order to align the forms properly.

ARITHMETIC ERRORS

The occurrence of the following message is a result of an attempt to perform a DIV (divide) operation using a divisor of zero. The program will be automatically discontinued (DS-ED) by the MCP.

```
<program-name> = <mix index> : DIVIDE BY ZERO
```

The occurrence of the following message is a result of an attempt to perform a SQRT (square root) operation on a negative argument. The program will request a reply from the operator; he may respond with "GO" (<mix index> AXGO ), in which case the RESULT FIELD will be set to zero and the program will continue; or, he may respond with "STOP" (<mix index> AXSTOP ), in which case the LR indicator will be turned ON, and all final detail, total calculations, and output will be performed.

```
<program-name> = <mix index> : SQRT
```

## VECTOR (ARRAY, TABLE) ELEMENT ERRORS

The occurrence of the following is a result of an attempt to reference a vector element that is out-of-bounds (i.e., 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 program will automatically be discontinued (DS-ED) by the MCP.

```
<program-name> = <mix index> : INVALID SUBSCRIPT
```

Note: Subscript is terminology used for vector element.

## SEQUENCE ERRORS

The occurrence of the following message indicates indexed data key out of sequence. Any response will force EOJ. Program data file will have to be re-sorted or program loading file or updating file will require re-working.

```
<program-name> = <mix index> : KEYSEQ
```

# APPENDIX F

# B 1700 RPG TO COBOL TRANSLATOR (COFIRS II)

The B 1700 RPG to COBOL Translator converts RPG source programs to COBOL source language disk files. It uses files created during the syntax checking and logic generation phases of the B 1700 RPG compiler, thus eliminating the need for a separate syntax checking program. It will translate any RPG 1 or RPG II construct that is acceptable to the B 1700 RPG compiler.

This close relationship to the B 1700 RPG compiler requires the user to be familiar with Burroughs B 1700 RPG as it is defined in the earlier chapters of this manual. For a list of differences between B 1700 RPG and other systems RPG see Section 1.

Three steps are required for the complete translation of an RPG program to B 1700 COBOL. The first is the execution of the B 1700 RPG compiler with special $ card options indicating this is a translation run. This causes intermediate work files to be saved for the use of the translator, and object code is not generated. COFIRS is then executed. It uses the intermediate work files created by the B 1700 RPG compiler to generate COBOL source code on disk.

The newly generated COBOL program is then compiled to provide executable object code. The compiled COBOL program will generally require approximately the same amount of memory for execution, and execution time will be approximately the same as the original RPG program would have required if compiled and executed on the B 1700 computer.

## COFIRS II OPERATION

The following three steps are required for the COFIRS II operation:

a. Section 12 describes the control card syntax required by the B 1700 RPG compiler phase of the translation. Briefly, it is as follows:

```
? COMPILE <program-name> WITH RPG LIBRARY
? DATA RPG/CARD
  $ option cards
  source specification cards
? END
```

The $ option cards to be included in the RPG source deck for use by COFIRS II are coded in the same manner as other $ option cards used by the B 1700 RPG compiler, with the $ in column seven and the COFIRS II option beginning in column nine. Only one option may be entered per card, but several $ cards may be entered as required by the user (see Section 11). The $ options PACKID, FAMILY, FILEID, AREAS, RPERA, and RSIGN are all applicable as needed in the RPG source program. The $ options MAP and XMAP are not applicable and will be ignored in the COFIRS II translation.

The following $ options are available for use exclusively for COFIRS II translations:

| | |
|---|---|
| $ XLATE | This option is required for translation of an RPG program to COBOL on the B 1700. It indicates to the B 1700 RPG compiler that this is a translation run and that object code is not to be developed for the RPG program. |
| $ SPERSE | This option will cause RPG statements to be interspersed with the generated COBOL source statements as comments. |
| $ XLIST | This option causes the translated COBOL statements to be printed as they are developed during the execution of COFIRS II. |

b. After the RPG compiler goes to End-of-Job, it is necessary to manually execute COFIRS II.

```
$ EXECUTE COFIRS
```

COFIRS II creates the COBOL source file on disk with the file name RPGCOB.

c. In order to compile the newly translated COBOL program, it is necessary either to change the name of the RPGCOB file to COBOLW/SOURCE or to label equate it. A possible set of compilation control cards might be:

```
? COMPILE <program-name> COBOL LIBRARY
? FILE SOURCE NAME = RPGCOB;
? DATA CARDS
   $ MERGE
? END
```

The <u>B 1700 Systems COBOL Reference Manual</u>, Form No. 1057197, describes the $ options available to the user for COBOL compilations. The <u>B 1700 System Software Operational Guide</u>, Form No. 1068731, describes the use of the FILE statement for label equation. RPGCOB is a disk file consisting of 80-character records blocked two.

The results of converting an RPG program into COBOL are discussed below:

a. The generated data-names are described in the following example:

03 REPORT-X-UA-4-5-154-RED is a field in a file description,

where:

| | |
|---|---|
| REPORT | is the file name used on the File Description Specifications in the RPG program. |
| UA | represents the data format of this field (UA, SA, UN, or SN are possible). |
| 4 | is the digit displacement of this field in the record. |
| 5 | is the width of this field in digits or bytes, depending on the PICTURE associated with this field. |
| 154 | is the relative position of this field in an internal table. |
| RED | indicates that this is a redefinition of another field. |

b. The actual data-names from the RPG program will be found in the WORKING-STORAGE SECTION with -F appended. These fields are filled in the INPUT-MOVE portion of the COBOL program and are used in the various calculations as required by the Calculation Specifications in the RPG program.

c. Compile-time vectors are entered in the same manner as if compiling RPG. These tables become WORKING-STORAGE SECTION entries containing the vector values entered at compile-time. The individual areas within this vector are referenced by an entry such as the following:

```
03 EX-TABAC-F    PC S9(11) CMP OCCURS 5.
03 PX-TABAC-F    PC  9 (4) CMP.
```

where:

EX — is the prefix used to designate that this is an element in a vector.

TABAC — is the vector name from the Extension Specifications of the RPG program.

PX — is the prefix used for the subscript field for this table.

d. RPG Indicators are prefixed by IND-, e.g., indicator 01 becomes IND-01.

e. References to UDATE will cause a single access to TODAYS-DATE which is then stored in the field UDATE-F.

If PAGE is used in the RPG program, a field, PAGE-F, will be declared and logic necessary to increment it is developed.

f. TAG names used in the RPG Calculation Specifications will appear as paragraph names in the COBOL program with -F-TAG appended, e.g., ENDDET-F-TAG.

g. B 1700 COFIRS II allows up to 15 files, 512 fields, and 64 vectors in a program for translation, subject to limitations in the RPG Compiler.

h. The Program Identification inserted in columns 73-80 is taken from columns 75-80 of the RPG Control Card. If this field is left blank, the default is RPGOBJ. The new COBOL source file name is RPGCOB and is sequenced by 100.

i. The following defaults are used if the appropriate information is not provided in the RPG source program:

1) All files:

| Option | Default |
|---|---|
| Family Name | Internal file name from the File Description Specifications. |
| File ID | None. |

2) Disk files:

| Option | Default |
|---|---|
| Areas | 25. |
| Records per Area | 500. |
| Disk Pack ID | None (defaults to System Disk). |

All these options, as well as other B 1700 RPG options, may be included in the RPG source program to be translated.

j. System Requirements:

B 1700 Central Processor
Minimum of 32 KB Memory
Card Reader
Line Printer
Disk
MCP I or MCP II
RPG Compiler
COBOL Compiler
COFIRS Object Code
SDL Interpreter
COBOL Interpreter

# APPENDIX G

# TAGSORT WITH RPG

TAGSORT is a method of sorting records in an input file by creating a tagfile by which the input file is sorted. The tagfile is placed on disk and may be referenced by any RPG program which contains the original input file for which the tagfile was created.

The method for creating a tagfile is discussed in the B 1700 System Software Operational Guide, Form No. 1068731.

Basically, the tagfile 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 tagfile 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.

To access the tagfile in an RPG program, both the tagfile and the original input file must be declared on the Input Specifications. The access method will be indexed. FACTOR 1 must contain the tagfile name (if no fields within the tagfile are specified) or the field name containing the relative record number within the tagfile. FACTOR 2 contains the name of the original input file, and the OPERATION field contains the CHAIN construct.

In the example shown in table G-1, the input file DISKFI is used to build a tagfile called TAGFILE. Refer to the B 1700 System Software Operational Guide, Form No. 1068731.

Table G-1. Building a Tagfile

| Contents of DISKFI | | TAGSORT Program | Contents of TAGFILE |
|---|---|---|---|
| RECD NO | | | |
| 00001 | 456 2222 | FILE IN DISKFI (DISK (100) 180 1) | 00000002 |
| 00002 | 123 0000 | OUT TAGFILE (DISK (270) 4 45)<br>KEY (1 3 A A) | 00000005 |
| 00003 | 879 1111 | TAGSORT | 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 tagfile. Record 5 has the second smallest key (258), so 5 is placed in the second position of the tagfile, etc.

In the example shown in table G-2, an RPG program called TAG containing DISKFI accesses TAGFILE and produces the resultant output.

Table G-2. Accessing a Tagfile

Program called TAG

```
FTAGFILE IPE                     DISK
FDISKFI  IC                      DISK
FPRINTIT O                       PRINTER

ITAGFILE AA  01
I                                   P   1   40RECNO
IDISKFI  BB  02
I                                       1    30KEY
I                                       5    80DATA


C          RECNO    CHAINDISKFI


OPRINTIT D       01
O                                  10 'KEY'
O                      KEY   X     15
O                                  40 'DATA'
O                      DATA  X     46
```

Printer Output From TAG

```
KEY  123          DATA  0000
KEY  258          DATA  4444
KEY  327          DATA  7777
KEY  456          DATA  2222
KEY  551          DATA  6666
KEY  653          DATA  3333
KEY  785          DATA  5555
KEY  879          DATA  1111
KEY  965          DATA  8888
```

# 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.

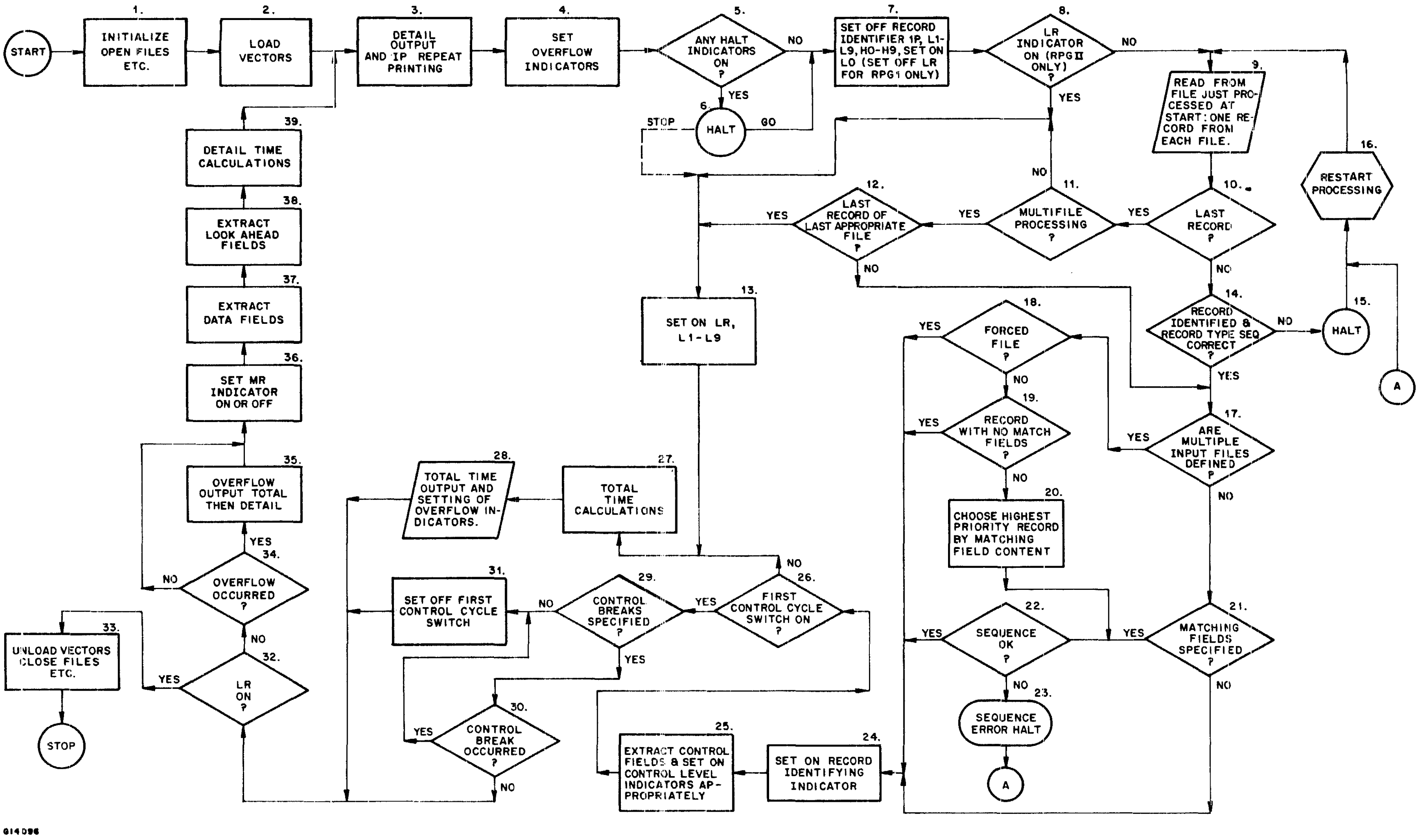
The specific steps taken in one program cycle are shown in figure H-1. The item numbers in the following description refer to the number in the figure. A program cycle begins with step 3 and continues through step 39.

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 1P REPEAT PRINTING
4.
SET OVERFLOW INDICATORS
5.
ANY HALT INDICATORS ON ?
NO
YES
6.
HALT
STOP
GO
7.
SET OFF RECORD IDENTIFIER 1P, L1-L9, HO-H9, SET ON LO (SET OFF LR FOR RPG1 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 APPROPRIATE FILE ?
YES
NO
13.
SET ON LR, L1-L9
14.
RECORD IDENTIFIED & RECORD TYPE SEQ CORRECT ?
NO
YES
15.
HALT
16.
RESTART PROCESSING
A
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
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.
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

614096

Figure H-1. RPG Program Cycle

3) All control field storage areas are initialized to hexadecimal zeroes.

4) Indicators LO and IP are set ON, all other indicators are set OFF. Under the RPG1 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 1P (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 H0 - 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. 1P, L1-L9 and H0-H9 are set OFF. L0 is set ON. Under the RPG1 dialect LR is also set OFF.

8. Under the RPG1 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

# INDEX (Cont)

INDEX (Cont)

INDEX (Cont)

INDEX (Cont)

cut along dotted line

# BURROUGHS CORPORATION
# DATA PROCESSING PUBLICATIONS
# REMARKS FORM

TITLE: BURROUGHS B 1700 SYSTEMS REPORT PROGRAM GENERATOR Reference Manual

FORM: 1057189
DATE: 2-75

CHECK TYPE OF SUGGESTION:

□ADDITION □DELETION □REVISION □ERROR

GENERAL COMMENTS AND/OR SUGGESTIONS FOR IMPROVEMENT OF PUBLICATION:

cut along dotted line

FROM: NAME ______________________ DATE __________
TITLE ______________________
COMPANY ______________________
ADDRESS ______________________
______________________

STAPLE

FOLD DOWN SECOND FOLD DOWN

Postage
Will Be Paid
by
Addressee

No
Postage Stamp
Necessary
If Mailed in the
United States

BUSINESS REPLY MAIL
First Class Permit No. 817, Detroit, Mich. 48232

Burroughs Corporation
Burroughs Place
Detroit, Michigan 48232

attn: Systems Documentation
Technical Information Organization, TIC-Central

FOLD UP FIRST FOLD UP

Wherever There's
Business There's

Burroughs