Burroughs

Language Manual

B 1000 Systems Report Program Generator (RPG)

(Relative to the Mark 11.0 System Software Release)

Priced Item
Printed in U.S.A.
March 1984

1152063

Burroughs

Language Manual

B 1000 Systems Report Program Generator (RPG)

(Relative to the Mark 11.0 System Software Release)

Priced Item
Printed in U.S.A.
March 1984

1152063

Burroughs cannot accept any financial or other responsibilities that may be the result of your use of this information or software material, including direct, indirect, special or consequential damages. There are no warranties extended or granted by this document or software material.

You should be very careful to ensure that the use of this software material and/or information complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Correspondence regarding this publication should be forwarded using the Remarks form at the back of the manual, or may be addressed directly to Corporate Documentation-West, Burroughs Corporation, 1300 John Reed Court, City of Industry, California 91745, U.S.A.

Burroughs

PUBLICATION CHANGE NOTICE

PCN No.: 1152063-001 Date: September 1984
Publication Title: B 1000 Report Program Generator (RPG) Language Manual (February 1984)
Other Affected Publications: None

Supersedes: N/A

Description

This PCN includes changes to the Mark 12.0 System Software Release documentation.

Replace Pages

iii
xv
xxi
xxv
13-7
13-9
D-33
1
5 thru 11

Delete Pages

D-35 thru D-37

LIST OF EFFECTIVE PAGES

Page	Issue	Page	Issue
Title	Original	12-46	Blank
ii	Original	12-47 thru 12-52	Original
iii thru iv	PCN-001	13-1 thru 13-6	Original
v thru xiv	Original	13-7 thru 13-9	PCN-001
xv thru xvi	PCN-001	13-10	Blank
xvii thru xx	Original	14-1 thru 14-5	Original
xxi thru xxii	PCN-001	14-6	Blank
xxiii thru xxiv	Original	15-1 thru 15-91	Original
xxv thru xxvi	PCN-001	15-92	Blank
xxvii thru xxix	Original	A-1 thru A-2	Original
xxx	Blank	B-1 thru B-2	Original
1-1 thru 1-4	Original	C-1 thru C-2	Original
2-1 thru 2-6	Original	D-1 thru D-32	Original
3-1 thru 3-4	Original	D-33 thru D-34	PCN-001
4-1 thru 4-22	Original	E-1 thru E-4	Original
5-1 thru 5-30	Original	F-1 thru F-3	Original
6-1 thru 6-2	Original	F-4	Blank
7-1 thru 7-13	Original	G-1	Original
7-14	Blank	G-2	Blank
8-1 thru 8-5	Original	H-1 thru H-3	Original
8-6	Blank	H-4	Blank
9-1 thru 9-7	Original	H-5 thru H-6	Original
9-8	Blank	1 thru 2	PCN-001
10-1 thru 10-37	Original	3 thru 4	Original
10-38	Blank	5 thru 12	PCN-001
11-1 thru 11-58	Original	13 thru 16	Original
12-1 thru 12-45	Original		

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)

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)

LIST OF ILLUSTRATIONS (Cont)

LIST OF ILLUSTRATIONS (Cont)

LIST OF TABLES

LIST OF TABLES (Cont)

INTRODUCTION

Burroughs RPG (Report Program Generator) is a machine-independent programming language suitable for implementation in a wide variety of data processing applications. B 1000 RPG embraces RPG as implemented on IBM 360/20 and RPG II as implemented on IBM S/3. In addition, Burroughs has taken advantage of the sophisticated operating system of the B 1000 to allow optional extensions to the above-mentioned implementations.

Burroughs B 1000 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 is denoted by "RPG I".

Burroughs B 1000 RPG offers the following advantages to the user:

1. Simple, generative syntax for ease of program implementation.
2. Accelerated programmer training and simplified retraining requirements.
3. Ease of conversion through standard implementation.
4. Ease of program modification.
5. Standardized documentation.

This manual describes Burroughs B 1000 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 edition contains changes throughout.

List of Applicable B 1000 Publications

The following is a list of publications related to the B 1000 RPG System which are referenced in this manual.

Publication Title	Form Number
B 1000 Systems System Software Operation Guide Volume 1	1151982
B 1000 Systems System Software Operation Guide Volume 2	1152097
B 1000 Systems COBOL Reference Manual	1057197
B 1000 Systems Network Definition Language (NDL) Language Manual	1073715

Publication Title	Form Number
B 1000 Systems Data Management System II (DMSII) Reference Manual	1127222
B 1000 Systems Sort Language Manual	1152071
B 1000 System Burroughs Network Architecture (BNA) Installation and Operation Manual	1151974

HOW TO USE THIS MANUAL

The information in this manual is presented in a logical hierarchy organized from general topics to specific subtopics. The headings in this manual provide a guide to this organization hierarchy of specifying the relative level of generality (or specificity) of the material associated with the heading.

Because the headings appear in the Table of Contents, a general outline of the major topics in the manual, arranged according to their levels in the logical hierarchy, may be found in the Table of Contents.

Major topics in this manual are discussed in separate sections. Section titles appear in all capital letters at the top of the first page of a section. A section title appears as follows:

SECTION 7
SECTION TITLE

Major topics within each section of the manual are indicated by first-level headings. First-level headings appear completely in capital letters indented one space on the page. Usually a first-level heading appears at the top of a new page in the section. A first-level heading appears as follows:

FIRST-LEVEL HEADING

Subtopics of the major topics within a section are indicated by second-level headings. Second-level headings appear with initial capital letters. A second-level heading appears as follows:

Second-Level Heading

Subtopics of the lowest level in the manual are indicated by third-level headings. Third-level headings appear in initial capital letters. Third-level headings appear as follows:

Third-Level Heading

PRESENTATION OF EXAMPLE PROGRAMS

Throughout this manual examples of the RPG language are presented. Contained in each presentation is a heading for each specification shown and a column-ruler line at the bottom to help identify the columns in which entries are specified. The left-most number on each specification line identifies the RPG source statement number. Figure i-1 shows one example of an RPG source program.

```
03 IINFILE  NS  01
04 I                                              1  90 RECORD
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


           Output-Format Specifications

01 OOUTFILE D         01
02 O                            RECORD    90
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure i-1. Example of RPG Language Program

SECTION 1
RPG OPERATION

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

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 1000 compiler.

Data Base Specifications

This specification provides information about the DMSII data base that is to be accessed by the RPG program.

File Description Specifications

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

Extension Specifications

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

Line Counter Specifications

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

Telecommunications Card Specifications

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

Input Specifications

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

Calculation Specifications

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

Output-Format Specifications

These specifications are used to specify the type and arrangement of data that are 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 can appear anywhere in a RPG Source Program, unless otherwise specified (see section 11).

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

1. Control Card Specifications, form number 1107588.
2. File Description Specifications, form number 1107596.
3. Extension Specifications/Line Counter Specifications, form number 1107604.
4. Data Communications Specifications, form number 1107620.
5. Input Specifications, form number 1107562.
6. Calculation Specifications, form number 1107612.
7. Output-Format Specifications, form number 1107570.
8. Dollar Card Specifications, form number 1055845.

RPG SOURCE PROGRAM DECK

The coded information contained on the specification forms is recorded in a disk file which constitutes the source program. The arrangement of these records in the source deck is illustrated in figure 1-1.

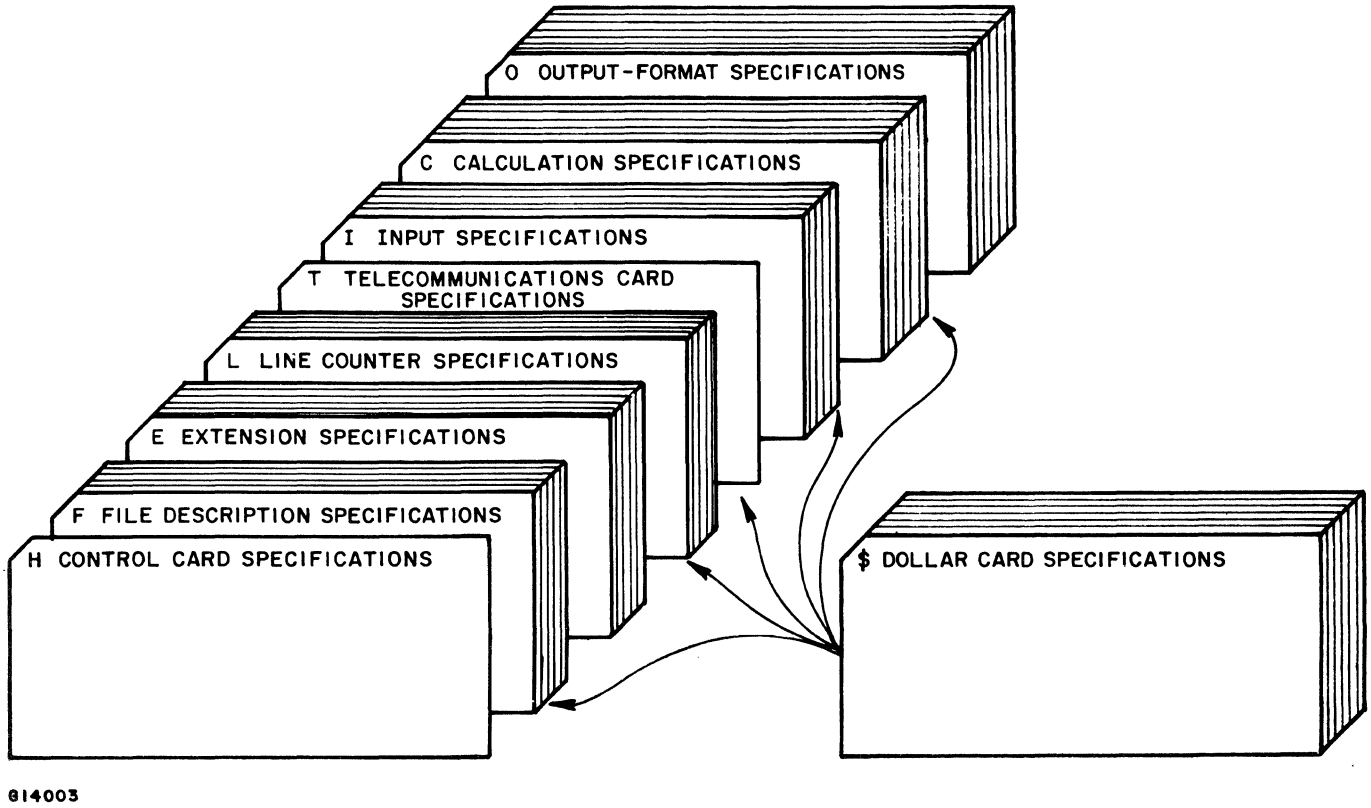


Figure 1-1. RPG Source Program Deck

If source corrections become necessary, appropriate changes can be made and the program recompiled. Thus, the source program file 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 I or RPG II.

The following is a list of exceptions found in B 1000 RPG:

1. The Control Card requires simplified recoding before compiling programs from systems other than B 1000.
2. B 1000 RPG allows 511 characters for alpha fields and 23 for numeric.
3. IBM's RPG uses standard right signed numeric fields, whereas Burroughs B 1000 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 1000 RPG allows the right sign option for system compatibility. (Refer to $RSIGN option in section 11.)
4. Numeric fields must be edited if the C zone punch is not wanted on output. Otherwise, a positive unedited numeric field contains 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 assigns a F to the left-most character, changing the output to appear as 1013.
5. The number 96-column devices do not allow packed decimal format.

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

1) Variable length records.
2) Sterling data format.
3) Alternate collating sequence.
4) Inquiry programs.
5) Printer keyboard output files.
6) External assembler subroutines.
7) Blank after literals.
8) Redefined field lengths.
9) Factor 1 and Factor 2 both literals.
10) Card print feature of IBM's 360/20.
11) File translation.

SECTION 2
RPG LANGUAGE ELEMENTS

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

CHARACTER SET

The minimum RPG character set consists of the following characters:

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

The following RPG characters are optional and can 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 38 characters: 0 through 9, A through Z, hyphen (-), and underscore (_).

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)
&	ampersand
–	minus sign

DEFINITION OF NAMES

A name must be left-justified in the field and must begin with an alphabetic character. It is ended by the space or by the end of the field, whichever comes first. All characters in a name, except the first, can be any combination of alphabetic, numeric, hyphen (-), and underscore (_) characters. Blank characters cannot 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. File names cannot exceed eight characters and the first seven characters must be unique among file names, for example, DISKFILE and DISKFIL are considered by the compiler to be equal. The compiler truncates the eighth character.

Vector Names

A vector name is used to identify a data item which is actually a table or an array. The table is 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 Calculations Specifications to which a GOTO operation branches, 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 5 below). The rules for the formation of a numeric literal are:

1. There must be at least one digit in a numeric literal.
2. 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.
3. Only one sign character and/or one decimal point can be contained in a numeric literal.
4. The maximum total length of a numeric literal is 10 characters, including sign and decimal point.
5. Decimal commas must be used in place of decimal points if the Inverted Print Options I or J are used (Control Card, Column 21).
6. 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 can 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:

1. There must be at least one character in an alphanumeric literal.
2. The maximum length of an alphanumeric literal used in the Calculation Specifications is eight characters, and in the Output-Format Specifications is 24 characters.
3. Alphanumeric literals cannot be used for arithmetic operations.

The following are examples of alphanumeric literals:

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

DEFINITION OF RESERVED WORDS

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

Special Words

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

The following special words are reserved for use as variables:

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

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

Operation Codes

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

Common Field Definitions

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

1-2 PAGE

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

3-5 LINE

The LINE entry defines a sequential number for the individual source records on each specification form. Normally, only numeric characters in the range 000-999 are specified; however, any EBCDIC characters (including blank characters) are valid entries.

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

```
02 FCARDIN  IPEAF  80  80            READER
03 FDISKOUT O    1800  90            DISK
04 FMASTER  UC    540 180  4 !       DISK
05 F
06 F
07 F
-------------------------------------------------------------------
035FPRINT   O     132 132     OF    LPRINTER
025FTAPEIN  ISEA  180 180            TAPE
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 2-1. Insertion of Coding Lines

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

6 FORM TYPE

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

Valid entries for this field are:

Entry	Definition
H	Header Card (Control Card Specification).
D	Data Base Specification.
F	File Description Specification.
F	Index-Sequential Key Field Specification.
A	Attribute 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 line allows the entire line to the right of column 7 (which contains an asterisk [*] character) 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 can 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.

```
02 F* THIS IS AN EXAMPLE OF HOW COMMENT LINES
03 F* ARE CODED. ALL LINES WITH AN ASTERISK (*)
04 F* IN COLUMN 7 ARE TREATED AS DOCUMENTARY,
05 F* AND ARE IGNORED BY THE RPG COMPILER.
06 F*
07 F*****************************************
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 2-2. Comment Line Coding

The dollar sign ($) character in column 7 designates the line to be a Dollar Card Specification which allows the RPG Compiler to accommodate machine and system dependent features. A dollar sign ($) character in column 7 overrides column 6. Dollar Card Specifications can be specified at any point in the RPG Source Program Deck. Section 13 describes the use of Dollar Sign 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 appears on the source program listing. Thus, it can be used for documentation to identify different portions of the program, if desired.

SECTION 3
CONTROL CARD SPECIFICATIONS

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

FIELD DEFINITIONS

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

1-2 PAGE

Refer to section 2 for a complete description.

3-5 LINE

Refer to section 2 for a complete description.

6 FORM TYPE

This field must be coded with the letter H.

15 DEBUG

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

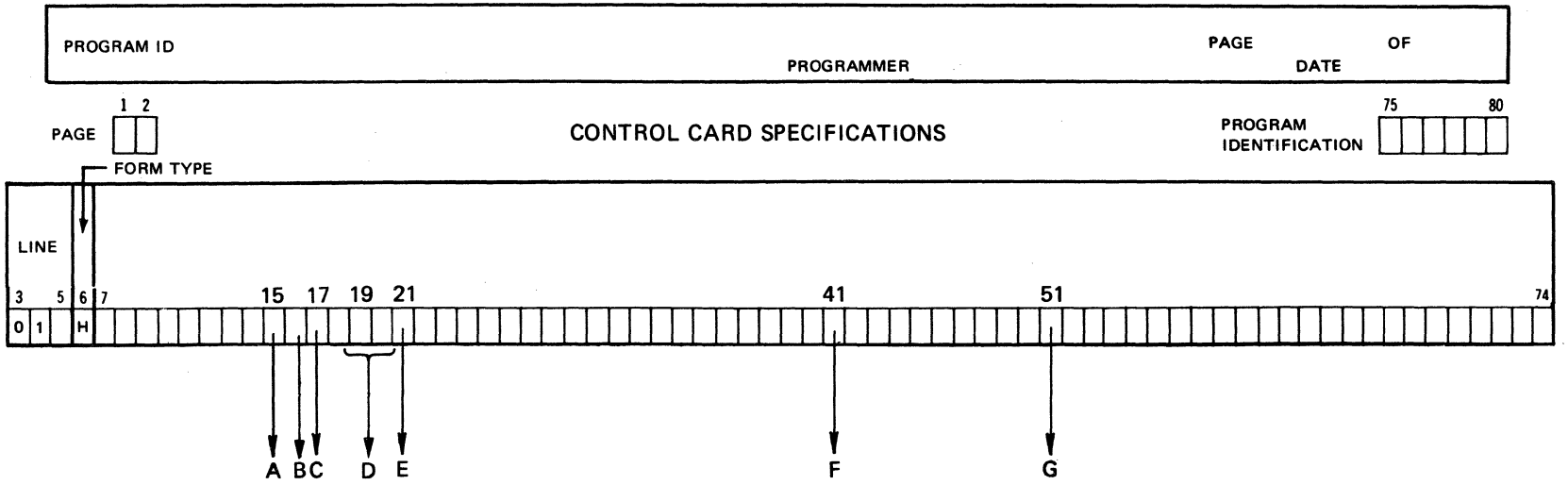
1. Column 15 of the Control Card Specifications must be coded with a 1.
2. The operation code for DEBUG must appear in the Calculation Specifications.

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

16 B-INDEXED FILES

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

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



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

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

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

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

E. 21 Specifies the punctuation (INVERTED PRINT) to be used for numeric literals. Entries: Blank or 1.

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

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

G14006

Figure 3-1. Control Card Specifications Summary Sheet

17 Sign Position

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

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

The sign position specified by this field can be overridden by the RSIGN Dollar Card Specifications (refer to section 13) if different sign positions are required within the program.

19-20 SORT MEMORY SIZE

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

The SORT memory size specification in columns 19-20 of the Header Specification, if used with indexed-sequential files, is given a warning message and ignored. It can be used for indexed (tag file) files.

21 INVERTED PRINT

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

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

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

Table 3-1. Inverted Print Specifications

Column 21 Entry	Numeric Literal with a Comma or a Period as a Decimal Point	Edit Codes with a Comma or a Period as a Decimal Point	Zero Suppress to the Left or Right of the Decimal Point	UPDATE with a Slash or a Period
Blank	5678.90	7,654.32	.60	MM/DD/YY
D	5678.90	7,654.32	.60	DD/MM/YY
I	5678,90	7.654,32	,60	DD.MM.YY
J	5678,90	7.654,32	0,60	DD.MM.YY

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 causes 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 can 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 I 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 I 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 I dialect is specified and vice versa.

75-80 PROGRAM IDENTIFICATION

Refer to section 2 for a complete description.

SECTION 4
FILE DESCRIPTION SPECIFICATIONS

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

The B 1000 system provides the facility for the object program to use the record and block lengths from the header record of existing disk files. This function can be inhibited by use of the dollar option REFORM.

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

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

FIELD DEFINITIONS

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

1-2 PAGE

Refer to section 2 for a complete description.

3-5 LINE

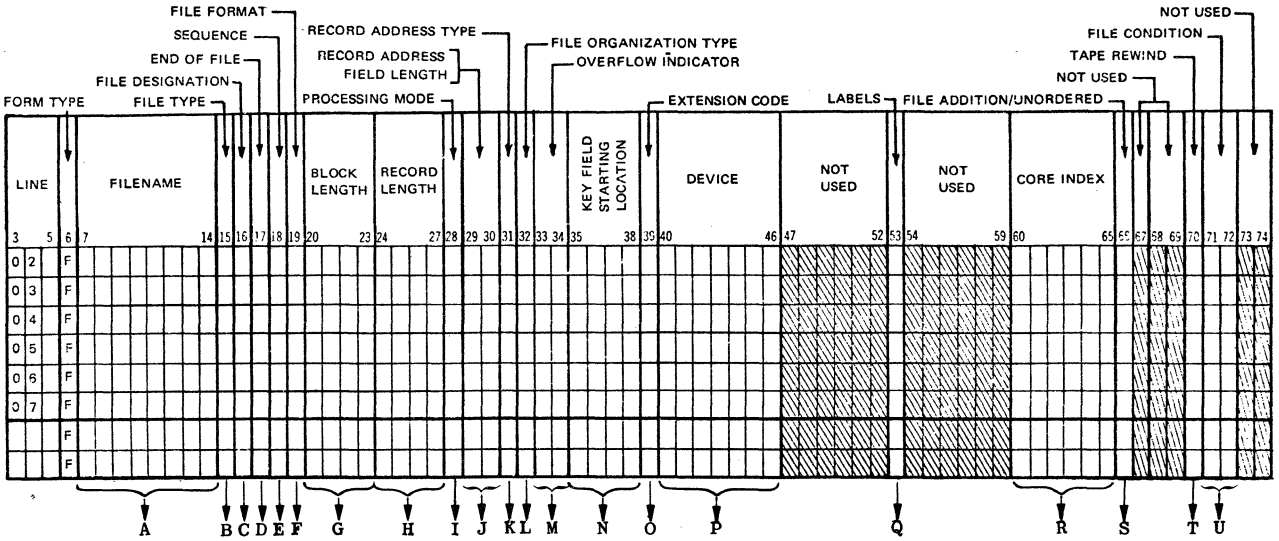
Refer to section 2 for a complete description.

6 FORM TYPE

This field must contain the letter F.

7-14 FILENAME

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



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

B. 15 Specifies how each file is to be used. Entries: 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 to be checked. Entries: Blank or A, or D.

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

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

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

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

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

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

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

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

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

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

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

Q. 53 Entries only meaningful to the B 1000. Entries: U (unlabelled file), F (special forms), or B (printer backup).

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

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

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

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

G14007

Figure 4-1. File Description Specifications Summary Sheet.

15 FILE TYPE

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

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

I Input Files

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

O Output Files

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

U Update Files

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

C Combined Files

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

1. Cards that are read by the program and are subsequently punched and/or printed as output. The punching is into the same cards that have been read, and output can occur only once per cycle.
2. Remote files that are written to (SEND) and read from (RECV) during calculations.

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

D Display Files

Display files are used to write a field or a record directly on the Operator Display Terminal (ODT). The DSPLY operation code must be used in the Calculation Specifications in order to perform the write operation. Display files are described only on the File Description Specifications form. The size of the display file is limited to 50 bytes.

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 and display files, excluding chained output files. Acceptable entries for this field are:

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

P Primary File

A primary file is the principal file from which the program reads input records, and can 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 syntax error is emitted.

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

S Secondary Files

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

C Chained File

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

Chained indexed files can be input or update files only.

T Table File

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

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

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

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

D Demand File

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

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

Demand files are processed as follows:

1. During input mode, the RPG program reads a record in calculations and does not depend on the RPG program cycle to make a record available.
2. During output mode, the RPG program allows exception records to be written during calculations. The desired output operation is performed, and program execution continues with the next calculation statement. The demand file does not have to wait for the RPG cycle to write the exception record.

R Record Address File

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

A record address file can be either:

1. A limits file that contains records consisting of lower and upper bound record keys that are used to process indexed files sequentially within limits, or
2. An addrout file that consists of 4-byte, relative-record numbers used to randomly process a disk file.

17 END OF FILE

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

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

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

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

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

18 SEQUENCE

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

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

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 can make one of the following entries:

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

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

19 FILE FORMAT

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

Entry	Definition
Blank or F	Fixed record length.

20-27 BLOCK AND RECORD LENGTH

These fields are used to specify the block and record sizes for the file. The minimum and maximum record and block length allowed depend upon the device to which the file is assigned. This information is detailed for the B 1000 in table 4-1.

For disk files with the letter U (update) in the FILE TYPE field (column 15), the RECORD LENGTH field (columns 24-27) should match the record length in the disk file; otherwise undesirable results can occur when the records are read and written. The RPG compiler and operating system do not verify that the record lengths of the disk file and the file in the RPG program are equal.

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

Device	Block Length		Record Length	
	Minimum	Maximum	Minimum	Maximum
CARD (80 col)	Same as record length	Same as record length	1	80
CARD (96 col)	Same as record length	Same as record length	1	96
PRINTER	Same as record length	Same as record length	1	132
DISK				
Data files	1	9999	1	8192
Addrout files	4	180	4	4
TAPE	8	9999	1	8192
CONSOLE	Same as record length	Same as record length	1	50

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

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

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

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

Table 4-3. Default Block and Record Lengths

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

The value specified in the BLOCK LENGTH file must be an integral multiple of the value specified in the RECORD LENTH field. For unblocked files (note that card and printer files are unblocked), the values specified in the BLOCK LENGTH and RECORD LENGTH must be equal. For card and printer files, the value specified in the RECORD LENGTH field can be less than the maximum record length for the device. That is, printer files can be specified as 96, 120, or 132 characters in length.

For printer files, if a value is specified for the BLOCK LENGTH field, that value must be the same as in the RECORD LENGTH field.

28 PROCESSING MODE

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

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

29-30 RECORD ADDRESS FIELD LENGTH

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

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

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

1. Index-Sequential files – 29 characters.
2. Indexed files – 29 characters.
3. B-Indexed files – 99 characters.

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

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

31 RECORD ADDRESS TYPE

The RECORD ADDRESS TYPE field provides additional information about the format of the keys and/or the processing mode of indexed disk files and record address files. The acceptable entries for this field are:

Entry	Definition
Blank	Not an index file, a Limits file, or an Addrout file. Also, the file is not processed by an addrout file. Not valid for files with the $ IXSEQ (default) option specified.
A	Index or Limits file with alphanumeric keys. Valid for files with the $ IXSEQ (default) option specified.
I	Addrout file or processed by an addrout file.
K	Index or Limits file with alphanumeric keys. Not valid for files with $ IXSEQ (default) option specified.

Entry	Definition
N	Index file with numeric keys in alphanumeric format. Not valid for files with the $ IXSEQ (default) option specified.

NOTE

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

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

32 FILE ORGANIZATION TYPE

The FILE ORGANIZATION TYPE field identifies file organization or indicates whether multiple buffers are required. Valid entries for this field are:

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

NOTE

B 1000 RPG organizational techniques are such that file accessing and file organization are independent of each other. This is not true for other RPG implementations.

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

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

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

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

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

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

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 (that is, the printing of special lines when the overflow file is fetched) is desired for that file. Acceptable entries for this field are:

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

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

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

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

35-38 KEY FIELD STARTING LOCATION

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

39 EXTENSION CODE

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

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

40-46 DEVICE

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

Entry	Definition
READER	80-column card reader
MFCU1	96-column MFCU – primary hopper.
MFCU2	96-column MFCU – secondary hopper.
PUNCH	80-column card punch.
PRINTER or PRINTER2	Line printer. (A maximum of 5 printer files can be specified).
TAPE	Magnetic tape.
DISK	Disk file.
CONSOLE*	Operator Display Terminal.
BSCA	Telecommunications file.
REMOTE	Telecommunications file.
DATACOM	Telecommunications file.

*Only one CONSOLE file can be specified and must be specified as a display file (see DSPLY in section 9).

Any card device name (that is READER, PUNCH, MFCU1, etc.) is considered only to denote a card file. The actual type (input, output, combined) is determined by the entry in column 15 of the File Specifications. Table 4-4 lists the devices available for each of the file types.

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

Table 4-4. Device Assignment for Files.

File	Type	Devices
Primary or Secondary Input Files	Card	MFCU1 or MFCU2 READER
	Disk	DISK
	Tape	TAPE
Demand Files	Card	MFCU1 or MFCU2 READER
	Data Communications	REMOTE
	Disk	DISK
	Tape	TAPE

Table 4-4. Device Assignment for Files (Cont)

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

The B 1000 system also accepts the following additional device names in the DEVICE (columns 40-46) field.

READ01	PUNCH20
MFCM1	PUNCH42
MFCM2	PRINTUF
READ20	PRINTLF
READ40	DISK11
READ42	DISK11F
CRP	DISK45
CRP20	SPO
DATA96	

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

Data communications files can be input, output, or combined files, and are used to reference Network Definition Language (NDL) files associated with the network controller. The filename specified in the FILENAME field (columns 7-14) of the File Description Card entry must be the same as the filename of the NDL file being referenced, or the names should be file-equated with a File statement. For information concerning the Network Definition Language, refer to the B 1000 Systems Network Definition Language Reference Manual.

REMOTE Specifications

When the device REMOTE is specified in the DEVICE (columns 40-45) field, the remote devices are accessed on a demand basis. The fields of the file description specifications shown in table 4-5 must be coded with the applicable entry when REMOTE is specified. The remaining fields must be coded as they would be for any demand file.

Table 4-5. File Description Specifications – REMOTE

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

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

1. Exception conditions.
2. Incomplete I/O operations.
3. End-of-file conditions.

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

DATACOM OR BSCA

When DATACOM or BSCA is specified starting in column 40, the fields of the file description specification shown in table 4-6 must be coded as shown.

Table 4-6. File Description Specifications – DATACOM or BSCA

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

53 FILE OPEN

The FILE OPEN field specifies the open option for the file. The following are valid entries for this field.

Entry	Definition
blank	Defaults to the letter U.
B	The letter B applies to printer and card files only. For printer files, the printer file is not written directly to the line printer, instead it is written to a printer backup file. The printer backup file can be printed at any time after the program closes the printer file, or after the program goes to end of job. For card files, the card file is not written directly to the card punch, but is written to a card punch backup file instead. The card punch backup file can be punched at any time after the program closes the card file or after the program goes to end of job.
F	The letter F applies to printer files and causes the RPG program to be temporarily suspended while waiting for special forms to be loaded on the line printer by the operator. A special forms message is displayed on the Operator Display Terminal (ODT).
L	The letter L applies to input, update, or output-with-add disk files and specifies that this file is unavailable for other programs while the disk file is open by this program.
R	The letter R applies to input, update, or output-with-add disk files. It specifies that this file is only available to other programs that open this file input while the disk file is open by this program.
U	The letter U specifies that this file is expected or provided as an unlabeled file. If the file is a disk file, access to this file by other programs is not restricted while the file is open by this program.

60-65 CORE INDEX

The core index is a rough table in memory which contains entries representing the key fields of the data file on disk. The use of the core index can significantly reduce the time needed to process an indexed file because it allows a more direct access to the specific record required. The program must search only a portion of the file instead of all entries in the file preceding the required record. The core index field must be blank when files other than chained index files or index files processed by the SETLL operation code (the letter L in column 28) in the File Description Specifications are specified.

This field is used only by files accessed using the CHAIN operator (that is index files) to specify the number of bytes of memory to be set aside for indexes. The entry must be right-justified, and leading zeros can be omitted. The maximum size that can be entered is 999999.

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 (FIB) to determine the total size of the file. The program divides the file into even partitions for the allowable number of keys in memory, and reads every nth record filling the core table in memory with just the data key fields. If no entry is made, a minimum (default) number of 10 of these keys (20 keys if the keys are in packed or numeric form) is automatically specified.

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

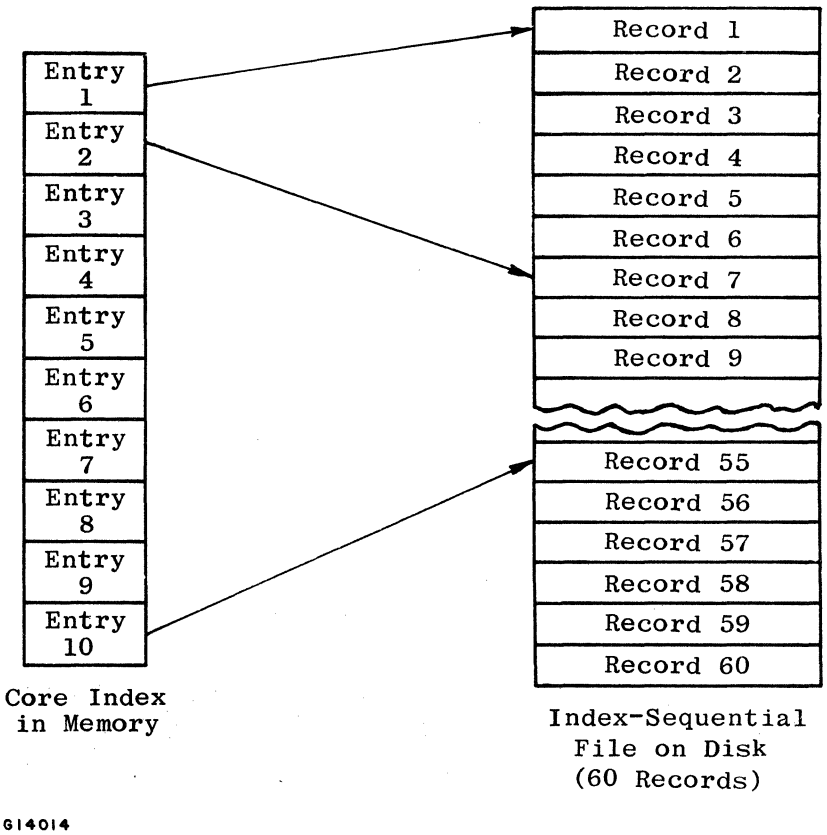


Figure 4-2. Core Index Selection of Data Keys

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

If a value is specified in the CORE INDEX field (columns 60-65) on the File Description Specifications for an index-sequential file, a warning message is generated and the value is not used.

66 FILE ADDITION/DELETION/UNORDERED

This field applies only to sequential, indexed ($ TAG), and index-sequential ($ IXSEQ) files and indicates:

1. New records are to be added to an existing file.
2. Unordered records are to be loaded into an output file.
3. Records are to be deleted from an index-sequential file.

When an entry is made in the FILE ADDITION/DELETION/UNORDERED field (column 66) for an indexed or index-sequential file, the indexed or index-sequential file is to be ordered in ascending sequence according to the key specified in the RECORD ADDRESS FIELD LENGTH field (columns 29-31) and the KEY FIELD STARTING LOCATION field (columns 35-38) of the File Description Specifications.

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

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

The FILE ADDITION/DELETION/UNORDERED field must be left blank when indexed files that are processed by record address files are specified.

The file-handling method for index-sequential files is different than for indexed ($ TAG) file processing when adding records to the file. For indexed file processing, a record that is added to a sequentially processed file (primary, secondary, or demand) cannot be read during the same program execution. In index-sequential file handling, the added record can be read if the added record has a key value greater than the current record.

Valid entries for the FILE ADDITIONS/DELETION/UNORDERED field are:

Entry	Definition
Blank	All files other than sequential output files and indexed ($ TAG) files that are processed by record address files. If the file is an indexed ($ TAG) output file, the records must be loaded in ascending key sequence.
A	Records are to be added to an existing sequential output file or to any index file.
B	Records can be added to and/or deleted from an existing index-sequential ($ IXSEQ) file. Does not apply to index disk ($ TAG) files.
D	Records are to be deleted from an existing index-sequential ($ IXSEQ) file. Does not apply to indexed ($ TAG) files.
U	Records are to be loaded for an indexed ($ TAG) file or index-sequential ($ IXSEQ) file in unordered sequence creating a new disk file.

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

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

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

The coding options for columns 15 and 66 are shown in table 4-7.

Table 4-7. Columns 15 and 66 Coding Options

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

70 TAPE REWIND

The TAPE REWIND 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 for this field are:

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

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 operating system (MCPII).

Card Output

All options are ignored. The output areas are released, the trailer label is punched, and the unit is returned to the operating system (MCPII).

Tape Input

The effects of the various options for tape input are:

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

Tape Output

The effects of the various options for tape output are:

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

Printer Output

All options are ignored. A page is ejected, a trailer label is written, and the printer is returned to the operating system (MCPII).

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 Disk Directory. A new file is one created by the program, and does not appear in the Disk Directory. A new file can only be referenced by the program which creates it.

Entry	Effect
Blank	Same as specifying the letter U.
N	Use of this option is not permitted with disk files.
P	An old file is removed from the disk and deleted from the Disk Directory, or a new file is removed from disk.
R	An old file is removed from the disk and deleted from the Disk Directory. The new file is entered in the Disk Directory and made available.

Entry	Effect
U	For an old file, the file remains in the Directory and is made available. A new file is entered in the Disk Directory (thereby making it an old file) and made available.

For index-sequential files ($ IXSEQ), if the letter P is specified in this column and the index-sequential file is redefined using the $ FILEID and/or $ FAMILYID compiler options, the file is not purged and the program is hung. The following message is then displayed on the Operator Display Terminal (ODT):

FILE NOT REMOVED -- IN USE

For indexed files ($ TAG), if the letter P is specified and the data file contains zero records, the tag file is purged but the data file is not. If the data file contains one or more records, both the tag and data files are purged. To ensure that the data file is purged, specify the $ OPEN option for the File Description Specifications of the indexed file. This causes the tag and the data files to be opened at beginning of job (BOJ); otherwise, only the tag file is opened, and if there are no records in the tag file, the data file is not opened.

71-72 FILE CONDITION

The FILE CONDITION 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 can be read or written into the file. Valid entries for this field are:

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

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

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

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

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

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

EXECUTE <program name> SWITCH <switch number> <switch value>

The following EXECUTE statement turns ON external indicator U1:

EXECUTE RPG/TEST SWITCH 1 1

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

MODIFY <program name> SWITCH <switch number> <switch value>

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

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

75-80 PROGRAM IDENTIFICATION

Refer to section 2 for a complete description.

SECTION 5
FILE ORGANIZATION AND PROCESSING

The file organization and processing of RPG programs is described in this section. Index file organization type include: index-sequential ($ IXSEQ), indexed ($ TAG), and B-indexed files. Included in file processing are the following file processing types: consecutive, sequential by key, sequential within limits, use of limits file, limits file requirements, and file restrictions.

FILE ORGANIZATION

The following types of file organization can be used with B 1000 RPG:

1. Sequential
2. Indexed
3. Direct

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

Sequential Files

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

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

Figure 5-1 shows a card file and the order in which the information from the cards is stored in a sequential disk file. The contents of cards A, B, C, D, E, F, G, and H are stored in disk segments 1 through 8 as entries A, B, C, D, E, F, G, and H respectively.

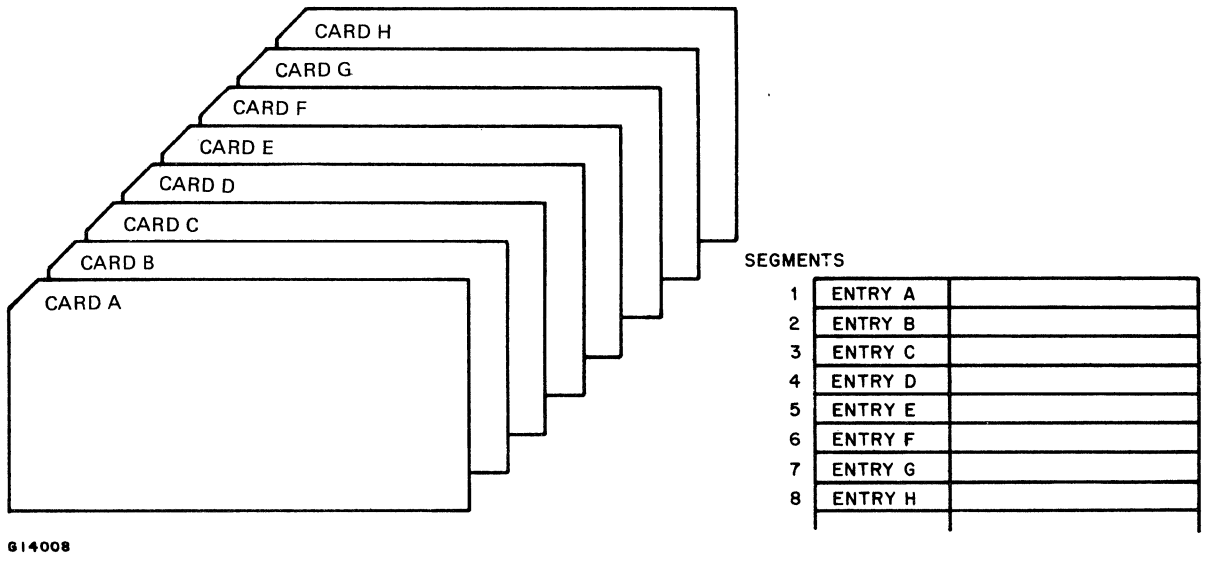


Figure 5-1. Sequentially Organized Card and Disk Files

Indexed Files

The following three types of indexed files can be used with B 1000 RPG:

1. Index-Sequential ($ IXSEQ)
2. Indexed ($ TAG)
3. B-Indexed (Letter B in column 16 of the Control Card Specifications)

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

A detailed description of each type of indexed file follows:

Index-Sequential ($ IXSEQ) Files

An index-sequential file is a disk file that is comprised of a data file, multiple key (access) files, and one global file. The data file contains the actual user information. The key file(s) is a sequential file and consists of key and address pairs, where the address points into the data file. The global file contains system information that associates the data and key files. The records in the data file are in random sequence order and are accessed by use of the key file. The following shows the naming convention used by the operating system (MCPII) to handle the three file types:

File Type	File Name
Global File	<filename>
Data File	00<filename>
Key File 1	01<filename>
Key File 2	02<filename>
Key File 3	03<filename>
.	.
.	.
.	.
Key File 99	99<filename>

Index-Sequential File Specifications

An index-sequential file can be specified by entering the IXSEQ (the default for index files) compiler option immediately prior to the File Description Specifications for a file. The File Description Specifications has space for declaring one key access. Additional key accesses can be specified on special file description specifications called Index-Sequential Key Field Specifications that immediately follow the File Description Specification.

Refer to figure 5-2 for an example of the Index-Sequential Key Field Specifications. Each additional key access requires one line on the form. Each of the Index-Sequential Key Field Specifications contain columns for describing the key length, data type, processing mode, starting location, name, access key, and access mode. These are documented in the following paragraphs.

Field Definitions

Figure 5-2 can be used in conjunction with the following field definitions for the Index-Sequential Key Field Specifications.

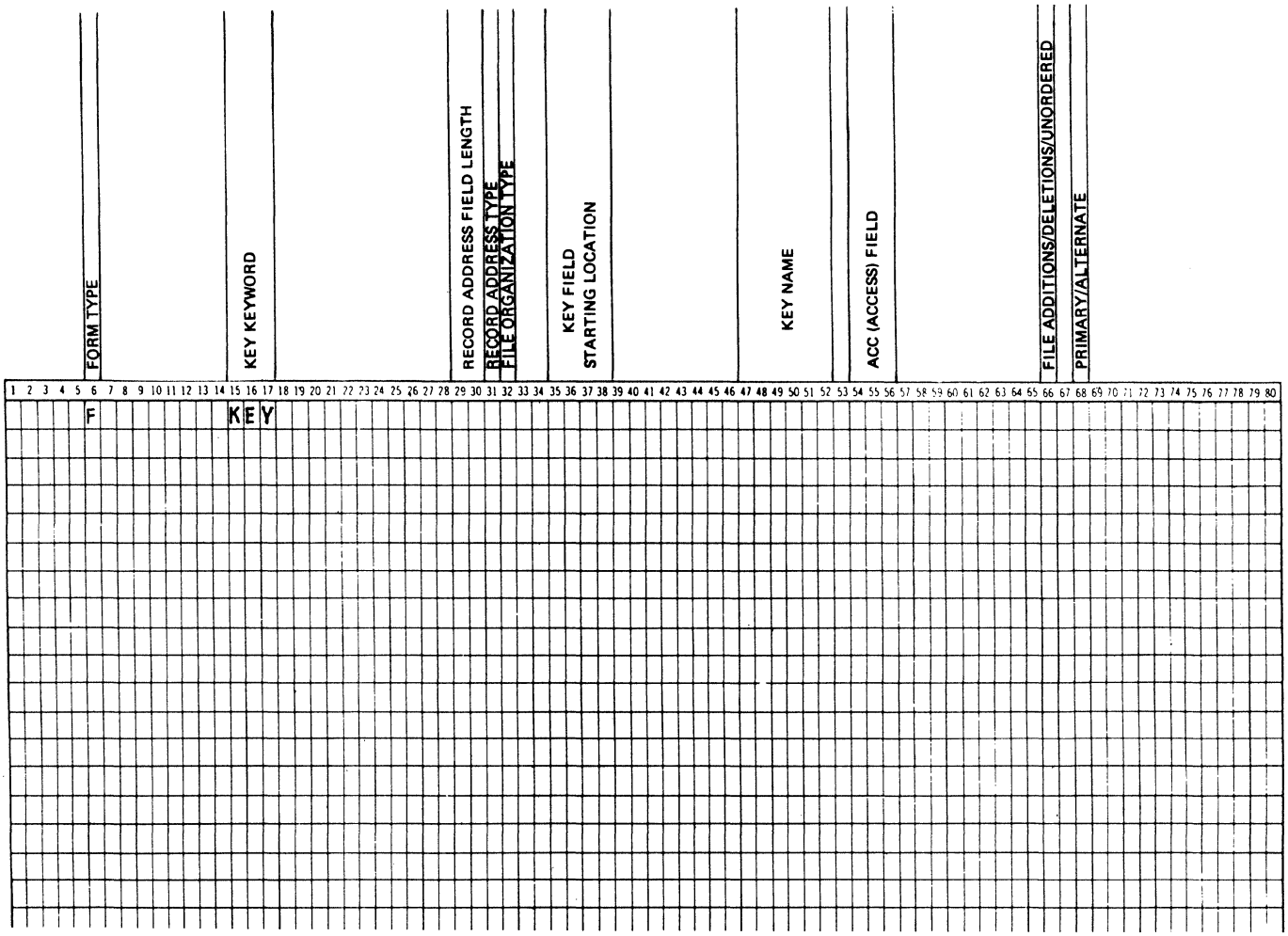


Figure 5-2. Index-Sequential Key Field Specifications

KEY KEYWORD Field

The KEY KEYWORD field (columns 15-17) of the Index-Sequential Key Field Specifications must contain the keyword KEY for additional key specifications that follow the File Description Specifications of the index-sequential file.

RECORD ADDRESS FIELD LENGTH Field

The key length is specified in the RECORD ADDRESS FIELD LENGTH field (columns 29-30) and can range from 1 to 29.

RECORD ADDRESS TYPE Field

The key data type is specified in the RECORD ADDRESS TYPE field (column 31) and can be the letter A or blank (defaults to the letter A).

FILE ORGANIZATION TYPE Field

The file organization is specified in the FILE ORGANIZATION TYPE field (column 32) and must be the letter I. This field is only valid on the File Description Specifications for the file.

KEY FIELD STARTING LOCATION Field

The key starting location is specified in the KEY FIELD STARTING LOCATION field (columns 35-38).

KEY NAME Field

The key name is specified in the KEY NAME field (columns 47-52) and identifies the name of the key. If multiple key specifications are described, this field can be left blank for only one of these key specifications.

ACC (Access) Field

The ACC field (columns 54-56) specifies that this key is the access key for various input and delete operations. The keysymbol ACC is specified and only one key specification can be the access key. If the ACC field is left blank, the key on the File Description Specifications is assumed to be the access key. Specifying ACC causes this key to be used whenever a CHAIN or DELET operation code does not include a key name in the RESULT FIELD field.

FILE ADDITIONS/DELETION/UNORDERED Field

The valid entries for the FILE ADDITIONS/DELETION/UNORDERED field (column 66) are as follows:

Entry	Definition
Blank	All files other than sequential output files and indexed ($ TAG) files that are processed by record address files. If the file is an indexed ($ TAG) output file, the records must be loaded in ascending key sequence.
A	Records are to be added to an existing sequential output file or to any index file.
B	Records can be added to and deleted from an existing index-sequential ($ IXSEQ) file. Does not apply to indexed ($ TAG) files.
D	Records are to be deleted from an existing index-sequential ($ IXSEQ) file. Does not apply to indexed ($ TAG) files.
U	Records are to be loaded for an index file in unordered sequence (creating a new index file).

The following describes how to add and delete records from index-sequential ($ IXSEQ) files.

Adding Records

The letter A (or B for index-sequential files) must be entered in column 66 of the Index-Sequential Key Field Specifications if there is an ADD entry in columns 16-18 of the Output-Format Specifications record description of the indexed disk file. If the letter A (or B for index-sequential files) is entered in this field for any file other than an update file, all output records must have ADD specified in columns 16-18 of the Output-Format Specifications. However, the Output-Format Specifications are optional. Records can only be added to an existing disk file.

Deleting Records

The letter D or B must be entered in column 66 if there is a DEL entry in columns 16-18 of the Output-Format Specifications record description of the index-sequential ($ IXSEQ) file. The record deleted is the current record of the key with ACC in columns 54-56 of the key specifications. No key can be specified for this type of delete operation.

Records can also be deleted from index-sequential ($ IXSEQ) files with the DELET operation code. Refer to the DELET operation code in section 9 for additional information.

PRIMARY/ALTERNATE KEY Field

The PRIMARY/ALTERNATE KEY field (column 68) specifies the primary key, alternate keys, and whether duplicate keys are allowed for a record in the index-sequential ($ IXSEQ) file. There can be only one primary key. All other keys must have the letter D or N specified in column 68. The primary key does not have to be the first key specified.

Valid entries for this field are:

Entry	Description
Blank	A blank entry specifies that the key is the primary key.
D	The letter D specifies that the key is an alternate key and duplicate keys are allowed.
N	The letter N specifies that the key is an alternate key and duplicate keys are not allowed.

Accessing CHAIN Files Sequentially

Accessing CHAIN files sequentially is accomplished by using the READ operation code. The READ operation code also allows the program to read records with duplicate keys in an index-sequential ($ IXSEQ) file. This is accomplished by first specifying the CHAIN operation code using the access key desired. Once the record is found, that is, the indicator specified in columns 54-55 is set off, duplicate records can be read by specifying the READ operation code. If the last CHAIN operation results in a not found condition, that is, the indicator specified in columns 54-55 is set on, subsequent READ operations do not return a record since no current record exists for the access key used. The same is true for updating. The last record that has been read, for example, access key number 2, is the record eligible for updating as long as there is no subsequent not found condition for a CHAIN operation on this key. A not found condition on another access key does not affect the current record for access key number 2.

Differences Between Key Comparison of Indexed Files and Index-Sequential Files

An RPG-issued comparison of the tag file key ignores the zone field when searching for a record with the given key value. When the index-sequential file is used to search for a record, the operating system (MCPII) does the key compare operation, and performs a bit-by-bit comparison that includes the zone field. Because of this difference in the way the system performs the comparison of the two files, an RPG program that accesses an index-sequential file can function differently when using the indexed (tag file) file. Examples of the differences between index-sequential comparison and indexed (tag file) files are given in the following subparagraphs.

Example Set 1:

For the example in this set, assume that the index-sequential file is created in such a way that the zone field for the sign byte of the key is a hexadecimal @F@ character.

1. The user program declares that the file has an alphanumeric key, and the program performs a CHAIN operation to the file using a numeric argument. This program works the same when using the index-sequential file as it did when using the indexed (tag file) file.
2. The user program declares that the file has an alphanumeric key, and the program performs a CHAIN operation to the file using a numeric argument. This program works the same when using an index-sequential file as it did when using the indexed (tag file) file.
3. The user program declares that the file has an alphanumeric key, and the program performs a CHAIN operation to the file using numeric as well as alphanumeric arguments. This program cannot find some records when using the index-sequential file that it found when using the indexed (tag file) file.

All CHAIN operations with numeric arguments function in the same manner. However, if the alphanumeric arguments are created by moving a numeric value to the alphanumeric argument, a CHAIN operation that uses that argument does not find the record by using the index-sequential file. With the B 1000 RPG compiler, a MOVE operation of a numeric value to an alphanumeric field causes a hexadecimal @C@ character to be stored in the zone position of the right-most character. By using the indexed (tag file) method, the RPG compiler has issued a compare operation that has ignored the zone portion of the character when searching for a record with the given key value. When using the index-sequential method, the operating system (MCPII) performs a bit-by-bit comparison, because the file on disk has hexadecimal @F@ characters in the zone portions. In that case, no record is found. The user has to manipulate the zone portion to force it to a hexadecimal @F@ character.

Example Set 2:

For the examples in this set, assume that the index-sequential file is created in such a way that the zone field for the sign byte of the key is a hexadecimal @C@ character.

1. The user program declares that the file has an alphanumeric key, and the program performs a CHAIN operation to the file using a numeric argument. Records that were found when using the indexed (tag) file are not found when using the index-sequential file.

   The numeric argument has a zone portion that contains a hexadecimal @F@ character. As the indexed (tag file) method does not compare the zone portion when searching for the key, the index-sequential method works. Again, the operating system (MCPII) is doing a bit-by-bit comparison and does not find any records.

2. The user program declares that the file has an alphanumeric key, and the program performs a CHAIN operation to the file using an alphanumeric argument. This program works the same with either type of index-sequential file.

3. The user program declares that the file has an alphanumeric key, and the program performs a CHAIN operation to the file using numeric and alphanumeric arguments. Records that were found when using the indexed (tag) file are not found when using the index-sequential file. The reason is the same as in item 1 of this set.

   When the argument is alphanumeric, the argument was created by moving a numeric value to the alphanumeric key, and the keys on disk are right signed, then the same records are found when using the index-sequential file as when using the indexed (tag file) file.

This difference can be resolved when the file is created or converted to this format by doing the following:

1. If the file is being created, specify the X edit code on the Output-Format Specifications for the key field. The X edit code forces the hexadecimal @F@ character in the zone portion of the key field.
2. If the file is being converted from using hexadecimal @C@ to @F@ characters in the zone portion of the key field, specify the X edit code on the Output-Format Specifications when writing the key field to data file.

Indexed

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

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

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

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

B-Indexed

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

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

Direct Files

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

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

Figure 5-3 shows a card file and a direct file on disk that was created from the card file. The relative records number of each card is the relative record number within the disk file that the record occupies.

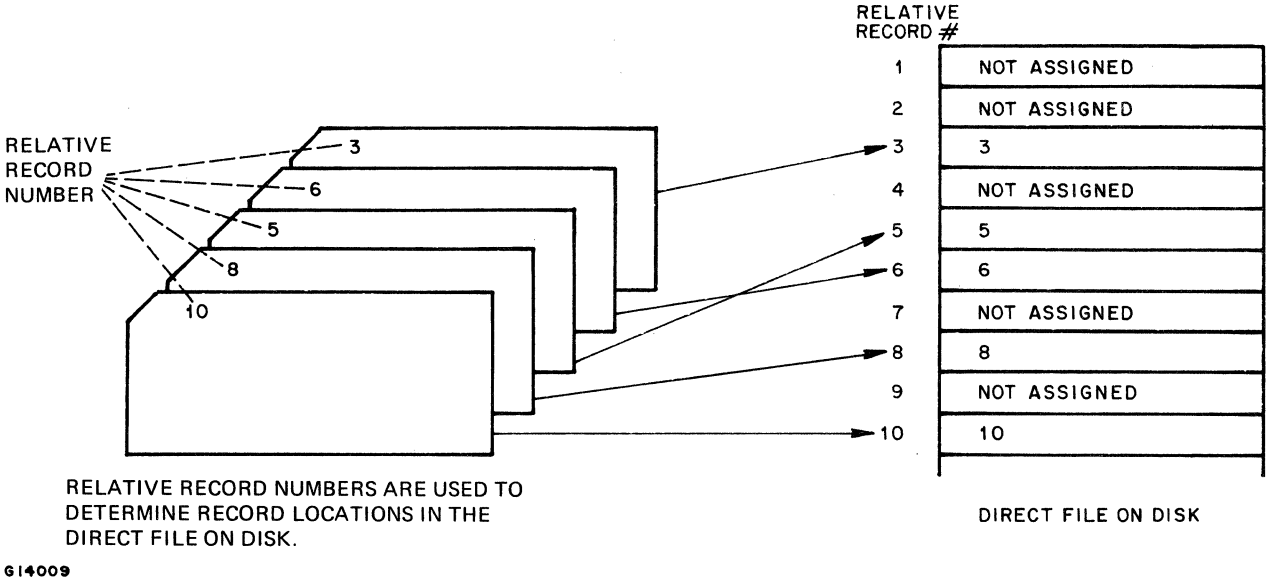


Figure 5-3. Direct File Organization

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

FILE PROCESSING

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

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

1. Consecutive.
2. Sequential by key.
3. Sequential within limits.
4. Random.

Consecutive

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

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

Index-sequential files cannot be processed consecutively without a key.

Sequential By Key

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

Sequential Within Limits

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

1. A limits file.
2. A SETLL operation code.

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

Use of a Limits File

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

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

A and J
D and J
A and I
F and Z
A and Z

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

Limits File Requirements and Restrictions

The restrictions and format requirements for records of a limits file follow:

1. Each limits file record can contain only one set of limits (lower and upper bounds).
2. The same set of limits (lower and upper bounds) can appear in more than one record of a limits file.
3. The lower bound key must begin in position 1 of the record. The upper bound key must immediately follow the lower bound key.
4. The lower bound key and the upper bound key in a limits file record can be the same. The result is that only the one data record described by the keys is read.
5. The length of limits file records must be twice the length of the record key specified in the index file. Therefore, the addition of leading zero or blank characters may be necessary. If the key length specified in the indexed file is 10 bytes, the length of the limits file records is 20 bytes.
6. When the limits file is on disk, either A or P (whichever is applicable) must be specified for the record address type in column 31 of the file description of the limits file. A further description of the use of A and P follows in item g. When the limits file is not on disk, column 31 must remain blank, and alphanumeric keys are assumed.
7. The keys of the limits file can have a different format than the keys of the indexed file that is to be processed within limits. For example, the indexed file can have packed keys and the associated limits file can have alphanumeric keys.

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

   a. The length of alphanumeric key fields must be twice the length specified for packed key fields, minus 1 or 2.
   b. At execution time of the RPG program, the format of the key in the limits file is changed to the same format as specified for the key in the associated indexed file.
   c. Only the digit portion of alphanumeric keys is significant.

8. The normal conventions for specifying the position of signs (left or right) within packed key fields applies to both indexed files and limits files. The files can have their signs in different positions within their key fields. The appropriate Dollar Card entries must be included in the file description specifications when necessary.
9. When an indexed file is to be processed within limits with the use of a limits file and an external indicator is specified in columns 71 and 72 of the file description, the external indicator must be coded for the indexed file.
10. When an indexed file is to be processed within limits with the use of a limits file and an end-of-file specification is desired, an E should be specified in column 17 of the file description specifications for the limits file.

File Description Specifications – Indexed File

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

Table 5-1. File Description Specifications – Indexed File

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

File Description Specifications – Limits File

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

Table 5-2. File Description Specifications – Limits File

Column Number	Field Name	Entry	Function or Description
15	File type	I	Input.
16	File designation	R	Record address file.
17	End of file	E	End of file. This entry is optional.

Table 5-2. File Description Specifications – Limits File (Cont)

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

Extension Specifications – Limits File

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

Table 5-3. Extension Specifications – Limits File

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

Figure 5-4 shows an example program which illustrates the use of limits file processing. The indexed file named DATA is processed by means of the limits file named LIMITS. The program reads a record in the limits file to obtain the first limits pair. A limits pair contains the range in which the file named DATA is to be processed. There are two values which make up a limits pair. The first value is the low key and the second value is the high key. Processing of the data file begins with the low key and ends with the high key. Once the high key is processed, the next record in the limits file is read to obtain the next limits pair. Processing of the data file continues until the high key of the last record in the limits file is processed.

The program in figure 5-4 updates records in the file named DATA by moving the literal UPDATE to the field named VALUE.

Table 5-4 lists the contents of the file named LIMITS.

Table 5-4. Contents of the File Named LIMITS

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

NOTE

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

Since the keys are alphanumeric, the zone portion of the right-most character contains the sign. Refer to appendix B for the hexadecimal equivalents of B 1000 characters.

Table 5-5 lists the contents of the indexed file named DATA before and after the program in figure 5-4 is executed.

Table 5-5. Contents of the File Named DATA

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

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

NOTE

Since the keys are alphanumeric, the zone portion of the right-most character contains the sign. Refer to appendix B for the hexadecimal equivalents of B 1000 characters.

```
        File Description Specifications

01 FDATA    UPE F 180  10L 2AI      1 DISK
02 FLIMTS   IR  F 180   4  2A        EDISK
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

           Extension Specifications

01 E    LIMITS  DATA
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

                Input Specifications

01 IDATA    NS  01
02 I                                            1   2 KEY
03 I                                            3   8 VALUE
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

              Calculation Specifications

 01 C                    MOVE 'UPDATE'  VALUE
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

              Output-Format Specifications

 01 ODATA    D         01
 02 O                            VALUE      8
 --*----1----*----2----*----3----*----4----*----5----*----6----*----
```

Figure 5-4. Example Program Illustrating the Use of Limits

Use of SETLL Operation Code

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

NOTE

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

Calculation Specifications – SETLL

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

Table 5-6. Calculation Specifications – SETLL

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

* This entry must be left-justified.

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

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

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

File Description Specifications – SETLL

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

Table 5-7. File Description Specifications – SETLL

Column Number	Field Name	Entry	Function or Description
15	File type	I U	Input file. Update file.
16	File designation	D	Demand file.
17	End of file	Blank	
28	Processing mode	L	Limits processing.

Table 5-7. File Description Specifications – SETLL (Cont)

Column Number	Field Name	Entry	Function or Description
31	Record address type	A P	Alphanumeric keys. Packed keys.
40	Device	DISK	The indexed file is assigned to disk.
66	File addition/ Unordered	Blank	Additions to the indexed file are not allowed.
71-72	File condition	U1-U8	External indicator. This entry is optional.

Record Processing Using SETLL

There is one exception to the normal convention of reading a demand file, as follows:

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

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

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

1. The record with the lowest record key if a new SETLL instruction is not executed.
2. The first record in the file whose key is equal to or greater than the low record key established by a new SETLL instruction.

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

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

1. When the end-of-file record is read for the indexed file, the following message is displayed on the ODT:

   REOF

2. Following the display of REOF, the operator must enter either the GO or ST system control instruction on the console printer. Entry of GO causes the program to continue, then reading can be performed and the records processed consecutively starting with either:

   1) The record with the lowest record key if a new SETLL operation code is not executed.
   2) The first record in the file whose key is equal to or greater than the low record key established by a new SETLL operation code.

Random

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

1. Read and modified during the time in which detail or total calculations are being performed, and rewritten in the file during the detail or total output phase, or
2. Rewritten during the time in which detail or total calculations are being performed by use of the EXCPT operation code.

Random Processing – Sequential and Direct Files

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

Random Processing – Indexed Files

When indexed files are processed randomly, records are selected on the basis of record key values. An indexed file that is an input file can be processed randomly by the use of relative record numbers. Index-sequential files cannot be processed randomly by using relative record numbers.

Random Processing – Addrout Files

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

Any sequential, indexed, or direct organized disk file that is designated as a primary or secondary file can be randomly processed by using an addrout file. An index-sequential file cannot be randomly processed by using an addrout file. During program execution, a record containing a relative record number is read from the addrout file, and the relative record number is then used to locate and read a record from the data file that is being randomly processed. Only the records of the data file whose relative record numbers are in the addrout file are processed. File processing continues until an end-of-file condition occurs for the addrout file.

Advantages of Using Addrout Files

The advantages of using addrout files for random processing of data files are as follows:

1. A data file can be sorted in many different sequences by using various control fields from the records in the data file.
2. Addrout files can be used in the processing of files that are organized sequential, indexed, or direct.
3. Addrout files require less disk space than comparable tagfiles composed of both record key fields and relative record numbers.
4. More than one addrout file can be used with a program.

Creating Addrout Files

An addrout file can be created by sorting a file with the tagsort option of the SORT compiler. Specify the same blocking factor in the tagsort parameters as specified in the File Description Specifications for the addrout file. A blocking factor of 45 is preferred. If the file is not blocked, each 4-byte relative record number occupies one disk segment. For additional information about the SORT compiler and tagsort, refer to the B 1000 Systems SORT Reference Manual.

File Description Specifications – Data Files

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

Table 5-8. File Description Specifications – Data Files

Column Number	Field Name	Entry	Function or Description
15	File type	I U	Input file. Update file.
16	File designation	P S	Primary file. Secondary file.
17	End of file	E	If an end-of-file test is desired, it should be coded for the addrout file. This entry is optional.
19	File format	F Blank	Fixed length records. Fixed length records.
20-23	Block length	Blank nnnn	File is to be unblocked. Must be a multiple of the record length.
28	Processing mode	R	Addrout processing.
31	Record address type	I	Addrout processing.
40	Device	DISK	The data file is assigned to disk.

Table 5-8. File Description Specifications – Data Files (Cont)

Column Number	Field Name	Entry	Function or Description
66	File addition/ Unordered	Blank	Additions to the data file are not allowed.
71-72	File condition	U1-U8	External indicator. This entry is optional.

File Description Specifications – Addrout File

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

Table 5-9. File Description Specifications – Addrout File

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

Extension Specifications – Addrout File

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

Table 5-10. Extension Specifications – Addrout File

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

Table 5-11 shows which combinations of file processing methods and file organization types can be used with primary, secondary, demand, and chained files. Allowable combinations are indicated by an X.

Table 5-11. Valid Combinations of File Types and File Processing

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

Figures 5-5 through 5-14 show coding examples for a variety of file types.

```
                File Description Specifications

01 F* SEQUENTIAL, BY KEY, NO ADDITIONS
02 F          IP  F          nnAI      1 DISK
03 F          IS  F          nnAI      1 DISK
04 F          ID  F          nnAI      1 DISK
05 F          UP  F          nnAI      1 DISK
06 F          US  F          nnAI      1 DISK
07 F          UD  F          nnAI      1 DISK
08 F* SEQUENTIAL, BY KEY, WITH ADDITIONS
09 F          IP  F            AI      1 DISK                         A
10 F          IS  F            AI      1 DISK                         A
11 F          ID  F            AI      1 DISK                         A
12 F          UP  F            AI      1 DISK                         A
13 F          US  F            AI      1 DISK                         A
14 F          UD  F            AI      1 DISK                         A
15 F* RANDOM, BY KEY, NO ADDITIONS
16 F          IC  F            AI      1 DISK
17 F          UC  F            AI      1 DISK
18 F* RANDOM, BY KEY, WITH ADDITIONS
19 F          IC  F            AI      1 DISK                         A
20 F          UC  F            AI      1 DISK
21 F* UNORDERED LOAD
22 F          O   F            AI        DISK                         U
23 F* ORDERED LOAD
24 F          O   F            AI        DISK
25 F* FILE ADDITIONS
26 F          O   F            AI        DISK                         A
--*----1----*----2----*----3----*----4----*----5----*----6----*-----
```

Figure 5-5. Example of File Description Entries for Indexed Disk Files

```
                File Description Specifications

01 F* CONSECUTIVE PROCESSING
02 F          IP  F                      DISK
03 F          IS  F                      DISK
04 F          ID  F                      DISK
05 F          IT  F                     EDISK
06 F          UP  F                      DISK
07 F          US  F                      DISK
08 F          UD  F                      DISK
09 F* RANDOM, CHAIN WITH RELATIVE RECORD NUMBER
10 F          IC  F                      DISK
11 F          UC  F                      DISK
12 F* LOAD (CREATE) FILE
13 F          O   F                      DISK
14 F* FILE ADDITIONS ONLY
15 F          O   F                      DISK                           A
--*----1----*----2----*----3----*----4----*----5----*----6----*-----
```

Figure 5-6. Example of File Description Entries for Sequential Files

```
          File Description Specifications

01 F* CONSECUTIVE PROCESSING
02 F         IP  F                      DISK
03 F         IS  F                      DISK
05 F         ID  F                      DISK
06 F         UP  F                      DISK
07 F         US  F                      DISK
08 F         UD  F                      DISK
09 F* RANDOM, CHAIN WITH RELATIVE RECORD NUMBER
10 F         IC  F                      DISK
11 F         UC  F                      DISK
12 F* RANDOM, FILE CREATE, CHAIN WITH RELATIVE RECORD NUMBER
13 F         OC  F                      DISK
--*----1----*----2----*----3----*----4----*----5----*----6----*-----
```

Figure 5-7. Example of File Description Entries for Direct Files

```
          File Description Specifications

01 F* PROCESSING METHODS FOR CARD FILES
02 F         IP  F                       READER
03 F         IS  F                       MFCU1
04 F         ID  F                       MFCU2
05 F         IT  F                      EREADER
06 F         CP  F                       MFCU1
07 F         CS  F                       MFCU2
08 F         CD  F                       READER
09 F         O   F                       PUNCH
--*----1----*----2----*----3----*----4----*----5----*----6----*-----
```

Figure 5-8. Example of File Description Entries for Card Files

```
          File Description Specifications

01 F* PROCESSING METHODS FOR TAPE FILES
02 F         IP  F                       TAPE
03 F         IS  F                       TAPE
04 F         ID  F                       TAPE
05 F         IT  F                      ETAPE
06 F         O   F                       TAPE
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 5-9. Example of File Description Entries for Tape Files

```
          File Description Specifications

01 F* PROCESSING METHODS FOR CONSOLE FILES
02 F         ID  F                       CONSOLE
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 5-10. Example of File Description Entries for Console Files

```
          File Description Specifications

01 F* PROCESSING METHODS FOR PRINTER FILES SUPPLYING LINE
02 F* COUNTER SPECIFICATIONS
03 F         O   F                    LPRINTER
04 F* USING SYSTEM DEFAULT FORM LENFTH AND OVERFLOW LINE
05 F         O   F                     PRINTER
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 5-11. Example of File Description Entries for Printer Files

```
          File Description Specifications

01 F* ADDROUT FILES
02 F          IR  F          IT         EDISK
03 F* INDEXED FILES PROCESSED BY ADDROUT FILE
04 F          IP  F       R  IT          DISK
05 F          IS  F       R  IT          DISK
06 F          UP  F       R  IT          DISK
07 F          US  F       R  IT          DISK
08 F          ID  F       R  IT          DISK
09 F          UD  F       R  IT          DISK
10 F* SEQUENTIAL AND DIRECT FILES PROCESSED BY ADDROUT
11 F          IP  F       R  IT          DISK
12 F          IS  F       R  IT          DISK
13 F          UP  F       R  IT          DISK
14 F          US  F       R  IT          DISK
15 F          ID  F       R  IT          DISK
16 F          UD  F       R  IT          DISK
17 F* LIMITS FILE
18 F          IR  I                     EDISK
19 F* INDEXED FILES PROCESSED BY LIMITS FILE
20 F          IP  F       L     AI       DISK
21 F          IS  F       L     AI       DISK
22 F          UP  F       L     AI       DISK
23 F          US  F       L     AI       DISK
24 F          ID  F       L     AI       DISK
25 F          UD  F       L     AI       DISK
--*----1----*----2----*----3----*----4----*----5----*----6----*-----
```

Figure 5-12. Example of File Description Entries for Record Address Processing

The following figures illustrate RPG coding examples for indexed and direct files. Figure 5-13 illustrates the updating of an existing file; figure 5-14 illustrates the creation of a new file.

```
            File Description Specifications

01 FINPUT    IPE F  80  80    2        READER
02 FDIRECT   UC  F 430  10    2        DISK
03 FINDEXED  UC  F 170  17  4 I    11 DISK                    400
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


                Input Specifications

01 IINPUT    AA  99
02 I                                              1   5ODIRKEY
03 I                                             11  14 INXKEY
04 I                                              1  10 SHORT
05 I                                              1  17 LONG
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


             Calculation Specifications

01 C            DIRKEY    CHAINDIRECT
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


            Output-Format Specifications

01 ODIRECT   D         99
02 O                            SHORT      10
03 OINDEXED  D         99
04 O                            LONG       17
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 5-13. Indexed and Direct Files – File Update

```
            File Description Specifications

01 FINPUT   IPE F  96  96            MFCU1
02 FDIRECT  OC  F 180   5            DISK
03 FINDEXED O   F 180  18  4AI    11 DISK                      U
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


                Input Specifications

01 IINPUT   NS  01
02 I                                          1   5ODIRKEY
03 I                                         11  14 INXKEY
04 I                                          6  10 SHORT
05 I                                          1  18 LONG
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


             Calculation Specifications

01 C   01       DIRKEY    CHAINDIRECT
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


            Output-Format Specifications

01 ODIRECT  D         01
02 O                            SHORT       5
03 OINDEXED D         01
04 O                            LONG       18
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 5-14. Indexed and Direct Files – File Creation

The following figures illustrate RPG code examples for index-sequential ($ IXSEQ) files. Figure 5-15 illustrates coding used to create an index-sequential file, figure 5-16 illustrates coding used to add records to an index-sequential file, figure 5-17 illustrates coding used to delete records from an index-sequential file, and figure 5-18 illustrates coding used to list records from an index-sequential file.

Figure 5-15 illustrates coding used to create an index-sequential file.

```
                Control Card Specifications

01 H
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

                File Description Specifications

02 FIN      IPE  1800  90             DISK
03  $ IXSEQ
04 FIXFILE  O    1800  90  8AI     1 DISK   FIRST UACC            U
05 F        KEY             2A      9        SECOND                  D
06 F        KEY             5A     14        THIRD                   N
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

                  Input Specifications

07 IIN      NS  01
08 I                                          1   8 KEY1
09 I                                          9  10 KEY2
10 I                                         14  18KEY3
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

           Output-Format Specifications

11 OIXFILE  D   01
12 O                          KEY1       8
13 O                          KEY2      10
14 O                          KEY3      18
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 5-15. Index-Sequential File – File Creation

```
          Control Card Specifications

01 H
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


          File Description Specifications

02 FINPUT   IPE  1800  90            DISK         U
02 FINPUT   IPE  1800  90            DISK         U
02 FINPUT   IPE  1800  90            DISK         U
03  $ IXSEQ
04 FIXFILE  UC   1800  90  8AI     1 DISK   FIRST UACC          A
05 F        KEY            2A      9        SECOND                D
06 F        KEY            5A     14        THIRD                 N
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


             Input Specifications

07 IINPUT   NS  01   1 CA
08 I                                          1   1 CODE
09 I                                          2   9 KEY1
10 I                                         10  11 KEY2
11 I                                         12  16 KEY3
12 IINPUT   NS  99
13 I                                          1   1 DUMMY
14 IIXFILE  NS  02
15 I                                          1   8 KEYA
16 I                                          9  10 KEYB
17 I                                         14  18 KEYC
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


          Output-Format Specifications

18 OIXFILE  DADD     01
19 O                          KEY1        8
20 O                          KEY2       10
21 O                          KEY3       18
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 5-16. Index-Sequential File – File Addition

```
          Control Card Specifications

01 H
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


          File Description Specifications

02  $ IXSEQ
03 FIXFILE  UPE  1800  90  8AI      1 DISK    FIRST UACC          D
04 F        KEY              2A       9       SECOND                D
05 F        KEY              5A      14       THIRD                 N
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


            Input Specifications

06 IIXFILE  NS  01
07 I                                          1   8 KEY1
08 I                                          9  10 KEY2
09 I                                         14  18 KEY3
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


            Calculation Specifications

10 C           KEY1      COMP 'AAAAAAAA'                  10
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


           Output-Format Specifications

11 OIXFILE  DDEL     10
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 5-17. Index-Sequential File – File Deletion

```
          Control Card Specifications

01 H
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

          File Description Specifications

02  $ IXSEQ
03 FIXFILE  IPE  1800  90  8AI      1 DISK   FIRST U
04 F        KEY             2A      9        SECOND ACC            D
05 F        KEY             5A     14        THIRD                 N
06 FLINE    O         132             PRINTER
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

               Input Specifications

07 IIXFILE  NS  01
08 I                                          1   8 KEY1
09 I                                          9  10 KEY2
10 I                                         14  18 KEY3
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

             Output-Format Specifications

13 O                          KEY2       30
14 O                          KEY3       40
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 5-18. Index Sequential File – File Listing

SECTION 6
ATTRIBUTE SPECIFICATIONS

Attribute Specifications describe file attributes for the file specified by the immediately preceding File Description Specifications in an RPG program. The field definitions and file attributes are described in the following paragraphs.

FIELD DEFINITIONS

The following paragraphs describe the field definitions of the Attribute 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

The letter A must appear in this field.

SPECIFICATION CONTINUATION

This field must be left blank.

ATTRIBUTE NAME AND VALUE

The ATTRIBUTE NAME AND VALUE field contains the attribute name followed by the value of the attribute. The attribute name and value can be specified in free-form format.

FILE ATTRIBUTES

The file attributes allowed in the Attribute Specifications are described in the following paragraphs.

HOSTNAME File Attribute

The HOSTNAME file attribute specifies the name of a remote Burroughs Network Architecture (BNA) host system. For the HOSTNAME file attribute, the value field must begin with an alphabetic character and cannot exceed 17 characters in length. The value is inserted left-justified and, if necessary, is padded with blank characters on the right to fill 17 characters. The value of the HOSTNAME file attribute must be contained within the single quotation mark (') characters and can contain only the letters A through Z, the numbers 0 through 9, the underscore (_), and the hyphen (-) characters. The following is an example that uses the HOSTNAME file attribute, where 'B6800' is the name of a remote host system. Refer to the B 1000 Systems Burroughs Network Architecture (BNA) Installation and Operation Manual for additional information on the B 1000 BNA system.

Example:

```
                File Description Specifications

01 FDISKFIL IPE F1800  90            DISK
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


                Attribute Specifications

02 A  HOSTNAME 'B6800'
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

SECTION 7
VECTORS AND EXTENSION SPECIFICATIONS

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

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

FIELD DEFINITIONS

Figure 7-1 can be used in conjunction with the following field definitions for the Extension Specifications.

1-2 PAGE

Refer to section 2 for a complete description.

3-5 LINE

Refer to section 2 for a complete description.

6 FORM TYPE

The letter E must appear in this field.

11-18 FROM FILENAME

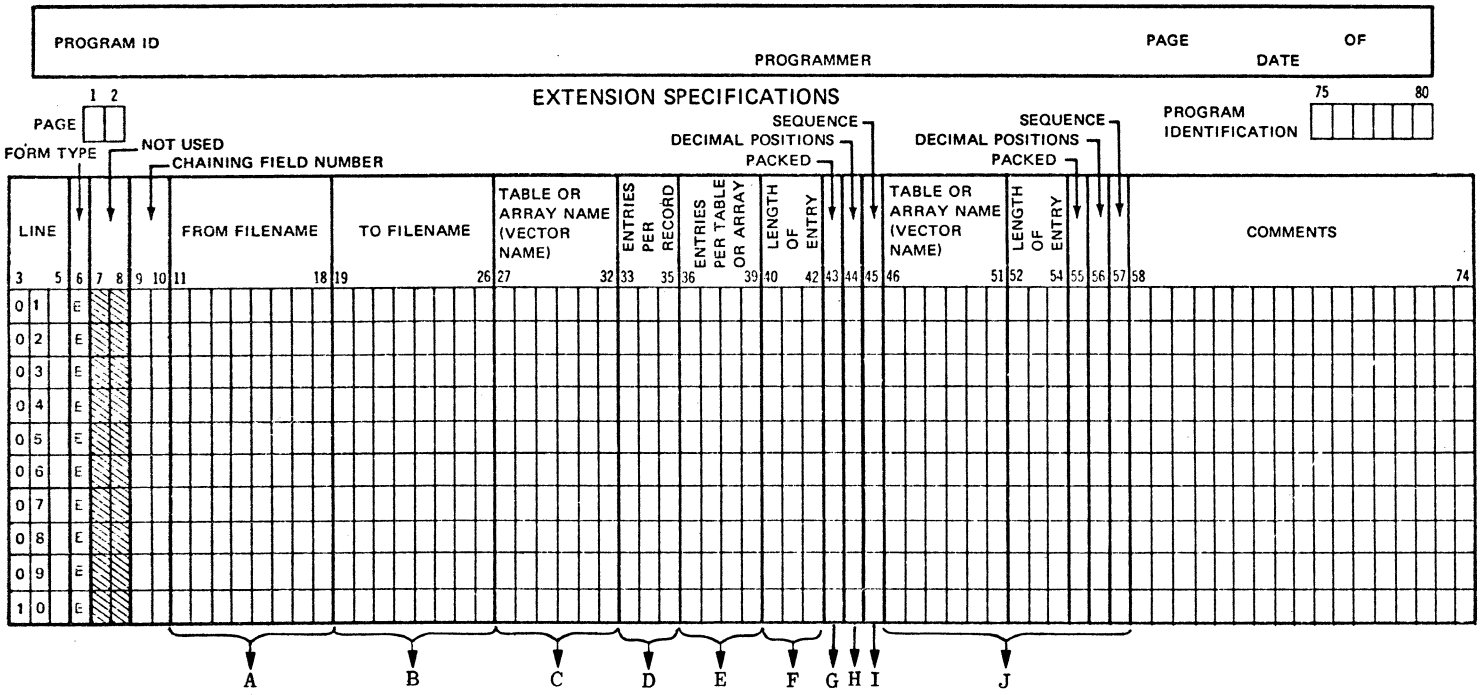
The FROM FILENAME field specifies the name of a record address file or a pre-execution-time vector file, and must contain the filename of every record address file and every pre-execution-time vector file to be used by the program. The file must be specified on the File Description Specifications as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

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

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

19-26 TO FILENAME

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



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

B. 19-26 Contains the file identification of the file to be processed by a record address file or the name of the file to which the vector is 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 input record.

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

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

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

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

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

G14022

Figure 7-1. Extension Specifications Summary Sheet

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

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

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

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

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

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

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

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

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

27-32 VECTOR NAME

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

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

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

33-35 ENTRIES PER RECORD

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

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

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

The FROM FILENAME and ENTRIES PER RECORD entries are used to determine the type of vector and therefore how it is loaded. Table 7-1 is a guide for determining vector type.

Table 7-1. 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

The ENTRIES PER VECTOR field specifies 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 9999 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 can be omitted.

40-42 LENGTH OF ENTRY

The LENGTH OF ENTRY field specifies 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 can 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, as seen from the following, is dependent on the format of the data.

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

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

43 PACKED

The PACKED field specifies the external format of the vector data. Valid entries are:

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

Except for compile-time vectors, any vector can be packed and can be either right or left signed as specified in the Control Card or by the dollar option RSIGN. Compile-time vectors must be in either unpacked decimal or alphanumeric format.

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

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

Binary compile-time vectors are not allowed.

44 DECIMAL POSITIONS

The DECIMAL POSITIONS field specifies:

1. The number of decimal positions contained in each element of the vector named in the first VECTOR NAME field (columns 27-32). If the elements have no decimal positions, a zero must be entered in column 44. This field must not be blank for a numeric vector or if column 43 contains a P.
2. Whether a data item is in numeric or alphanumeric format. Column 40 (LENGTH OF ENTRY) defines the length of the item. For alphanumeric data items, the entry specifies the number of bytes. For numeric data items, including those in packed or binary format, the entry specifies the number of digits. For binary data items, either the number 4 or 9 must be specified.

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

Example:

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

Valid entries for this field are:

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

45 SEQUENCE

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

Valid entries for this field are:

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

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

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

58-74 COMMENTS

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

75-80 PROGRAM IDENTIFICATION

Refer to section 2 for a complete description.

VECTORS

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

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

Table and Array Differences

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

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

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

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

TYPES OF VECTORS

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

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

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

Required Entries for Extension Specifications

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

Figure 7-2 illustrates which fields require entries for each of the three types of vectors.

```
                Extension Specifications

01 E THE FOLLWOING SPECIFICATION LINES DEFINE THE ENTRIES NECESSARY
02 E TO DESCRIBE THE THREE TYPES OF VECTORS. FIELDS CONTAINING ONLY
03 E BLANKS BELOW MUST BE LEFT BLANK WHEN CODING THE VECTOR
04 E DESCRIPTION. FIELDS FILLED WITH THE LETTER "R" ARE REQUIRED;
05 E THOSE FILLED WITH THE LETTER "O" ARE OPTIONAL.
06 E
07 E              OOOOOOOORRRRRRRRRRRRRRRRRROOOOOOOOOOOOOCOMPILE-TIME
08 E     RRRRRRRROOOOOOOORRRRRRRRRRRRRRRRRROOOOOOOOOOOOOPRE-EXECUTION
09 E                        RRRRRR    RRRRRRR RROOOOOOOOOOOOODYNAMIC
10 E
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 7-2. Entries Necessary to Describe a Vector

Columns 27-45 are used as needed to specify 1) the name assigned to identify the vector (VECTOR NAME), 2) the number of vector elements occurring in each input record (ENTRIES PER RECORD), 3) the size of the vector (ENTRIES PER TABLE OR ARRAY and LENGTH OF ENTRY), 4) whether the input data is in binary, packed or unpacked decimal format (PACKED), 5) the number of decimal positions in each entry (DECIMAL POSITIONS), and 6) the order in which the elements are sequenced (SEQUENCE).

Additional entries are required if:

1. The vector is to be loaded at pre-execution time from the file named in the FROM FILENAME field.
2. The vector is to be written to the output file named in the TO FILENAME field at end of job.

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

For dynamic/execution-time vectors, the FROM FILENAME, TO FILENAME, and the ENTRIES PER RECORD fields must be left blank because dynamic vectors cannot be written at end of job.

Compile Time Vectors

For a compiler-time vector load, the data to be loaded is read into the program storage area reserved by the entries in the Extension Specifications at the same time that the source program is compiled. Physically, the data is placed in a card file called RPG/VECTOR, which immediately follows the source deck (see figure 7-3).

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

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

Compile-Time Vector Load

To load a compile-time vector, it is necessary only to include the data cards for the vector in the proper order in the file named RPG/VECTOR (figure 7-3). The compiler automatically reads this file after the source program has been read and loads the data into the vectors declared.

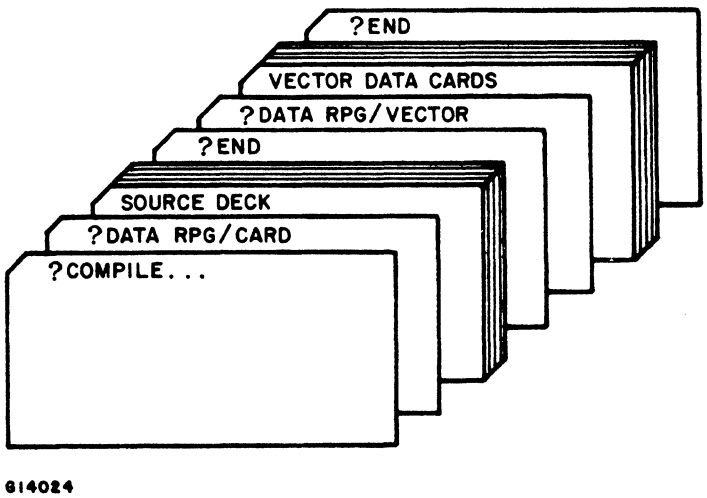


Figure 7-3. Setting Up a Data Deck For a Compile-Time Vector

If more than one compile-time vector has been declared, care must be exercised in setting up the data deck. The compiler expects data for the vectors to be entered in the same order in which they were declared in the Extension Specifications form. The compiler reads data cards and stores the elements in the first vector until it is full, then proceeds to do the same for the second vector, and so forth. If more or fewer records are present than are necessary to exactly fill a vector, data is placed into the wrong vector. See figure 7-4 for an example of a compile-time vector load declaration.

```
            Extension Specifications

01 E             PUNCH1  ARRAY1 25  50  3  A
02 E                     ARRAY2 15  60  4  1
03 E                     TABLE1  4 100 15  ATABLE2  5 2A
04 E
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 7-4. Compile-Time Vector Load

Pre-Execution Time Vectors

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

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

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

Pre-Execution Time Vector Load

To load a pre-execution time vector, the data to be loaded must be placed in the file described in the File Description Specifications to which the vector is assigned (FROM FILENAME). Data is read in at the beginning of program execution and placed in the vector until the vector is full. If more than one vector (not in alternating format) is assigned to a single file, some special considerations must be taken into account (see figure 7-5). Vectors are still loaded in the same order that they are specified in the Extension Specifications, so that files are 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, then 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

01 FFILE1    IT  F  80  80            EREADER
02 FFILE2    IT  F  80  80            EREADER
03 FFILE3    O   F  80  80             PUNCH
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


                Extension Specifications

04 E     FILE1            TABLE1  3 100 18
05 E     FILE2            ARRAY1 20  80  4 20
06 E     FILE1    FILE3   ARRAY2  5  15 10  AARRAY3  5 1A
07 E     FILE1            TABLE2 10 100  6P1
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 7-5. Pre-execution Time Vector Load

Dynamic/Execution Time Vectors

Dynamic vectors are loaded during program execution through entries in the Input Specifications or Calculation Specifications. Certain fields of input records or the results of calculation operations can 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 program control.

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

Dynamic/Execution Time Vector Load

To load a dynamic vector, the data elements can be obtained from fields within input records or from the result of operations in the Calculation Specifications. Figure 7-6 provides coding examples on the File Description Specifications and the Extension Specifications for dynamic vector loading.

```
                File Description Specifications

01 FFILE1   IP  F  80  80              READER
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


                Extension Specifications

02 E                        AR1        100  5  A
03 E                        AR2         10  6  1A
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 7-6. Dynamic Vector Loading

Input Specifications Load

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

```
                Input Specifications

01 IFILE1   AA  01   1 CT
02 I                                          2   6 AR1,1
03 I                                          8  67 AR2
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 7-7. Dynamic Vector Load – Input Specifications

NOTE

When a record read from FILE1 has the letter T in position 2, the 01 indicator is turned ON. The five characters beginning in position are loaded into the first element of the AR1 array. The 60 characters beginning in position 8 are loaded into the entire AR2 array.

Calculation Specifications Load

Any operation which specifies a vector (with or without an index) as the RESULT FIELD field causes the designated vector element (or entire array, if no index is specified) to be loaded with the result of the operation. See figure 7-8 for an example of a load via the Calculation Specifications. Refer to section 9 for the operations which can be specified for vectors.

```
                Calculation Specifications

01 C             AR1,1      ADD  AR2         AR2
02 C* ADD THE FIRST ELEMENT OF ARRAY AR1 TO EVERY ELEMENT OF
03 C* ARRAY AR2.
04 C                        Z-ADDO           IX       30
05 C* DEFINE THE NUMERIC FIELD "IX" (3 DIGITS LONG, 0 DECIMAL
06 C* POSITIONS) AND SET IT TO ZERO.
07 C             LOOP       TAG
08 C* DEFINE THE LABEL "LOOP"
09 C                1       ADD  IX          IX
10 C                10      COMP IX                            0202
11 C* IF THE VALUE OF FIELD IX IS LESS THAN OR EQUAL TO 10,
12 C* THEN TURN ON INDICATOR 02.
13 C   02       AR2,IX     DIV  3.1415      AR2,IX
14 C* IF INDICATOR 02 IS ON, DIVIDE 3.1415 INTO THE ELEMENT OF
15 C* AR2 INDEXED BY THE VALUE OF FIELD IX.
16 C   02                  GOTO LOOP
17 C* IF INDICATOR 02 IS ON, THEN BRANCH TO LABEL "LOOP".
18 C             'GOOD BYE'DSPLYODT
19 C* DISPLAY THE MESSAGE "GOOD BYE".
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 7-8. Dynamic Vector Load – Calculation Specifications

Rules for Loading a Vector

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

1. For a vector to be loaded at pre-execution time, entries are necessary in the FROM FILENAME and ENTRIES PER RECORD fields.
2. For a vector to be loaded at compile time, the FROM FILENAME field must be blank and an entry must be made in the ENTRIES PER RECORD field.
3. Vector loading is not implied if the FROM FILENAME, TO FILENAME, and ENTRIES PER RECORD fields are blank. Vectors whose specifications contain blanks in those fields can be loaded as a result of Input or Calculation Specifications, which are execution-time loads.

Vector Output

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

Dynamic vectors cannot specify a TO FILENAME, and thus cannot be automatically written out at end of job.

Also, an entire array can be written during output time by specifying the array name without an index as field names (VARIABLE NAME) in the Output-Format Specifications. If an entire array is to be output in this way, the end position specified must allow sufficient space for all elements of the array, allowing for any editing. For editing of a whole array see section 12.

Vectors in Alternating Format

Vectors specified as occurring in alternating format have related elements contained in alternating format on each input record (see figure 7-9). The two vectors need not be of the same size (LENGTH OF ENTRY), type (numeric or alphanumeric), or sequence order, but they must have the same number of elements contained on each input record, and each vector must contain the same number of elements (NUMBER OF ENTRIES PER TABLE OR ARRAY). Each pair of elements on the input record is considered one entry. The first element of each pair belongs to the vector described in columns 27-45; the second element belongs to the vector described in columns 46-57.

```
            Extension Specifications

01 E     VECTOR1          VEC1    10  10  5 2
02 E     VECTOR2          VEC2    10  10  2
03 E                      VEC3    10  10  3 2 VEC4     2
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


              Input Records

VEC1:
00000111112222233333444445555566666777778888899999
----*----1----*----2----*----3----*----4----*----5----*----6----*---

VEC2:
AABBCCDDEEFFGGHHIIJJ
----*----1----*----2----*----3----*----4----*----5----*----6----*---

VEC3:
000AA111BB222CC333DD444EE555FF666GG777HH888II999JJ
----*----1----*----2----*----3----*----4----*----5----*----6----*---
```

Figure 7-9. Vectors in Alternating Format

SECTION 8
LINE COUNTER SPECIFICATIONS

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

1. Specify form length. The default is 66 lines.
2. Specify the overflow line.
3. Define the line – channel equations that are associated with the carriage control format tape that must be installed on the printer when the program is executed.

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

LINE – CHANNEL EQUATIONS

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

CHANNEL

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

LINE NUMBER – CHANNEL NUMBER

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

FIELD DEFINITIONS

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

1-2 PAGE

Refer to section 2 for a complete description.

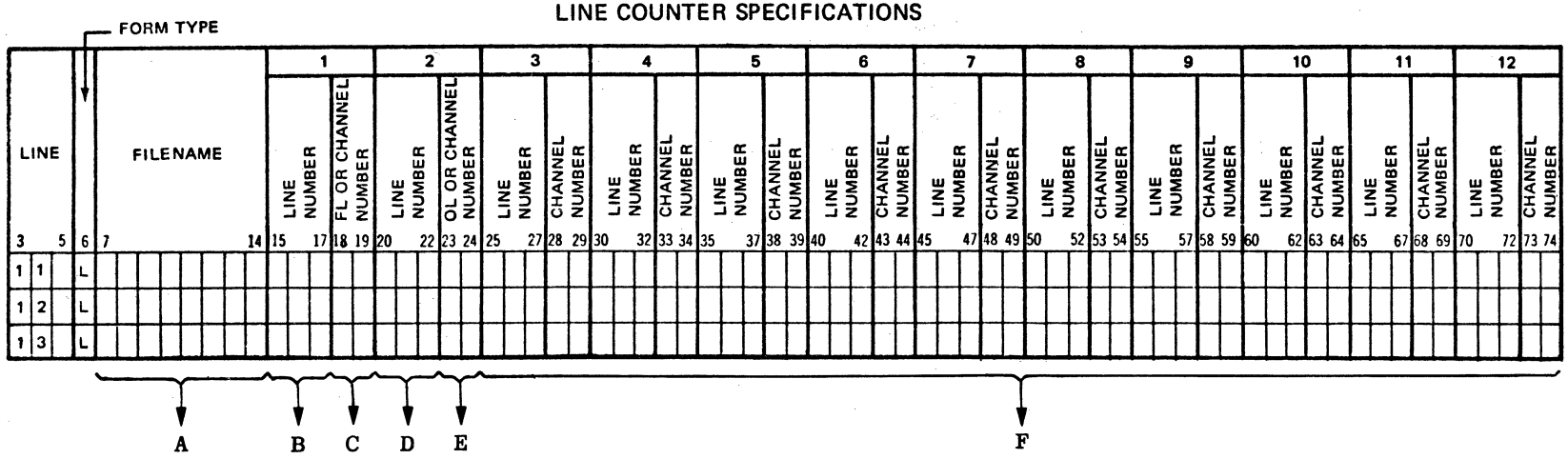
3-5 LINE

Refer to section 2 for a complete description.

6 FORM TYPE

This field must contain the letter L.

LINE COUNTER SPECIFICATIONS



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

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

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

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

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

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

GI4029

Figure 8-1. Line Counter Specifications Summary Sheet

7-14 FILENAME

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

15-19 LINE NUMBER, FL OR CHANNEL NUMBER

The LINE NUMBER field has two possible meanings, depending upon the entry in FL OR CHANNEL NUMBER field (columns 18-19).

If the FL OR CHANNEL NUMBER field (columns 18-19) contains the entry FL (Forms Length), the entry in the LINE NUMBER field (columns 15-17) specifies the length (in print lines) of each page. The LINE NUMBER field entry must be between 1 and 112, inclusive, and be right-justified in the field. Leading zero (0) characters are optional (see figure 8-2).

If the FL OR CHANNEL NUMBER field (columns 18-19) contains a numeric entry between 1 and 12, inclusive, then the entry in the LINE NUMBER field (columns 15-17) specifies the number of lines from the top of the form to be associated with the CHANNEL NUMBER entry designated in columns 18-19. The LINE NUMBER entry must be between 1 and 112, inclusive. Both entries must be right-justified in their respective fields. Leading zeros are optional (see figure 8-2).

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).

```
            Line Counter Specifications

01 L          1001 5012
02 L          50FL 1001 6012
03 L          80FL 600L
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

  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 8-2. Line Counter Specifications Code Example

20-24 LINE NUMBER, OL OR CHANNEL NUMBER

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

RPG II Dialect Only

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

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

1. Detail lines still to be printed as part of the current program cycle are completed.
2. Total lines are printed.
3. Total lines conditioned by the overflow indicator for this file are printed.
4. Heading and detail lines conditioned by the overflow indicator for this file are printed.

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

RPG I or RPG II Dialect

If columns 23-24 contain a numeric entry between 1 and 12, inclusive, the entry in columns 20-22 must specify the number of lines from the top of the form to be associated with the CHANNEL NUMBER entry (columns 23-24). The LINE NUMBER field entry must be between 1 and 112, inclusive. Both entries must be right-justified in their respective fields, and leading zeros are optional (see figure 8-2).

25-74 LINE NUMBER AND CHANNEL NUMBER

The remainder of the form is divided into ten 5-character fields, each consisting of a 3-character LINE NUMBER field and a 2-character CHANNEL NUMBER field. All fields are optional and must be left blank if they are not to be used (see figure 8-2).

The CHANNEL NUMBER fields can contain a numeric entry between 1 and 12, inclusive. This CHANNEL NUMBER entry is associated with the corresponding LINE NUMBER field entry (between 1 and 112, inclusive) and is used to relate a channel number to a particular line on each page of the output forms. The same channel numbers must not be specified more than once on the same Line Counter Specification.

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

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

75-80 PROGRAM IDENTIFICATION

Refer to section 2 for a complete description.

PRINTER CHANNEL SKIPPING

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

CHANNEL SKIPPING – RPG II

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

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

Example:

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

CHANNEL SKIPPING – RPG I

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

CHANNEL SKIPPING – RPG I and RPG II

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

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

Printer Backup

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

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

1. Columns 15-19 can optionally contain the form length and columns 18-19 can optionally contain "FL". If these entries are omitted, the default value of 66 is assumed for the form length.
2. If the form length is specified in columns 15-19, then columns 20-74 can be used for line – channel equations that can specify a maximum of 11 channels.
3. If a form length is not specified in columns 15-19, the default value of 66 is assumed and columns 15-74 can be used to specify up to 12 channels.
4. The user must specify channel 1 and channel 12. Channel 12 defines the overflow line.
5. The user must provide line – channel specifications for any channel used in the Output-Format Specifications.
6. "OL" is not specified.
7. A skip to channel 12 is permitted.

SECTION 9
TELECOMMUNICATIONS CARD SPECIFICATIONS

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

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

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

DATA COMMUNICATIONS FILES

Data communications files are required when the RPG program is to communicate with a remote device, and they are used to transmit and/or receive information contained in corresponding NDL files of the network controller. The filename of the NDL file must be entered in columns 7-14 of the Telecommunications Card. The data communications files used with B 1000 RPG are of the following two types:

1. REMOTE
2. DATACOM and BSCA

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

REMOTE FILES

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

A description of the required fields to be coded for a Telecommunications Card follows. This description is followed by figure 9-1, which illustrates the use of the Control Card Specifications form for coding Telecommunications entries for REMOTE files.

1-2 PAGE

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

3-5 LINE

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

6 FORM TYPE

This field must contain the letter T.

7-14 FILENAME

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

15

Column 15 must be blank.

16-18 NUMBER OF STATIONS

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

19-21 MAXIMUM MESSAGES

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

Example:

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

1. The terminal remains in transmit mode following the attempted transmission of the third message since the input queue is full.
2. When the program performs a read operation, a message is read from the input queue, thereby making space available in the input queue for the third message.
3. The third message is then transmitted from the terminal buffer to the input queue, and the terminal returns to local mode.

22-27 STATION NUMBER

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

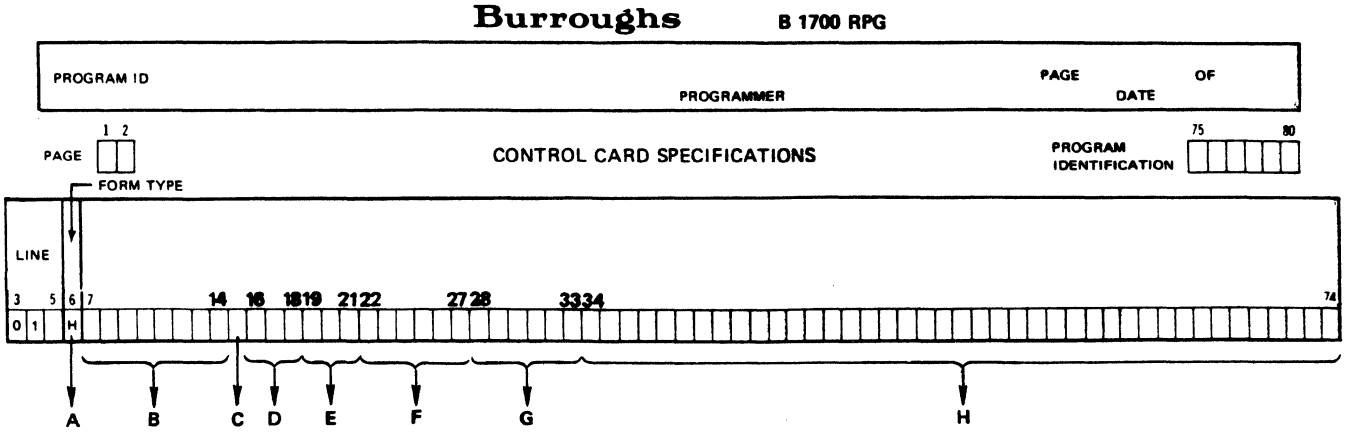
28-33 MESSAGE LENGTH

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

34-80

This field must remain blank.

Figure 9-1 illustrates the use of the Control Card Specifications form for coding Telecommunications entries for REMOTE files.



A. 6 Must contain the letter T.

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

C. 15 Must remain blank.

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

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

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

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

H. 34-80 Must remain blank.

G12038

Figure 9-1. Telecommunications Entries – REMOTE Files

DATACOM OR BSCA FILES

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

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

Figure 9-2 illustrates the use of the Control Card Specifications form for coding Telecommunications entries for DATACOM or BSCA files.

The following paragraphs describe the field definitions for DATACOM or BSCA files on the Telecommunications Specification.

1-2 PAGE

Refer to section 2 for a complete description.

3-5 LINE

Refer to section 2 for a complete description.

6 FORM TYPE

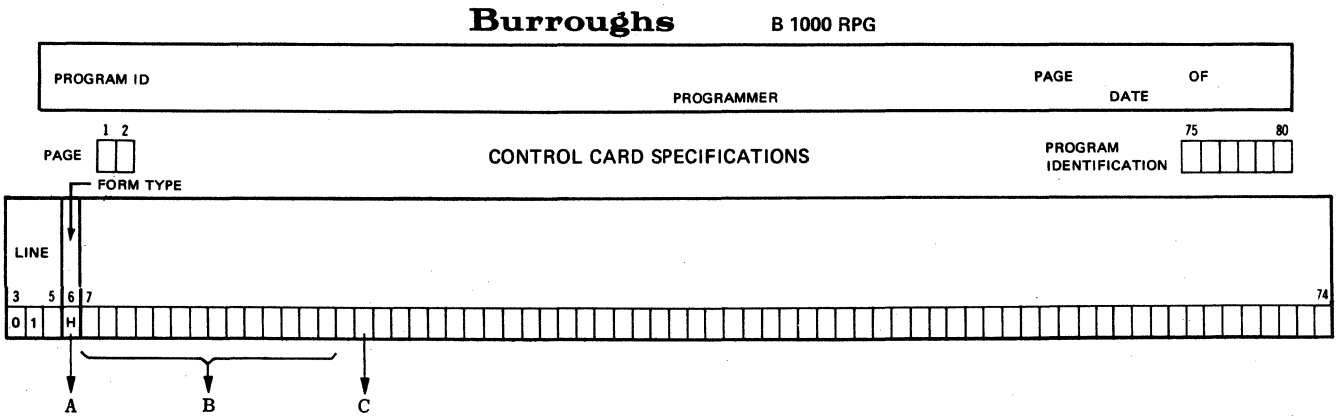
This field must contain the letter T.

7-14 FILENAME

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

15 CONFIGURATION

This field is unused. Any entry other than blank, P, M, or S causes a syntax error to be generated.



A. 6 Must contain the letter T.

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

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

G14033

Figure 9-2. Telecommunications Entries – DATACOM or BSCA

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

The program displays three fields. The first two fields are in forms mode. Enter any 8-digit number into the first two fields, press the transmit (XMT) key with the cursor in the home position, and the result is displayed in the third field. The primary file DUMMYFI is specified to satisfy the RPG compiler requirements for a primary file. The file must reside on disk.

```
                File Description Specifications

01 FDUMMYFI IPE F                    DISK
02 FRMTE    CD  F19201920            REMOTE
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

              Telecommunications Card Specifications

01 TRMTE     001002STANUMMESLEN
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

                   Input Specifications

02 IDUMMYFI NS  01
03 I                                          1    1 DUMMY
04 IRMTE    NS  02
05 I                                          1   80FLD1
06 I                                          9  160FLD2
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

              Calculation Specifications

01 C  N99               EXSR SETUP
02 C            AGAIN   TAG
03 C                    MOVE '100'     MESLEN
04 C                    SEND RMTE                    1011
05 C                    RECV RMTE                 20
06 C            FLD1    COMP BYE                  LR
07 C  NLR       FLD1    COMP END                  LR
08 C  NLR       FLD1    ADD  FLD2      FLD3    90
09 C  NLR               GOTO AGAIN
10 CSR          SETUP   BEGSR
11 CSR                  MOVE 'BYE'     BYE     80
12 CSR                  MOVE 'END'     END     80
13 CSR                  SETON                     99
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 9-3. Example of an RPG Data Communications Program (Sheet 1 of 3)

Calculation Specifications (continued)

```
14 C* THE FOLLOWING HEXADECIMAL BEFINES ARE VALID FOR THE
15 C* TD830, MT983, AND MT985 SERIES TERMINALS.  IF A DIFFERENT
16 C* TERMINAL IS TO BE USED, CONSULT THE REFERENCE MANUAL OF
17 C* THE TERMINAL FOR THE CORRECT BIT REPRESENTATIONS.
18 C*
19 C* END OF TEXT - ETX
20 CSR                    BITOF'012345'  ETX      1
21 CSR                    BITON'67'      ETX
22 C* STAY IN RECEIVE MODE - SR
23 CSR                    BITOF'012456'  SR       1
24 CSR                    BITON'37'      SR
25 C* LINE FEED - LF
26 CSR                    BITOF'01346'   LF       1
27 CSR                    BITON'257'     LF
28 C* HORIZONTAL - HT
29 CSR                    BITOF'012346'  HT       1
30 CSR                    BITON'57'      HT
31 C* CARRIAGE RETURN - CR
32 CSR                    BITOF'01236'   CR       1
33 CSR                    BITON'457'     CR
34 C* HOME CURSOR AND CLEAR SCREEN - HCLR
35 CSR                    BITOF'012367'  HCLR     1
36 CSR                    BITON'45'      HCLR
38 C* HOME CURSOR - HC
39 CSR                    BITOF'0167'    HC       1
40 CSR                    BITON'2345'    HC
41 C* LEFT BRACKET - LB
42 CSR                    BITOF'012'     LB       1
43 CSR                    BITON'34567'   LB
44 C* RIGHT BRACKET - RB
45 CSR                    BITOF'0127'    RB       1
46 CSR                    BITON'3456'    RB
47 C* FORMS MODE - FM
48 CSR                    BITOF'0134'    FM1      1
49 CSR                    BITON'2567'    FM1
50 CSR                    BITOF'347'     FM2      1
51 CSR                    BITON'01256'   FM2
52 CSR                    MOVELFM1       FM       2
53 CSR                    MOVE FM2       FM
54 C*
55 CSR                    SETON                     99
56 CSR                    MOVE '001'     STANUM
57 CSR                    ENDSR
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 9-3. Example of an RPG Data Communications Program (Sheet 2 of 3)

```
                Output-Format Specifications

01 ORMTE     E
02 O                              HCLR        1
03 O                              LB          2
04 O                              FLD1   Z    10
05 O                              RB          11
06 O                                          21 'ADDED TO'
07 O                              LB          23
08 O                              FLD2   Z    31
09 O                              RB          32
10 O                                          39 'EQUALS'
11 O                              FLD3   Z    49
12 O                              CR          50
13 O                              CR          51
14 O                              CR          52
15 O                                          78 'ENTER "BYE" OR "END" TO '
16 O                                          88 'GO TO EOJ.'
17 O                              FM          90
18 O                              ETX         91
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 9-3. Example of an RPG Data Communications Program (Sheet 3 of 3)

SECTION 10
INPUT SPECIFICATIONS

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

1. Record type descriptions (columns 7-42) which define the various input records and their relationship to other records in the file. Columns 43-70 must be blank.
2. Field descriptions (columns 43-70) which define each field within the records. Columns 7-72 must be blank.

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

FIELD DEFINITIONS

Figure 10-1 can be used in conjunction with the following field definitions for the Input Specifications.

1-2 PAGE

Refer to section 2 for a complete description.

3-5 LINE

Refer to section 2 for a complete description.

6 FORM TYPE

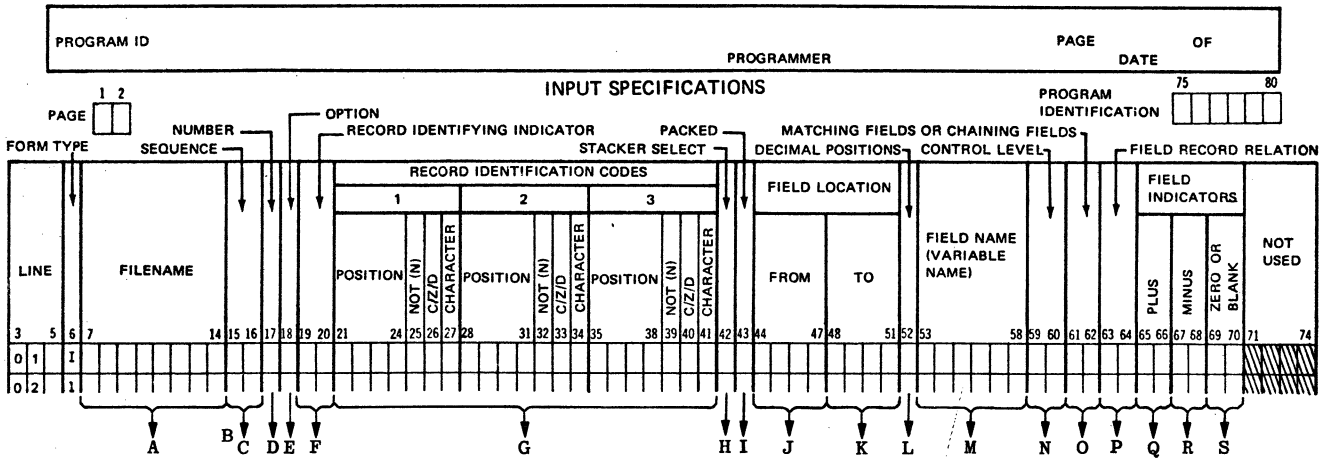
This field must contain the letter I.

7-14 FILENAME

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

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

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



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), the spread card designator(TR), or specifies look ahead records(**).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

G14034

Figure 10-1. Input Specifications Summary Sheet

14-16 AND/OR LINES

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

AND Line

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

OR Line

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

```
                    Input Specifications

01 IINPUT   011 L1  77 CD  78 CE  79 CP
02          AND     80 CT
03          OR       1 D9
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 10-2. AND/OR Relationships – Record Identification Codes

NOTE

The line in figure 10-2 describe a record type which can be identified by the code DEPT appearing in positions 77-80 of the input record. The AND line specifies additional characters as part of the record identification code. The OR line allows an alternative code, that is, the digit 9 in the first position to identify the record type. This allows both DEPT in positions 77-80 or the digit 9 in position 1 to identify the input record type. Both conditions cause record identifying indicator L1 to be turned ON.

15-16 SEQUENCE

The SEQUENCE field specifies 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 is emitted), it specifies that the record types need not be in any special order.

Within each file, all record types having alphabetic entries in the SEQUENCE field must be specified before those with numeric entries. Refer to figure 10-4 for an example of how to code the SEQUENCE field when both alphabetic and numeric entries are desired. All chained and demand files must have an alphabetic entry in this field.

A sequence group is data file records which are defined by a numeric entry specified in columns 15-16 of the Input Specifications 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 automatically checks 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 halts. The system operator can order the program to resume, at which time it ignores the record that is out of sequence and reads 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 10-3, 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 can 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 occurs each time the first record of a new group is read.

```
                  Input Specifications

01 I*
02 I* NOT SEQUENCE CHECKED
03 IDISKIN  AA  01  15 D5
04 I*
05 I* SEQUENCED CHECKED
06 IDISKIN  011 L1  80 CX
07 I        02N 02  80NCX
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 10-3. Input Sequence Checking

Refer to figure 10-4 for an example of how to code the SEQUENCE field when both alphabetic and numeric entries are desired.

```
                  Input Specifications

01 IDISKIN  NS  10  15 D5
02 I                                      4  13 FIELD1
03 I        011 20  80 CA
04 I                                      4   9 FIELD2
05 I        02N 30  80NCA
06 I                                      4   9 FIELD3
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 10-4. Example of Alphabetic and Numeric Entries in SEQUENCE Field

NOTE

Line 01 specifies a record type description that is not sequence checked for a sequence group of records. If position 15 of the input record contains the digit 5, indicator 10 is turned ON and the input data is moved to the FIELD1 field.

Line 03 specifies a record type description that can only appear exactly once

in the sequence group of records. If position 80 contains the character A, indicator 20 is turned ON and the input data is moved to the FIELD2 field.

Line 05 specifies a record type description that can appear one or more times in this sequence group of records. If position 80 does not contain the character A, indicator 30 is turned ON and the input data is moved to the FIELD3 field.

17 NUMBER

The NUMBER is used only if sequence checking is to be done, that is, the SEQUENCE field contains a numeric entry. An entry in the NUMBER field indicates whether more than one record of the designated type can appear in each group of a sequenced input file (see figure 10-4).

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

Valid entries for the NUMBER field are:

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

Figure 10-4 shows the alphabetic and numeric entries in the SEQUENCE field of the Input Specifications.

18 OPTION

The OPTION field is used only if sequence checking is to be done, that is, the SEQUENCE field contains a numeric entry. An entry in this field indicates whether certain record types are optional. A letter O entry specifies that a record of this type can or can not be present in each group of a sequenced file. If this field is left blank, each group can contain one or any number of records of this type, depending on the entry in column 17 (see figure 10-4).

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

Entry	Definition
Blank	Record type must be present in each group.
O	Record type is optional and cannot be present in each group.

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

Example Coding of Sequence, Number, and Option Fields

Figures 10-4A, B, and C illustrate coding of the sequence, number, and option fields on the Input Specifications. Following each figure is a description of the specified record sequence. In conjunction, tables 10-1, 10-2, and 10-3 list possible sequences for which records from SEQ1, SEQ2, and SEQ3 can appear in for figures 10-4A, B, and C, respectively. The records from the SEQ1, SEQ2, and SEQ3 files are read sequentially.

```
                    Input Specifications

01 ISEQ1     011 01    1 C1
02 I                                          1    1OFIELD1
03 I         021002    1 C2
04 I                                          1    1OFIELD2
05 I         031003    1 C3
06 I                                          1    1OFIELD3
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 10-4a. Input Specifications for DISKIN (SEQ1)

Figure 10-4a is used in conjunction with table 10-1.

The sequence that records from SEQ1 (as specified in figure 10-4a) can appear in is as follows:

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

Records with a 2 or 3 in column one are optional.

Table 10-1 lists one possible sequence in which records can appear from SEQ1. The asterisk (*) characters denote out of sequence records. This table is used with figure 10-4a.

Table 10-1. SEQ1 Data File

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

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

```
                    Input Specifications

01 ISEQ2     01N 01    1 C1
02 I                                          1    10FIELD1
03 I          021002    1 C2
04 I                                          1    10FIELD2
05 I          031003    1 C3
06 I                                          1    10FIELD3
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 10-4b. Input Specifications for DISKIN (SEQ2)

Figure 10-4b is used in conjunction with table 10-2.

The sequence that records from SEQ2 (as specified in figure 10-4b) can appear in is as follows:

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

Table 10-2 lists one possible sequence in which records can appear from SEQ2. Asterisks denote out of sequence records. This table is used with figure 10-4b.

Table 10-2. SEQ2 Data File

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

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

```
                  Input Specifications

01 ISEQ2     01N 01    1 C1
02 I                                          1    10FIELD1
03 I         021002    1 C2
04 I                                          1    10FIELD2
05 I         031 03    1 C3
06 I                                          1    10FIELD3
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 10-4c. Input Specifications for DISKIN (SEQ3)

Figure 10-4c is used in conjunction with table 10-3.

The sequence that records from SEQ3 (as specified in figure 10-4c) can appear in is as follows:

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

Table 10-3 lists one possible sequence in which records can appear from SEQ 3. The asterisk (*) characters denote out of sequence records. This table is used with figure 10-4c.

Table 10-3. SEQ3 Data File

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

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

19-20 RECORD IDENTIFYING INDICATOR

The RECORD IDENTIFYING INDICATOR field can be used for the following purposes:

1. To assign an indicator to each record type.
2. To indicate look-ahead fields.
3. To specify spread cards.

Entry	Definition
01-99	Record-identifying indicator.
L1-L9	Control level indicator.
LR	Last record indicator.
H0-H9	Halt indicator.
TR	Spread cards.
**	Look-ahead field.

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

Record Identifying Indicator 01-99

Each input file can 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 can 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. For cycle driven files, only one record-identifying indicator is set ON. For non-cycle driven files, such as chain or demand files, there can be more than one record-identifying indicator ON at any one time if the RECORD IDENDIFYING CODES field causes a different record-identifying indicator to be set ON.

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 can be assigned to more than one type.

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

Control Level Indicator L1-L9

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

Last Record Indicator LR

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

Halt Indicator H0-H9

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

Look-Ahead Field **

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

Look-ahead fields can be used to:

1. Determine when the last record in a particular control group is being processed.
2. Extend the use of the matching record function.

Rules for Look-Ahead Fields

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

1. Look-ahead fields can be specified for input, update, or combined files that are primary or secondary files, regardless of whether or not they are processed by record address files.
2. Look-ahead fields cannot be specified for combined files, demand files, or for files that specify spread card records.
3. One set of look-ahead fields can be specified per file, and the field descriptions apply to all records in that file.
4. Look-ahead fields cannot be used as result fields in calculation operations.
5. The name given to a look-ahead field must not occur on any other Input or Extension Specification.
6. If the look-ahead field occurs on an Output Specification, blank after must not be specified.
7. When a program needs to access information before and after the record is selected for processing, the field must be described twice with different names (once as a look-ahead field and once in the normal way). Refer to figure 10-5.
8. The ** line cannot follow a record type description that has a numeric sequence entry.
9. Columns 17-18 and 21-74 must be left blank.
10. Any combination of alphabetic characters or blanks can be entered in columns 15-16 except ND and R*w (*w = blank).
11. The fields themselves are described on the lines following the *a*a line. When fields are described, columns 7-42 and 59-74 must be blank.
12. When the last record of a file is being processed, any look-ahead fields for that file contain all "9's" (signed numeric or alphanumeric according to field type).

Figure 10-5 provides an example of how to code look-ahead fields.

```
                    Input Specifications

01 IFILEIN  AA  01
02 I                                          1  19 FIL1
03 I                                         20  33 FIL2
04 I                                         34  40 FIL3
05 I          AB  **
06 I                                          1  19 NXFIL1
07 I                                         20  33 NXFIL2
08 I                                         34  40 NXFIL3
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 10-5. Coding Look-Ahead Fields

Use of the Look-Ahead Feature with Input Files

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

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

When the look-ahead feature is used with an update or combined file, the look-ahead field usually references the current record being processed. The look-ahead file references only the next record in the file when the current record was not read from the file. Therefore, when an update or combined file is the only file being read, the look-ahead field always references the current record.

Processing Two Input Files Using the Look-Ahead Feature

Figure 10-6 shows processing of records from two input files, one of which is a primary file and the other a secondary file. All primary records are processed before any secondary records are available. Therefore, it is necessary to use the look-ahead feature in order to use data from the secondary file while processing the primary file.

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

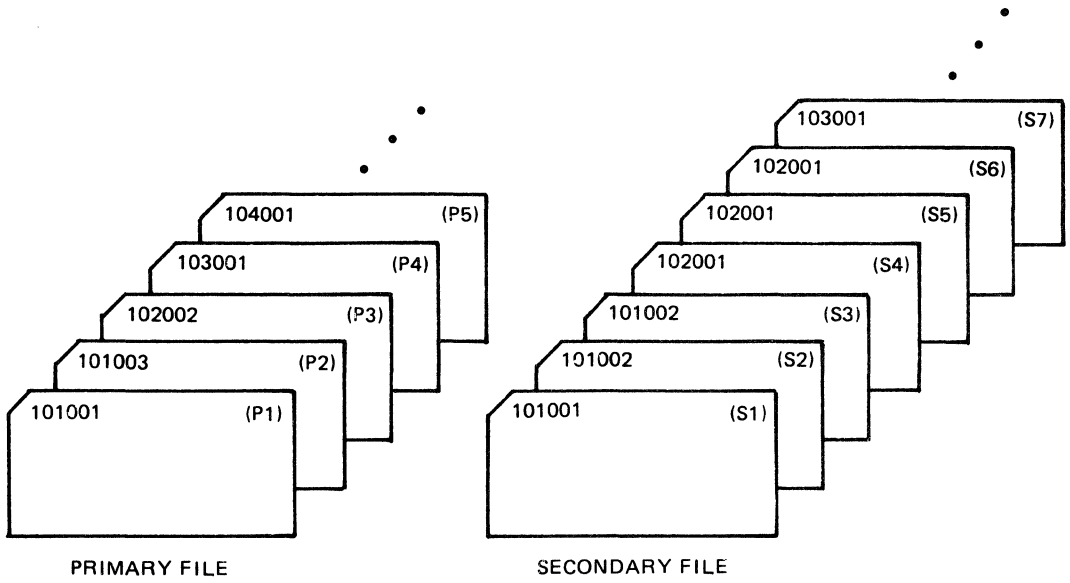


Figure 10-6. Records Available for Look-Ahead: Two Input Files

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

Figure 10-7 shows processing records from an update file and a secondary input file.

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

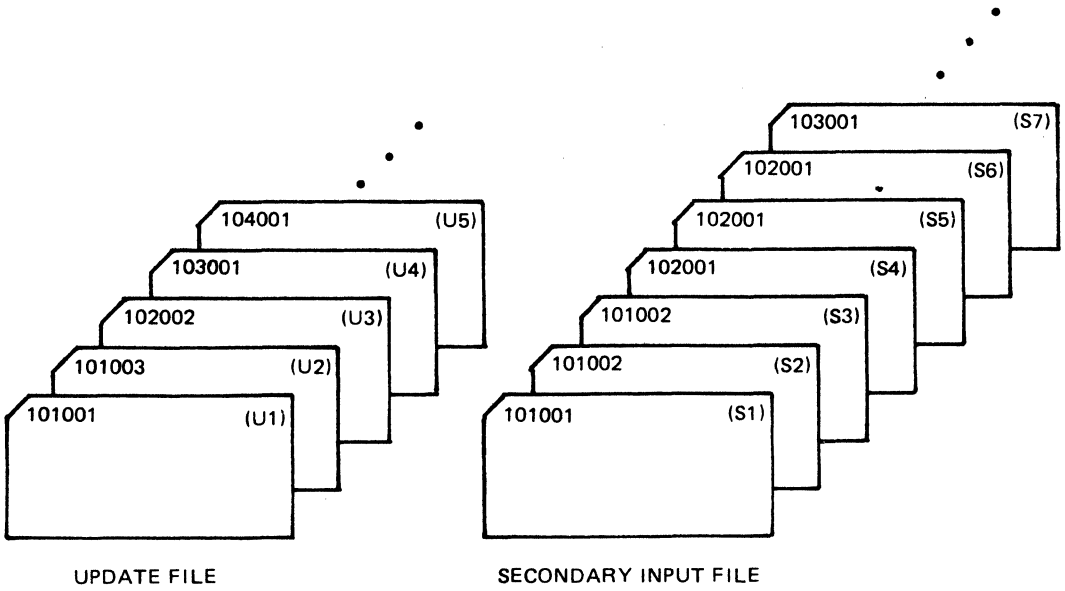


Figure 10-7. Records Available for Look-Ahead: One Update File, One Input File

Spread Card Indicator TR

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

Figure 10-8 illustrates the difference between a regular data deck and a data deck in spread card format. Note that six cards are required for invoice number 1 in regular format, but only one spread card is required for the identical information. For each spread card, the header field contains INVOICE NUMBER and the trailer fields contain the set PART# QTY PRICE.

Figure 10-9 shows the Input Specifications for the spread card data deck shown in figure 10-8.

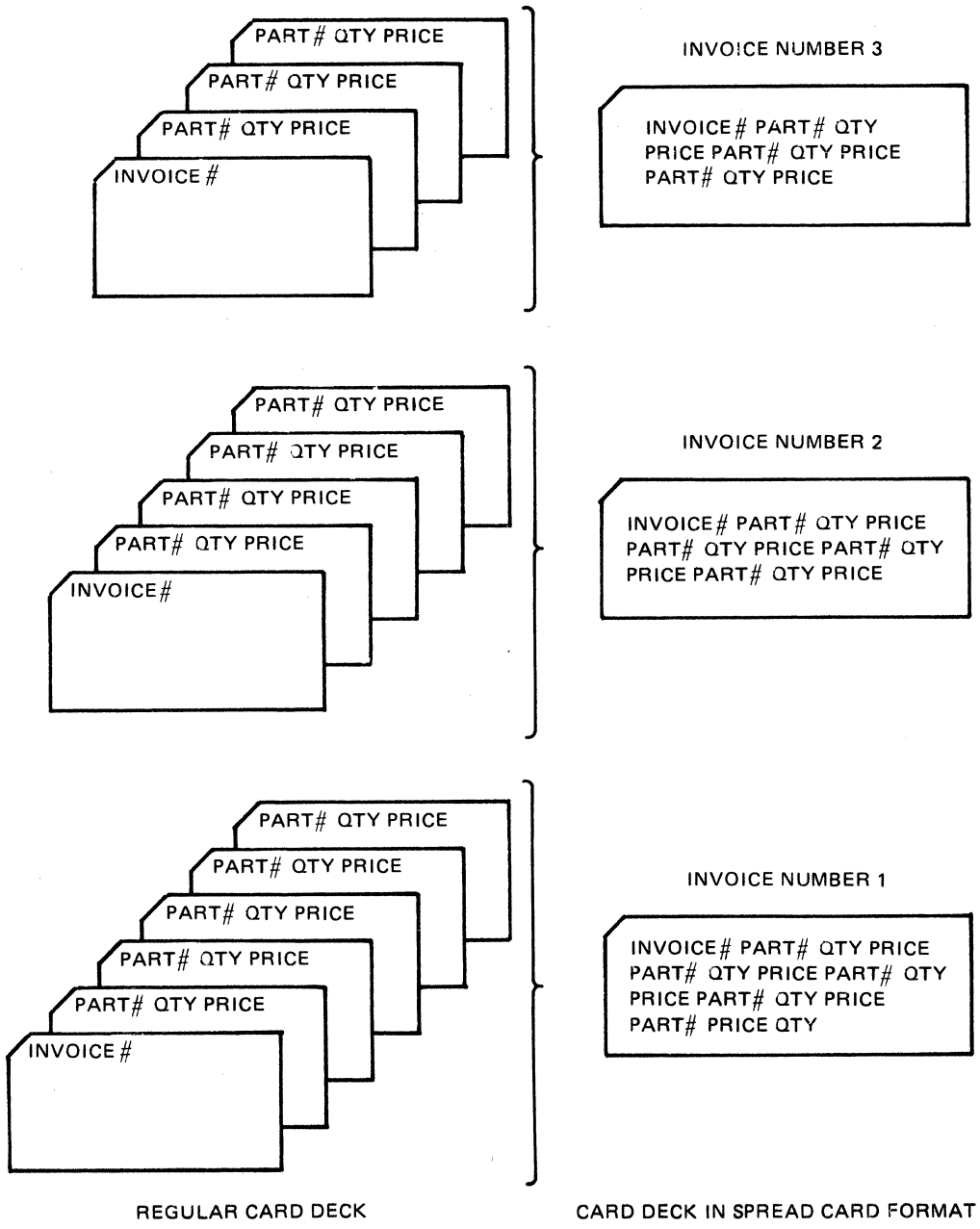


Figure 10-8. Comparison of Regular and Spread Card Data Decks

```
                    Input Specifications

01 ICARDIN  NS  20
02 I                                          1  140INVCNO
03 I              TR
04 I                                         16  200PARTNM
05 I                                         22  240QTY
06 I                                         26  300PRICE
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 10-9. Coding for Spread Cards

Rules for Spread Cards

The following rules must be observed regarding spread cards.

1. Spread cards can be specified for primary or secondary input card files only.
2. Look-ahead fields are not allowed.
3. If sequencing is specified in columns 15-16, the letter N must be entered in column 17.
4. The header field specification is optional, but when specified, is coded as follows:

   1) Describe the header fields on separate specifications lines immediately following the file and record type entries.
   2) Describe only the header fields that are used within the program.

5. Enter "TR" in columns 19-20 to specify that the trailer fields are described on the specification lines that follow. All other entries on the "TR" line must be left blank.
6. It is necessary to describe only the trailer fields that are to be used in the program. However, the fields that specify the beginning and ending position of each trailer portion must be described.
7. All trailer fields must be of the same length and contain the same fields. Therefore, it is necessary to describe only the first trailer portion.
8. Trailer field entries are made in columns 44-58, and all other columns are left blank.

Processing Spread Cards

Spread cards are processed as follows:

1. The header portion of the spread card record and one trailer portion are processed each program cycle and are treated as one logical record.
2. The process described in step 1 continues until:

   1) All trailer portions of a record have been processed, or
   2) A trailer portion of a record is encountered whose fields are all blank.

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

21-41 RECORD IDENTIFICATION CODES

The RECORD IDENTIFICATION CODES field describes 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.

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

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

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

The POSITION fields 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

The NOT fields indicate whether the specified character must be present in the record at the designated position. Valid codes are:

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

26,33,40 C/Z/D

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

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

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

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

1 1 0 0 0 0 0 1

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

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

The zone of the ampersand (&) character is treated similarly to the zone of the characters A through I. The zone of the minus sign (−) character is treated similarly to the zone of the J through R characters The treatment of these characters holds true regardless of the internal codes actually used for the ampersand (&) and minus sign (−) characters. The zone of the ampersand (&) character specifies that the record identification code is the standard positive zone. The zone of the minus sign (−) character specifies that the record identification code is the standard negative zone.

In figure 10-10, only the records of customers whose last names begin with the letters A through I are 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.

```
                    Input Specifications

01 ICUSTFIL AA  12  10 ZA
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

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

In figure 10-11, 5-digit employee numbers are checked to see that all 5 digits are numeric. The zone for all numeric characters is the same.

```
                    Input Specifications

01 IEMPLYFI AA  12   1 Z1   2 Z1   3 Z1
02 I        AND      4 Z1   5 Z1
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 10-11. C/Z/D Coding Example 2

In figure 10-12, only persons whose names start with C, L, or T are processed, since the digit portion of the characters L and T is the same as C.

```
                    Input Specifications

01 IRECFILE AA  12   7 DC
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 10-12. C/Z/D Coding Example 3

NOTE

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

27,34,41 CHARACTER

The CHARACTER field specifies the value to test for when using record-identifying codes.

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

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

Additional Record Identification Codes

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

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

1. Multiple record types have the same fields but the fields are in different positions.
2. Multiple record types have the same field descriptions.
3. A field occurs in one type of record but not in another.
4. Various combinations of 1, 2, and 3.

42 STACKER SELECT

This field indicates the stacker into which the input card is to be placed after being read. Only card input or combined files can be stacker selected. Input files can be stacker selected only in the Input Specifications; combined files can 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, then the Output-Format Stacker Specifications overrides the stacker specified in the Input Specifications. Valid entries for this field are:

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

Card types identified by OR lines can 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 goes to the default stacker. AND lines cannot 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 goes to the default stacker.

This entry must be left blank for input files with multiple I/O areas, otherwise a warning is emitted that the results cannot be those the user intended.

43 PACKED OR BINARY FIELD

This field specifies that a numeric field is in packed decimal format or binary format. Valid entries are:

Entry	Description
Blank	Field is in unpacked decimal format or is alphanumeric.
P	Field is in packed decimal format.
B	Field is in binary format.

When the input field named in columns 53-58 is in packed decimal format, column 43 must contain the letter P. When the input field is in binary format, column 43 must contain the letter B.

Whether in packed decimal or unpacked decimal format, the data can be signed at the most significant or least significant position as specified by means of column 17 of the Control Card or by use of the dollar option RSIGN. The object program automatically converts all numeric data internally during execution with the sign at the most significant position.

UNPACKED DECIMAL FORMAT

Unpacked decimal format means that each byte of storage contains one character. Each byte is divided into a 4-bit zone portion and a 4-bit digit portion. The format for unpacked decimal (left signed) is shown in figure 10-13.

POSITIVE SIGN	0001		1001		0111		0110
ZONE	DIGIT	ZONE	DIGIT	ZONE	DIGIT	ZONE	DIGIT

G18624

Figure 10-13. Unpacked Format for Decimal Number 1976 (Left Signed)

The format for unpacked decimal (right signed) is shown in figure 10-14.

	0001		1001		0111	POSITIVE SIGN	0110
ZONE	DIGIT	ZONE	DIGIT	ZONE	DIGIT	ZONE	DIGIT

G18625

Figure 10-14. Unpacked Format for Decimal Number 1976 (Right Signed)

When processing numeric data in the unpacked format, the zone portion is included for each digit in the number, but only the zone in the right-most (right signed) or left-most (left signed) digit serves as the sign, that is, determines if the number is positive or negative.

Packed Decimal Format

Packed decimal format means that each byte of storage can contain two decimal numbers. The sign is included for the number, but the zone portion is omitted for each digit in the number. Each byte consists of two 4-bit digit positions. Within each number, either the left-most or right-most byte contains a digit and the sign, depending on whether left or right signs are desired.

When describing external data in RPG, the smallest element is a byte or character. Packed data is always character-aligned on external storage media such as magnetic tape or disk.

Unpacked decimal format means that each byte of storage contains one character. Each byte is divided into a 4-bit zone portion and a 4-bit digit portion:

ZONE	DIGIT	ZONE	DIGIT	ZONE	DIGIT	ZONE	DIGIT

Each digit portion holds one digit of the number. On conversion to packed decimal format, the zone portions are dropped, except for the sign position.

Packed data input always causes the field that is to contain the data to be of an odd size, since even-numbered digits are padded with a zero. All digits except the sign digit are considered data.

The following example illustrates the format for packed decimal data:

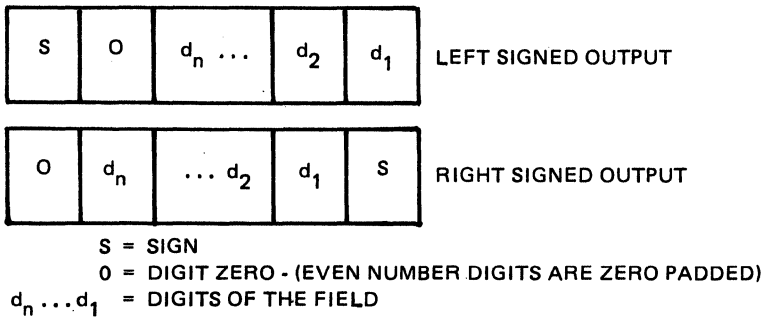


Table 10-4 shows the corresponding packed field length in bytes for unpacked fields of one byte through 15 bytes in length.

Binary Format

Binary format means that two bytes of storage can contain up to four decimal numbers, and that nine decimal numbers can be contained in four bytes of storage.

Rules for Binary Format

The following rules must be observed when binary format is used:

1. Each binary field must be either two bytes or four bytes in length.

Table 10-4. Packed Equivalents for Unpacked Fields

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

2. Binary fields cannot be used as control fields or matching fields.
3. Each two-byte field consists of a 1-bit sign followed by a 15-bit numeric value. The numeric value must be within the range −9,999 and +9,999 inclusive.

4. Each four-byte field consists of a 1-bit sign and a 31-bit numeric value. The numeric value must be within the range −999,999,999 and +999,999,999 inclusive.
5. When a two-byte or four-byte binary field is assigned a value greater than it can store, the result is truncation of the left-most (high order) digits of the decimal number.
6. For binary fields, the left-most bit is the sign bit and is used to indicate whether the number is positive or negative. If the sign bit is a 0 (OFF), the number is positive. If the sign bit is a 1 (ON), the number is negative. Figure 10-15 shows the format of a two-byte binary field.

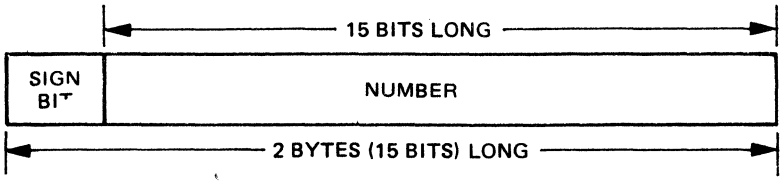


Figure 10-15. Format of Two-Byte Binary Field

7. The decimal value of a binary field can be determined by adding the decimal equivalents of the binary bits that are ON. The sign bit is not included in the addition. Figure 10-16 shows how the decimal equivalents of binary bits are obtained.

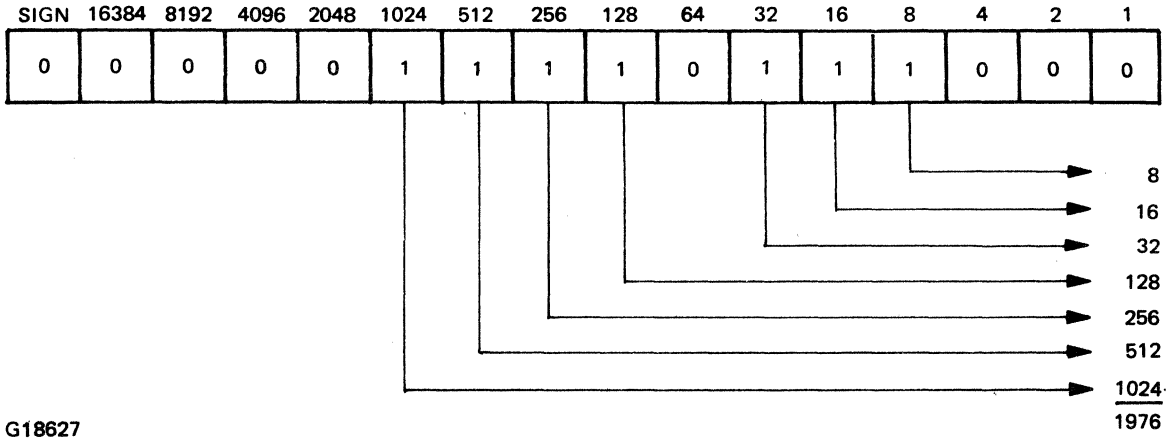


Figure 10-16. Binary Representation of the Decimal Number 1976

44-51 FIELD LOCATION

The FIELD LOCATION field describes the location of data fields within a record and is divided into two subfields that specify the beginning (FROM) and ending (TO) positions of the data field. A field of only one character has the same position number entered in both subfields. Both entries must lie between 1 and the RECORD LENGTH. The TO entry must be greater than or equal to the FROM entry. Entries in the FROM and TO subfields must be right-justified; leading zeros are optional. Figure 10-17 shows example entries in the FIELD LOCATION fields.

```
                                Input Specifications

                                                  | FIELD    |
                                                  |LOCATION  |
                                                  | Fields   |
                                                  |FROM TO   |
01 IFILEIN
02 I                                                1   9OEMPNO
03 I                                               10  23 NAMEL
04 I                                               24  42OHOURS
05 I                                               53  55OCODE
06 I                                               56  76 TITLE
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 10-17. FIELD LOCATION Fields Coding Example

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 is read in from element 1 up to as many elements as fits into the locations specified. The decimal positions must contain the same entry as specified on the Extension Specifications for that array.

52 DECIMAL POSITIONS

The DECMIAL POSITIONS field specifies the number of positions to the right of the implied decimal point in a numeric field. This entry cannot be blank for a numeric field; if the data field contains only integral values, a 0 should be entered to indicate no decimal positions. Valid entries are:

Entry	Definition
Blank	Alphanumeric field.
0-9	Number of decimal positions in a numeric field.

Any field to be used for arithmetic operations or to be edited, must be numeric. The number of decimal positions specified cannot exceed the length of the field (as specified in the FIELD LOCATION field).

If the FIELD NAME entry (columns 53-58) on the Input Specifications specifies an array name, then the decimal positions field must contain the same entry as specified in the decimal positions field of the Extension Specifications for that array.

53-58 FIELD NAME (VARIABLE NAME)

The FIELD NAME field specifies an identifier (name) to an input data field. All fields referenced by the program must be named. Names must be assigned in accordance to the rules for forming field names as described in section 2. A previously defined vector name can 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 can 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 can be used. The OR relationship, illustrated in figure 10-18, shows two record types which have identical fields in the same record positions. Refer to the AND/OR LINES field (columns 14-16) in this section.

```
                Input Specifications

01 IFILENAM AA  21  80 CA
02 I                                          1   5 FIELD1
03 I                                         10  25 FIELD2
04 I                                         41  47 FIELD3
05 I                                         60  69 FIELD4
06 I          BB  31  80 CB
07 I                                          1   5 FIELD1
08 I                                         10  25 FIELD2
09 I                                         41  47 FIELD3
10 I                                         60  69 FIELD4
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

                Input Specifications

01 IFILENAM AA  21  80 CA
02 I        OR  31  80 CB
03 I                                          1   5 FIELD1
04 I                                         10  25 FIELD2
05 I                                         41  47 FIELD3
06 I                                         60  69 FIELD4
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 10-18. OR Relationship – Identical Fields Within Different Record Types

NOTE

The record descriptions above both contain the same fields. The two descriptions can be combined using an OR line as shown in figure 10-18.

Special Words

The following special words are reserved for use as variable names in columns 53-58 of the Input Specifications:

PAGE
PAGEn (where n = 1–8)

If page numbering is desired on output, the special word PAGE or PAGEn for two more printer files indicates that page numbering is to be done. Page field coding is illustrated in figure 10-19.

This feature allows a page number to be entered through the field called PAGE in an input record. The page number printed is 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 can 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 10-19). The special word PAGE can be used in calculations like any other field.

The same PAGE entry can be used for two different output files, but this is not recommended.

Figure 10-19 is an example of PAGE field coding on Input Specifications.

```
                  Input Specifications

01 IINPUT   PG  91    1 CX
02 I                                          2   4OPAGE
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 10-19. PAGE Field Coding

NOTE

The specification lines 01 and 02 describe an input file which contains a record with a PAGE field. The PAGE field is defined as three characters in length with no decimal digits to the right of the decimal point, beginning in position 2 of the input record. The value contained in the PAGE field of the input record is assigned to the PAGE variable when the record is read.

The PAGE field can be used in the Calculation Specifications as any other field. On the Output-Format Specifications, the PAGE field is automatically incremented by 1 before the output record in which the PAGE field is specified is printed.

59-60 CONTROL LEVEL

The CONTROL LEVEL field assigns control level indicators to primary or secondary files. Any field other than a whole array or a look-ahead field can be assigned a control level indicator, in which case it is known as a Control Field. Control fields are checked each program cycle for a change in information; when data in the field changes, a control break occurs. A group of records with the same information in the control field is known as a Control Group. Valid entries for this field are:

Entry	Definition
L1-L9	Control level field assigned.
Blank	No control level field assigned.

A control break occurs when a record containing a control field is read and the information in that control field is different from the information in the same control field of the previous record. When a control break occurs, the designated control level indicator turns ON, along with all control level indicators lower than it. For example, if control level indicator L5 is turned ON, L4, L3, L2, and L1 are also automatically turned ON. Control level indicator L0 is always ON and cannot be assigned to a control field. However, L0 can be used to condition total calculations or total output.

A control level indicator can be turned on or off by SETON or SETOF operation code or can be used as a record identifying indicator. However, in such cases, control level indicators lower than the one specified are not turned on or off automatically.

Control level indicators are used to condition operations, such as:

1. Calculations that must be performed when a control group changes (totals, etc.).
2. Operations that must be performed on the first card of a new control group.
3. Summary punching or total printing that must be performed for each control group.

Control level indicators can be used in input, calculation, and output specifications. The following rules must be observed when assigning control level indicators:

1. The same control level indicator can be used in different record types or files; however, the control fields associated with that indicator must be of the same length. (See figure 10-20).
2. Field names have no effect upon the control level indicator assigned; therefore, control fields in different record types can have both the same name and the same indicator assigned.
3. The maximum size of a control field is 255 characters.
4. Within one record type, control fields can overlap.
5. Numeric control fields are treated as though they had no decimal positions.
6. For numeric control fields, only the digit portion of each character is compared; thus, negative numbers are treated similarly to positive numbers.
7. All control fields with the same control level indicator are considered numeric if any one of those fields is described as numeric.
8. Control levels need not be written in any special sequence, and gaps are permitted in the control levels assigned.
9. Control fields are initialized to binary zeros.
10. 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).
11. 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.
12. Control level indicators cannot be assigned to a binary field.

```
                    Input Specifications

01 IINPDATA ZY  09   2 CQ
02 I                                          70  75 MINOR L1
03 I                                          76  80 NAME
04 I                                          61  70OMAJOR L2
05 I                                          55  65 TOTAL L3
06 I          XX  08   2NCQ
07 I                                          11  20OMAJOR L2
08 I                                          21  30 FILLER
09 I                                          31  36 MINOR L1
10 I                                           5  15 TOTAL L3
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 10-20. Control Fields in Two Record Types

NOTE

The specification lines 01-10 describe two record types, both of which have CONTROL LEVEL fields defined. Even though the CONTROL LEVEL fields are specified in different positions within the input records, the value specified for the LENGTH and DECIMAL POSITIONS fields are the same for each pair. Also note that the MAJOR (L2 indicator) and TOTAL (L3 indicator) fields overlap each other.

Split Control Fields

If the same control level indicator is assigned to more than one field within the same record type, the control field created is known as a Split Control Field. All fields so designated (those having the same control level within the same record type) are combined by the program in the order specified in the Input Specifications and are treated as one control field. Split control fields are illustrated in figure 10-21.

```
                    Input Specifications

01 ISPLITIN BB  04    1 D1
02 I                                          13  16 TYPE01L3
03 I                                          19  25 TYPE03L3
04 I                                          10  12 TYPE02L3
05 I                                          26  18 TYPE10
06 I                                          29  29 TYPE11L5
07 I                                          30  31 TYPE12L5
08 I          CC  05    IND1
09 I                                           2   4 PART01L5
10 I                                           5  10 DEPT01L3
11 I                                          11  18 DEPT02L3
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 10-21. Split Control Fields

NOTE

The specification lines 01-11 describe split control fields in two record types. The L3 control field is split in both record types; however, the total length of all L3 fields is the same. The L5 control field is split in record type 04 and the total length of all L5 control fields in record type 04 is equal to the length of the single L5 control field in record type 05.

The following special rules must be observed for split control fields:

1. The same control level indicator can be used for split control fields in different record types if the field names used are different. The length of various portions of a split control field in one record type can be different than the corresponding portions in another record type; in fact, a control field can be split in one record type but not in another. However, the total length of the control fields (whether split or not) must be the same in both record types. For further information, see FIELD RECORD RELATION field (columns 63-64).
2. If one portion of a split control field is numeric, the entire field is considered numeric.
3. Any one portion of a numeric split control field cannot exceed the maximum size allowed for a numeric field, which is 15 characters. However, the total length of all fields assigned to one control level indicator (within each record type) can be as large as 255 characters.
4. A mixture of packed and unpacked control fields is allowed. For example, consider that three different record types can have:

   1) A control level specified with a field size of seven bytes.
   2) Split control fields of three and four bytes.
   3) A split control field consisting of two bytes containing three packed digits, one byte, and three bytes containing three unpacked digits.

5. No other specification lines can come between lines describing split control fields.

61-62 MATCHING FIELDS

The MATCHING FIELDS field designates matching fields for multifile processing and sequence checking. Valid entries are:

Entry	Definition
Blank	No matching fields and no sequence checking specified.
M1-M9	Matching fields and sequence checking when two or more input, update, or combined files specify the same match fields. Sequence checking when only one input, update, or combined file specifies matching fields.

Matching Fields

Designation of matching fields allows comparison of records from a primary file with records of one or more secondary files to determine if the records match. A maximum of nine different fields can be specified by field designators M1-M9, which are used to specify which fields in each record are to be matched.

Rules for Matching Fields

The MR (matching record) indicator is turned ON when the contents of the primary file match field is the same as the contents of the match field of any secondary file. The matching record indicator is set ON or OFF before detail calculations during the RPG program cycle.

M1-M9 are not indicators; they cause MR to be set ON when the specified records match. MR is used to condition those operations that are to be executed when the records match.

Rules for Matching Fields

The following rules must be observed when assigning matching field values:

1. 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 causes 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.
2. 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.
3. All fields given the same matching field value (M1-M9) must be of the same length.
4. Overlapping of different Match Fields within one record type is allowed; however, the length of any individual Match Field must not exceed 255 characters.
5. All records to be matched must contain the same Match Fields (M1-M9); otherwise, a match cannot be obtained.

6. When more than one Match Field is designated for a record type, all fields specified are combined in order by descending sequence of matching field values and are treated as one contiguous Match Field. The high order field is M9.
7. 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.
8. Numeric Match Fields are treated as though they had no decimal positions.
9. For numeric Match Fields, only the digit portion of each character is compared; thus, negative numbers are treated similarly to positive numbers.
10. All Match Fields with the same matching field value are considered numeric, if any one of the fields is numeric.
11. In order for the matching record indicator to be turned ON, the absolute values of the data in the match fields in the secondary file must equal the absolute values of the data of the match fields in the primary file. Therefore, when more than one matching field value is used for matching records, the contents of all match fields must match before the MR indicator can be turned ON.
12. Field names have no effect upon the matching field value assigned.
13. Whole arrays must not be designated as match fields.
14. Records in primary and secondary files without match fields are processed before records with match fields specified.
15. Match fields for each match field value must have the same length.

Multiple File Processing Using Matching Records

At the beginning of RPG program execution, one record is read into the input buffer of each primary and secondary file. One of these records is selected for processing according to the rules following this paragraph. After the selected record is processed, the next record is automatically read from the same file. During the next program cycle, the current record is compared with the records remaining from the previous cycle in order to select the next record.

Rules for Multiple File Processing Using Matching Records

The following rules apply to multiple file processing using matching records:

1. If the next record from any primary or secondary file has no match fields, then that record is selected.
2. If the records do not match and the records are in ascending sequence, then the record with the lowest match field value is selected. If records are in descending sequence, then the record with the highest match field value is selected.
3. If more than one record satisfies either of the rules described in items 1 and 2, the record from the highest priority file is selected. The primary file is of highest priority, followed by the secondary files in the order they are specified on the File Description Specifications.
4. When a record from the primary file matches a record from the secondary file, the primary file record is processed first.
5. When more than one secondary file is declared, all matching records from a secondary file are processed before control is passed to the next secondary file.
6. Matching records allow the program to enter data from the primary record into the matching secondary record, since the primary record is processed first. Transfer of data from secondary records into matching primary records can be done through the use of look-ahead fields.
7. The MR indicator is ON during the processing of any record containing a match field, providing the current match field value originally occurred in a primary record. Otherwise, the MR indicator in is the OFF state.
8. The MR matching indicator, is turned OFF for one program cycle when a record selected by the FORCE operation is processed. If the next record not selected by the FORCE operation matches the last record with match fields specified, then the MR indicator is turned ON.

9. If the primary file has the letter E in column 17 of the File Description Specifications but the secondary files do not, then any secondary file records matching the last primary record plus any interspersed secondary file records without match fields are processed before the LR indicator is turned ON.

When the coding shown in figure 10-22 is compiled and executed, the two matching fields are combined in the ascending order of the matching field values, M1 and M2. Notice that an end of file (the letter E in column 17 of the File Description Specification) is specified for the secondary file. This causes the job to end after the last record in the file, TIMECDS, is read.

Processing of matching fields proceeds as follows:

1. When a record from the primary file matches a record from the secondary file, the primary file record is processed first.
2. When records do not match, the record with the lowest (ascending files) or highest (descending files) Match Field value is processed first.
3. A record type which has no matching field specifications is processed immediately after the record it follows, and the MR indicator is not turned ON. If such a record is the first one in a file, it is processed first (even if it is not in the primary file).
4. Matching records allow the program to enter data from the primary record into the matching secondary record, since the primary record is processed first. Transfer of data from secondary records into matching primary records can be done through the use of look-ahead fields.
5. When additional secondary files are declared, all matching records are processed in one secondary file before passing control to the next secondary file. The precedence of the secondary files is determined by their order of appearance on the File Description Specifications.

In figure 10-22, the two matching fields are combined in the ascending order of the matching field values, M1 and M2. Notice that an end-of-file detection (the letter E in column 17 of the File Description Specifications) has been specified for the secondary file. This causes the job to end after the last record in the file, TIMECDS, has been read.

```
                    File Description Specifications

01 FEMPLYCR IP A   96  96            MFCU1
02 FTIMECDS CSEA   96  96            MFCU2
03 FPAYROLL O      96  96            PRINTER
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


                      Input Specifications

01 IEMPLYCR      10   1 CN   2 CA
02 I                                          3  15 NAME
03 I                                         16  18ODEPTNO  M2
04 I                                         19  22OEMPLNO  M1
05 I                                         23  292PAYRAT
06 ITIMECDS BB   20   1 CR   2 CC
07 I                                          3  15 NAME
08 I                                         16  18ODEPTNO  M2
09 I                                         19  22OEMPLNO  M1
10 I                                         40  462HRSWRK
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 10-22. Coding Matching Fields

The example shown in figure 10-23 illustrates the order in which two matching fields are processed. An end-of-file detection has been specified for the primary file.

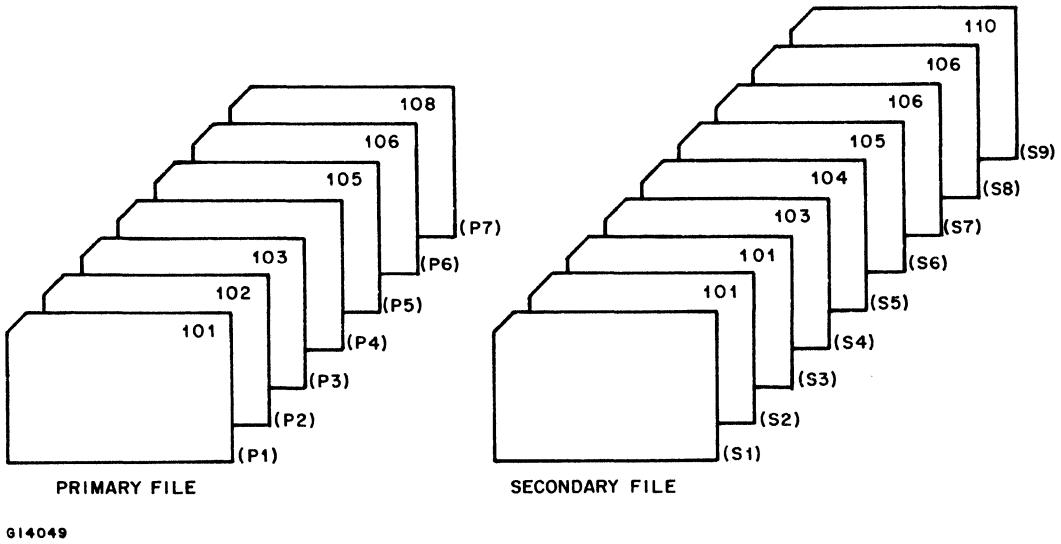


Figure 10-23. Record Selection From Two Matching Files

The files (figure 10-23) are processed in the following order:

S1 Records with no matching fields are processed before records with matching fields regardless of file type.

P1, S2, S3 The MR indicator is ON. All matching secondary records are processed after the primary record.

P2

P3 The MR indicator is ON.

P4 Records with no matching fields are processed first.

S4 The MR indicator is ON.

S5 When no records match, the record with the lowest sequence number is processed, regardless of file type.

P5, S6 The MR indicator is ON.

P6, S7, S8 The MR indicator is ON.

P7 Last record processed.

S9 When an end-of-file detection is specified for a primary file, secondary records are not processed unless the MR indicator is turned ON.

The example shown in figure 10-24 illustrates the order in which records with three matching fields are processed.

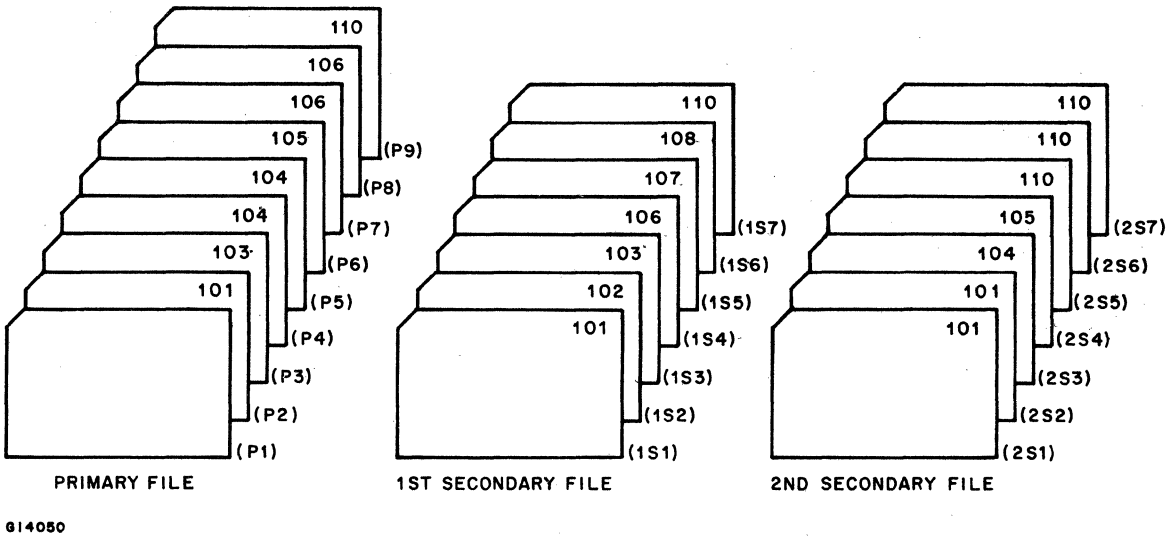


Figure 10-24. Record Selection From Three Matching Files

The files (figure 10-24) are processed in the following order:

P1

P2
1S1
2S1
2S2

The MR indicator is ON since all Match Fields match.

1S2

When records do not match, the lowest matching field is processed next.

P3
1S3
P4
P5
2S3
P6
2S4
P7
P8
1S4
1S5
1S6

P9
1S7
2S5
2S6
2S7

The MR indicator is ON. 2S7 is the last record processed.

Sequence Checking

Sequence checking of records within a file is performed when even matching field designators (M1-M9) are assigned. The sequence checking is performed in conjunction with matching records if two or more input, update, or combined files have match field designators specified.

If only one input, update, or combined file has an entry in columns 61-62, then sequence checking only is performed on the data in the fields to which M1-M9 have been assigned. A maximum of nine fields (M1-M9) within the record can be selected for sequence checking. When a record is encountered which has a field or fields out of sequence, the program halts. The system operator can cause program operation to be resumed, in which case the record in error is ignored and the next record is read from the same file.

The following rules must be observed when assigning matching field designators for sequence checking:

1. All fields designated for sequence checking must be in the same order, either ascending or descending.
2. When more than one field is designated for sequence checking, all fields specified are combined in order by ascending sequence of matching field designators (M1-M9) and are treated as one contiguous field.
3. Split sequence fields are not allowed; thus, the same matching field designation should not be used more than once in the record, unless field record relation indicators are used.
4. Numeric fields are treated as though they had no decimal positions.
5. For numeric fields, only the digit portion of each character is compared; thus, negative numbers are treated the same as positive numbers.
6. All sequence fields are considered numeric if any one of the fields is numeric.

The example shown in figure 10-25 illustrates coding procedures for sequence checking.

```
                Input Specifications

01 IVOTERLS NS  01  80 CX
02 I                                          4   8 STATE   M4
03 I                                         10  15 COUNTY  M3
04 I                                         20  30 CITY    M2
05 I                                         33  35 PRECNT  M1
06 I                                         40  44 PARTY
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 10-25. Coding for Sequence Checking

NOTE

In the specifications lines 01-06, the first four fields of each record are sequence checked, but the last field is not.

63-64 FIELD RECORD RELATION

The FIELD RECORD RELATION field permits data to be made available for processing only if the conditions specified in the FIELD RECORD RELATION fields are satisfied. For example, if the indicator is ON, the data is made available just prior to detail calculations. If the indicator is OFF, the field remains in its previous condition.

The following rules must be observed when assigning FIELD RECORD RELATION indicators.

1. A field record relation indicator need not necessarily be the same as any record identifying indicator specified for the file.
2. Fields within one record type which specify the same field record relation indicator can be entered in any order; however, the most efficient data storage is obtained when they are written as a group on specification lines following one another.
3. All portions of a split control field must be assigned the same field record relation indicator and must be written as a group on specification lines following one another.

If field record relation indicators are not to be assigned, this field should be left blank (see figure 10-26). The acceptable entries are as follows:

Entry	Definition
Blank	No field record relations.
01-99	Record identifying indicator assigned to a record type.
L1-L9	Control level indicator defined elsewhere.
MR	Matching record indicator.
U1-U8	External indicator defined elsewhere.
H0-H9	Halt indicator defined elsewhere.

Each of the entries is discussed in the following paragraphs.

Figure 10-26 shows the use of the FIELD RECORD RELATION fields on the Input Specifications.

```
                    Input Specifications

01 ICARDIN  ZZ  01  80 CA
02 I        OR  02  80 CB
03 I        OR  03  80 CC
04 I        OR  04  80 CD
05 I                                    1   51FIELD0L2
06 I                                   71  722FIELD1L2
07 I                                   61  64 FIELD2L3  02
08 I                                   65  70 FIELD3L4  02
09 I                                    7  12 FIELD4L4  03
10 I                                   13  22 FIELD5L4  03
11 I                                   41  514FIELD6    03
12 I                                   10  13 FIELD7L4  04
13 I                                   14  20 FIELD8L4  04
14 I                                   21  25 FIELD9L4  04
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 10-26. Field Record Relations – Using the OR Relationship

01-99 Record Identifying Indicators

When several record types have been defined in an OR relationship, all fields defined apply to all record types. In many cases, however, not all of the record types defined have exactly the same fields. The FIELD RECORD RELATION field allows the programmer to specify that some fields apply only to certain record types and not to others. If the FIELD RECORD RELATION field is left blank, the associated field applies to all record types to which it is subordinate. However, by placing the same entry found in the Record Identifying Indicator field of one record type in the FIELD RECORD RELATION field, the field is identified as applying only to the corresponding record type.

Control Fields and Match Fields can also be related to a particular record type in an OR relationship by a FIELD RECORD RELATION entry.

When two or more Control Fields or two or more Match Fields have the same control level indicator or matching field value, respectively, only one of these cannot have a field record relation indicator assigned. This applies to a group of specifications, if it is a split control field specification. Thus, specifications with field record relation indicators are used if that indicator is ON. When none of the field record relation indicators are ON, only the specification without any field record relation indicator is used.

L1-L9, MR Control Level or Matching Record Indicator

The use of control level or matching record indicators signifies that the data from the field is to be made available only if a control level break or matching record condition has occurred on this record.

The following restrictions apply:

1. A control level indicator must not be used on an input specification line to show the field record relation of a field that is designated in columns 59-60 of the same line as a part of or as the control group.
2. The matching record indicator (MR) must not be used on an input specification line to show the field record relation of a field that is designated in columns 61-62 of the same line as a part of, or all of, the matching record input control group.

To overcome these restrictions, the fields have to be defined twice, once to specify the control and/or matching fields, and again on another specification line (but with a different Variable Name) to specify the field record relations desired.

External Indicators

External indicators are used primarily to condition files in the File Description Specifications (see columns 71-72, FILE CONDITION). However, they can also be used to condition fields even though file conditioning is not specified. Data from the associated field is accepted only when the specified indicator is ON.

H0-H9 Halt Indicators

The halt indicators are used to show a field/record relationship and specify that the data in the field is to be made available only when the given halt indicator is ON. This would usually be used with a record specified with the halt indicator in columns 19-20, RECORD IDENTIFYING INDICATOR to load a signal pertaining to the reason or conditions of the halt operation.

65-70 FIELD INDICATORS

The FIELD INDICATOR field is composed of three subfields that allow the data of the input field to be tested as follows:

1. If the input field is defined as numeric, the data is tested for a positive, negative, or zero condition.
2. If the input field is defined as alphanumeric, the data is tested for greater than blank, less than blank, or blank, according to the EBCDIC collating sequence.

Valid entries for this field are:

Entry	Definition
Blank	No field indicators used.
01-99	Field indicator.
H0-H9	Halt indicator.

The FIELD INDICATOR field is subdivided as described in the paragraphs that follow.

65-66 PLUS

Any valid indicator specified in this field turns ON if the corresponding data field is greater than zero (numeric field only) or greater than blank (alphanumeric field).

MINUS

Any valid indicator specified in this field turns ON if the corresponding data field is less than zero (numeric field only) or less than blank (alphanumeric field).

69-70 ZERO OR BLANK

Any valid indicator specified in this field turns ON if the corresponding data field is zero (numeric field only) or blank (alphanumeric field).

Rules for Assigning Indicators

The following rules must be observed when assigning and using field indicators:

1. All field indicators are OFF at the beginning of the program and remain OFF until the condition being tested is satisfied on the input record just read.

   If the condition being tested is not satisfied, the indicator is turned OFF.
2. A field can be assigned more than one indicator; however, only the indicator specified for the condition with a true result are turned ON. All other indicators assigned to the field are turned OFF.
3. The state of a field indicator assigned to fields in different record types is always determined by the last record selected.
4. 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.
5. When different field indicators are assigned to fields in different record types, a field indicator remains ON (or OFF) until another record of the same type is selected.

6. If a halt indicator specified in the FIELD INDICATOR field is turned ON as a result of the corresponding condition being true, the program halts after the input record which caused it to turn ON has been completely processed.
7. Field indicators cannot be assigned to whole arrays.

75-80 PROGRAM IDENTIFICATION

Refer to section 2 for a complete description.

SECTION 11
CALCULATION SPECIFICATIONS AND OPERATION CODES

Calculation Specifications indicate the sequence and timing of the data manipulation that belongs to the following two broad categories:

1. Arithmetic, logical, and manipulative operations on defined data elements (fields), and
2. Input/output control of some kinds of files.

The Calculation Specifications describe the operations to be performed on the data and specify the order in which the operations are to be performed. Calculation Specifications are of three types, which are optional, but if they occur, must appear in the following order:

1. Detail calculations.
2. Total calculations.
3. Subroutines.

Subroutines must not be the only Calculation Specifications entries because subroutines can only be accessed from detail or total calculations.

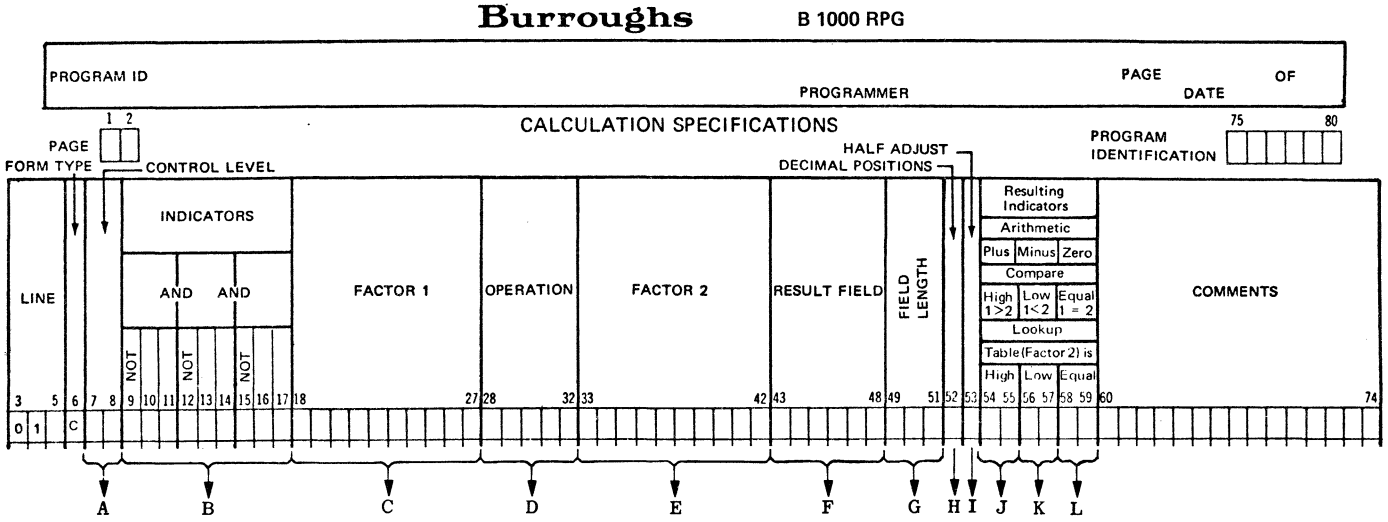
Within each grouping, operations are performed in the order they are written.

Each specification line describes one operation and is divided into three functional parts:

1. Conditions under which the operation is to be performed (columns 7-17).
2. The kind of operation to be performed and the data which is to be operated upon (columns 18-53).
3. Tests to be made upon the results of the operation (columns 54-59).

FIELD DEFINITIONS

Figure 11-1 can be used in conjunction with the following field definitions for the Calculation Specification.



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, 0V 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, 0V 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, 0V or blank.

G14052

Figure 11-1. Calculation Specifications Summary Sheet

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

The letter C must appear in this field.

7-8 CONTROL LEVEL

The valid entries for this field are listed below, and each entry is described in the paragraphs that follow.

Entry	Definition
Blank	Calculation operation is part of detail calculations or can be part of a subroutine.
L0-L9	Calculation operation is performed in total calculations when the designated control break occurs or the specified indicator is set ON (LO is always ON).
LR	Calculation operation is performed after the last record has been processed (or if set ON).
SR	Calculation operation is part of a subroutine (documentational, not required for subroutines).
AN,OR	Establishes AND and OR relationship between lines of indicators.

An entry of L0-L9 or LR indicates that the operation is performed during total calculations when the specified indicator is ON. The first entry of L0-L9 or LR in columns 7-8 determines the start of total calculations. All detail calculations (if any are specified) must precede any total calculations. Any operations with blanks in columns 7-8 occurring before any total operation, or any BEGSR operation, are treated as detail operations and are performed during detail calculations in every cycle depending upon the conditions specified in columns 9-17.

Subroutines, if used, must be specified after all detail and total lines. For a subroutine, column 7-8 can contain SR, OR, AN, or blank. The first BEGSR operation determines the start of subroutines.

L0-L9

Control level indicators L0-L9 are used to condition operations that are to occur at a control break. If a control level indicator is specified in columns 7-8, the operation described on the same specification line is done only when the designated indicator is ON. When a control break for a certain level occurs, all lower control level indicators also turn ON. However, when a control level indicator used as a record identifying indicator turns ON to indicate a specific record type, or when a control level indicator is turned ON by a SETON operation, all lower level indicators remain OFF. Control level indicators need not be specified in any particular order. Operations are executed in the order in which they are specified, providing the control level indicator, columns 7-8, is ON.

The LO indicator is turned ON during every cycle of the program after detail output. If no other control level indicators are assigned, but it is desired to perform total calculation and total output operations, the LO indicator can be used to condition those operations. LO indicator coding is illustrated in figure 11-2.

```
                Calculation Specifications

01 C    40       WAGE       ADD  TOTWAG     TOTWAG   82
02 C    50       WAGE       ADD  TOTWAG     TOTWAG
03 C                        SETOF                     70
04 C    40                  SETON                     70
05 CLO  50 70    TOTWAG     ADD  TOTAL      TOTAL    82
06 CLR           TOTWAG     ADD  TOTAL      TOTAL
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 11-2. LO Indicator Coding Example

NOTE

The L0 indicator can be used to condition detail operations so that they are performed at total calculation time (ahead of normal detail calculations).

LR Indicator

The LR indicator automatically turns ON when the last record of the appropriate file has been read and processed. (All control level indicators L1-L9 also turn ON). This indicator is used to condition those operations that are to be performed at end of file. The LR indicator can also be turned ON by its specification as a calculation resulting indicator. In that case, if the LR indicator is turned ON, the lower control levels are not set ON as a result. When the LR indicator is used as a resulting indicator in Calculation Specifications, the operation should be conditioned by NLR to avoid the inadvertent resetting of the LR indicator.

SR Entry

The SR entry is not an indicator; rather, it is used to indicate that the specification on the same line is part of a subroutine. All subroutine lines must be specified after all other calculation lines. Subroutines must be specified after all detail and total lines are specified. For a subroutine specification line, columns 7-8 can contain SR, OR, AN or blank. The BEGSR operation determines the start of a subroutine.

AN-OR Entrys

This field can also 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 can 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 can contain an L0-L9, LR, or SR entry if the entire group is conditioned by a control level indicator or is part of a subroutine. Each AND/OR line, and the previous line, must contain at least one indicator in columns 9-17.

```
                Calculation Specifications

01 C   02 03 04
02 CAN 05 06   WAGE       ADD  TOTAL     TOTAL   82
03 C   10
04 COR 13
05 COR 14
06 COR 15      WAGE       ADD  TOTAL     TOTAL   82
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 11-3. AND/OR Relationship Coding Example

NOTE

The AN-OR lines specify the conditions under which the add operations are to be performed. These lines extend the conditions which can be specified by the three parts of the INDICATORS field.

9-17 INDICATORS

This field is divided into three subfields such that up to three indicators on each line can be specified to condition a calculation operation. Each subfield is divided into two parts, as follows:

1. NOT (one column).
2. INDICATOR (two columns).

The NOT portion specifies that the associated indicator must be OFF in order for the operation to occur. If this condition is desired, an N must be entered in the NOT position; otherwise, the NOT portion must be left blank.

The INDICATOR portion specifies the indicator to be tested for ON (NOT = blank) or OFF (NOT = N). Only the following entries are allowed in this portion of the INDICATOR field:

Entry	Definition
Blank	Operation not conditioned by any indicator.
01-99	Operation conditioned by indicator used elsewhere in the program.
L0-L9	Operation conditioned by control level indicator previously assigned.
LR	Operation conditioned by last record indicator.
MR	Operation conditioned by matching record indicator.
H0-H9	Operation conditioned by halt indicator used elsewhere in the program.

Entry	Definition
U1-U8	Operation conditioned by external indicator previously defined.
OA-OG, OV	Operation conditioned by overflow indicator previously defined.

All three indicators on one line are in an AND relationship. All indicators on one line (or grouped lines), plus the Control Level Indicator (if used) must be ON or OFF as specified, in order for the associated operation to take place.

Each of the indicators is discussed in the paragraphs that follow.

Indicators 01-99

1. To condition calculation operations to be performed only for specific input record types (record identifying indicators).
2. To condition calculation operations to be performed only when an input field satisfies certain conditions (field indicators).
3. To condition calculation operations according to the results from previous operations specified in the Calculation Specifications (resulting indicators).

Control Level Indicators (L0-L9)

The control level indicators are normally defined in the CONTROL LEVEL field of the Input Specifications but can be used as calculation resulting indicators. For either use, the control level indicators entered in columns 9-17 signify an operation to be performed only during the first cycle and after a control break.

The following information is applicable when control level indicators are specified in the INDICATORS field:

1. If the operation is performed during detail calculations, then it occurs only while processing the first record of the new group.
2. If the operation is performed during total calculations, then it occurs only when the control break specified in columns 7-8 and the control break specified in the INDICATORS field are both satisfied. When both control breaks are satisfied, the last record of a control group has been processed.
3. If the operation is part of a subroutine and is called from detail calculations, then it occurs only while processing the first record of the new group.
4. If the operation is part of a subroutine and is called from total calculations, the operation occurs only if the EXSR is of an equal or higher control level than the control level indicator specified in the INDICATORS field.

Last Record Indicator (LR)

The last record indicator is used to condition operations that are to be performed at end of job.

Matching Record Indicator (MR)

The matching record indicator is used to indicate whether an operation is to be performed only relative to the status of the matched records found.

The following rules must be observed when using the MR indicator:

1. When used during detail calculations, MR refers to the status of the match fields of the record last selected.
2. When used during total calculations, MR refers to the status of the match fields of the record prior to the last record selected.
3. When used in a subroutine that is called during detail calculations, MR refers to the status of the match fields of the record last selected. When used in a subroutine that is called during total calculations, MR refers to the status of the match fields of the record prior to the last record selected.

Halt Indicators (H0-H9)

Halt indicators previously assigned in the Field Indicators field of the Input Specifications or Resulting Indicators field of the Calculation Specifications can be used to condition operations that are to be performed only when an error condition occurs.

Since the program does not halt until the record in error has been completely processed, action must be taken to prevent reporting the erroneous results. By using the halt indicators in conjunction with an N entry in the NOT portion of the INDICATORS field, an operation can be inhibited when the specified halt indicator is ON.

External Indicators (U1-U8)

External indicators condition operations that are to be performed only when the specified indicator has been set at program execution time. Refer to the description of the FILE CONDITION field in the section on File Description Specifications for a complete description of external indicators.

Overflow Indicators (OA-OG, OV)

These indicators, previously assigned in the File Description Specifications, condition operations that are to be performed when the overflow line of a printer file has been reached.

18-27 FACTOR 1

This field supplies data to be operated on by the Operation Code specified in columns 28-32.

Allowable entries are:

1. The name of any field previously defined or to be defined later.
2. A literal (alphanumeric or numeric).
3. A label (tag name, subroutine name, END subroutine name).
4. A vector name or an element of a vector. A vector element is composed of the vector name, followed by a comma, followed by the desired index value.
5. The special words UDATE, UMONTH, UDAY, UYEAR, PAGE, and PAGEn.

Entries, including numeric literals, in this field must be left-justified.

28-32 OPERATION

This field specifies the proper operation to be performed using FACTOR 1 and FACTOR 2. The operation code must be left-justified in the field.

Operations are performed in the order in which they appear on the Calculation Specifications sheet; however, all operations conditioned by control level indicators specified in the CONTROL LEVEL field must appear after those operations not conditioned by control level indicators and prior to all subroutines.

Each of the operation codes is discussed in detail in the second portion of this section of the manual.

33-42 FACTOR 2

This field supplies data to be operated on by the Operation Code specified in columns 28-32. Allowable entries are:

1. The name of any field defined elsewhere in the program.
2. A literal (alphanumeric or numeric).
3. The name of a subroutine (EXSR operation only).
4. A vector name or an element of a vector. A vector element is composed of the vector name, followed by a comma, followed by the desired index value.
5. The special words UDATE, UMONTH, UDAY, UYEAR, PAGE, and PAGEn.
6. A label (GOTO operation only).
7. A filename (DEBUG, DSPLY, CHAIN, READ, RECV, SEND or FORCE operation only).

Entries in this field must be left-justified.

43-48 RESULT FIELD

The use of this field is dependent upon the particular operation being performed. The entry in this field is used to name the field, vector, or vector element that is to be used to store the results of the operation specified on this specification line.

All entries in the RESULT FIELD field must be left-justified. The entry can be the name of a vector or vector element defined elsewhere in the program, or the name of a field described elsewhere in the program or described on this specification line. Special words, except those in the following list, can also be entered in the RESULT FIELD field.

The following special words must never be used in the RESULT FIELD field.

1. UDATE
2. UMONTH
3. UDAY
4. UYEAR

Also, literals must never be specified in the RESULT FIELD field.

Definition of a new field is also allowed and is accomplished by entering the new field name, along with FIELD LENGTH and DECIMAL POSITIONS entries, for that field. Refer to section 2 for a complete description of field names.

49-51 FIELD LENGTH

The FIELD LENGTH field specifies the length of an entry specified in the RESULT FIELD field that has not been previously defined. Numeric and alphanumeric fields are limited to the size specified in section 2.

If this entry is left blank, the field must be defined elsewhere. This field must be blank if the RESULT FIELD field is blank. It is allowable to enter the length of a field that has been previously defined; however, the length and number of decimal positions specified must be the same as that previously defined.

Entries in this field must be right-justified, and leading zeros are optional.

52 DECIMAL POSITIONS

This field specifies the number of positions to the right of the implied decimal point in a numeric result field. If the RESULT FIELD field is alphanumeric or the FIELD LENGTH field is blank,then this field must be left blank. The valid entries are:

Entry	Definition
Blank	Alphanumeric field.
0-9	Number of decimal positions in a numeric field.

When defining numeric fields, the number of decimal positions must be the number of digits in the numeric field; if the numeric field contains only integral values, then the number 0 must be entered to indicate there are no digits to the right of the decimal point (.) character.

The number of decimal positions specified must not exceed the length of the RESULT FIELD, as specified by the FIELD LENGTH entry.

53 HALF ADJUST

This field specifies whether or not the contents of the RESULT FIELD 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 field. Then all digits to the right of the last decimal position are dropped. Valid entries for this field are:

Entry	Definition
Blank	Do not half adjust.
H	Half adjust.

Entries in this field are allowable only for certain arithmetic operations.

54-59 RESULTING INDICATORS

The indicators U1-U8, MR, or IP must not be entered in this field. Any other RPG indicator or blanks are valid entries.

Valid entries for this field are:

Entry	Definition
01-99	Any numeric indicator.
L1-L9	Any control level indicator.
LR	Last record indicator.
H0-H9	Any halt indicator.
OA-OG,OV	Any overflow indicator.

Indicators specified in these columns are set OFF immediately before the operation is performed. Immediately after the operation, indicators are set ON to indicate the result of the operations. The use of the result indicators is dependent on the type of operation specified.

Resulting indicators must not be specified when the RESULT FIELD field contains an array name, the only exception being a LOKUP operation.

Figure 11-4 illustrates the use of resulting indicators and various operation codes. The specific uses of the result fields are further described in the discussion on the various operation code types in this section.

```
          Calculation Specifications

01 C* ARITHMETIC OPERATIONS:
02 C   01      FIELD1    SUB  FIELD2    FIELD3       22
03 C* IF FIELD3 BECOMES NEGATIVE, THEN INDICATOR 22 IS TURNED ON.
04 C*
05 C* COMPARE OPERATIONS:
06 C           FIELD2    COMP 124                  232425
07 C* IF FIELD2 IS GREATER THAN 124, THEN INDICATOR 23 IS TURNED ON.
08 C* IF FIELD2 IS LESS THAN 124, THEN INDICATOR 24 IS TURNED ON.
09 C* IF FIELD2 IS EQUAL TO 124, THEN INDICATOR 25 IS TURNED ON.
10 C*
11 C* LOOKUP OPERATIONS:
12 C           ARGUMENT  LOKUPTABLES                 89
13 C* IF THE VALUE CONTAINED IN THE ARGUMENT FIELD IS FOUND IN
14 C* TABLES, THEN INDICATOR 89 IS TURNED ON.
15 C*
16 C* CHAIN OPERATIONS:
17 C           KEY       CHAINDISKIN               99
18 C* IF THE DESIGNATED RECORD IS NOT FOUND, THEN INDICATOR 99 IS
19 C* TURNED ON.
20 C*
21 C* SETTING INDICATORS:
22 C                     SETON                     5152
23 C                     SETOF                     535455
24 C* INDICATORS 51 AND 52 ARE TURNED ON AND INDICATORS 53, 54,
25 C* AND 55 ARE TURNED OFF.
26 C*
27 C* READ OPERATIONS:
28 C                     READ FILEIN                  H8
29 C* IF THE END-OF-FILE RECORD IS READ, THE H8 INDICATOR IS
30 C* TURNED ON.
31 C*
32 C* TEST ZONE OPERATIONS:
33 C                     TESTZ          ALPHA      010203
34 C* IF THE LEFTMOST CHARACTER OF THE FIELD ALPHA IS A-I OR 8,
35 C* THEN THE 01 INDICATOR IS TURNED ON AND INDICATORS 02 AND 03
36 C+ ARE TURNED OFF.
37 C* IF THE LEFTMOST CHARACTER OF THE FIELD ALPHA IS J-R, OR -,
38 C* THEN THE 02 INDICATOR IS TURNED ON AND INDICATORS 01 AND 03
39 C* ARE TURNED OFF.
40 C* IF THE LEFTMOST CHARACTER OF THE FIELD ALPHA IS ANY OTHER
41 C* CHARACTER, THE INDICATOR 03 IS TURNED ON AND INDICATORS
42 C* 01 AND 02 ARE TURNED OFF.
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 11-4. Resulting Indicators Coding Example

60-74 COMMENTS

This field is available for inclusion of comments and documentary remarks and can 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 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, logical, or input/output operation is to be performed on the associated operands.

Table 11-1 summarizes each of the operation codes discussed in the following paragraphs.

Table 11-1. Summary of Operation Codes

Operation Code and Definitions	Control Level Indicators	Factor 1	Factor 2	Result Field	Field Length	Decimal Positions	Halfadjust	Resulting Indicators
Arithmetic Operations:								
ADD Adds the contents of the FACTOR 1 field to the contents of the FACTOR 2 field and stores in the contents of the RESULT FIELD field.	0	R	R	R	0	0	0	0
DIV Divides the contents of the FACTOR 2 field by the contents of the FACTOR 1 field and stores in the contents of the RESULT FIELD field.	0	R	R	R	0	0	0	0
MULT Multiplies the contents of the FACTOR 1 field by the contents of the FACTOR 2 field and stores in the contents of the RESULT FIELD field.	0	R	R	R	0	0	0	0
MVR Moves the remainder of the preceding divide operation to the contents of the RESULT FIELD field.	0	B	B	R	0	0	B	0
SQRT Derives the square root of the contents of the FACTOR 2 field and stores in the contents of the RESULT FIELD field.	0	B	R	R	0	0	B	B

Table 11-1. Summary of Operation Codes (Cont)

Operation Code and Definitions	Control Level Indicators	Factor 1	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust	Resulting Indicators
SUB Subtracts the contents of the FACTOR 1 field by the contents of the FACTOR 2 field and stores in the contents of the RESULT FIELD field.	0	R	R	R	0	0	0	0
XFOOT Sums the elements of an array and stores in the contents of the RESULT FIELD field.	0	B	R	R	0	0	0	0
Z-ADD Clears the contents of the RESULT FIELD field and adds the contents of the FACTOR 2 field to the RESULT FIELD field.	0	B	R	R	0	0	0	0
Z-SUB Clears the contents of the RESULT FIELD field and subtracts the contents of the FACTOR 2 field from the contents of the RESULT FIELD field.	0	B	R	R	0	0	0	0
Move Operations:								
MOVE Moves the contents of the FACTOR 2 field to the contents of the RESULT FIELD field, right-justified.	0	B	R	R	0	0	B	0
MOVEA Moves data (left-justified) into, or out of an alphanumeric array.	0	B	R	R	B	B	B	B
MOVEL Moves the contents of the FACTOR 2 field to the contents of the RESULT FIELD field, left-justified.	0	B	R	R	0	0	B	0

Table 11-1. Summary of Operation Codes (Cont)

Operation Code and Definitions	Control	Level	Indicators	Factor 1	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust	Resulting Indicators
Move Zone Operations:										
MHHZO Moves the high-order (left-most) zone of the FACTOR 2 field to the high-order (left-most) zone of the RESULT FIELD field. The FACTOR 2 and RESULT FIELDS fields must be alphanumeric.	0	0	B	R	R	0	B	B	B
MHLZO Moves the high-order (left-most) zone of the FACTOR 2 field to the low-order (right-most) zone of the RESULT FIELD field. The FACTOR 2 field must be alphanumeric.	0	0	B	R	R	0	0	B	B
MLHZO Moves the low-order (right-most) zone of the FACTOR 2 field to the high-order (left-most) zone of the RESULT FIELD field. The RESULT FIELD field must be alphanumeric.	0	0	B	R	R	0	B	B	B
MLLZO Moves the low-order (right-most) zone of the FACTOR 2 field to the low-order (right-most) zone of the RESULT FIELD field.	0	0	B	R	R	0	0	B	B
Compare Operations:										
COMP Compares the contents of the FACTOR 1 field to the contents of the FACTOR 2 field.	0	0	R	R	B	B	B	B	R
TESTZ Tests the value of the zone (right-most character for alphanumeric fields and the left-most position of numeric fields) of the RESULT FIELD field.	0	0	B	B	R	0	B	B	R
TESTN Tests the contents of the RESULT FIELD field for numeric characters.	0	0	B	B	R	0	B	B	R

Table 11-1. Summary of Operation Codes (Cont)

Operation Code and Definitions	Control Level	Indicators	Factor 1	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust	Resulting Indicators
Binary Field Operations:									
BITON Sets the bits to ON for the contents of the RESULT FIELD field as indicated by the bit locations specified in the FACTOR 2 field.	0	0	B	R	R	0	B	B	B
BITOF Sets the bits to OFF for the contents of the RESULT FIELD field as indicated by the bit locations specified in the FACTOR 2 field.	0	0	B	R	R	0	B	B	B
TESTB Tests the bits indicated by the FACTOR 2 field for bits that are ON in the contents of the RESULT FIELD field.	0	0	B	R	R	0	B	B	B
Setting Indicators:									
SETOF Sets to OFF the indicators specified in the RESULTING INDICATIONS field.	0	0	B	B	B	B	B	B	R
SETON Sets to ON the indicators specified in the RESULTING INDICATIONS field.	0	0	B	B	B	B	B	B	R
Program Branching Operations:									
GOTO Branches to label specified in the FACTOR 2 field.	0	0	B	R	B	B	B	B	B
TAG Identifies the name in the FACTOR 1 field as the destination of a GOTO	0	B	R	B	B	B	B	B	B
Transfer of Control Operation:									
ZIP Initiates a system command.	0	0	B	R	B	B	B	B	B
Lookup Operations:									
LOKUP Searches the table or array specified in the FACTOR 2 field for the contents specified in the FACTOR 1 field and turns ON the approriate indicators in the RESULTING INDICATORS field.	0	0	R	R	0	0	0	B	R

Table 11-1. Summary of Operation Codes (Cont)

Operation Code and Definitions	Control Level	Indicators	Factor 1	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust	Resulting Indicators
Subroutine Operations:									
BEGSR Identifies the FACTOR 1 field as the destination for an EXSR operation code.	0	B	R	B	B	B	B	B	B
ENDSR Identifies the end of a subroutine.	0	B	0	B	B	B	B	B	B
EXSR Calls the subroutine specified in the FACTOR 2 field.	0	0	B	R	B	B	B	B	B
Control of Input and Output:									
CHAIN Reads randomly from a disk file.	0	0	R	R	B	B	B	B	EI
DSPLY Displays on the ODT the contents of the FACTOR 1 field, and optionally accepts a message from the operator and stores the message in the RESULT FIELD field.	0	0	0	R	0	0	0	B	B
EXCPT Causes output to be performed during detail- or total-time calculations.	0	0	B	B	B	B	B	B	B
FORCE Reads the next input record ahead of the cycle.	B	0	B	R	B	B	B	B	B
READ Reads a record from a demand file.	0	0	B	R	B	B	B	B	EI
Debug Operations:									
DEBUG Prints the indicators that are ON and the contents of the field specified in the RESULT FIELD field.	0	0	0	R	0	B	B	B	B
Time:									
TIME Obtains the current system date and time.	0	0	B	B	R	0	0	B	B

Table 11-1. Summary of Operation Codes (Cont)

Operation Code and Definitions	Control Level	Indicators	Factor 1	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust	Resulting Indicators
Data Communications:									
RECV Reads from the remote file specified in the FACTOR 2 field.	O	O	B	R	B	B	B	B	O
SEND Writes to the remote file specified in the FACTOR 2 field.	O	O	B	R	B	B	B	B	O
Process Within Limits:									
SETLL Sets the lower limit for a demand file processed sequentially.	O	O	R	R	B	B	B	B	B

NOTE

O – Optional
R – Required
B – Blank
EI – Only the Equal Indicator can be used. The indicator is turned ON if the end-of-file record is read on the demand file.

Table 11-2. RESULT FIELD Contents for Various Field Lengths and Decimal Positions – MULT Operation Code

Field Length	Decimal Positions					
	0	1	2	3	4	5
1	2	.6				
2	92	2.6	.61			
3	192	92.6	2.61	.611		
4	0192	192.6	92.61	2.611	.6111	
5	00192	0192.6	192.61	92.611	2.6111	.61116

NOTE
The contents of the RESULT FIELD field is the result of multiplying 89.67 by 2.148.

Arithmetic Operations

In the descriptions of arithmetic operations in the following paragraphs, an indexed vector or a table name is valid anywhere that a field name is valid except as noted. Arithmetic operations are allowed only on numerically defined data items. FACTOR 1 and FACTOR 2 fields must contain the names of numeric fields or numeric literals. The RESULT FIELD field must be numeric. All results are signed and decimal alignment is performed. If the RESULT FIELD field is not large enough to hold the result of an arithmetic operation, the result is truncated at either or both ends following decimal alignment. If the RESULT FIELD field is too large to hold the result of an arithmetic operation, then leading and trailing zeroes are inserted following decimal alignment.

The FACTOR 1, FACTOR 2 and the RESULT FIELD fields can all be different fields or any two or all three can be the same field. The HALF ADJUST field can be specified for all operations except SQRT and MVR, that is, the DIV operation code followed by the MVR operation code. Resulting indicators can be specified for all operations if the RESULT FIELD field is not a whole array.

The use of result indicators in connection with arithmetic operations is as shown below:

1. Plus (Columns 54-55). Any indicator entered in this field is turned on if the result of an arithmetic operation is positive.
2. Minus (Columns 56-57). Any indicator entered in this field is turned on if the result of an arithmetic operation is negative.
3. Zero (Columns 58-59). Any indicator entered in this field is 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 the FACTOR 2 field to the contents of the FACTOR 1 field and stores the sum in the RESULT FIELD field. The FACTOR 1 and FACTOR 2 fields are not affected by this operation, unless one of them is also designated as the RESULT FIELD field. An entry can be specified in the HALF ADJUST field.

SUB

This operation subtracts the contents of the FACTOR 2 field from the contents of the FACTOR 1 field and places the difference in the RESULT FIELD field. The FACTOR 1 and FACTOR 2 fields are not affected by this operation, unless one of them is also designated as the RESULT FIELD field.

Note that subtracting two fields which are the same gives the same result as clearing the RESULT FIELD field to zero; this method can be used for clearing fields to zero.

MULT

This operation multiplies the contents of the FACTOR 1 field by the contents of the FACTOR 2 field and stores the product in the RESULT FIELD field. The FACTOR 1 and FACTOR 2 fields are not affected by this operation, unless one of them is also designated as the RESULT FIELD field. An entry can be specified in the HALF ADJUST field.

Table 11-2 provides an example of the contents of the RESULT FIELD 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. If an entry is not specified in the HALF ADJUST field, fields not permitted are blank. 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 the FACTOR 1 field by the contents of the FACTOR 2 field and places the quotient in the RESULT FIELD field. The FACTOR 1 and FACTOR 2 fields are not affected by this operation, unless one of them is also designated as the RESULT FIELD field.

Any remainder resulting from the divide operation is lost unless the next operation specified is the MVR (move remainder) operation. If so, the result of the divide operation cannot be half adjusted. The RESULT FIELD field cannot contain a whole array when an a MVR operation code is specified as the next operation.

If the FACTOR 2 field 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 field. The MVR operation must immediately follow the DIV operation, and the FACTOR 1 and FACTOR 2 fields must be left blank. The RESULT FIELD field for the MVR operation must be greater than or equal to the length of the FACTOR 2 field for the previous DIV operation. Both the DIV and the MVR operation codes can 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 operation, otherwise the result of the MVR operation is undefined. The HALF ADJUST field must be blank.

The following considerations should be made when specifying the size of the RESULT FIELD field:

1. The number of significant decimal places in the remainder is the larger of:

   1) The number of decimal positions in the FACTOR 1 field of the previous DIV operation.
   2) The sum of the decimal positions in the FACTOR 2 field and the RESULT FIELD field of the previous DIV operation.

2. The maximum integer positions in the remainder is greater than or equal to the integer positions in the FACTOR 2 field of the previous DIV operation.
3. The RESULT FIELD field must not be a whole array.

Figure 11-5 provides a DIV and MVR operation coding example.

```
                  Calculation Specifications

01 C            FIELDA    DIV  FIELDB    FIELDC  62
02 C                      MVR             REMAIN  5
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 11-5. DIV and MVR Coding Example

SQRT

The operation derives the square root of the contents of the FACTOR 2 field and places it in the RESULT FIELD field. The FACTOR 1 field must be left blank.

To obtain reasonable precision from the SQRT operation, the following must be kept in mind:

1. For every digit to the left of the decimal point in the RESULT FIELD field, there must be two digits to the left of the decimal point in the FACTOR 2 field.
2. For every digit to the right of the decimal point in the RESULT FIELD field, there must be two digits to the right of the decimal point in the FACTOR 2 field.

The result of the SQRT operation is automatically adjusted; therefore, a HALF ADJUST entry for SQRT is not allowed.

If the FACTOR 2 operand is negative, the program displays a SQRT error message and halts. The operator can resume processing, in which case the RESULT FIELD field is set to zero.

Table 11-3 provides an example of the contents of the RESULT FIELD field for the square root operation. In the example, all fields that are permitted are shown, although some contain incomplete results. Fields not permitted are blank.

Table 11-3. RESULT FIELD Contents for Various Field Lengths and Decimal Positions – SQRT Operation Code

Field Length	Decimal Positions					
	0	1	2	3	4	5
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

NOTE
The contents of the RESULT FIELD field is the result of performing a square-root operation on 2994.42.

XFOOT

This operation crossfoots (sum) the elements of a numeric array. All the elements of the array specified by the FACTOR 2 field are summed and the total placed in the RESULT FIELD field. The FACTOR 1 field must be left blank. The RESULT FIELD field must not be a whole array.

Resulting indicators can be specified. If the RESULT FIELD field is an element of the array named in FACTOR 2, the value of that element before the XFOOT operation is used in obtaining the total. HALF ADJUST can be specified.

Z-ADD

This operation sets the RESULT FIELD field to zeros and adds the contents of the FACTOR 2 to the RESULT FIELD field. The FACTOR 1 field must be left blank.

The FACTOR 2 field is not affected by this operation unless it is named in the RESULT FIELD field. The HALF ADJUST field can be specified.

Z-SUB

This operation sets the RESULT FIELD field to zeros, then subtracts the contents of the FACTOR 2 field from the RESULT FIELD field. The FACTOR 1 field must be left blank. This operation is used to change the sign of the field designated by the FACTOR 2 field. The HALF ADJUST field can be specified.

Note that this is the same as the Z-ADD operation, except the sign is changed.

In the example shown in figure 11-6, if the contents of FIELDA equals 678.9321, then the contents of FIELDB equals 000678.93, and the contents of FIELDC equals −000678.93.

```
                    Calculation Specifications

01 C                          Z-ADDFIELDA    FIELDB  82
02 C                          Z-SUBFIELDA    FIELDC  82
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 11-6. Z-ADD and Z-SUB Coding Example

Move Operations

Move operations transfer the contents of the FACTOR 2 field to the RESULT FIELD field. The FACTOR 2 field is not affected by the move operation. The FACTOR 1 field must be left blank. Resulting indicators can be specified to indicate the status of the RESULT FIELD field (plus, minus, or zero/blank) in the same manner as field indicators on Input Specifications. If the FACTOR 2 and the RESULT FIELD field are the same length, the MOVE and MOVEL operation codes operate in the same manner. The FACTOR 2 field can be a field, vector, vector element, or literal; however, a RESULT FIELD field can only be a field, vector, or vector element.

An array name can be entered in the FACTOR 2 field if the RESULT FIELD field also specifies an array. In this case, the designated move operation is performed on each element of the array designated in the FACTOR 2 field with the result being placed in the corresponding elements of the array designated in the RESULT FIELD field. The operation is terminated when the end of the shorter array is reached. If the RESULT FIELD field is an array and the FACTOR 2 field is not, then the designated move operation is performed on the FACTOR 2 field, with the result being placed in all elements of the RESULT FIELD array.

For the purpose of the MOVE, MOVEL operation codes, the "sign" of the variable is defined as (1) the algebraic sign if the variable is numeric, or (2) the low order zone if the variable is alphanumeric.

Move operations can be specified between fields of different data types. This converts an alphanumeric to a numeric and vice versa. Decimal points are ignored in all move operations. The HALF ADJUST field must be blank for the MOVE or MOVEL operation codes.

Decimal positions are ignored when moving data from one numeric field to another numeric field. For example, if X is a field with three decimal positions and Y is a field with two decimal positions, then a move operation of 10.000 from X to Y yields 100.00 in Y.

MOVE

This operation moves characters from the FACTOR 2 field to the RESULT FIELD field, starting with the right-most character and continuing until either the source field is exhausted, or the destination field is filled. The sign of the FACTOR 2 field is moved to the sign position of the RESULT FIELD field.

When an alphanumeric-to-numeric move is specified, the digit portion of each character is moved to the RESULT FIELD field . Blank characters are transferred as zeros. The zone portion of the right-most alphanumeric character is used as the sign of the RESULT FIELD field.

When a numeric-to-alphanumeric move is specified, each digit is converted to its corresponding internal character code as it is moved to the RESULT FIELD field. The sign of the numeric field is placed in the zone portion of the right-most character. If the numeric field is positive, then the zone portion of the right-most character contains the systems standard positive sign (hexadecimal C). If the numeric field is negative, then the zone portion of the right-most character contains the systems standard negative sign (hexadecimal D). If writing to the printer, characters A thru I, or ? are printed in the right-most character of the field. The characters A thru I represent digits 1 through 9, respectively. The question mark (?) character represents digit 0. Refer to appendix B for more information concerning the hexadecimal values for the B 1000 character set.

MOVE operations are shown in table 11-4.

Table 11-4. MOVE Operation Code Examples

RESULT FIELD Field Larger than the FACTOR 2 Field

FACTOR 2		RESULT FIELD		
Alphanumeric	AB4SK	Before: After:	123456789 1234AB4SK	Alphanumeric
Alphanumeric	AB4SK	Before: After:	+123456789 −123412422	Numeric
Numeric	−9876543	Before: After:	+123456789 −129876543	Numeric
Numeric	+9876543	Before: After:	ABCDEFGHI AB9876543	Alphanumeric

Table 11-4. MOVE Operation Code Examples (Cont)

RESULT FIELD Field Smaller than the FACTOR 2 Field

FACTOR 2		RESULT FIELD		
Alphanumeric	GEBKLM4SK	Before: After:	56789 LM4SK	Alphanumeric
Alphanumeric	GEBKLM4SK	Before: After:	+56789 −34422	Numeric
Numeric	−9876543	Before: After:	+56789 −76543	Numeric
Numeric	+9876543	Before: After:	ABCDE 7654L	Alphanumeric

RESULT FIELD and FACTOR 2 Fields Same Length

FACTOR 2		RESULT FIELD		
Alphanumeric	AB4SK	Before: After:	56789 AB4SK	Alphanumeric
Alphanumeric	AB4SK	Before: After:	+56784 −12422	Numeric
Numeric	−98765	Before: After:	+56789 −98765	Numeric
Numeric	+98765	Before: After:	ABCDE 8765M	Alphanumeric

NOTE
Letter K equals −2 (negative 2)
Letter L equals −3 (negative 3)
Letter M equals −4 (negative 4)

MOVEL Operation Code

This operation moves characters, left-justfied from the FACTOR 2 field to the RESULT FIELD field, starting with the left-most 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 field.

When an alphanumeric-to-numeric move is specified, only the digit portion of each character is moved to the RESULT FIELD field. Blank characters are transferred as zeros.

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

MOVEL operations are shown in table 11-5.

Table 11-5. MOVEL Operations

FACTOR 2		RESULT FIELD		Characters Transferred	Sign Transferred
Type	Length	Type	Length		
Numeric	Equal	Numeric	Equal	Digits	To Sign Position
Numeric	Shorter	Numeric	Longer	Digits	No Transfer
Numeric	Longer	Numeric	Shorter	Digits	To Sign Position
Numeric	Equal	Alphanumeric	Equal	Digits	To Right-most Zone
Numeric	Shorter	Alphanumeric	Longer	Digits	To Right-most Zone
Numeric	Longer	Alphanumeric	Shorter	Digits	No Transfer
Alphanumeric	Equal	Numeric	Equal	Digits	To Sign Position
Alphanumeric	Shorter	Numeric	Longer	Digits	No Transfer
Alphanumeric	Longer	Numeric	Shorter	Digits	To Sign Position
Alphanumeric	Equal	Alphanumeric	Equal	Bytes	To Right-most Zone
Alphanumeric	Shorter	Alphanumeric	Longer	Bytes	To Right-most Zone
Alphanumeric	Longer	Alphanumeric	Shorter	Bytes	No Transfer

MOVEA Operation Code

The MOVEA (move array) operation code moves the data from the left-most position of the field specified in the FACTOR 2 field to the left-most position of the field specified in the RESULT FIELD field. When the end of either the FACTOR 2 field or the RESULT FIELD field is reached, data transfer is terminated. When the RESULT FIELD field is longer than the FACTOR 2 field, the remainder of the data in the RESULT FIELD field remains unchanged. The data contained in the FACTOR 2 field is never changed.

The MOVEA operation code can be used with either vector type, tables, or arrays.

It is possible to move the following with the MOVEA operation code:

1. Adjacent array elements into a field.
2. A field into adjacent array elements.
3. Adjacent elements in one array into the corresponding elements of another array.
4. The array in either the FACTOR 2 or the RESULT FIELD fields can be indexed. The index can be either absolute or variable.

    1) For move operations from the FACTOR 2 field, the data transfer begins at the element indicated by the value of the index.
    2) For moves to the RESULT FIELD field, the data reception begins at the element indicated by the value of the index.

5. A literal to a field.
6. A literal to an array.
7. A literal to an array element.
8. A table name or element when referenced by the FACTOR 2 or the RESULT FIELD fields.
9. An array can be moved to itself by referencing the same array in both the RESULT FIELD and FACTOR 2 fields.

Requirements for MOVEA Operation

The following requirements must be observed for the MOVEA operation:

1. The data format of both the FACTOR 2 and RESULT FIELD fields must be alphanumeric.
2. The OPERATION CODE, FACTOR 2, and RESULT FIELD fields must contain entries.
3. Control levels and conditioning indicators (columns 7-17) can contain valid entries. The remaining portion of the specification line must be left blank.

Restrictions for the MOVEA Operation Code

The following restrictions must be observed when using the MOVEA operation code:

1. The FACTOR 2 or RESULT FIELD fields cannot reference numeric data.
2. The RESULT FIELD field cannot contain a literal.

Coding Examples for MOVEA Operation Code

The six figures (figures 11-6a through f) on the following pages show the use of the MOVEA operation code and the contents of the FACTOR 2 and RESULT FIELD field before and after execution of the MOVEA operation code.

```
                Calculation Specifications

01 C                        MOVEAFIELD1    ARRAY
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

 Contents:

  FIELD1                          ARRAY
  ------                          -----
 12345678        Before MOVEA: ABC DEF GHI JKL
                 After MOVEA:  123 456 781 JKL
```

The field FIELD1 is an eight-character alphanumeric field.
The array ARRAY is an alphanumeric array containing four elements of three characters each.

Figure 11-6a. Field to Array Move – No Indexing

```
                Calculation Specifications

01 C                        MOVEAARRAY,4   FIELD1
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

 Contents:

   ARRAY                        FIELD1
   -----                        ------
ABC DEF GHI JKL  Before MOVEA: 12345678
                 After MOVEA:  JKL45678
```

The array ARRAY is an alphanumeric array containing four elements of three characters each.
The field FIELD1 is an eight-character alphanumeric field.

Figure 11-6b. Array to Field Move – Array Indexed by Literal

```
                Calculation Specifications

01 C                        MOVEAARRAY2    ARRAY
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

 Contents:

  ARRAY2                          ARRAY
  ------                          -----
123 456 789      Before MOVEA: ABC DEF GHI JKL
                 After MOVEA:  123 456 789 JKL
```

The arrays ARRAY2 and ARRAY have different numbers of elements, but all elements are of the same length.

Figure 11-6c. Array to Array Move – No Indexing

```
                    Calculation Specifications

01 C                          MOVEAARR2,N    ARRAY
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

 Contents:

 ARRAY2                              ARRAY
 ------                              -----
01 23 45 67 89    Before MOVEA: ABC DEF GHI JKL
                  After MOVEA:  678 9EF GHI JKL
```

The FACTOR 2 field is indexed by a variable (N) whose value is 4.
The arrays ARR2 and ARRAY have elements of different lengths.

Figure 11-6d. Array to Array Move – FACTOR 2 Field Indexed

```
                    Calculation Specifications

01 C                          MOVEAARR2      ARR3,3
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

 Contents:

  ARR2                               ARR3
  ----                               ----
01 02 03 04 05    Before MOVEA: AB CD EF GH IJ
                  After MOVEA:  AB CD 01 02 03
```

The RESULT FIELD field is indexed by a literal.

Figure 11-6e. Array to Array Move – RESULT FIELD Field Indexed

```
                    Calculation Specifications

01 C                          MOVEAARR2      ARRAY
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

 Contents:

  ARR2                               ARR3
  ----                               ----
01 23 45 67 89    Before MOVEA: ABC DEF GHI JKL
                  After MOVEA:  012 345 678 9KL
```

The arrays ARR2 and ARRAY have elements of different lengths.

Figure 11-6f. Array to Array Move – No Indexing

Move Zone Operations

Move zone operations are used to move the zone portion of only one character. The FACTOR 2 field is not affected by this operation, and only the zone portion of one character of the RESULT FIELD field is affected. The FACTOR field must be left blank. The HALF ADJUST and RESULTING INDICATORS fields must also be left blank.

An array name can be entered in the FACTOR 2 field, if the RESULT FIELD field also specifies an array. In this case, the designated move zone operation is performed on each element of the array designated in the FACTOR 2 field, with the result being placed in the corresponding elements of the array designated in the RESULT FIELD field. The operation is terminated when the end of the shorter array is reached.

If the RESULT FIELD field is an array, but the FACTOR 2 field is not, then the zone from the FACTOR 2 field is moved into the appropriate position in all elements of the array.

For the purpose of move zone operations, the low-order zone portion of a variable is defined as:

1. The algebraic sign if the variable is numeric, or
2. The low-order zone if the variable is alphanumeric.

A high-order zone must refer only to an alphanumeric field. For a numeric field, only the sign can be referenced.

The systems standard positive sign is forced into all positive numeric fields that are the result of a move zone operation. When the zone portion of a positive numeric field is moved to the zone portion of an alphanumeric field by a move zone operation, the sign byte of the result field contains alphanumeric 0-9. When the zone of a negative numeric field is moved to the zone of an alphanumeric field by a move zone operation, the sign of the result field contains the systems standard negative sign (D).

Move zone operations are shown in table 11-6.

Table 11-6. Move Zone Operations

Instruction	Contents of FACTOR 2	RESULT FIELD	Definition
MHHZO	Alpha	Alpha	Move high-order zone portion of FACTOR 2 to high-order zone of RESULT ZONE.
MHLZO	Alpha	Alpha	Move high-order zone portion of FACTOR 2 to low-order zone of RESULT FIELD.
	Alpha	Numeric	Move high-order zone portion of FACTOR 2 to the sign position of RESULT FIELD.
MLHZO	Alpha	Alpha	Move low-order zone portion of FACTOR 2 to high-order zone of RESULT FIELD.

Table 11-6. Move Zone Operations (Cont)

Instruction	Contents of FACTOR 2	RESULT FIELD	Definition
MLLZO	Numeric	Alpha	Move sign position of FACTOR 2 to high-order zone portion of RESULT FIELD.
	Alpha	Alpha	Move low-order zone of FACTOR 2 to low-order zone of RESULT FIELD.
	Alpha	Numeric	Move low-order zone portion of FACTOR 2 to sign position of RESULT FIELD.
	Numeric	Alpha	Move sign position of FACTOR 2 to low-order zone portion of RESULT FIELD.
	Numeric	Numeric	Move sign position of FACTOR 2 to sign position of RESULT FIELD.

Each of the move zone operations is discussed in the following paragraphs.

MHHZO Operation Code

This operation moves the zone from the high-order (left-most) position of the FACTOR 2 field to the high-order (left-most) position of the RESULT FIELD field. Both the FACTOR 2 and RESULT FIELD fields must be alphanumeric (see figure 11-7).

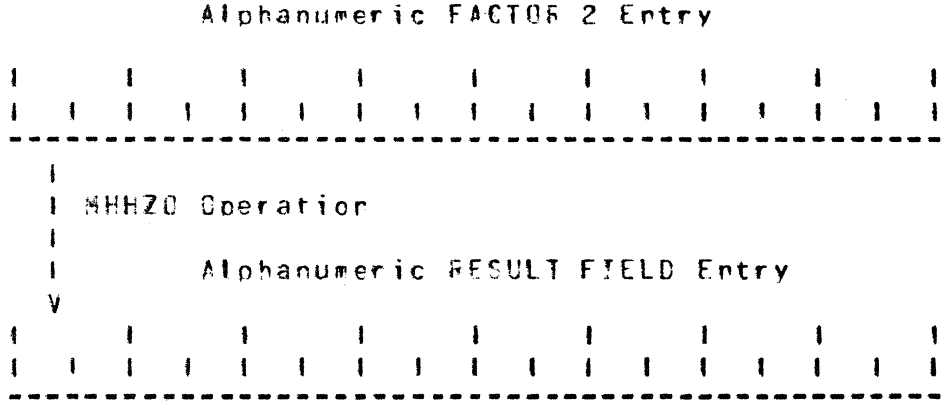


Figure 11-7. MHHZO Move Zone Operation

MHLZO Operation Code

If the FACTOR 2 field is alphanumeric and the RESULT FIELD field is alphanumeric, this operation moves the zone from the high-order (left-most) position of the FACTOR 2 field to the low-order (right-most) position of the RESULT FIELD field. If the FACTOR 2 field is alphanumeric and the RESULT FIELD field is numeric, this operation moves the high-order (left-most) position of the FACTOR 2 field to the sign position of the RESULT FIELD field. The FACTOR 2 field must be alphanumeric (see figure 11-8).

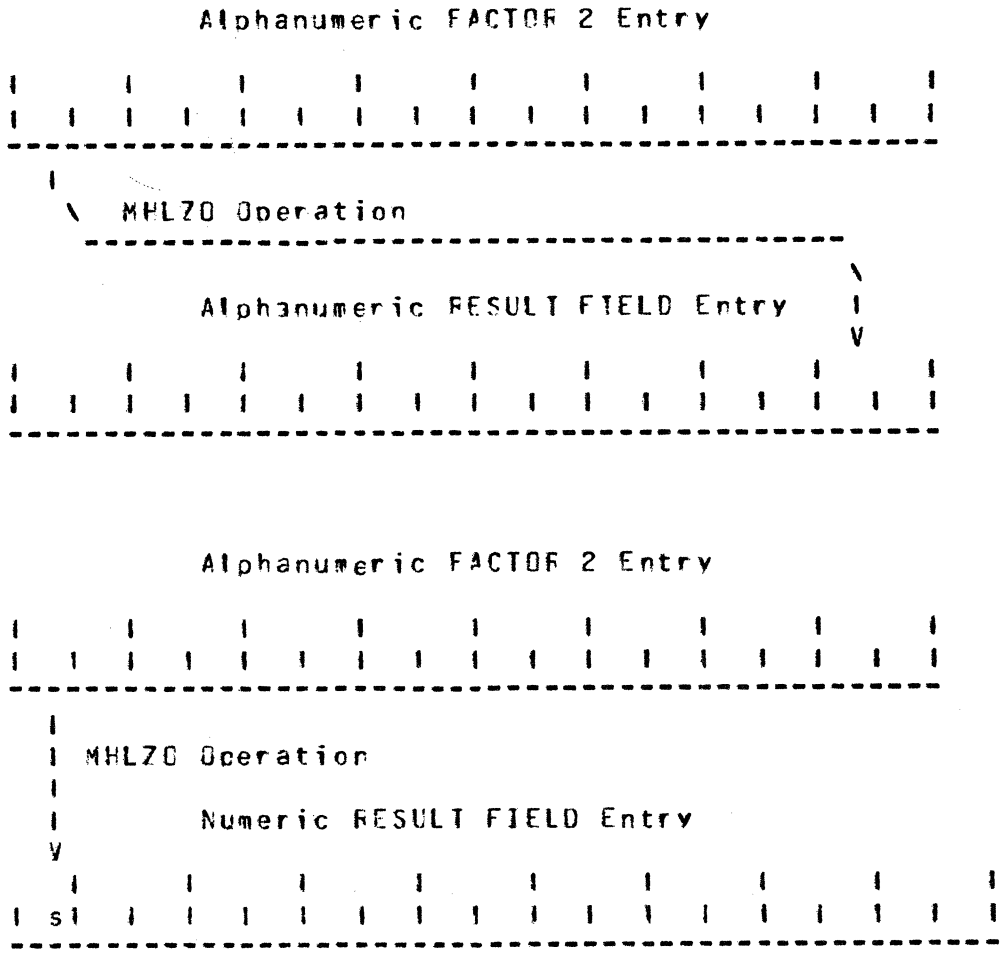


Figure 11-8. MHLZO Move Zone Operations

MLHZO Operation Code

If the FACTOR 2 field is alphanumeric and the RESULT FIELD field is alphanumeric, then this operation moves the zone from the low-order (right-most) position of the FACTOR 2 field to the high-order (left-most) position of the RESULT FIELD field. If the FACTOR 2 field is numeric and the RESULT FIELD field is alphanumeric, then this operation moves the zone from the sign position of the FACTOR 2 field to the high-order (left-most) position of the RESULT FIELD field. The RESULT FIELD field must be alphanumeric (see figure 11-9).

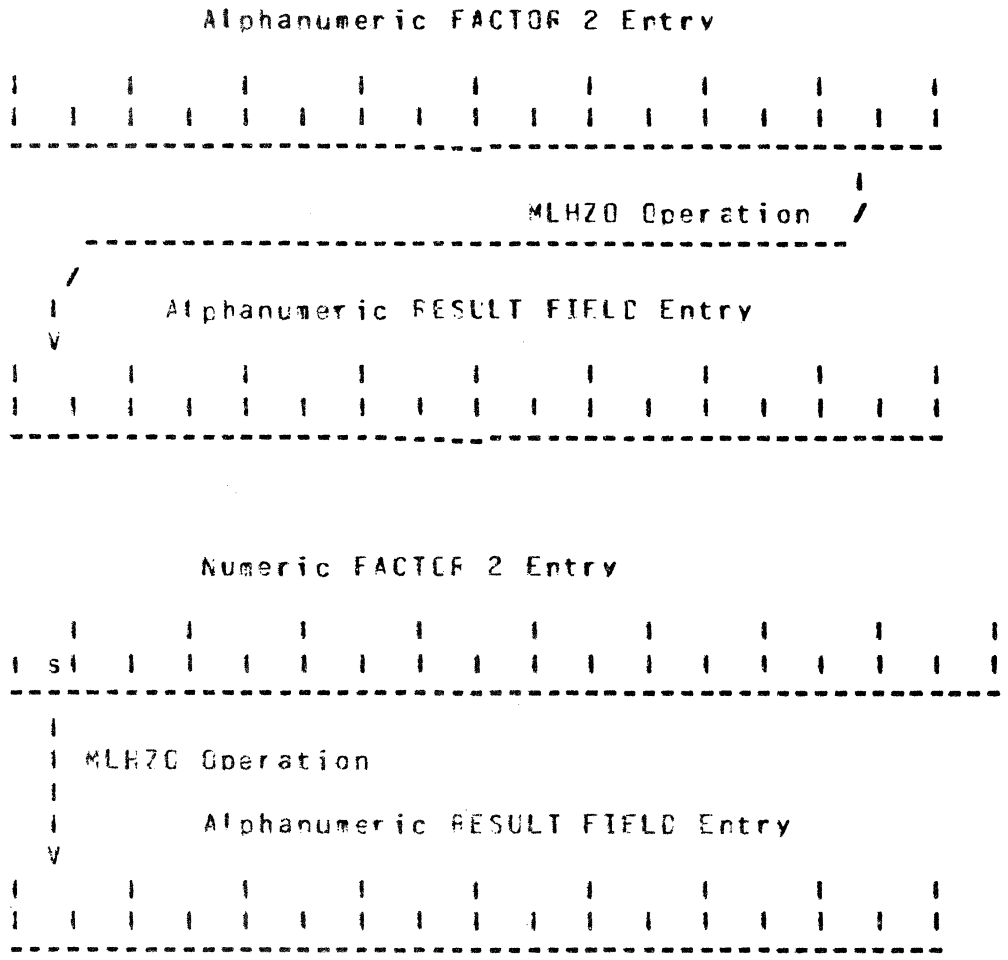


Figure 11-9. MLHZO Move Zone Operations

MLLZO Operation Code

If the FACTOR 2 field is alphanumeric and the RESULT FIELD field is alphanumeric, then this operation moves the zone from the low-order (right-most) position of the FACTOR 2 field to the low-order (right-most) position of the RESULT FIELD field.

If the FACTOR 2 field is alphanumeric and the RESULT FIELD field is numeric, then this operation moves the zone from the low-order (right-most) position of the FACTOR 2 field to the sign position of the RESULT FIELD field.

If the FACTOR 2 field is numeric and the RESULT FIELD field is alphanumeric, then this operation moves the zone from the sign position of the FACTOR 2 field to the low-order (right-most) position of the RESULT FIELD field.

If the FACTOR 2 field is numeric and the RESULT FIELD field is numeric, then this operation moves the zone from the sign position of the FACTOR 2 field to the sign position of the RESULT FIELD field (see figure 11-10).

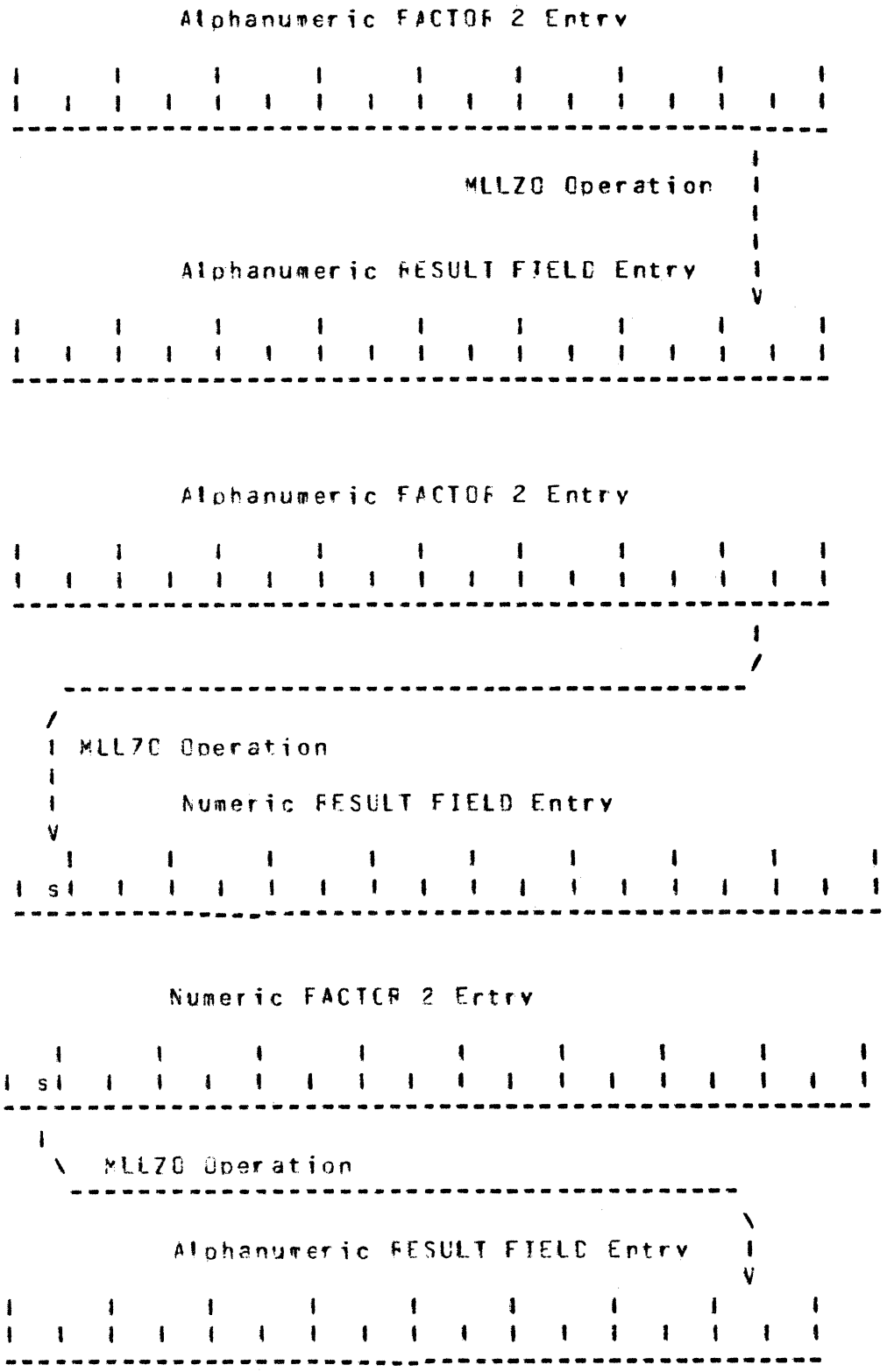


Figure 11-10. MLLZO Move Zone Operations (Sheet 1 of 2)

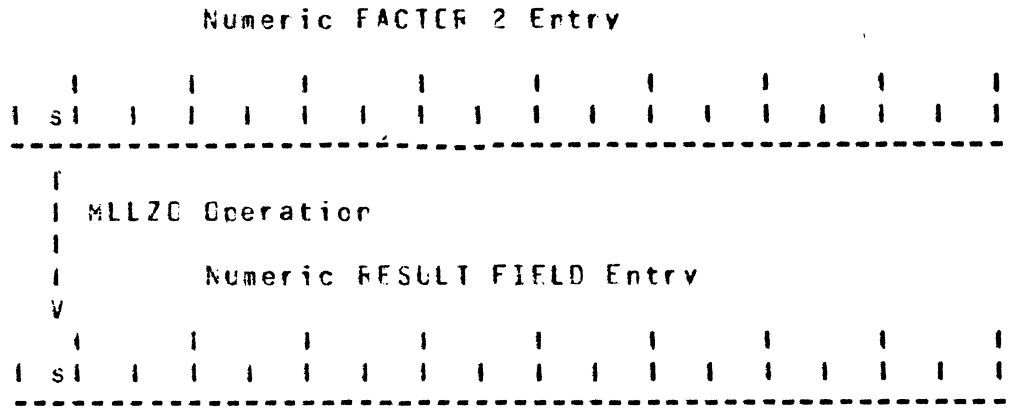


Figure 11-10. MLLZO Move Zone Operations (Sheet 2 of 2)

Compare Operations

These operations test specified fields for certain conditions. The results of these operations are shown by the setting of the specified RESULTING INDICATOR field (01-99, L1-L9, LR, H0-H9, OA-OA-OG, OV). The FACTOR 1 and FACTOR 2 fields are not affected by the operation. At least one RESULTING INDICATOR field must be specified. The HALF ADJUST and RESULT FIELD fields must be left blank. Neither the FACTOR 1 or FACTOR 2 field can be a whole array.

COMP Operation Code

This operation compares the contents in the FACTOR 1 field with the contents in the FACTOR 2 field, causing the indicators specified in the RESULTING INDICATORS field to be set as follows:

1. HIGH (columns 54-55) – FACTOR 1 > FACTOR 2
2. LOW (columns 56-57) – FACTOR 1 < FACTOR 2
3. EQUAL (columns 58-59) – FACTOR 1 = FACTOR 2

Both fields must be of the same data type.

Comparison of Numeric Fields

The comparison of numeric fields is based on their respective values considered purely as signed numeric quantities. The length of the fields, in terms of digits, is not itself significant. Both fields are automatically aligned on their decimal points, and leading or trailing zeros are supplied, as needed, to make the lengths identical.

Comparison of Alphanumeric Fields

The comparison of alphanumeric fields is accomplished by placing the contents of the FACTOR 1 and FACTOR 2 fields, left-justified, in work areas of equal length, and padding the right-most positions of the shorter field with blank characters. A character-to-character comparison is then made beginning at the left-most position of each field. Moving to the right, the comparison continues until:

1. An unequal condition is found between corresponding characters, or
2. The end of the fields is reached.

When the unequal condition is found, if the character that is highest in the collating sequence is in the FACTOR 1 field , then the high (1 greater than 2) indicator is turned ON, if specified. If the

character that is highest in the collating sequence is in the FACTOR 2 field , the low (1 is less than 2) indicator is turned ON, if specified. Otherwise, the equal (1 is equal to 2) indicator is turned ON, if specified.

When the end of the fields is reached, an equal condition is indicated.

TESTN Operation Code

The test numeric operation tests the contents of the alphanumeric field, indexed vector name, or unindexed table name specified in the RESULT FIELD field (columns 43-48) for numeric characters or blanks. A whole array cannot be specified. If all characters in the alphanumeric field specified are numeric, the indicator specified in columns 54-55 (high) is set ON. Each character in the field must be unsigned, with the exception of the low-order character, which must contain:

1. The systems standard positive or negative sign, or
2. Be unsigned, or
3. Contain a valid plus or minus character to be considered numeric.

If the alphanumeric field specified contains numeric characters and leading blanks, the indicator specified in columns 56-57 (low) is set ON. The indicator specified in columns 58-59 (equal) is set to ON if the field contains all blanks. The same indicator can be specified for more than one condition and is set if the condition exists.

The TESTN operation code determines the sign of the low-order (right-most) character within an alphanumeric field as follows:

Character Sign	Character Being Tested
Plus	A through I, +0, +
Minus	J through R, −0, −
Unsigned	0 through 9

The TESTN operation code determines the character status of the other characters within an alphanumeric field, with the exception of the low-order character, as follows:

Character Status	Character Being Tested
Numeric	0 through 9
Non-numeric	All other characters

NOTE

The TESTN operation code applies to alphanumeric fields only. If an arithmetic operation is to be performed, then this field must be moved to a numeric field.

Table 11-7 lists the various results of the TESTN operation code for a 4-byte alphanumeric field. A "b" indicates a blank character.

Table 11-7. Results of the TESTN Operation Code

Contents of Alphanumeric Field	Indicator Results		
	Columns 54-55	Columns 56-57	Columns 58-59
ABCD	OFF	OFF	OFF
bABC	OFF	OFF	OFF
bbbA	OFF	ON	OFF
Abbb	OFF	OFF	OFF
123D	ON	OFF	OFF
b12C	OFF	ON	OFF
bb1C	OFF	ON	OFF
bbbb	OFF	OFF	ON
1234	ON	OFF	OFF

TESTZ Operation Code

This operation tests the zone portion of the left-most character of the RESULT FIELD field, setting the specified RESULTING INDICATOR field (01-99, L1-L9, LR, H0-H9, OA-OG, OV) to the results of the test. The FACTOR 1, FACTOR 2, and HALF ADJUST fields must be left blank. The RESULT FIELD field must not be a whole array.

If the RESULT FIELD field is alphanumeric, the character under test sets 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 field is numeric, the PLUS field (columns 54-55) or MINUS field (columns 56-57) indicator is set according to the sign of the field. A zero indicator must not be specified for a numeric field. Note that the TESTZ operation code is essentially a zone portion test and differentiates 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 operation code) against zero, instead of the TESTZ operation code

Binary Field Operations

Three operation codes, BITON, BITOF, and TESTB are used. The FACTOR 2 field can contain either of the following:

1. Bit values 0-7. One or more bits (maximum of eight) can be set ON, set OFF, or tested per operation. The bits are numbered from left to right and are enclosed in apostrophes. The order

of specification of the bits is not restricted. Bits not specified in the FACTOR 2 field are not changed.

2. The name of a one-position, alphanumeric field or table, or array element. In this case, the bits which are on in the field or array or table element are set ON, set OFF, or tested in the RESULT FIELD field ; bits which are not ON are not affected.

BITON Operation Code

The BITON operation code causes bits identified in the FACTOR 2 field to turn ON (set to 1) in a previously defined field named as the RESULT FIELD field. The BITON operation code must appear in columns 28-32. Conditioning indicators can be used in columns 7-17. Any entry under the FIELD LENGTH field must be 1.

The FACTOR 1, DECIMAL POSITIONS, HALF ADJUST, and RESULTING INDICATORS fields must be left blank (see figure 11-11).

```
                  Calculation Specifications

01 C                        BITON'1234'     BITA
02 C                        BITONITEMA      BITB
03 C
04 C                        BITOFITEMB      ARR,XY
05 C                        BITOF'123'      ARR,XY
06 C                        BITOFARR,XY     TBL,20
07 C
08 C                        TESTB'057'      BITC        101212
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 11-11. Binary Field Operations Coding Examples

BITOF Operation Code

The BITOF operation code causes bits identified in the FACTOR 2 field to turn OFF (set to zero) in a previously defined field named as the RESULT FIELD field.

The BITOF operation code must appear in columns 28-32. All other specifications are the same as those for the BITON operation code (see figure 11-11).

TESTB Operation Code

The TESTB operation code causes bits identified in the FACTOR 2 field to be tested for an ON or OFF condition in the previously defined field named in a RESULT FIELD field, setting the specified resulting indicator to the result of the test. All other specifications are the same as those for the BITON and BITOF operation codes.

At least one RESULTING INDICATOR field must be specified with the TESTB operation code; as many as three can be named for one operation. Two indicators can be the same for one TESTB operation code, but not three. If the FACTOR 2 field contains bits which are all OFF, no resulting indicators are turned ON. Resulting indicators have the meanings described in the following paragraphs.

Columns 54-55

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

Columns 56-57

An indicator in these columns is turned ON if two or more bits were tested and found to be of mixed status; that is, some bits ON and other bits OFF. It is important to ensure that the field named in the FACTOR 2 field contains more than one bit which is ON if an indicator appears in columns 56-57.

Columns 58-59

An indicator in these columns is turned ON if each bit specified in the FACTOR 2 field is ON (set to 1) in the RESULT FIELD field.

The following explanations refer to figure 11-11.

1. Bits 1234 are turned ON in the field named BITA.
2. Bits that are ON in the field named ITEMA cause the corresponding bits in the field named BITB to be turned ON.
3. Bits that are OFF in the field named ITEMB cause the corresponding bits in the array element ARR,XY to be turned OFF. Then bits 1, 2, and 3 are turned off in array element ARR,XY, and finally the bits that are off in ARR,XY are turned off in the array element TBL,20.
4. If bits 0, 5, and 7 are OFF in the field named BITC, indicator 10 is turned ON. If bits 0, 5, and 7 are of mixed status in the field named BITC, indicator 12 is turned ON. If bits 0, 5, and 7 are ON in the field named BITC, indicator 12 is turned ON.

Setting Indicators

Up to three indicators can be set ON or OFF with one operation. These can be entered in the RESULTING INDICATORS field. The FACTOR 1, FACTOR 2, HALF ADJUST and RESULT FIELD fields must be left blank. The following rules must be observed when setting indicators:

1. The following indicators cannot be set ON or OFF: 1P, MR, LO, or U1-U8.
2. Setting a control level indicator (L1-L9) ON or OFF does not affect any other control level indicator.
3. All control level and record identifying indicators, except LO are automatically turned OFF after detail output operations are completed, regardless of any previous SETON or SETOF operation code.
4. If any halt indicators (H0-H9) are set ON and are not turned OFF before the detail output operations finish, then the program halts.
5. If the LR indicator is turned ON by a SETON operation code which is conditioned by a control level indicator (columns 7-8 on Calculations Specifications form), then the program stops after all total output operations are completed. If the LR indicator is turned ON by a SETON operation code which is not conditioned by a control level indicator, the program stops after the next total output operation is completed.

SETON Operation Code

The SETON operation code sets the indicators entered in the RESULTING INDICATORS field ON.

SETOF Operation Code

This SETOF operation code sets the indicators entered in the RESULTING INDICATORS field OFF.

Program Branching Operations

Operations within the Calculation Specifications are normally performed in the order in which they are written. Branching operations allow variation of the order of operation; thus, conditional branching and repetitive operations are possible.

GOTO Operation Code

The GOTO operation code causes the program to branch (go to) to some other instruction rather than "falling through" to the next sequential operation. Branching both forward and backward is allowed. Branching into or out of subroutines is not permitted. Branching from total calculations to detail calculations, though permitted in RPG I, can produce unexpected results when used in RPG II.

The FACTOR 2 field must contain a label which must be defined elsewhere in the program as a TAG. This label must follow the rules for formation of labels as described in section 2. Columns 18-27 and 39-59 must be blank.

TAG Operation Code

The TAG operation code identifies the point where a GOTO operation code branches. The FACTOR 1 field must contain a unique label (TAG) which must follow the rules for formation of labels as described in section 2.

Conditioning by a control level indicator in columns 7-8 is permissible if the TAG is part of total calculations. The INDICATOR field (columns 9-17) must be left blank. Columns 33-59 and 24-27 must be blank.

Transfer Control Function

This operation causes the operating system (MCPII) to execute a control instruction within the operating RPG object program. The system command that can be specified is system dependent. Refer to the B 1000 System Software Operation Guide, Volume 1.

ZIP Operation Code

The ZIP operation code causes the operating system (MCPII) to execute the control instruction specified in the FACTOR 2 field. The FACTOR 2 field can be a field name or vector name, previously defined, or a meaningful alphanumeric literal enclosed in apostrophes.* The operation can be conditioned by the conditioning indicator. All other fields must be left blank. The information contained in the FACTOR 2 field must be a valid B 1000 operating system command. The contents of the FACTOR 2 field must not exceed a maximum of 511 alpha characters.

The ZIP operation code can programmatically schedule object programs contained in the disk directory, or the ZIP operation code can perform any system command that can be perform through the Operator Display Terminal (ODT) or card reader.

In the example shown in figure 11-12, the field named 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, then the operating system retrieves JOB10 from the disk directory and places it in the system mix for subsequent operation. The program containing the ZIP operation code proceeds to the next sequential instruction following the ZIP operation, without waiting for the execution of the program named JOB10.

* When ZIPPING to a program do NOT use apostrophes. RA 10/7/92

```
                Calculation Specifications

01 C   02 03 04          ZIP  DATA
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 11-12. The ZIP Operation Using a FIELD NAME

The example of a literal in the FACTOR 2 field, shown in figure 11-13, causes the same sequence of operations.

```
                Calculation Specifications

01 C   02 03 04          ZIP  DATA
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 11-13. The ZIP Operation Using a Literal

Look-up Operations

The following paragraphs describe the look-up operations.

LOKUP

The LOKUP operation code searches a table or array for a particular data item. The table or array name is entered in the FACTOR 2 field. The search word is entered in the FACTOR 1 field.

Search Word

The search word, sometimes called the search argument, is the data item for which the program tries to find a match in the table or array named in the FACTOR 2 field.

The search word in the FACTOR 1 field can be:

1. An alphanumeric or numeric literal.
2. A field name.
3. A table name.
4. A table element.
5. An array element.

When the FACTOR 1 field references a table name, it refers to the element in the table that was last selected in a previous LOKUP operation. It does not refer to the whole table.

When the FACTOR 1 field references a table element, it refers to the actual table element and not necessarily to the element in the table that was last selected in a previous LOKUP operation.

Resulting Indicators

The RESULTING INDICATORS fields must be used with the LOKUP operation code. These indicators specify the type of search to be performed. For example:

1. When an indicator is assigned to the LOW field (columns 56-57), the table or array is searched for an item that is nearest to, yet lower in sequence than, the search word. When the search is successful, the indicator is turned ON.
2. When an indicator is assigned to the HIGH field (columns 54-55), the table or array is searched for an item that is nearest to, yet higher in sequence than, the search word. When the search is successful, the indicator is turned ON.

3. When an indicator is assigned to the EQUAL field (columns 58-59), the table or array is searched for an item equal to the search word.

Indicator Assignment

At least one RESULTING INDICATOR field must be specified, and up to two RESULTING INDICATORS fields can be specified if desired. However, when two RESULTING INDICATORS fields are specified, one of them must be assigned to EQUAL field. The RESULTING INDICATORS fields can be assigned as follows:

EQUAL
EQUAL, HIGH
EQUAL, LOW
HIGH
LOW

A LOKUP operation code causes a syntax error when:

1. No resulting indicator is specified.
2. Indicators are assigned to both the HIGH and LOW fields.

Rules for LOKUP

The following rules apply to the LOKUP operation code:

1. At least one RESULTING INDICATOR field must be assigned.
2. The search word and the data items in the table or array must be of the same data type (alpha or numeric) and of the same length.
3. Decimal points are ignored for numeric data.
4. A table or array should be searched for high, low, high and equal, or low and equal only if it is specified as ordered on the Extension Specifications.
5. When multiple indicators are assigned, one of them must be assigned to the EQUAL field, with equal condition taking precedence. For example, searching for high or equal conditions, the equal condition takes precedence.

```
                Calculation Specifications

                                                        >   =
01 C            '5'       LOKUPARY,X                    20  21
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

      The array contains:  1 2 3 4 5 6 7

      Indicator 21 turns ON and value of the X field
      is 5, containing '5'.
```

6. Resulting indicators are turned ON only when the search is successful and, conversely, are turned OFF for an unsuccessful search.
7. When a LOKUP operation code is performed on a table or array, the search starts with:

   1) The first element of the table or array when the FACTOR 2 field contains an unsubscripted table or array name.
   2) The element reference when the FACTOR 2 field contains a subscripted table or array name.

8. A subscripted table name cannot be specified in the result field for a LOKUP operation code.

9. An array name cannot be specified in the RESULT FIELD field for a LOKUP operation code.
10. Short tables or arrays should only be searched for an equal condition because the compiler-supplied "filler" can cause the LOKUP operation code to produce unexpected results.
11. Any search for other than an equal condition (low or high, or low and equal, or high and equal) is allowed only if the vector was specified as having ascending or descending sequence on the Extension Specifications. The search algorithm assumes that the vector is in the specified sequence; if the programmer allows the vector to get out of sequence, the results can be undesirable.

Single Table LOKUP Operations

When the LOKUP operation code is specified for an item in a single table, entries must be made in the FACTOR 1 field (search word). The FACTOR 2 field (table to be searched) and at least one resulting indicator must be specified in the RESULTING INDICATORS fields to designate the type of search operation. In addition, control level and conditioning indicators can be specified.

When a table item is found that satisfies the type of search specified, that is, HIGH, LOW, or EQUAL fields, then a copy of that table item is stored in a special hold field, and the appropriate resulting indicator is turned ON. With each successful LOKUP operation, the contents of the table element is stored in the special hold field, thereby overwriting the information that was previously stored there. When the LOKUP operation is unsuccessful, the contents of this special hold field are left unchanged.

For a successful LOKUP operation, the internal pointer into the table specified in the FACTOR 2 field points to the element that satisfied the condition of the LOKUP operation. If the LOKUP operation is not successful, no resulting indicators are turned ON and the internal pointer into the table specified in the FACTOR 2 field is set to the first element in the table.

Two Table LOKUP Operation

When the LOKUP operation code is specified with two related tables, only one table is actually searched. Entries must be made in the FACTOR 1 field (search word), the FACTOR 2 field (table to be searched), the RESULT FIELD field (contains the name of the related table from which data is made available), and at least one of the HIGH, LOW, or EQUAL RESULTING INDICATORS fields. In addition, control level and conditioning indicators can be specified.

When the LOKUP operation code is specified with related tables, the two tables must be capable of holding an equal number of elements. Whenever the searched table (the FACTOR 2 entry) is longer than the related table (the RESULT FIELD entry), then the LOKUP operation is terminated when the end of the shorter table is reached.

Following a successful LOKUP operation, the proper resulting indicator is turned ON. The internal pointers into the tables specified in the FACTOR 2 and RESULT FIELD fields are set to point to the corresponding elements, and the contents are moved into the special hold area. If the LOKUP operation was unsuccessful, no resulting indicators are turned ON and the special hold areas remain unchanged. The internal pointers in each table are set to point to the first element of each associated table.

Referencing Unsubscripted Table Entries

When an unsubscripted table name is specified in any operation except LOKUP, the table name refers to the information that was stored in the special hold area by the last successful LOKUP operation. Table items can then be referenced when performing other operation codes.

When an unsubscripted table name is specified in the RESULT FIELD field of an operation code other than the LOKUP operation code, the information that was stored in the special hold area is changed by the calculation operation. In addition, the actual corresponding table element in the table is changed. This is one way of changing the contents of a table during program execution.

When an unsubscripted table name is specified in the FACTOR 1 field (search word) of a LOKUP operation code, the contents of the special hold area is used. The result of the last successful LOKUP operation can be used as the search word or argument for other LOKUP operations.

When an unsubscripted table name is specified for an operation code prior to a successful LOKUP operation, the current contents of the special hold area is referenced. At beginning of job for the program, the internal pointer is initialized to one and the the special hold area is initialized to zeros if the field is numeric or to blank characters if the field is alphanumeric.

Referencing Subscripted Table Entries

When a subscripted table name is specified with any operation code, the information referenced is the actual entry in the table. The contents of the special hold area are not affected in any way.

LOKUP Operation with an Array

Specifying the LOKUP operation code with an array is the same as for a table, except that when an array is specified in a two-vector LOKUP operation, the array name must appear in the FACTOR 2 field, and only an unsubscripted table name can be specified in the RESULT FIELD field.

The following rules apply to the handling of arrays when specified in the LOKUP operation code.

1. Arrays do not have special holding areas as do tables; therefore, when a LOKUP operation is successful, the indicators only reflect that the information is in the array. The information found is not immediately available to the program as with LOKUP operations with tables.
2. When referencing an array, if only the array name is specified (no subscript), the LOKUP operation begins with the first element in the array.
3. When referencing an array and a subscripted array name is specified, the LOKUP operation begins at the element specified by the subscript. The subscript can be a numeric field name or a literal. When a LOKUP operation is successful, the appropriate resulting indicator is turned ON. The type of subscript specified has the following effect:

    1) When the subscript is a literal, a successful LOKUP operation only verifies that the information is in the array. The actual information or its location within the array is not available to the program as with LOKUP operations on tables.
    2) When the subscript references a numeric field name, a successful LOKUP operation verifies that the information is in the array. In addition, the element number of the array in which the information is contained is automatically placed in the field that originally contained the beginning subscript. The information can now be made available to the program again by referencing the array using the same field as the subscript.

When the LOKUP operation is unsuccessful, the results are as follows:

1. For an unsubscripted array, all resulting indicators are turned OFF.
2. For an array subscripted with a literal, all resulting indicators are turned OFF and the literal is not changed.
3. For an array subscripted by a numeric field name, all resulting indicators are turned OFF and the value of the numeric field name is set to a value of 1.

LOKUP Operation Algorithm

The algorithm used for the LOKUP operation code is table 11-8. The array or table type references the letter A or D entries in columns 45 or 57 of the Extension Specifications. These entries determine the sequence of the information, which is either ascending (the letter A) or descending (the letter D). A LOKUP operation for any search relation other than EQUAL requires that the table or array must be in either ascending or descending sequence order.

Table 11-8. LOKUP Operation Code Algorithm

Array or Table Type	LOKUP Search Relation	Algorithm for Actual Search Performed by the RPG Interpreter
Ascending	LSS	GTR or EQL
Ascending	LEQ	GTR
Ascending	EQL	EQL
Ascending	GEQ	GEQ
Ascending	GTR	GTR
Descending	LSS	LSS
Descending	LEQ	LEQ
Descending	EQL	EQL
Descending	GEQ	LSS
Descending	GTR	LSS or EQL

The LOKUP search relation references the entries made in the RESULTING INDICATORS fields (columns 54-59) of the Calculation Specifications. The entries made in these fields specify the type of LOKUP operation to be performed as specified in the following table:

LOKUP Search Operation	Definition
LSS	Less than
LEQ	Less than or equal to
EQL	Equal to
GEQ	Greater than or equal to
GTR	Greater than

The algorithm references the type of search actually performed by the RPG interpreter for each type of LOKUP operation.

Example:

An ascending table is to be searched for an item that is less than (LSS) a particular value. The table is searched for the first item that is equal to (EQL) or greater than (GTR) the particular value. If the LOKUP operation is successful, the RPG interpreter then references the table item immediately preceding the found item.

When a table or array is not specified as being in sequence, the only LOKUP operation that is permitted is a search for an EQUAL condition. Each element in the table is compared with the data item specified for an equal comparison.

Subroutine Handling

Subroutines in RPG are used to combine calculation operations in a logical order. Special operation codes are provided to delimit the beginning and end of a subroutine, and to call a subroutine for execution from some point in the program. Specifications lines within a subroutine must contain the letters SR, OR, AN, or blank in the CONTROL LEVEL field (columns 7-8), and all subroutines must be the last operations specified in the Calculation Specifications. The recursive use of subroutines is allowed but not recommended, for example, the operation EXSR SUB1 with the SUB1 subroutine. Subroutines cannot be nested in the program. For example, the BEGSR operation code cannot appear in a subroutine. Refer to figure 11-14 for an example showing the use of subroutines in a program.

```
                  Calculation Specifications

01 CL1                        EXSR SUB1
02 C                            .
03 C                            .
04 C                            .
05 CSR            SUB1        BEGSR
06 CSR                          .
07 CSR                          .
08 CSR                          .
09 CSR                        EXSR SUB2
10 CSR                          .
11 CSR                          .
12 CSR                          .
13 CSR                        ENDSR
14 CSR            SUB2        BEGSR
15 CSR                          .
16 CSR                          .
17 CSR                          .
18 CSR                        ENDSR
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 11-14. Subroutine Coding Example

The following paragraphs describe the BEGSR, ENDSR, and EXSR operation codes used in specifying subroutines.

BEGSR Operation Code

The BEGSR operation code specifies the beginning of a subroutine. The FACTOR 1 field must contain the name of the subroutine, which must follow the rules for the formation of labels as described in section 2. Columns 33-59 and 24-27 must be left blank.

ENDSR Operation Code

The ENDSR operation code specifies the end of a subroutine. The FACTOR 1 field can contain a label. This label is used like the TAG operation code as a point for a GOTO operation code within the subroutine to branch, thus allowing an exit from different locations within the subroutine. Columns 33-59 and 24-27 must be left blank.

EXSR Operation Code

The EXSR operation code specifies the subroutine to perform and can appear anywhere within the Calculation Specifications. When execution of the subroutine is completed, control returns to the next line following the EXSR operation code. The EXSR operation code can be specified within a subroutine.

The EXSR operation code can be conditioned by an indicator, allowing the subroutine to be performed (called) only when all the conditions are satisfied. The FACTOR 2 field must contain the name of the subroutine being performed, which must be the same name entered in the FACTOR 1 field of a BEGSR operation code. The FACTOR 1 field and columns 39-59 must be left blank.

PROGRAMMED CONTROL OF INPUT AND OUTPUT

Within the normal B 1000 RPG program cycle, a record is read, calculations are performed (using the data from that input record), and an output record is written. The CHAIN, CLOSE, DELET, DSPLY, EXCPT, OPEN, READ, RECV, and SEND operation codes allow greater control over input and output, and provides the capability to read and write records at times other than those normally available as part of the RPG program cycle. The FORCE and SETLL operations permit some programmatic control over the selection of records for processing.

CHAIN Operation Code

The CHAIN operation code provides a method of accessing a data file on disk in an order other than the physical sequence of the records. The CHAIN operation code can be specified anywhere in the Calculation Specifications in columns 28-33.

When the CHAIN operation code is specified, the following requirements must be satisfied:

1. The FACTOR 1 field must contain a key specifier which cannot be a whole array.
2. The FACTOR 2 field must contain the name of the file the program is chaining to.
3. An indicator should be specified in columns 54-55 to turn ON if the record is not found. If it is desired that the systems operator be informed when no record is found, the field can be left blank.

When the CHAIN operation code is specified, the following fields must be left blank:

1. FIELD LENGTH
2. HALF ADJUST
3. DECIMAL POSITIONS
4. RESULTING INDICATORS – EQUAL

When the CHAIN operation code is specified, optional entries are allowed in the following fields:

1. RESULT FIELD
2. CONTROL LEVEL INDICATOR
3. CONDITIONING INDICATORS

The RESULT FIELD field is used for index-sequential ($ IXSEQ) files and specifies which key to use for the search. If the RESULT FIELD field is left blank, the access key (key specified in the ACC field on the File Description Specification, or if none, the key on the File Description Specification) is used.

When the chained file is conditioned with an external indicator, the CHAIN operation should be conditioned with the same external indicator. If this is not done, the CHAIN operation is automatically suppressed when the external indicator is OFF in order to avoid changing the status of the indicator specified in columns 54-55.

If column 31 on the File Description Specifications contains either the letter A or K and FACTOR 1 is a numeric identifier, the key comparison is performed in numeric mode. If more than one chaining operation is performed on the same file and any one of the operations is numeric, then all chaining operations on that file are numeric.

The following paragraphs describe how the CHAIN operation code functions for each of the applicable file types.

Chained Indexed Files

The field length of the FACTOR 1 field must be the same as the key length specified for the file in the File Description Specifications. Note that if the FACTOR 1 field is a literal, leading zeros or trailing blank characters are required to make the field length and key length equal. The file name entered in the FACTOR 2 field must have been defined in the File Description Specifications as a chained disk file. If the file name specified in the FACTOR 2 field was defined in the File Description Specifications with the letter P or N in column 31, then the FACTOR 1 field must be numeric.

Input Files

The value of the data specified in the FACTOR 1 field is compared with the index in order to locate the data record whose key field contains the same value as the FACTOR 1 field. All specified record-identifying indicators and field indicators are set appropriately.

If an equal condition is not found in the index, the indicator specified in columns 54-55 is set ON. If no indicator is specified, the system operator is notified by a message on the ODT.

Update Files

A chained indexed update file is processed in the same manner as chained indexed input files, with the following exception: when a record is found, the record location is retained. This allows the record to be written back into the same location from which it was read without having to chain to the record again.

Output Files

Output files are not allowed.

Chained Direct Files

The FACTOR 1 field must be numeric and the FACTOR 2 field must be defined in the File Description Specifications as a chained direct file.

Record Identification in Direct Files

When chaining to a direct file, if the key is equal to or greater than 1 and equal to or less than the total number of records in the file, there is a found condition. Such a situation can present problems.

In a file with a potential of more records than have actually been written, there are gaps in the file. For example, records 1, 5, and 1000 are all that a 1000-record, direct file currently contains. Under these conditions, when chaining to relative record number 6, a record found condition implies that the key is greater than 0 and less than or equal to the potential number of records in the file.

To ensure that the record being read is, in fact, a record and not an empty location, the programmer must take care to design proper record identifying codes in the direct files.

Input Files

The value of the data specified in the FACTOR 1 field must be a numeric (literal or field name) integer between 1 and n, where n is not greater than the total number of records in the file and is the relative record number of the desired record.

If the key (numeric integer) is within the bounds of the file, the record is found. Refer to Record Identification in Chained Direct Files in this section for additional information. If the key is beyond the bounds of the file, the indicator specified in columns 54-55 is turned ON; however, if no such indicator is specified, the operator is notified by a message on the console printer.

Update Files

The chained direct update file is processed in the same manner as the chained direct input file, with the following exception: When a record is found, the record location is retained. Therefore, the record can be written back into the location from which it was read without chaining again.

Output Files

In order to create a direct file, it must be specified in the File Description Specifications as output and chained in columns 15-16. At program execution time, when the key (relative record number) is used to chain, the record location specified is made available for the writing of the record as defined on the Output-Format Specifications. Refer to Record Identification in Direct Files in this section for additional information.

The highest key (relative record number) that is written when the file is first created determines the maximum file size.

Chained Sequential Files

Any disk file can be accessed as a sequentially created disk file by providing FACTOR 1 of the chain with relative record numbers for keys. To accomplish this, the disk file must also be defined on the File Description Specifications as a chained direct file, regardless of how it was originally created. However, this is an unusual procedure as it requires extensive knowledge of the location within the file of particular records.

Input Files

Chained sequential input files are handled the same as chained direct input files. Refer to the subsection titled Chained Direct Files in this section for detailed information.

Update Files

Chained sequential update files are handled the same as chained direct update files. Refer to the subsection titled Chained Direct Files in this section for detailed information.

CLOSE Operation Code

The CLOSE operation code explicitly closes a file with a file designation of chain, indicated by the letter C in the FILE DESIGNATION field (column 16) of the File Description Specifications. If any records are added to the file, a sort operation is performed every time the file is closed. If the CLOSE operation code is specified for a file that does not have a file designation of chain, the RPG compiler generates the following syntax error.

CLOSE NOT VALID FOR THIS FILE TYPE

The CLOSE operation code is only valid for disk files.

The FACTOR 2 field contains the name of the disk file that is to be closed.

The FACTOR 1 and RESULT FIELD fields must be blank.

The close type defaults to the type of close specified in the TAPE REWIND field (column 70) of the File Description Specifications.

When the file is conditioned with an external indicator (U1-U8), the CLOSE operation code should be conditioned with the same external indicator. If the CLOSE operation code is not conditioned by the external indicator, the program can abort. The CLOSE operation code is suppressed when the external indicator is off.

Figure 11-14a shows one way in which the OPEN and CLOSE operation codes can be used. This example program adds records to an existing file. If the number of added records reaches 100, the program closes the file and then reopens the file. This causes the tag file to be sorted/reorganized to include the added records, and additions to the file can be made more quickly.

```
00100 $ SECURE
00200H
00300FTRANS     IPE  1800  90                    DISK            U
00400 $ TAG
00500 $ RPERA          500
00600FCHAIN     UC   1800  90      5AI         1 DISK            L
00700FDISPLAY   D      50  50                    CONSOLE
00800ITRANS     NS  01   1 CA
00900I                                                 1      1 TYPE
01000I                                                 2      6 KEY
01100I                                                 7     90 DATA
01200ITRANS     NS  02   1 CC
01300I                                                 1      1 TYPE
01400I                                                 2      6 KEY
01500I                                                 7     90 DATA
01600ITRANS     NS  03   1NCA       1NCC
01700I                                                 1     90 DUMMY
01800ICHAIN     NS  04
01900I                                                 1     90 RECORD
02000C  N99                 Z-ADDO                  COUNT      50
02100C  N99                 SETON                              99
02200C   01                 EXSR ADDSUB
02300C   02                 EXSR CHGSUB
02400C   03                 EXSR INVSUB
02500C* SUBROUTINE TO ADD RECORDS TO THE CHAIN FILE.  IF THE NUMBER OF
02600C* ADDED RECORDS IS GREATER THAN 100, THE FILE IS CLOSED AND
02700C* REOPENED.
02800CSR        ADDSUB      BEGSR
02900CSR        KEY         CHAINCHAIN                         10
03000CSRN10     •BAD ADD'   DSPLYDISPLAY
03100CSR 10     COUNT       ADD  1      COUNT
03200CSR 10     COUNT       COMP 100                           20
03300CSR 20                 CLOSECHAIN
03400CSR 20                 OPEN CHAIN
03500CSR 20                 Z-ADDO      COUNT
03600CSR                    ENDSR
03700C* SUBROUTINE TO CHANGE AN EXISTING RECORD IN THE CHAIN FILE.
03800CSR        CHGSUB      BEGSR
03900CSR        KEY         CHAINCHAIN                         10
04000CSR 10     •BAD-CHG'   DSPLYDISPLAY
04100CSR                    ENDSR
04200C* SUBROUTINE TO DISPLAY AN INVALID RECORD TYPE.
04300CSR        INVSUB      BEGSR
04400CSR        •INV-REC'   DSPLYDISPLAY
04500CSR                    ENDSR
046000CHAIN   DADD      01
047000                          KEY        5
048000                          DATA      90
049000CHAIN   D         02
050000                          KEY        5
051000                          DATA      90
----*----1----*----2----*----3----*----4----*----5----*----6----*----7--
```

Figure 11-14a. Example Program Using OPEN and CLOSE Operation Codes

Example ODT Messages from Executing Example Program:

```
(RPG) (RPG)/OPENO =83 BOJ. PP=4, MP=4 TIME = 14:58:44.1
(RPG) SORT/VSORT =84 BOT. (#83) PP=4, MP=4 TIME = 14:58:54.8
"U/(RPG)/TAGCHAIN" REMOVED
% (RPG) SORT/VSORT =84 : 100 RECORDS SORTED
(RPG) SORT/VSORT =84 EOT. (#83) TIME = 14:59:10.1
(RPG) sort/VSORT =85 BOT. (#83) PP=4, MP=4 TIME = 14:59:19.8
"U/(RPG)/TAGCHAIN" REMOVED
% (RPG) SORT/VSORT =85 : 200 RECORDS SORTED
(RPG) SORT/VSORT =85 EOT. (#83) TIME = 14:59:43.3
(RPG) SORT/VSORT =87 BOT. (#83) PP=4, MP=4 TIME = 14:59:53.7
"U/(RPG)/TAGCHAIN" REMOVED
% (RPG) SORT/VSORT =87 : 300 RECORDS SORTED
(RPG) SORT/VSORT =87 EOT. (#83) TIME = 15:00:11.5
(RPG) SORT/VSORT =89 BOT. (#83) PP=4, MP=4 TIME = 15:00:50.2
"U/(RPG)/TAGCHAIN" REMOVED
% (RPG) SORT/VSORT =89 : 500 RECORDS SORTED
(RPG) SORT/VSORT =89 EOT. (#83) TIME = 15:01:05.4
(RPG) (RPG/OPENO =83 EOJ. TIME = 15:01:11.6
```

DELET Operation Code

The DELET operation code deletes either the current record or the record specified by a key value for an index-sequential ($ IXSEQ) chain file.

The FACTOR 1 field can contain the value of the key to be deleted. If the FACTOR 1 field is left blank, the current record in the index-sequential (IXSEQ) chain file is deleted.

The FACTOR 2 field contains the name of the index-sequential file and must be an index-sequential ($ IXSEQ) chain file.

The RESULT FIELD field can optionally contain the key which is to be deleted. The FACTOR 1 field must not be blank.

The HIGH field (columns 54-55) in the RESULTING INDICATORS field can contain an indicator. If the delete operation is successful, that is, there is a record to delete, then this indicator is set off. If there is no record to delete, this indicator is set on. If no indicator is specified and there is no record to delete, the following message is displayed on the ODT and the program is suspended and is waiting for a response from the operator.

```
<program-name> = <program-mix-number> : DELERR
```

If the operator desires to continue the program, the following command can be entered:

```
<program-mix-number>AXGO
```

If the operator desires to terminate the program, the following command can be entered:

```
<program-mix-number>AXSTOP
```

The following Calculation Specifications causes the current record of an index-sequential chain file to be deleted.

```
01000C                      DELETFILENAM                10
----*----1----*----2----*----3----*----4----*----5----*----6----*----7--
```

Indicator 10 is set on if there is no record to be deleted.

The following Calculation Specifications causes the record whose access key is equal to AAAAA in the index-sequential chain file FILENAM to be deleted.

```
02000C            •AAAAA'   DELETFILENAM                20
----*----1----*----2----*----3----*----4----*----5----*----6----*----7--
```

Indicator 20 is set on if there is no record to be deleted.

The following Calculation Specifications causes the record whose key is equal to 11111 by way of the key ALTKEY of an index-sequential chain file FILENAM to be deleted.

```
02000C            •11111'   DELETFILENAM   ALTKEY      30
----*----1----*----2----*----3----*----4----*----5----*----6----*----7--
```

Indicator 30 is set on if there is no record to be deleted.

DSPLY Operation Code

This operation causes data to be displayed on the Operator Display Terminal (ODT) or provides for certain low volume data entries to be entered by the system operator through the console printer. The file name of the file assigned to the console printer must be entered in the FACTOR 2 field. The HALF ADJUST and RESULTING INDICATORS fields must be left blank. Control level and conditioning indicators can be assigned. The FACTOR 1 field is optional and can be used to name a data item, which can be a field name or a vector element, or to specify a literal (numeric or alphanumeric). An unindexed table name must not be specified in the FACTOR 1 field. The RESULT FIELD field is optional and can be used to specify the name of a data item (field name or vector element), or an array name. An unindexed table name must not be specified in the RESULT FIELD field. If the RESULT FIELD field is used, the system operator must respond on the console printer. If both the FACTOR 1 and the RESULT FIELD fields are blank, a syntax error is emitted.

The DSPLY operation code functions in the following manner:

1. If the RESULT FIELD field is blank and the FACTOR 1 field contains a data item, the data item is printed on the console printer and the program proceeds to the next operation. Refer to figure 11-18, line 2.
2. If the RESULT FIELD field contains a data item and the FACTOR 1 field is blank or contains a data item, the data is printed on the console printer and an accept (AX) message is generated by the operating system (MCPII), to which the system operator must respond. The contents of the response are placed in the field specified in the RESULT FIELD field. For the format of the accept (AX) message, refer to the B 1000 Systems System Software Operation Guide, Volume 1. Refer to figure 11-15, line 5.
3. If both the RESULT FIELD and FACTOR 1 fields contain data items, the data from both is printed on the console printer and an accept (AX) message is generated by the operating system (MCPII), to which the system operator must respond. The contents of the response are placed in the field specified in the RESULT FIELD field. Refer to figure 11-15, line 8.
4. Display output is crunched by the operating system (MCPII).

```
            Calculation Specifications

01 C   10      FIELDA    DSPLYDSPYOUT
02 C*
03 CL2 20                DSPLYDSPYOUT   FIELDA
04 C*
05 C   30      FIELDA    DSPLYDSPYOUT   FIELDB
--*----1----*----2----*----3----*----4----*----5----*----6----*----
```

Figure 11-15. DSPLY Operation Coding Examples

DSPLY Operation with an Array

When an entire array is loaded or dumped with the DSPLY operation code, only one element of the array is loaded or dumped with each accept (AX) message. For example, to load an 8-element array would require entering eight accept (AX) messages. These extra accept (AX) messages are generated by the RPG compiler when the entire array is specified.

When a single element of an array is specified, a single accept (AX) message is generated.

EXCPT Operation Code

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 is written, depending on conditioning indicators. If this operation is specified, Exception Time Output Specifications are required. The EXCPT operation code can have CONTROL LEVEL fields and conditioning INDICATORS fields assigned; all other fields must be left blank.

FORCE Operation Code

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 that supplies the next record for processing.

The force operation enables programmatic selection of the file from which the next record is to be taken for processing, and overrides the normal record selection process that occurs during input. The FORCE operation only selects the file that supplies the next record for processing and does not actually read any record during calculations.

The FORCE operation applies only to input, update, or combined files designated as PRIMARY or SECONDARY files. FACTOR 2 must contain the name of the file to be forced; all other fields (except conditioning INDICATORS) must be left blank. This operation must not occur within total calculations or in subroutines.

The MR indicator is always off while a forced record is being processed, and the forced record is treated as if it had no match fields specified.

The programmer should exercise care when specifying more than one FORCE operation code during the same program cycle. If more than one FORCE operation code is conditioned to occur, only the last one affects record selection, and all of the others are ignored. If the forced file is at end of file, normal record selection determines the next record to be selected for processing.

OPEN Operation Code

The OPEN operation code explicitly opens a file with a file designation of chain, indicated by the letter C in the FILE DESIGNATION field (column 16) of the File Description Specifications. If the file is already open, the RPG compiler ignores the OPEN operation code. If the OPEN operation code is specified for a file that does not have a file designation of chain, the RPG compiler generates the following syntax error:

OPEN NOT ALLOWED FOR THIS FILE TYPE

The OPEN operation code is only allowed for disk files whose file type is input, output, or update, that is, the FILE TYPE field (column 15) of the File Description Specifications must contain either the letters I (input), O (output), or U (update). When the OPEN operation code is performed, the open operation defaults to the type of open specified in the FILE TYPE (column 15) and the FILE OPEN field (column 53) of the File Description Specifications.

If the OPEN operation code is specified for a file defined in the File Description Specifications as output, indicated by the letter O in the FILE TYPE field (column 15), a new file is created. Therefore, if the RPG program writes several records to this file, closes the file with the CLOSE operation code, and later opens the file, another file is created. The previous records are not saved in the new file.

If a file is not open and a read or write operation is performed for the file, then the operating system (MCPII) automatically opens the file.

If the file contains additions, is closed (tag file is sorted) and reopened,then the in-core table is reloaded using the new end-of-file (EOF) pointer. This reorganization feature can be useful when adding a large number of records to a file. If a large number of records have been added to a chain file and more records are to be added, then it is recommended that the file be closed and reopened as an intermediate setup during the addition process. If this is not done, the amount of time that the RPG program takes to locate a record significantly increases as each new record is added. This time can be critical in a data communications environment where a user at a terminal must wait for the program to search for a record prior to adding it to the file.

The internal mechanism used by the RPG program is to first search the existing file for the record. This is accomplished by using a very efficient binary search technique. If the record is not found, the RPG program then sequentially searches through all the records that have been added. A sequential search is performed in this stage because the records are stored in the order in which they were added. This sequential search can consume a significant amount of processor time, especially if there are a significantly large number of records that have been added.

Closing the file and then reopening it at various intervals prevents the long delays in adding records to the file. The programmer should be aware that once the file is closed, the program hangs and waits for the sort operation to complete. When the sort completes, the program is activated again and continues processing. In a data communications environment, it is recommended that the close operation be performed during non-peak times to avoid users waiting for the sort operation to complete.

When the file is conditioned with an external indicator (U1-U8), the OPEN operation code should also be conditioned by the same external indicator. The OPEN operation code is ignored when the external indicator is off.

Refer to the CLOSE operation code for an example program that illustrates one use of the OPEN operation code.

READ Operation Code

The READ operation code causes a record to be read from a demand file during calculations. This differs from the FORCE operation code because the FORCE operation code causes input during the next program cycle, whereas the READ operation code causes input during the current program cycle. It also differs from the CHAIN operation, because CHAIN is used to read records randomly, whereas the READ operation code reads records sequentially.

The FACTOR 2 field must contain the name of the file to be read. The FACTOR 1, RESULT FIELD, FIELD LENGTH, DECIMAL POSITIONS, and HALF ADJUST fields must be left blank. The READ operation code can be conditioned by the CONTROL LEVEL fields or by conditioning the INDICATORS field in columns 7-17.

Whenever the READ operation code is specified, an indicator should be specified in columns 58-59 of the Calculation Specifications. The indicator is turned ON when an end-of-file condition occurs for the indexed file. Any subsequent attempt to read from the same file causes the indicator to be turned ON. When no indicator is used and an end-of-file condition occurs, the program automatically displays an REOF message on the ODT. The system operator can then enter either STOP or GO on the console printer. If STOP is entered, the program terminates as if LR had occurred. If GO is entered, the program continues.

It is good programming policy to always use an indicator in columns 58-59 with the READ operation code.

The following rules must be observed when using the READ operation code:

1. Only demand files designated as INPUT, UPDATE, or COMBINED can be read by the READ operation code.
2. Sequence-checking in the Input Specifications is not allowed for demand files.
3. Control levels, matching fields, and look-ahead fields are not allowed for demand files.
4. If a demand file is conditioned by an external indicator (U1-U8) which is not set, the READ operation code is ignored (the end-of-file indicator in columns 58-59 is not set ON).
5. When read operations are performed on demand files for which sequential within limits processing has been specified, the entire file can be read as many times as desired even though end-of-file record has been read. Refer to the subsection titled Sequential Within Limits in section 4A for additional information about that type of file processing.

RECV Operation Code

This operation is used to read messages from REMOTE data communication files during calculations. Only files specified as REMOTE and designated as demand in the File Description Specifications can be accessed with the RECV operation.

The RECV operation code is similar to the READ operation code but allows additional indicators in columns 54-59 to cause the following to be reported to the RPG program:

1. Exception conditions.
2. Incomplete I/O operations.
3. End-of-file condition.

The FACTOR 2 field must contain the name of the remote file to be read. The FACTOR 1, RESULT FIELD, FIELD LENGTH, DECIMAL POSITIONS, and HALF ADJUST fields must be left blank.

Indicators can be specified in columns 54-59 and are described as follows:

Columns 54-55

An indicator can be specified in columns 54-55 to cause exception conditions to be reported. The indicator is turned on only when the following conditions are satisfied:

1. The established number of retries per read operation is attempted by the network controller without success, and
2. The TERMINATE ERROR statement is included in the NDL network controller and is invoked during the attempted RECV operation.

NOTE
When a TERMINATE ERROR condition occurs, the incoming message to the RPG program is not processed.

When a specified indicator in columns 54-55 is turned on, there are several possibilities that can be programmed by the user. Some of these possibilities are:

1. Re-issue a RECV operation for the same terminal.
2. Make an entry in a log file.
3. Display a message on the console printer to notify the B 1000 operator.
4. Stop attempting to read from a terminal if repeated errors occur, or go to end of job.

If no indicator is specified in columns 54-55 and a TERMINATE ERROR condition occurs, the message is not processed and the RPG program is not informed of the error unless the program has previously cleared the record identifying indicators.

Columns 56-57

This field can be used to specify the type of read discipline desired for a RECV operation. If an indicator is specified in columns 56-57, incomplete I/O operations are reported to the RPG program. If no message is queued on an attempted RECV operation, incomplete I/O is reported and control is returned to the RPG program.

If no indicator is specified in columns 56-57 and a RECV operation is executed, control is not returned to the RPG program until a message is read.

Following are several items that could be programmed for depending upon the results of incomplete I/O reporting:

1. Notify the operator that specific terminals do not respond.
2. Send inquiry messages to the terminals.
3. Have the RPG program perform other functions before attempting to read the message again.
4. Determine if there is activity on the data communications transmission lines.

Columns 58-59

This field is used to indicate if end-of-file conditions are to be reported to the RPG program. The end-of-file indicator can be turned on by:

1. Entry of the Quit Controller (QC) input message on the console printer by the B 1000 operator, or
2. An end-of-file message placed in the queue assigned to the file by a controlling Message Control System (MCS).

SEND Operation Code

The SEND operation code writes records (messages) to REMOTE data communications files during calculations. Only files specified as device type REMOTE and designated as demand in the File Description Specifications can be accessed with the SEND operation.

The FACTOR 2 field must contain the name of the REMOTE file to which the message is to be sent. The FACTOR 1, RESULT FIELD, FIELD LENGTH, DECIMAL POSITIONS, and HALF ADJUST fields must be left blank.

The SEND operation code is similar to the EXCPT operation code but allows additional indicators in columns 56-59 to cause the following to be reported to the RPG program:

1. Incomplete I/O operations.
2. End-of-file conditions if an invalid key is used.

The indicators that can be specified in columns 56-59 are described as follows:

Columns 56-57

This field can be used to specify the type of write discipline desired for a SEND operation code. If an indicator is specified in columns 56-57 and the SEND operation code is attempted when the message queue is full, then incomplete I/O condition is reported to the RPG program. The output message queue size is controlled by the network controller.

Several actions that could be taken following incomplete I/O reporting are:

1. The RPG program can send the message again.
2. The operator can be notified that a terminal does not respond to inquiry.
3. Stop attempting to send messages to a particular terminal.

Columns 58-59

This field can be used to indicate if end-of-file conditions are to be reported. If an indicator is specified in columns 58-59, end of file is reported for output and combined (input and output) files if an invalid key is used for a SEND operation.

If no indicator is specified in columns 58-59, end of file is not reported and use of an invalid key results in the display of an program abort message on the ODT.

Prior to the first SEND operation code, a value must be moved to the station number field (columns 22-27 on the Telecommunications Specifications Line). This value is three characters long and represents the relative station number in the file of logical stations defined in the network controller program.

Prior to every SEND operation code, a value must be moved to the message length field (columns 28-33 on the Telecommunications Card Specifications line). This value is four characters long and represents the largest ending position value defined on the Output-Format Specifications line for the REMOTE file.

Figure 11-16 shows an example program that writes the message 'HI THERE' to the station from which the program was executed:

```
            Control Card Specifications

01 H
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


          File Description Specifications

01 FPRIMARY IPE F 180 180            DISK
02 FREMFILE 0   F19201920            REMOTE
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


      Telecommunications Card Specifications

01 TREMFILE  001002STATONMESLEN
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


             Input Specifications

01 IPRIMARY AA  01
02 I                                        1   1 DUMMY
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


            Calculation Specifications

01 C                    MOVE '001'     STATON  3
02 C                    MOVE '0010'    MESLEN  4
03 C                    SEND REMFILE
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


             Output-Format Specifications

01 OREMFILE E        01
02 O                                    10 'HI THERE'
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 11-16. Example RPG Showing SEND Operation Code

SETLL Operation Code

When this operation is specified in the Calculation Specifications, any indexed file that is designated as a demand file can be processed sequentially within limits.

WARNING

An indexed file designated as a demand file cannot be processed sequentially within limits by both a limits file and the SETLL operation code within the same program.

When the SETLL operation code is specified, the absolute value contained in a field or literal is used to establish the lower limit (bound) of the record key for the indexed file that is to be processed sequentially within limits. The upper limit (bound) automatically defaults to the highest record key that exists in the file.

The FACTOR 1 field must contain a field name, a vector element, a table name, or a literal representing the value of the lower limit being set. A whole array is not allowed. The type (alphanumeric or numeric) and length of the FACTOR 1 field should be equal to the type and length of the key for the file named as the FACTOR 2 field.

The FACTOR 2 field must contain the name of the file for which the lower limit is to be set. The named file must be specified on the File Description Specifications as an indexed demand file processed within limits but must not occur on Extension Specifications. For example, no record address file can be associated with this file.

The RESULT FIELD field can optionally contain the name of the alternate key in which to access the index-sequential ($ IXSEQ) file. If the RESULT FIELD field is left blank, the access key is used, that is, the key with ACC in columns 54-56 of the File Description Specifications. If no access key is specified, the key declared on the File Description Specifications for the index-sequential file is used.

Columns 49-59 of the Calculation Specifications must be left blank. Columns 7-17 can be used in the normal way.

Whenever the indexed demand file named as FACTOR 2 is read using the READ operation code, records are read sequentially by key. The SETLL operation code sets the lower key limit for the file. The next record read is the record whose key is next in sequence but greater than the lower key limit. Successive records are read in ascending key sequence until either end of file occurs or another SETLL operation code is executed.

Refer to section 4A of this manual for additional information concerning the SETLL operation code and file processing within limits.

TIME Operation Code

The TIME operation code provides the system time. The system date is provided as an option. System time and system date, as referred to here, are the dynamic time and date provided by the system and can change during the execution of an RPG program.

To use this operation, enter the TIME operation code in columns 28-32 of the Calculation Specifications. The RESULT FILED field, columns 43-48, must contain the name of a six-or twelve-digit numeric field which can be defined elsewhere in the program or in columns 49-52 of the Calculation Specifications. The numeric field named in columns 43-48 cannot be a whole array. The FACTOR 1, FACTOR 2, HALF ADJUST, and RESULTING INDICATORS fields must remain blank.

After this operation is performed:

1. A 6-digit Result Field contains the system time in the format hhmmss, where hh represents hours, mm represents minutes, and ss represents seconds.
2. A 12-digit Result Field contains both the system time and the system date, with the format of the system date being dependent on the entry in the Inverted Print field (column 21) of the Control Card Specification.

If the Inverted Print field contains a blank, the Result Field is provided in the format hhmmssmmddyy.

If the Inverted Print field contains a D, I, or J, the Result Field is in the format hhmmssddmmyy, where hh represents hours, mm represents minutes, ss represents seconds, mm represents month, dd represents day, and yy represents year.

SECTION 12
OUTPUT-FORMAT SPECIFICATIONS

The Output-Format Specifications state where a record is to be written. Within the file where the record is to be written, these specifications also define when to write, what data is to be written, and the location and format of the data within the record.

Depending on the peripheral equipment available, RPG can be used to write on line printers, character printers, tapes, disks, cards and remote devices. Although any of these devices can be used to create new files or records, disks are suitable for changing a record in place without changing the location of the record in the file. Therefore, the Output-Format Specifications can describe all of, or part of, a record.

The Output-Format Specifications are used to define where to write. Device in the File Description Specifications names the peripheral and filename associates a uniquely devised mnemonic with that device.

The Output-Format Specifications are functionally divided into the following two sections:

1. Record description, consisting of columns 7-31.
2. Field description, consisting of columns 23-70.

RECORD DESCRIPTION SECTION

The record description section contains the following information:

1. Where the record is to be written – filename, columns 7-14.
2. When the record is to be written – file type, column 15.

    1) Detail output, enter the letter D. Enter the letter H for heading information.
    2) Total output, enter the letter T.
    3) Exception output, enter the letter E. Exception output is written only as a result of the EXCPT operation code and can be executed during either detail calculations or total calculations.

3. Output conditioning indicators, columns 23-31. These indicators are used with the letters H, D, T, and E of column 15 to further control when the record is to be written.

FIELD DESCRIPTION SECTION

The field descriptions contain the following information:

1. When to write the data on the record. Output conditioning indicators, columns 23-31.
2. What data to write on the record.

    1) Variable name, columns 32-37.
    2) Edit code, columns 38, punctuate the data written.
    3) Blank after, column 39, clears the field after the write.
    4) Constant or edit word, columns 45-70, punctuates and/or specifies constant information to be written in the record.

3. Relative location where the data is to be written in the record-end position, columns 40-43.
4. Format of data to be written in the record.

    1) Edit codes, column 38, punctuate numeric variables.
    2) Packed, column 44, can specify packed format of numeric variables.
    3) Binary, column 44, can specify binary format of numeric variables.
    4) Constant or edit word, columns 45-70, specifies constant data and punctuation and constants, if required, of numeric variables.

Since several of the Output-Format Specifications fields have multiple usages which are conflicting, they are discussed in the text of this section.

Figure 12-1 illustrates and describes the Output-Format Specifications form.

The first Output-Format Specifications must be a record type description. Field description entries must start one line below the associated record type descriptions. A warning is emitted if a record type description has no associated field descriptions.

FIELD DEFINITIONS

Refer to figure 12-1 in conjunction with the following field definitions for the Output-Format 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

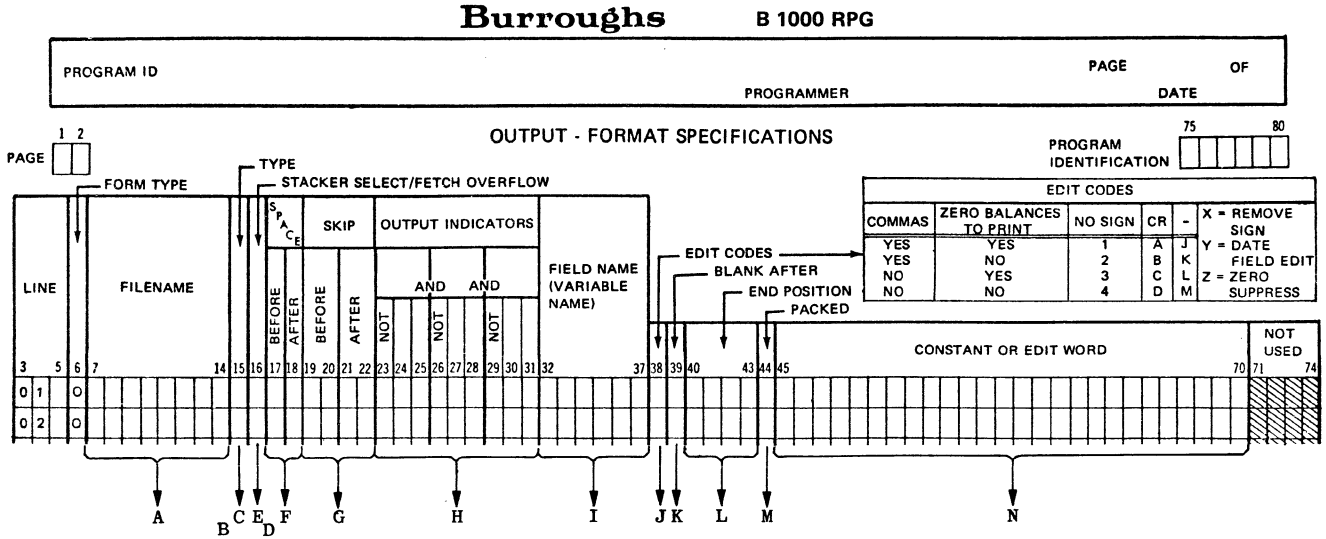
The letter O must appear in this field.

7-14 FILENAME

The FILENAME field specifies the name of the file to which the subsequent record type and field description entries belong. The file specified must have been previously described on the File Description Specifications form as output, update, combined, or input, with an A in column 66 on the File Description Specifications and ADD specified in columns 16-18 of the Output-Format Specifications. Every output file described in the File Description Specifications should also be described on the Output-Format Specifications form, but this is not required.

If this entry is blank on a record type description, the filename of the previous record is assumed. The first record type description must not have a blank filename entry.

For update files, only the fields to be changed must be specified in an output record. The remainder of the update record remains unchanged.



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 or deleted from an index sequential file. Entries: Blank, ADD, or DEL.

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, 0A-0G, 0V, 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 12-1. Output-Format Specifications Summary Sheet

15 TYPE

The TYPE field specifies when an output record is to be written. These records contain the following information:

1. Heading records contain such information as page headings and are treated as if they were detail records.
2. Detail records usually contain some type of data obtained directly from input records and calculation operations. Detail records are written once during every cycle, depending upon conditioning indicators.
3. Total records usually contain totals accumulated from a group of input records. Total records are written only during cycles in which a control break occurs, depending upon conditioning indicators.
4. Exception records are written during calculation time through use of the EXCPT or SEND operation codes.

Valid entries for the TYPE field are:

Entry	Definition
H	Heading records.
D	Detail records.
T	Total records.
E	Exception records (written during calculation time).

Record types can be specified in any order. However, if all output indicators are satisfied, the sequence of output is the order specified on the Output-Format Specifications, as follows:

1. Heading and detail output. Each cycle including 1P (first page) output on the first cycle only.
2. Exception output. During total calculation except during the first cycle.
3. Total output. Each cycle except the first.
4. Overflow output. Each cycle when a paper overflow condition has occurred.
5. Exception output. During detail calculations.

First page (1P indicator ON) output is heading or detail output conditioned with the 1P indicator and only occurs at the beginning of program execution because the 1P indicator is permanently turned OFF after the first detail output.

Overflow output is heading detail, or total output records conditioned with an overflow indicator.

For ease of writing and subsequent program maintenance, one of the following two methods is usually used for determining output record sequence:

1. The records within one file are specified, beginning with heading records and continuing with detail records, total records, and exception records.
2. The programmer can specify headings for all files, detail records for all files, total records for all files, and exception records for all files.

The two methods of determining the sequence of output specifications can be varied as required, so that the programmer can ensure the proper physical sequence of the records in the files.

Exception records can be specified for a combined file, although this is not recommended for files other than REMOTE. Exception output records conditioned on L0-L9 indicators or total output must not be specified for primary or secondary update files.

16-18 RECORD ADDITION/DELETION

The following paragraphs describe record addition and record deletion in the Output-Format Specifications.

Record Addition

If a record is to be added to an input, output, or update file, the following conditions apply:

1. The file must be a disk file.
2. The corresponding File Description Specifications must have the letter A or B in the ADDITIONS/DELETIONS/UNORDERED field (column 66).
3. The word ADD must be entered in columns 16-18 of the record type description field of the Output-Format Specifications.
4. The word ADD must not be specified on the AND/OR line.

Record Deletion

If a record is to be deleted from an input, output, or update sequential file, the following conditions apply:

1. The file must be an index-sequential ($ IXSEQ) disk file.
2. The corresponding File Description Specifications must have the letter B or D in the FILE ADDITION/DELETION/UNORDERED field (column 66).
3. The DEL key symbol must be entered in columns 16-18 of the record type description field of the Output-Format Specifications.
4. The DEL key symbol must not be specified on the AN/OR lines.
5. The file must be declared as a primary, secondary, or demand file on the File Description Specifications.

16 STACKER SELECT/FETCH OVERFLOW

This field specifies the following:

1. The stacker into which the output or combined file card is placed after it is processed, or
2. That the overflow routine can be called for possible execution prior to printing the record specified by this line.

Valid entries for this field are:

Entry	Definition
Blank	Cards automatically go to default stacker.
Numeric Entry	Stacker into which card type is stacked.
F	Fetch overflow (printer files only).

Stacker Selection

Output card files can only be stacker selected on the Output-Format Specification.

Combined card files can be stacker selected on either the Input Specifications or the Output-Format Specifications. At program execution time, if a record appears that is selected in both the Input and Output-Format Specifications, then it is selected according to the Output-Format Specification. The programmer should try to avoid such instances because the results can be confusing.

Stacker selection on the basis of matching records should be specified only for detail output lines, because the card is selected prior to total time.

If the numeric entry specifying stacker selection is greater than the potential of the stacked hardware device, then the record is selected to the normal (default) stacker.

Record types identified by OR lines can be stacker selected to a special stacker by an entry in the field. However, if the stacker select field is left blank, the record type selected by the OR line goes to the default stacker, that is stacker select entry on the previous line is not assumed. AND lines cannot have a stacker select entry.

Fetch Overflow

If the printing of a line could cause overflow, leaving insufficient space on the page to print the remaining detail, total output lines or lines conditioned by the fetch overflow indicator, fetch overflow should be specified (F in column 16).

The same sequence of events always occurs when the overflow line is reached. These events are described in detail in the subsection titled Printer File Handling in this section of the manual. Briefly, remaining detail lines, total lines, and overflow lines (lines conditioned by the overflow indicator) are printed on the page following the occurrence of overflow.

However, if the program is written to print overflow lines before the usual time, a fetch overflow routine can be specified. This can be initiated anytime after the overflow print line has been reached. When the overflow condition is caused in this manner, the following actions occur:

1. All total lines conditioned by the overflow indicator are printed. If skipping is specified on these print lines, it occurs as defined by the programmer.
2. Heading and detail print lines conditioned by the overflow indicator are printed. If skipping is specified on these print lines, it occurs as defined by the programmer.
3. The print line that fetched the overflow routine is printed.
4. Any detail or total print lines left to be printed in the current program cycle are printed.

For the printer file, the letter F in column 16 on the Output-Format Specifications invokes the fetch overflow routine. The letter F can be specified for any total, detail, or exception print line that is not conditioned by an overflow indicator.

If a print line causes the overflow indicator to turn ON, the next print line containing the letter F in column 16 causes the execution of the overflow routine. When that is complete, normal printing resumes with the statement that fetched the overflow routine (see figure 12-2).

```
          Output-Format Specifications

01 OPRINTOU H  206   OF
02 O
03 O         DF 2     10
04 O
05 O         T  1     L1
06 O
07 O         TF 1     L1
08 O
09 O         T  1     OF
10 O
11 O         T  1     OF
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 12-2. FETCH OVERFLOW Coding Example

In the example in figure 12-2, if the printing of line 03 causes the overflow indicator to turn ON, printing occurs 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.

Special forms are not automatically advanced to a new page. It is the responsibility of the programmer to specify a skip to the first printing line on a new page. This skip to the top of a page must be specified on a print line conditioned by the overflow indicator (see figure 12-3).

```
          Output-Format Specifications

01 OPRINTOU H  406   OF
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 12-3. Overflow Indicator with Skip Specified

Fetch overflow (the letter F in column 16) can 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 specified, fetch overflow pertains only to overflow print lines associated with the file in which the record specifying fetch overflow is defined.

Exception Lines

Since an overflow indicator cannot be specified on an exception print line (E in column 15), fetch overflow causes overflow output at exception time.

Fetch overflow (figure 12-4) causes the heading, detail, and total overflow printer lines to be printed when the overflow print line has been passed, that is if the conditioning indicators of the fetch line satisfied. The programmer can also force overflow by using the SETON operation code and setting the appropriate overflow indicator ON prior to issuing the EXCPT operation code.

```
          Output-Format Specifications

01 OPRINTOU H  304   OF
02 O
03 O         EF 2     10
04 O
05 O         EF 2     20
06 O
07 O         T  1     OF
08 O
09 O         T  1     OF
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 12-4. FETCH OVERFLOW with Exception Output Coding Example

17-18 SPACE

The SPACE field specifies forms spacing for printer files. It is divided into two subfields such that spacing before or after printing can be specified. If both the SPACE and SKIP fields are left blank, single spacing after printing is performed.

Valid entries for this field are:

Entry	Definition
Blank	No spacing specified.
0	Space Suppress.
1-9	Space 1-9 lines, as specified.

Refer to the subsection titled Printer File Handling in this section of the manual for additional information.

19-22 SKIP (RPG II DIALECT)

The SKIP field specifies forms skipping for printer output files. It is divided into two subfields such that skipping before or after printing can be specified. Valid entries for this field are:

Entry	Definition
0-99	Line number for skip-0-99.
A0-A9	Line number for skip-100-109.
B0-B2	Line number for skip-110-112.
Blank	No skipping specified.

Refer to the subsection titled Printer File Handling in this section for additional information.

A skip entry must not be greater than the form length as specified in the Line Counter Specifications or as defaulted.

If both skipping and spacing are specified on the same line, the operations are performed in the following order:

SKIP BEFORE
SPACE BEFORE
SKIP AFTER
SPACE AFTER

On the B 1000 system it is more efficient to perform skipping and spacing operations after printing a line, and it is recommended that the RPG program be programmed to use this feature.

Different SPACE and SKIP entries can 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 files other than printer files.

23-31 OUTPUT INDICATORS

The OUTPUT INDICATORS field is divided into three subfields such that up to three indicators on each line can be specified to condition an output operation. Each subfield is divided into two parts as follows:

1. NOT (one column long).
2. INDICATOR (two columns long).

The NOT field specifies that the associated indicator must be OFF in order for the operation to occur. If this condition is desired, an N must be entered in the NOT field. Otherwise, the NOT field must be left blank. No output line should be conditioned by all negative indicators (at least one of the indicators used should be positive). If all negative indicators condition a heading or detail line, the line is printed at the beginning of the program cycle when 1P (first page) lines are written.

The INDICATOR field specifies the indicator to be tested for ON (NOT = blank) or OFF (NOT = N). The following entries are allowed in this portion of the INDICATORS field:

Entry	Definition
Blank	Operation not conditioned by an indicator.
01-99	Operation conditioned by indicator used elsewhere in the program.
L0-L9	Operation conditioned by control level indicator previously assigned.
LR	Operation conditioned by last record indicator.
MR	Operation conditioned by matching record indicator.
H0-H9	Operation conditioned by halt indicator used elsewhere in the program.
U1-U8	Operation conditioned by external indicator previously set.

Entry	Definition
OA-OG, OV	Operation conditioned by overflow indicator previously set.
1P	Operation conditioned by first page indicator.

All three indicators on one line are in an AND relationship. All indicators on one line (or grouped lines) must be ON or OFF as specified in order for the associated operation to take place.

An indicator specified on the line describing the record type conditions the entire output record. An indicator used to condition a field within the record is placed on the same line as the field description (see figure 12-5).

```
            Output-Format Specifications

01 OPRINTOU D 1       14
02 O                          FIELD1    27
03 O                          FIELD2    50
04 O                  15N16   FIELD3    83
05 O*
06 ODETAIL1 D 1       14
07 O        OR        15 16 17
08 O        AND       18
09 O        OR        19
10 O                  20      LINE01    32
11 O                  21 22N23LINE02 B  56
--*----1----*----2----*----3----*----4----*----5----*----6----*----
```

Figure 12-5. Output Indicators Coding Example

NOTE

Lines 01 through 04 specify that indicator 14 must be ON for detail-time printing to occur for PRINTOU output. Also, indicator 15 must be ON and indicator 16 must be OFF for the FIELD3 field to be included in the output record.

Lines 06 through 11 illustrate how AND/OR specifications can be used to condition output records and condition specific fields in the output record.

The following paragraphs describe the uses of each of the output indicators.

01-99 Numeric indicators can be used as follows:

1. To condition output operations that are only to be performed for specific input record types (record identifying indicators).
2. To condition output operations that are to be done when an input field meets certain conditions (field indicators).
3. To condition output operations according to the results of previous operations in the Calculation Specifications (resulting indicators).

L0-L9 Indicators

The L0-L9 (control level) indicators 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, then the L0 indicator can be used to condition those operations.

LR Indicator

The LR (last record) indicator conditions output operations that are to be performed at end of job.

MR Indicator

The MR (matching records) indicator conditions output operations that are to be performed only when matching input records are found.

H0-H9 Indicators

The H0-H9 (halt) indicators previously assigned in the FIELD INDICATORS field (Input Specifications) or the RESULTING INDICATORS field (Calculation Specifications) can be used to condition output operations that are to be performed only when an error condition occurs.

Since the program does not halt until after the record in error has been completely processed, some operations must be prevented in order to avoid erroneous output. By using halt indicators in conjunction with an N in the NOT portion of the INDICATORS field, an operation can be inhibited when the specified halt indicator is ON.

U1-U8 Indicators

The U1-U8 (external switch) indicators condition output on the setting of the external switches specified to the program at execution time. If an output file is specified as conditioned by an external indicator in the File Description Specifications (EXTERNAL INDICATORS field), then every output record described for the file should be conditioned by the same external indicator, otherwise the object program builds the output record and performs any blank after operations specified but must suppress the write operation.

OA-OG, OV Indicators

The OA-OG,OV (overflow) indicators previously assigned in the OVERFLOW INDICATOR field in the File Description Specifications condition output operations that are to be performed when the overflow line on a printer file has been reached.

Overflow indicators which have not been previously assigned in the File Description Specifications (except OF and OV, which are compiler defined) cannot be used in the Output-Format Specifications. Forms advancing at end of page are handled automatically if no overflow indicators are assigned to the file. Any specification line not conditioned by an overflow indicator which designates a skip to the next page turns OFF all overflow indicators before the skip takes place.

Overflow indicators must not condition exception records but can condition fields within exception records.

No more than one overflow indicator can be associated with a group of output indicators in an AND or OR relationship, and it must be the same indicator assigned to the file in the File Description Specifications.

The overflow line can be sensed when printing total or detail time output. If an overflow indicator conditions output lines, the following steps occur when the overflow line is sensed during total time output:

1. The overflow indicator turns ON.
2. The remaining total lines not conditioned by overflow are printed.
3. All total lines conditioned by overflow are printed. If skipping is specified on these lines, it occurs as defined by the programmer.
4. Heading and detail output lines conditioned by overflow are printed. If skipping is specified on these lines, it occurs as defined by the programmer.
5. Heading and detail lines not conditioned by overflow are printed.
6. The overflow indicator turns OFF.

If the overflow print line is sensed during detail time output, the following steps occur:

1. The overflow indicator turns ON.
2. The remaining detail print lines not conditioned by an overflow indicator are printed.
3. All total lines not conditioned by an overflow indicator are printed.
4. All total lines conditioned by an overflow indicator are printed. If skipping is specified on these print lines, skipping is performed as specified by the programmer.
5. Heading and detail output lines conditioned by an overflow indicator are printed. If skipping is specified on these print lines, skipping lines is performed as specified by the programmer.
6. Heading and detail print lines not conditioned by an overflow indicator are printed.
7. The overflow indicator turns OFF.

When using the overflow indicator to condition overflow printing, remember:

1. Overflow indicators can be turned ON and OFF by the SETON and SETOF operation codes.
2. Spacing operations past the overflow line causes the overflow indicator to turn ON.
3. Spacing or skipping operations past the overflow print line to any print line, the new page does not turn the overflow indicator ON.
4. A skip operation to new page specified on a print line not conditioned by an overflow indicator causes the overflow indicator to turn OFF.

Control Level Indicators With Overflow Indicators

If it is desired to have headings identifying the type of information on each page or each page to contain information from only one control group, then control level indicators can be used in conjunction with overflow indicators. Together they condition when headings and/or group information are to be printed.

In the example in figure 12-6, the L1 control level indicator is used in conjunction with the overflow indicator in order to print headings on every page. Line 01 allows the headings to be printed at the top of a new page only when overflow occurs. Line 02 allows printing of headings on a new page only at the beginning of a new control group (L1). In this way, duplicate headings caused by both L1 and OF being ON at the same time does not occur.

```
          Output-Format Specifications

01 OPRINTOU H  406   OFNL1
02 O       OR        L1
03 O                                    10 'NAME'
04 O                                    20 'ADDRESS'
05 O                                    35 'PHONE'
06 O                                    48 'EMPLOYEE NO.'
--*----1----*----2----*----3----*----4----*----5----*----6----*----
```

Figure 12-6. Using Control Level Indicators With Overflow Indicators

Figure 12-7 shows the necessary coding for the printing of certain fields on every page.

```
          Output-Format Specifications

01 OPRINTOU D  406   OFNL1
02 O       OR        L1
03 O                          EMPNO     10
--*----1----*----2----*----3----*----4----*----5----*----6----*----
```

Figure 12-7. Coding for the Printing of Certain Fields

1P Indicator

The 1P (first page) indicator conditions heading and detail printer output before the first record is processed. The 1P indicator is set ON by the RPG program and is set OFF after detail-time output and before the first record is processed. It cannot be used to condition calculations, output at total time, or output at exception time.

All lines conditioned by the 1P indicator are written even before the first record from any input file is processed. Therefore, do not condition output fields (except PAGE, UPDATE, UYEAR, UMONTH, UDAY, and JDATE) which are based upon data from input records by the 1P indicator.

The 1P indicator can also be used in an OR relationship with the overflow indicators OA-OG, OV. This allows printing the same heading or detail information on all pages.

The 1P indicator is invalid for update and combined files.

The B 1000 operating system (MCPII) buffers logical line printer output in order to take advantage of the hardware architecture of the printing devices and to create only one physical output record for each logical record. This feature delays the printing of a record until the next logical record is written by a program. In RPG, this causes the output at 1P time to be delayed until the next print record is written and requires a dummy print record to be written in order for the system operator to successfully line up special forms on a line printer. A dummy print record causes the operating system to write the print record that is in the line printer buffer. After the RPG program writes the dummy print record immediately after the print record, an operator can line up the special form in the desired fashion. Figure 12-7a shows an example of specifying a dummy print record immediately following the special forms print record in the Output-Format Specifications.

```
          Output-Format Specifications

01 OLINE    H  2     1P
02 O                                   50 'SPECIAL FORMS RECORD'
03 OLINE    H  0     1P
04 O                                    1 ' '
--*----1----*----2----*----3----*----4----*----5----*----6----*----
```

Figure 12-7a. Special Forms Printing Specifications

In figure 12-7a, the dummy print record contains the number zero (0) in the SPACE AFTER field (column 18) of the Output-Format Specifications. This causes the line printer to leave the paper in place, that is, no line skip or space operation is performed. The RPG program automatically displays the message AGAIN? and waits for the operation to enter YES to print 1P conditioned lines again, or NO to continue processing.

```
            Output-Format Specifications

01 OREPORT1 H  3      1P
02 O                                    41 'REPORT HEADING #1'
03 OREPORT2 H  2      1P
04 O        OR        OF
05 O                                    35 'REPORT HEADING #2'
--*----1----*----2----*----3----*----4----*----5----*----6----*----
```

Figure 12-8. 1P Indicator Coding Example

NOTE

Lines 01 and 02 specify a heading line that is to be printed only once for the REPORT1 file, on the first page.

Lines 03 through 05 specify both the 1P and OF indicators in an OR relationship, allowing the heading line to be printed on the first page and all subsequent pages (during overflow printing for the REPORT2 file).

AND/OR Fields

If it is necessary to specify more than three indicators to condition an output operation, an AND line can 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 takes place.

OR lines (OR in columns 14-15) can 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 can be used together to condition an output record (but not a field). A maximum of three indicators in an AND relationship (on one line) can be used to condition a field. See figure 12-5 for an example of the usage of AND and OR lines.

There is no limit on the number of AND or OR lines that can be specified. However, it is recommended that the user not exceed 20 if compatibility with other Burroughs systems is desired. There must be at least one indicator on an AND/OR line and on the preceding line.

32-27 FIELD NAME (VARIABLE NAME)

The FIELD NAME field assigns an identifier (name) to an output data field. The identifier used must have been previously defined in the Input Specifications (VARIABLE NAME field), the Extension Specifications (VECTOR NAME field), or the Calculation Specifications (RESULT FIELD field). Also, any special words can be used. A separate line must be used for each field description. Fields can be listed in any order within each record type, since their location is determined by the entry in the END POSITION field. If fields overlap, only the last field specified appears intact in the output record (the exact results depend upon the degree of overlap).

Special Words

The following special words are reserved for use as variable names:

PAGE through PAGEn
UDATE
UMONTH
UDAY
UYEAR
JDATE
*PLACE
*PRINT

Each special word has a specification defined usage, as described in the following paragraphs:

Page Fields

When page numbering is to be done on output, the special words PAGE through PAGEn indicates that automatic page numbering is to be performed. When a PAGE field is named in this field without being previously defined in the Input or Calculation Specifications, it is assumed to be four characters in length with no decimal positions. On output, leading zeros are suppressed and the sign is not printed unless an edit word or edit code is specified.

The page number begins at zero unless otherwise specified, and is automatically incremented by one each time before the page field is written.

The page number can be reset at any point during the program by setting the PAGE field to zero before it is printed (see figure 12-9). This can be accomplished in two ways:

1. Use the BLANK AFTER field to cause the field to be cleared to zero after printing, or
2. Assign an output indicator to the PAGE field. If the indicator is ON, the field is set to zero before normal incrementation takes place.

```
            Output-Format Specifications

01 OPRINTOU H  3      L1
02 O                  20       PAGE      120
03 O*
04 OOUTPUT  H  3      L1
05 O                           PAGE    B 100
--*----1----*----2----*----3----*----4----*----5----*----6----*----
```

Figure 12-9. PAGE Specifications Coding Example

NOTE

When the indicator assigned to the PAGE field is ON, the value of the PAGE field is reset to 0 (zero). Before being printed, the value of the PAGE field is incremented by 1.

The BLANK AFTER field can also be used to cause the PAGE field to be reset to 0 (zero) after the line is printed.

The same PAGE through PAGEn entry can be used for two different output files, but this is not recommended. PAGE fields are not restricted to printer files.

Date Fields (UDATE, UMONTH, UDAY, UYEAR and JDATE)

Five special words allow the program to obtain the current value of the date as supplied through the B 1000 system. The following rules apply to date fields.

1. 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).

2. The other three fields, UMONTH, UDAY, and UYEAR, each gives a 2-digit field representing the month, day, and year, respectively.
3. These fields must not be changed by any operations within the program; thus, they are usually used only in compare, test, and output operations.
4. JDATE is a special word reserved for accessing the Julian date as obtained from the system. JDATE can be used anywhere that UDATE can be used. JDATE cannot be changed in any way by the RPG program. JDATE is five digits long with zero decimal places and has the format YYDDD.

*PLACE Special Word

The *PLACE special word allows writing of the same field or fields in more than one place in an output record without having to specify the field names and end positions more than once. The designated fields are written in the same relative positions ending in the position specified for the *PLACE entry. It is possible to obtain the same results in two ways (see figure 12-10).

1. Define each field and its corresponding end position for every time it is to appear in the output records, or
2. Use the *PLACE special word.

```
            Output-Format Specifications

01 OOUTPUT  D  1
02 O                          FIELD1     5
03 O                          FIELD2    15
04 O                          FIELD3    25
05 O                          FIELD1    35
06 O                          FIELD2    45
07 O                          FIELD3    55
--*----1----*----2----*----3----*----4----*----5----*----6----*----
```

NOTE

The above Output-Format Specifications can also be defined using the *PLACE special word, as shown below:

```
01 OOUTPUT  D  1
02 O                          FIELD1     5
03 O                          FIELD2    15
04 O                          FIELD3    25
05 O                          *PLACE    55
--*----1----*----2----*----3----*----4----*----5----*----6----*--
```

NOTE

The *PLACE special word duplicates all fields previously specified.

Figure 12-10. *PLACE Specifications Coding Example

Both methods produce identical results, but use of the *PLACE entry saves extra coding.

The following rules must be observed when using the *PLACE specification:

1. All fields within the record type written above the *PLACE entry are repeated according to the *PLACE specification. the one line above.
2. An end position must be given for every *PLACE specification.
3. An additional *PLACE special word (on a separate line) must be used every time the fields are to be repeated.
4. The *PLACE special word 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.
5. The end position specified for the *PLACE special word 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 occurs, with the *PLACE output overlapping the previous characters.
6. The end position specified for the *PLACE special word must not be lower than the highest previously specified field end position.
7. The left-most position of the fields to be moved by the *PLACE special word is always assumed to be position 1.
8. When the *PLACE special word is specified for card output, the fields and constants named above are repunched. Any printed output on the cards are not reprinted unless an * (asterisk) character is entered in column 40 of the same line as the *PLACE special word.
9. Only the conditioning indicators (columns 23-31), FIELD NAME (columns 32-37) and END POSITION (columns 40-43) fields can have entries.

*PRINT Special Word

The *PRINT special word causes 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 special word must be used only once for each record and appears after all fields on the card which are to be printed. The *PRINT special word can be conditioned by indicators in the OUTPUT INDICATORS field; all other fields must be left blank (see figure 12-11).

```
                Output-Format Specifications

01 OPUNCHOU D         01
02 O                              DATA1      10
03 O                              FIELD1     20
04 O                              FIELD2     34
05 O                              *PRINT
06 O                              DATA2      40
07 O                              FIELD3     44
--*----1----*----2----*----3----*----4----*----5----*----6----*----
```

Figure 12-11. *PRINT Specifications Coding Example

NOTE

The *PRINT entry causes the three fields above it to be printed as well as punched. The other two fields are only punched.

Having the *PRINT special word 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 special word:

1. Enter the field name to be printed in the VARIABLE NAME field, and
2. Enter an asterisk (*) character in column 40, and
3. Enter the end position for the field in columns 41-43 (limited to a maximum of 128), right-justified; leading zeros not required.

38 EDIT CODES

The EDIT CODES field specifies editing of an unpacked numeric output field. Editing operations have been provided, for which it is not necessary to write an edit word. These operations and their corresponding edit codes are summarized in the EDIT CODES portion of the Output-Format Specifications sheet.

Only unpacked numeric fields can be edited, and if an edit code is specified, then the constant or edit word field must be left blank, except when the check protect or floating dollar sign is required. In the latter case, enter an asterisk (*) or dollar sign ($) character, respectively, in column 45-47. Edit codes X, Y, and Z must not be specified with the asterisk (*) or dollar sign ($) characters. The Y edit code must only be specified with field lengths of three to six digits.

When an entire array is to be edited with an edit code, the following points must be taken into account to avoid specifying the end position less than the total size of the output data:

1. Two blank characters are inserted to the left of each element, and
2. Comma (,), decimal point (.), and minus sign (−) characters occupy one character position each. The credit sign (CR) characters occupy two character positions.

The following are the edit code rules:

1. Conditioning indicators on the record description line and the field description line determine when the field is written.
2. The variable name specified in columns 32-37 must be previously defined as:

   1) An entire numeric array, or
   2) A single element of a numeric table or array, or
   3) A numerically defined field, or
   4) A numerically defined special word.

3. The use of an edit code in column 38 causes various punctuation of the numeric field specified in columns 32-37.
4. Specifying the letter B in the BLANK AFTER field (column 39) clears the field after output but does not affect any editing specified on this line.
5. End position, columns 40-43, defines where the last character of the edited output is to appear.

NOTE

Syntax errors occur if the total length in bytes of the variable and all inserted decimal (.) and comma (,) characters, and status reporting is greater than the length in bytes from the specified end position to position one of the output record.

6. The PACKED field (column 44) must be blank.
7. The Constant or Edit Word portion of this specification line must remain blank, except for the following two exceptions:

1) If the check protect asterisk (*) character is required, enter an asterisk (*) character in columns 45-47.
2) If the floating dollar sign ($) character is required, enter a dollar sign ($) character in columns 45-47.

NOTE

The check protect asterisk (*) and floating dollar sign ($) characters cannot be specified with the X, Y, or Z edit codes.

8. Edit codes A, B, J, K, 1, and 2 provide the comma (,) option. Comma (,) characters must only be inserted in the edited output if significant non-zero integers exceed three in number.
9. Edit codes A, C, J, L, 1, and 3 provide the zero balance option. This option provides that at least one zero is written if the value of the field is zero. The number of zeros written is determined by the decimal positions assigned when the field was defined, as shown in the following example:

Decimal Position	Zero Balance Output
0	0
1	.0
2	.00
9	.000000000

10. A sign or symbol for a negative number can be requested or ignored.

1) When specified, A, B, C, and D provide CR for negative status report and J, K, L, and M provide minus sign (−) character for negative status report.
2) When ignored, 1, 2, 3, and 4 indicate no status report.

11. The X edit code removes the plus sign from the field with no zero suppression.
12. The Z edit code removes all signs and the suppression of leading zeros.
13. The Y edit code edits in date field format. The variable must be no less than three digits and no more than six digits in lengths.
14. An entire array can be edited with one edit code. Include the following when specifying the end position:

1) One space for each character of the array, plus
2) Two spaces to be inserted to the left of each element, plus
3) One space for each comma that can be inserted, plus
4) One space for each decimal point (.) character that is inserted, plus one of the following unless using edit codes 1, 2, 3, or 4.

a. One space for each minus sign with each element, or
b. Two spaces for each CR sign with each element.

Table 12-1. Edit Codes Table

Edit Code	Comma	Decimal Point	Zero Suppress	Sign for Negative Balance	Printout on Zero Balance*		
					International		Domestic United Kingdom
					I	J	
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 International format are printed or punched in two ways, depending on the entry made in INVERTED PRINT field (column 21)

of the Control Card Specifications.

** The X edit code removes only the positive sign.

*** The Y edit code suppresses only the left-most zero. The Y edit 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

The BLANK AFTER field specifies that a variable is to be reset after the output operation is finished. Alphanumeric fields are cleared to blank characters and numeric fields are cleared to zeros. Valid entries for this field are:

Entry	Definition
Blank	Field is not to be cleared.
B	Variable is to be cleared after being moved to the output buffer.

Entering the letter B in the BLANK AFTER field (column 39) causes the field named in the FIELD NAME field to be set to zero or blank if the field is numeric or alphanumeric, respectively. This is to be done immediately after output to the designated record work area.

When an indicator is assigned to a field to test for zero or blank in the Input or Calculation Specifications and the same field is used on Output-Format Specifications with the BLANK AFTER field, then that zero/blank indicator is affected as follows:

RPGII Dialect: No effect on zero/blank indicator.

RPG I Dialect: Specified indicator is turned ON after the field is blanked during the output operation.

RPG I and RPG II Dialects

If more than one indicator is assigned to the same field as a ZERO/BLANK indicator (on different Input or Calculation Specifications), then only the first one assigned is turned ON when the field is blanked out. When an array or array elements are output with BLANK AFTER, zero/blank indicators are not changed.

When the specified field is a table name, the element that satisfied the last successful LOKUP operation is blanked or zeroed.

40-43 END POSITION

The END POSITION field specifies the location of a field within an output record. Only the location of the right-most position is specified; the number of positions to the left of that location is determined by the length specified previously for the field. Space must also be left to allow for any editing symbols (edit codes or edit words) that can be entered. The end position must not exceed the record length specified for the file, except in the special case where an asterisk (*) character is specified in the END POSITION field.

To print fields on a card in a position other than the one assigned by using the *PRINT special word, enter an asterisk (*) character in the END POSITION field and proceed as follows:

1. Enter the field name to be printed in the FIELD NAME field, or enter a literal in columns 45-70. A variable can be edited by means of an edit code or edit word.
2. Enter the end position for the field in columns 41-43, right-justified. Leading zeros are not required.

Table 12-2 illustrates the results obtained when the various edit codes are used to edit a two decimal position field whose value is a negative (−3.12). The Output-Format Specifications specify an ending position of 12.

Table 12-2. 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	

NOTE
The asterisk (*) character represents a negative 0.

44 PACKED OR BINARY

The PACKED OR BINARY field specifies that a numeric output field is to be written in packed decimal or binary format. Packed decimal and binary fields should not be printed because the resultant printout is meaningless.

Valid entries for this field are:

Entry	Definition
Blank	Field is to be written in alphanumeric or unpacked decimal format.
P	Field is to be written in packed decimal format.
B	Field is to be written in binary format.

When the letter B is specified to indicate binary format, the length of the numeric field that is to be written cannot exceed nine bytes, and the following rules apply:

1. Numeric output fields that have a length of one digit to four digits inclusive are converted to 2-byte binary fields.
2. Numeric output fields that have a length of five digits to nine digits inclusive are converted to 4-byte binary fields.
3. Output in binary format is only allowed for disk, tape, or 80-column card devices.
4. Arrays, tables, array elements, and numeric fields can be specified.

Column 44 must be left blank under the following conditions:

1. When an asterisk (*) character appears in column 40 of the same Output-Format Specifications line.
2. For fields that precede a *PRINT special word for a card file.
3. For fields that precede a *PLACE specification for a card or printer file.
4. For alphanumeric fields.
5. When a constant is specified.
6. When an edit code is specified.
7. When an edit word is specified.

45-70 CONSTANT or EDIT WORD

The CONSTANT or EDIT WORD field 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 appear exactly as written in the Output-Format Specifications, in the position specified. The following rules must be observed when forming constants:

1. The VARIABLE NAME field must be left blank.
2. Constants must be left-justified, enclosed in apostrophe (') characters and no more than 24 characters in length.
3. If an apostrophe is to appear in the constant, two consecutive apostrophes must be coded for

each apostrophe required. (') character is required. In reference to the end position, the two apostrophes are to be counted as one.

4. Numeric data can be used as a constant but must still be enclosed in apostrophe (') characters. Note that this is an alphanumeric literal composed of numeric characters.

The example in figure 12-12 uses the Line Printer Spacing Form to illustrate the coding of output constants.

```
        Format of Desired Printer Output Listing

  ----*----1----*----2----*----3----*----4----*----5----*----6----*---
01
02
03
04                          MONTHLY  STATEMENT
05
06
07          ACCOUNT          OLD            PAYMENT        NEW
08          NUMBER         BALANCE                         BALANCE
09
10          XXXXXX          XXX.XX          XXX.XX         XXX.XX
11         (ACCNO)         (OLDBAL)        (PAYMNT)       (NEWBAL)
12
  ----*----1----*----2----*----3----*----4----*----5----*----6----*---


            Output-Format Specifications

 01 OBILLING H  304    1P
 02 O                                      43 'MONTHLY STATEMENT'
 03 O         H  1     1P
 04 O                                      27 'ACCOUNT          OLD'
 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                           ACCNO      15
 11 O                           OLDBAL1    29
 12 O                           PAYMNT1    44
 13 O                           NEWBALA    56
 --*----1----*----2----*----3----*----4----*----5----*----6----*----
```

Figure 12-12. Coding of Output Constants

Edit Words

Edit words give a program more adaptability then edit codes when editing numeric data fields. These data fields are made more meaningful and easier to read by inserting one or more of the following characters:

Comma (,)
Virgule (/)
Decimal point (.)
Dollar sign ($)
Asterisk protection (*a)
Zero (0)
Credit field identifier (CR)
Negative field identifier (–)
Constants

When a numeric data field is to be edited, an edit word, sometimes called an edit mask, must be placed in columns 45-70 of the Output-Format Specifications. The edit word must begin in Column 26, be enclosed in apostrophe (') characters, and be no more than 24 characters in length. The first apostrophe (') character must be coded in column 45, and the ending apostrophe (') character must be coded in the column following the last character of the edit word.

Edit Word Rules

The following rules must be observed when using edit words:

1. Column 38 must be blank when edit words are used.
2. A numeric data field name entry must be made in columns 32-37.
3. An end position must be specified in columns 40-43.
4. Edit words must be enclosed in apostrophe (') characters and must not exceed 24 characters in length.
5. Any valid, printable RPG character can be used in an edit word. The following characters have special uses when placed in specific positions within the edit word:

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

6. When an edit word is used, automatic suppression of all high-order zeros within the data field occurs unless a zero or asterisk is specified in the edit word. When zero is specified, the last leading zero in the field is replaced by an asterisk.
7. The number of replaceable characters in an edit word should be equal to the length of the field that is to be edited, with the following exceptions:

   1) When the floating dollar sign ($) character is used, an extra space can be left in the edit word to ensure that the dollar sign ($) character is printed when the numeric data field is full.
   2) An extra space can be left in the edit word when zero suppression is not wanted. All other editing is performed.

   The replaceable characters and their meanings are:

Replaceable Character	Meaning
0	Zero suppression.
*	Asterisk fill.
b	Blank.
$	Floating dollar sign (if it appears immediately to the left of zero suppress character).

8. Fixed dollar sign ($), decimal point (.), comma (,), ampersand (&), negative sign (CR and –) characters , and constant information are not replaceable characters.

9. Any zeroes or asterisk (*) characters following the left-most zero or asterisk are treated as constants and are not replaceable characters.
10. Any constant to the left of the zero suppression stop character, except the floating dollar sign ($), is suppressed unless a significant digit precedes the constant.
11. When a blank edit word is used, all leading zeroes are suppressed and any sign in the unedited field is removed. Negative values are not identified.
12. When no edit word is used, the data in the output record has the same format as the unedited data. When the numeric data field contains a negative number, the low-order sign position is printed as a alphabetic character (J-R) depending upon the value of the numeric data field.

Zero Suppression

The elimination of high-order zeroes when printing a numeric data field is known as zero suppression. Either full, none, or limited zero suppression can be accomplished by the use of an edit word.

To create the proper edit word a programmer must know:

1. The maximum size of the field to be edited.
2. The number of positions to be zero suppressed.

Full Zero Suppression Coding

The following are recommended procedures for full zero suppression coding:

1. The edit word length is equal to the maximum size of the field to be edited, or
2. Enter a zero in the right-most position of the edit word, and
3. Leave the remaining position in the edit word blank.

When full zero suppression is the only editing function desired, it is more efficient to use the Z edit code by making the appropriate entry in column 38 of the Output-Format Specifications.

No Zero Suppression Coding

The following does not apply to RPG I but is a recommended method of coding for no zero suppression.

1. Add one to the maximum size of the field to be edited.
2. Enter a zero in the left-most position of the edit word, and
3. Leave the remaining positions in the edit word blank.

Figure 12-13 illustrates some zero suppression examples:

Source Data	Constant or Edit Word	Output Record
0000000005	' , , 0. '	bbbbbbbbb.05
00000000	' , , 0 -'	bbbbbbbbb0b
-00000000123	' , , . -'	bbbbbbbbbb1.23-
-0000123456	' 0'	bbbb123456
+0000123456	'0 '	b000123456
0000000000	'$ 0 &CR GROSS'	$bbbbbbbb00bbbGROSS
0000000000	' , , , -OLD BAL'	bbbbbbbbbbbbb0LD BAL
000000	' , 0. '	bbbbb.00
000000	' , .0 '	bbbbbbb0
0000000000	' , , 0 * '	bbbbbbbbb0*00
001234	'0, ,0 '	b,012,034
-000000015	' , , . -'	bbbbbbbbbb15-
000000005	' , , 0 . -'	bbbbbbbb0.05b
	4 5 5 6 6 7 5----0----5----0----5----0 Columns 45 through 70	

Figure 12-13. Zero Suppression Coding Examples

Limited Zero Suppression Coding

The following conditions accomplish limited zero suppression when:

1. The edit word length is equal to the maximum size of the field to be re-edited.
2. Enter a zero in the last position that is to be zero suppressed by the edit word. Allow all other positions in the edit word to remain blank.

Editing with a Comma, Decimal Point, and Fixed Dollar Sign

When editing with a comma, decimal point, or fixed dollar sign, note the following:

1. The edit word length is equal to the maximum size of the field to be edited plus the number of editing characters that are to be inserted.
2. When a zero suppress "zero" is not included in the edit word, full suppression is done automatically.
3. All comma (,) and decimal point (.) characters to the left of a zero suppress "zero" are replaced by blank characters when they are not preceeded by a significant non-zero digit.
4. The fixed dollar sign ($) character is never suppressed.

Editing with an Asterisk

This special character protects printed data from being tampered with by unauthorized persons. This is accomplished by printing asterisk (*) characters in the print positions where zero (0) and comma (,) characters have been suppressed. This particular technique is frequently referred to as check protection.

Asterisk (*) fill/check protection follows the same rules as zero suppression. Zero suppression replaces all leading zero (0) and comma (,) characters with blank characters, while asterisk (*) fill/check protect replaces these characters with asterisk (*) characters.

Figure 12-14 shows examples of check-protect or asterisk-fill editing.

Source Data	Constant or Edit Word	Output Record
0000123456	' , , * . &-'	*****1,234.56bb
-0000123456	'* '	*000123456
+1234567890	'* '	1234567890
-0000123456	' *'	****123456
0000001234	' , *, * '	******,012*34
012345	'& ,* 0, '	***120,345
	4 5 5 6 6 7 5----0----5----0----5----0 Columns 45 through 70	

Figure 12-14. Asterisk Fill Coding Examples

Editing with a Floating Dollar Sign

A floating dollar sign is frequently used when writing checks. This method of check protection places a dollar sign immediately to the left of the first significant non-zero digit in the data field or the decimal point. Unnecessary commas and zeroes to the left of the dollar sign are replaced by spaces in the output area.

The following rules apply to editing with a floating dollar sign:

1. An extra space must be left in the edit word for the floating dollar sign because a full print position is required for the dollar sign when the field is full.
2. The floating dollar sign is placed immediately to the left of the "zero" used to signify zero suppression.
3. The floating dollar sign should never be placed immediately to the left of a decimal point.

Figure 12-15 illustrates examples of dollar sign coding with before and after images.

Source Data	Constant or Edit Word	Output Record
+0000000005	' , ,$0 . -'	bbbbbbbbb$0.05b
-0012345678	' , , $0. CR**'	bbb$123,456.78CR**
0000000000	'$, , 0. '	$bbbbbbbbbb.00-
+0000123456	'$ &- NET'	$bbbb123456bbbNET
-0000123456	'$ &- NET'	$bbbb123456b-bNET
0000123456	'$0 - NET'	$000123456bbNET
-0000123456	' $0 &CR'	bbbb$123456bCR
-1234567890	' $0 &CR'	$1234567890bCR
0000000005	' , ,$0 . &NET'	bbbbbbbbb$0.05bNET
0000000005	' , , $0. '	bbbbbbbbbb$0.05
-1234567890	' , , $0. -'	$12,345,678.90-
-0001234567	' , , $0. CR'	bbbb$12,345.67CR
0000001234	' , $0, . -SALES'	bbbbbb$,012.34bSALES
00123456	' , $,0 . '	bbb1$,234.56
	4 5 5 6 6 7 5----0----5----0----5----0 Columns 45 through 70	

Figure 12-15. Dollar Sign Coding Examples

Editing Negative Numbers

Negative numbers can be represented by the dash or minus sign (−) or the credit symbol (CR) characters.

These negative value symbols are placed on the right side of the edit word. The dash requires one additional position in the output area, while the credit symbol requires two additional positions.

When the field being edited is positive in value, the negative symbols are not printed. If a number is negative and the edit word does not contain negative editing symbols, then the number is edited as a positive number.

Figure 12-16 illustrates examples of negative sign coding.

Source Data	Constant or Edit Word	Output Record
+0000123456	' &CR&NET'	bbbb123456bbbbNET
+0000123456	' &CR NET'	bbbb123456bbbbNET
-0000123456	' &CR NET'	bbbb123456bCRbNET
-0000123456	' &- NET'	bbbb123456b-bbNET
0000123456	' &NET&CR'	bbbb123456bbbbbbb
-0000123456	' &NET&CR'	bbbb123456bbNETbCR
-1234567890	'$& , , 0. CR'BAL'	$b12,345,678.90CR
+0000123456	' , , *. *CR**'	*****1,234.56bbb**
-0000123456	' , , *. *CR**'	*****1,234.56*CR**
	4 5 5 6 6 7 5----0----5----0----5----0 Columns 45 through 70	

Figure 12-16. Negative Signs Coding Examples

Editing with a Virgule

The virgule (/) character edits date fields and provides separation of the day, month, and year in order to make the printed date more readable. As with all edit words, automatic zero suppression is performed.

When date field editing is desired, it is more efficient to use the Y edit code entry in column 38 if the field is from three to six digits in length.

Editing to Remove the Sign

When no edit word is specified and a numeric field is printed and punched, the output has a zone overscore over the low-order digit (right-signed) or the most significant digit (left-signed) of the field. This zone represents the sign of the field. When the field is positive, the letters A through I are printed and punched. If the field is negative, the letters J through R are printed and punched. The zone can be eliminated by:

1. Positive signs:

   1) A positive sign can be an edit word that contains any or none of the edit characters.
   2) A positive sign can be the X edit code entry in column 38. This is the most efficient way of removing the positive sign when no other editing is desired.

2. Negative signs:

   A negative sign can be an edit word that contains any or none of the edit characters.

For negative fields, an edit word must be used to remove the sign.

Figure 12-17 illustrates examples of sign removal.

Source Data	Constant or Edit Word	Output Record
+0000123456		0000123456
-0000123456		0000123456
0000000000	' '	bbbbbbbbbb
+0000123456	' '	bbbb123456
-0000123456	' '	bbbb123456
	4 5 5 6 6 7 5----0----5----0----5----0 Columns 45 through 70	

Figure 12-17. Sign Removal Coding Examples

Ampersand

The ampersand (&) character provides a blank space in the edit word when RPG II dialect is specified. Refer to figure 12-18 for ampersand coding examples using RPG II dialect.

Source Data	Constant or Edit Word	Output Record
-1234567890	'$& , , 0 . &- GROSS	$b12,345,678.90b-GROSS
-1234567890	'$& , , 0 . &-NET&PAY'	$b12,345,678.90-NET PAY
-0000123456	'$& ; , *. CR'	$******1,234.56CR
+000002	' 0 LBS.& OZ.TARE&-'	bbb0LBS.b02bbbbbbbbb
031274	' & & &LATER'	b3b12b74LATER
	4 5 5 6 6 7 5----0----5----0----5----0 Columns 45 through 70	

Figure 12-18. Ampersand Coding Examples (RPG II Dialect)

The ampersand (&) character appears as an ampersand when RPG I dialect is specified. Refer to figure 12-19 for ampersand coding examples using RPG I dialect.

Source Data	Constant or Edit Word	Output Record
0000123456	' &&PROFIT'	bbbb123456&&PROFIT
0000123456	' &CR&NET'	bbbb123456bbb&NET
1234567890	'$& , , 0 . &-NET&PAY'	$b12,345,678.90b-NET&PAY
	4 5 5 6 6 7 5----0----5----0----5----0 Columns 45 through 70	

Figure 12-19. Ampersand Coding Examples (RPG I Dialect)

Editing and Constants

The edit characters that have been described are the standard characters used for editing. It is, however, possible to use any character in an edit word. The specified character is printed on the output report in the position specified whenever a non-zero character has been printed to the left of the specified character.

Figure 12-20 illustrates the use of constants in the edit word.

Source Data	Constant or Edit Word	Output Record
0000123456	' &&PROFIT'	bbbb123456bbPROFIT
-0000123456	' , , . &CR NET'	bbbbb1,234.56bCRbbNET
0000123456	' , , DOLLARS CENTS'	bbbbb1,234DOLLARS56CENTS
000000	' , DOLLARS CENTS'	bbbbbbbbbbbbbCENTS
000000	' , 0 DOLLARS CENTS&CR'	bbbb0DOLLARS00bbbbbbbb
-000002	' 0LBS. 0Z.TARE&-'	bbbbLBS.020Z.TAREb-
063379790	'0 - - '	b63-37-9790
0042	'0 HRS. MINS.&0''CLOCK'	b0HRS.42MINS.b0'CLOCK
062283	' - - &LATER'	b6-22-83bLATER
062283	' / / '	b6/22/83
	4 5 5 6 6 7 5----0----5----0----5----0 Columns 45 through 70	

Figure 12-20. Examples of Constants in the Edit Word

75-80 PROGRAM IDENTIFICATION

Refer to section 2 for a complete description.

EDIT WORD AND EDIT CODE EXAMPLES – RPG II DIALECT

Table 12-3 shows examples using Edit Words and Edit Codes.

The first column on the left contains the number of the example. The second column contains the input data. The third column contains the edit word or edit code used. The fourth column contains the resulting output. A brief description of example is provided immediately after table 12-3.

Table 12-3. Examples of Edit Words and Edit Codes

Example Number	Source Data	Edit Word or Edit Code	Output Result
1	0000123456	(no edit word)	+000123456
2	-0000123456	(no edit word)	!000123456
3	0000000000	' '	(no output)
4	+0000123456	' '	1234546
5	-0000123456	' '	-0000123456
6	0000000005	' , , 0. '	.05
7	0000000000	' , , 0 -'	0
8	-0000000123	' , , . -'	1.23-
9	-0000123456	' 0'	123456
10	+0000123456	'0 '	000123456
11	0000000000	'$ 0 &CR GROSS'	$ 0 CR GROSS
12	0000000000	', , , -OLD BAL'	OLD BAL
13	000000	' , 0. '	.00
14	000000	' , .0 '	0
15	0000000000	' , , 0 * '	0*00
16	001234	'0, ,0 '	,012,034
17	-0000000015	' , , . -'	15-
18	0000000005	' , , 0 . - '	0.05
19	0000000005	' , ,$0 . -'	$0.05

Table 12-3. Examples of Edit Words and Edit Codes (Cont)

20	-0012345678	' , , $0. CR**'	$123,456.78CR**
21	0000000000	'$, , 0. '	$ .00
22	+0000123456	'$ &- NET	$ 123456 NET
23	-0000123456	'$ &- NET'	$ 123456 - NET
24	+0000123456	'$0 - NET'	$000123456 NET
25	-0000123456	' $0 &CR'	$123456 CR
26	-1234567890	' $0 &CR'	$1234567890 CR
27	0000000005	' , ,$0 . &NET'	$0.05 NET
28	0000000005	' , , $0. '	$.05
29	-1234567890	' , , $0. -'	$12,345,678.90-
30	-0001234567	' , , $0. CR'	$12,345.67CR
31	0000001234	' , $0, . -SALES'	$,012.34 SALES
32	00123456	' , $,0 . '	1$,234.56
33	0000123456	' , , * . &-	*****1,234.56
34	-0000123456	''* '	*000123456
35	+1234567890	'* '	1234567890
36	-0000123456	' *'	****123456
37	0000001234	' , *, * '	******,012*34
38	012345	'& ,* 0, '	***120,345
39	+0000123456	' &CR&NET'	123456 NET
40	+0000123456	' &CR NET'	123456 NET
41	-0000123456	' &CR NET'	123456 CR NET
42	-0C00123456	' &- NET'	123456 - NET
43	0000123456	' &NET&CR'	123456
44	-0000123456	' &NET&CR'	123456 NET CR
45	-1234567890	'$& , , . CR'	$ 12,345,678.90CR
46	+0000123456	' , , *. *CR**'	*****1,234.56 **
47	-0000123456	' , , *. *CR**'	*****1,234.56*CR**
48	-12345678	'$&, , 0 . &-GROSS'	$,234,567.89 -GROSS

Table 12-3. Examples of Edit Words and Edit Codes (Cont)

49	-1234567	'$& , 0 . &-NET PAY'	$ 12,345.67 -NET PAY
50	-0000123456	'$& , , *. CR'	$******1,234.56CR
51	+000002	' 0 LBS.& OZ.TARE&-'	OLBS. 02
52	+0002	'0 LBS.& - OZ. TARE'	OLBS.02 OZ. TARE
53	093083	' & & &LATER'	9 30 83 LATER
54	0000123456	' &&PROFIT'	123456 PROFIT
56	-1234567	' , DOLLARS CENTS'	12,345DOLLARS67CENTS
57	000000	' , DOLLARS CENTS'	CENTS
58	00000	' 0 DOLLARS CENTS&CR'	ODOLLARSOO
59	-000002	' OLBS. OZ.TARE&-'	LBS.02OZ.TARE -
60	063379790	'0 - - '	63-37-9790
61	06337979	'0 - - '	063-37-9790
62	0042	'0 HRS. MINS.&0''CLOCK'	OHRS.42MINS. O'CLOCK
63	093083	' - - &LATER'	9-30-83 LATER
64	093083	' / / '	9/30/83
65	093083	Y edit code	9/30/83
66	09308	Y edit code	9/30/8
67	0930	Y edit code	9/30
68	093	Y edit code	9/3
69	+0000000000	1 edit code	0
70	+0000000000	2 edit code	(no output)
71	+0000000000	A edit code	0
72	+0000000000	B edit code	(no output)
73	+0000000000	X edit code	0000000000
74	+0000000000	Z edit code	(no output)
75	-0000000000	1 edit code	0
76	-0000000000	2 edit code	(no output)
77	-0000000000	A edit code	0

Table 12-3. Examples of Edit Words and Edit Codes (Cont)

78	-0000000000	B edit code	(no output)
79	-0000000000	J edit code	0
80	-0000000000	K edit code	(no output)
81	-0000000000	X edit code	!000000000
82	-0000000000	Z edit code	(no output)
83	+0000000000	1 edit code	.00
84	+0000123456	1 edit code	1,234.56
85	+000012345	2 edit code	1,234.56
86	+0000123456	3 edit code	1234.56
87	+0000123456	4 edit code	1234.56
88	+0000123456	A edit code	1,234.56
89	+0000123456	B edit code	1,234.56
90	+0000123456	C edit code	1234.56
91	+0000123456	D edit code	1234.56
92	+0000123456	J edit code	1,234.56
93	+0000123456	K edit code	1,234.56
94	+0000123456	L edit code	1234.56
95	+0000123456	M edit code	1234.56
96	+0000123456	X edit code	0000123456
97	+0000123456	Z edit code	123456
98	-0000123456	1 edit code	1,234.56
99	-0000123456	2 edit code	1,234.56
100	-0000123456	3 edit code	1234.56
101	-0000123456	4 edit code	1234.56
102	-0000123456	A edit code	1,234.56CR
103	-0000123456	B edit code	1,234.56CR
104	-0000123456	C edit code	1234.56CR
105	-0000123456	D edit code	1234.56CR
106	-0000123456	J edit code	1,234.56-
107	-0000123456	K edit code	1,234.56-

Table 12-3. Examples of Edit Words and Edit Codes (Cont)

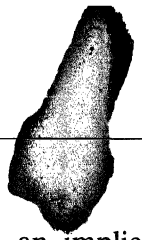
108	-0000123456	L edit code	1234.56-
109	-0000123456	M edit code	1234.56-
110	-0000123456	X edit code	!000123456
111	-0000123456	Z edit code	123456

1. With no edit word, the sign position is not converted to a numeric character. The first character is plus zero (+).
2. With no edit word, the sign position is not converted to a numeric character. The first character is a minus zero (!).
3. Blank edit word. Suppresses leading zero (0) characters.
4. Blank edit word. Suppresses leading zero (0) characters.
5. Blank edit word. Suppresses leading zero (0) characters.
6. This is the usual approach to editing an amount field. The decimal point (.) character is printed such that there are two decimal positions to the right of the decimal point (.) character. This is useful for printing dollars and cents. The comma (,) characters indicate each three positions left of the decimal point (.) character and the zero suppression (0) character immediately left of the decimal point (.) character causes zero suppression to the left.
7. The zero suppression begins where the zero suppression (0) character is specified and continues to the left.
8. The status portion is printed when the field is negative. Since no zero suppression (0) character is specified, all insignificant characters are suppressed.
9. The zero suppression (0) character specified for this position performs no function. The sign is lost as there is no status specified in the edit word.
10. The zero suppression (0) character specified for this position performs no function. The sign is lost as there is no status specified in the edit word.
11. From the right of the dollar sign ($) character to the left of the ampersand (&) character is ten characters. Eight zero (0) characters are suppressed. All zero fields are positive.
12. All nine zero (0) characters are suppressed. Only the extension portion of the edit word is printed.
13. This is the recommended way to edit an amount field. The decimal point (.) character is printed two places to the left in the output field to handle dollars and cents output results. The comma (,) character is printed three positions to the left of the decimal point (.) character and the zero suppression (0) character immediately left of the decimal point (.) character causes zero suppression to the left.
14. All insignificant zero (0) characters and punctuation are suppressed.
15. The first zero suppression (0) character is the zero (0) character. The asterisk (*) character is treated as a constant character.
16. Only the first zero suppression (0) character of the source data is suppressed. The second zero (0) character of the edit word is treated as a constant character.
17. Zero suppression of all insignificant zero (0) characters to the decimal point (.) character. Punctuation characters are also suppressed.
18. Zero suppression of all insignificant zero (0) characters to the decimal point (.) character. Punctuation characters are also suppressed.
19. Zero suppress with a floating dollar sign ($) character. The dollar sign ($) character is printed as far to the right as the zero suppression (0) character.

20. Floating dollar sign ($) character on a negative amount. Note that the edit word has one more replaceable position than the length of the source data.
21. The dollar sign ($) character does not float because it does not immediately precede the zero suppression (0) character. All superfluous punctuation is suppressed.
22. Zero suppression with the constant dollar sign ($) character and an extension portion.
23. Same as example 22 with status portion printing the minus sign (−) character.
24. Only the left-most position is zero suppressed and the dollar sign ($) character floats to the left of the first printed zero (0) character.
25. Floating dollar sign ($) character with the status portion (CR characters) being printed.
26. Floating dollar sign ($) character with the status portion (CR characters) being printed.
27. Floating dollar sign ($) character.
28. Floating dollar sign ($) character.
29. Note again that in order for the dollar sign ($) character to be printed, the edit word had to have one more replaceable character than the length of the source data.
30. Note again that in order for the dollar sign ($) character to be printed, the edit word had to have one more replaceable character than the length of the source data.
31. Same as example 29 with an extension. Note that the placement of the zero suppression (0) character can cause non-aesthetic results.
32. Since the dollar sign ($) and the zero suppression (0) characters are separated by the comma (,) character, the dollar sign ($) character is treated as a constant.
33. Check protect (*) character suppresses insignificant zero (0) and punctuation characters, filling each printable position with asterisk (*) characters.
34. The edit word is ten characters long but the asterisk (*) character occupies one position.
35. The asterisk (*) character is a replaceable character. When there are no insignificant characters, it is replaced.
36. The check protect (*) character suppresses insignificant zero (0) and punctuation characters, filling each position with an asterisk (*) character.
37. Same as example 33. The second asterisk (*) character is treated as a constant. In most cases this is undesirable and non-aesthetic.
38. The ampersand (&) character is a replaceable character (see example 45). The asterisk (*) character fills all insignificant positions. The zero (0) is treated as a constant.
39. Shows a 10-position field being edited by a 10-position edit word. The ampersand (&) characters are only for spacing and are not printed.
40. Shows a 10-position field being edited by a 10-position edit word. The ampersand (&) character is only for spacing and is not printed.
41. Same as example 40 with the status-reporting (CR) characters indicating a negative number.
42. Same as example 40 with the status-reporting (CR) characters indicating a negative number.
43. The extension, NET, does not print because the amount is positive. NET is interpreted by the generator as part of the sign status report, not the NET extension. Reversing the order, that is, CR&NET, causes the NET characters to be printed.
44. Same as example 43, except the amount is negative and causes the NET CR characters to be printed.
45. The dollar sign ($) character is treated as a constant and the ampersand (&) character is specified for spacing. Since the amount is negative, the CR characters are printed.
46. The first asterisk (*) character is treated as the check protect character, the second asterisk (*) character is treated as part of the sign-status reporting, and the third and fourth asterisk (*) characters are extensions.
47. Same as example 46 except the amount is negative and causes the CR characters to be printed.
48. The dollar sign ($) character is treated as a constant character and the ampersand (&) character is used for spacing. Zero suppression occurs and the minus sign (−) character is printed as the status reporting character.
49. Same as example 48, except the extension is printed.

50. The dollar sign ($) character is treated as a constant character and the ampersand (&) character is a replaceable character. The check protect (*) character is substituted for replaceable characters, insignificant zero (0) characters, and punctuation.
51. Suppresses the first three zero (0) characters of the source data, the fourth zero (0) character is printed. The LBS characters are treated as constants. The left ampersand (&) character prints as a blank character. The 02 characters are printed and the sign status report is positive, that is, "OZ.TARE –" is not printed.
52. Same as example 52 except the extension "OZ. TARE" is printed regardless of the TARE sign status report.
53. Special zero suppression is not specified so zero (0) characters are automatically suppressed. The ampersand (&) characters force the source data to be spaced. No sign status reporting is specified.
54. Automatic zero suppression with no sign status reported is performed. The PROFIT extension is printed.
55. Automatic zero suppression and sign status reporting (CR characters) is performed. The NET extension is printed.
56. Automatic zero suppression is performed, significant punctuation is printed, and space characters are printed across the DOLLARS and CENTS constants until the end of the source data is printed.
57. Automatic zero suppression and suppression of insignificant constants and punctuation in the edit word. The CENTS extension is printed.
58. Suppresses three zero (0) characters of the source data. Since the sign status reports are positive, the CENTS characters are not printed (extension characters must follow the sign status reporting characters to be printed).
59. Same as example 51. See example 52 for a more readable output.
60. One method of editing a social security number, with zero suppression specified.
61. If the left-most zero needs to be printed, specify the zero suppression (0) character in the edit word with one extra position on the left, and specify the zero suppression (0) character in the left-most position.
62. Zero suppression is specified only in the left-most position. Since the apostrophe (') character needs to be printed, two apostrophe (') characters must be specified for each that needs to be printed.
63. Shows the same type of editing as specified in example 60 with the LATER extension portion added. Automatic zero suppression is performed.
64. Another method of editing a data field using the virgule (/) character instead of the hyphen (-) character as in example 63.
65. The Y edit code is very useful in editing date fields. The largest field that can be edited is a 6-digit number.
66. The Y edit code on a 5-digit field left-justifies the data.
67. The Y edit code on a 4-digit field prints only one virgule (/) character.
68. The Y edit code on a 3-digit field prints only one virgule (/) character.
69. The 1 edit code of a 10-position field prints one zero (0) character on the right if the amount is zero.
70. The 2 edit code of a 10-position field prints no characters if the amount is zero.
71. The A edit code of a 10-position field prints one zero (0) character on the right if the amount is zero.
72. The B edit code of a 10-position field prints no characters if the amount is zero.
73. The X edit code of a 10-position field prints no characters if the amount is zero.
74. The Z edit code of a 10-position field prints no characters if the amount is zero.
75. The 1 edit code of a 10-position field prints one zero (0) character and no minus sign (-) character if the amount is a negative zero.
76. The 2 edit code of a 10-position field prints no characters if the amount is a negative zero.

77. The A edit code of a 10-position field prints one zero (0) character and no minus sign (-) character if the amount is a negative zero.
78. The B edit code of a 10-position field prints no characters if the amount field is a negative zero.
79. The J edit code of a 10-position field prints one zero character if the amount field is a negative zero.
80. The K edit code of a 10-position field prints no characters if the amount field is a negative zero.
81. The X edit code of a 10-position field prints the sign representation (!) character followed by nine zero (0) characters if the amount field is a negative zero.
82. The Z edit code of a 10-position field prints no characters if the amount field is a negative zero.
83. The 1 edit code of a 10-position field prints one decimal point (.) character followed by two zero (0) characters if the amount field is a positive zero.
84. The 1 edit code of a positive non-zero, 10-position field prints comma (,) and implied decimal point (.) characters.
85. The 2 edit code of a positive non-zero, 10-position field prints comma (,) and implied decimal point (.) characters.
86. The 3 edit code of a positive non-zero, 10-position field prints an implied decimal point (.) character.
87. The 4 edit code of a positive non-zero, 10-position field prints an implied decimal point (.) character.
88. The A edit code of a positive non-zero, 10-position field prints the comma (,) and implied decimal point (.) characters.
89. The B edit code of a positive non-zero, 10-position field prints the comma (,) and implied decimal point (.) characters.
90. The C edit code of a positive non-zero, 10-position field prints an implied decimal point (.) character.
91. The D edit code of a positive non-zero, 10-position field prints an implied decimal point (.) character.
92. The J edit code of a positive non-zero, 10-position field prints the comma (,) and implied decimal point (.) characters.
93. The K edit code of a positive non-zero, 10-position field prints the comma (,) and implied decimal point (.) characters.
94. The L edit code of a positive non-zero, 10-position field prints an implied decimal point (.) character
95. The M edit code of a positive non-zero, 10-position field prints an implied decimal point (.) character.
96. The X edit code of a positive non-zero, 10-position field removes the positive sign before printing the whole 10 character field.
97. The Z edit code of a positive non-zero, 10-position field removes the leading zero (0) characters before printing the amount.
98. The 1 edit code of a negative non-zero, 10-position field prints the comma (,) and implied decimal point (.) characters.
99. The 2 edit code of a negative non-zero, 10-position field prints the comma (,) and implied decimal point (.) characters.
100. The 3 edit code of a negative non-zero, 10-position field prints an implied decimal point (.) character.
101. The 4 edit code of a negative non-zero, 10-position field prints an implied decimal point (.) character.
102. The A edit code of a negative non-zero, 10-position field prints the comma (,), an implied decimal point (.), and the sign status report (CR) characters.



103. The B edit code of a negative non-zero, 10-position field prints the comma (,), an implied decimal point (.), and the sign status report (CR) characters.
104. The C edit code of a negative non-zero, 10-position field prints the implied decimal point (.) and the sign status report (CR) characters.
105. The D edit code of a negative non-zero, 10-position field prints the implied decimal point (.) and the sign status report (CR) characters.
106. The J edit code of a negative non-zero, 10-position field prints the comma (,), an implied decimal point (.), and the sign status report (-) characters.
107. The K edit code of a negative non-zero, 10-position field prints the comma (,), an implied decimal point (.), and the sign status report (-) characters.
108. The L edit code of a negative non-zero, 10-position field prints the implied decimal point (.) and the sign status report (-) characters.
109. The M edit code of a negative non-zero, 10-position field prints the implied decimal point (.) and the sign status report (-) characters.
110. The X edit code of a negative non-zero, 10-position field prints the sign representation (!) character in the left-most position followed by the remaining nine characters.
111. The Z edit code of a negative non-zero, 10-position field removes the leading zero (0) characters and prints the remaining significant digits.

PRINTER FILE HANDLING

An important objective of most RPG programs is the generation of a printed report. To help facilitate the production of an easily readable report with minimum programming, special features and functions have been incorporated into the Burroughs B 1000 RPG.

Page Formatting

The RPG I and RPG II dialects can both use the Line Counter Specifications and/or carriage control tapes to control paper movement through the printer. However, there are several differences in usage and specification between the two dialects.

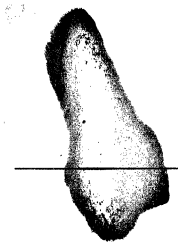
RPG I Dialect

The RPG I dialect uses the skip to channel option. Line Counter Specifications are required and must contain at least line-channel equations for channel 1 and channel 12.

The overflow line is the line specified in the line channel equations for channel 12. Up to 12 line-channel equations can be specified on the Line Counter Specifications form.

When a different forms length is desired, the length is entered in columns 15-19 and FL is entered in columns 18-19 of the Line Counter Specifications. Line-channel equations are required for channels 1 and 12. Up to nine additional line-channel equations can be specified.

Whenever a channel number is specified in columns 19-22 of the Output-Format Specifications, there must be a corresponding line-channel equation entry in the Line Counter Specifications. At compile time, when a skip to channel is found that does not have a corresponding line channel equation, a syntax message is emitted. Neither a syntax error nor a warning is issued if a line-channel equation is defined but is not referenced in the Output-Format Specifications.



RPG II Dialect

In RPG II dialect, Line Counter Specifications are optional. The default value for form length is 66 lines and the default value for the overflow line is line 60. When these default values are acceptable, as many as 12 line-channel equations can be specified on the Line Counter Specifications entry. If the defaults are not desirable, then both the form length and overflow line must be entered in the first two sections of the Line Counter Specifications and only 10 line-channel equations can be specified on the remainder of the specification line.

When the line-channel equations are supplied, skip to line is performed as skip to channel. If no line-channel equation is supplied, skip to line is performed by spacing in the usual manner.

Entries in columns 19-22 of the Output-Format Specifications are treated as line numbers.

End of Page

The physical end of a page is reached when the paper becomes positioned in the overflow area. For RPG I this is the area from channel 12 to one line prior to channel 1. In RPG II it is the first line after the overflow line to one line prior to line 1.

When the end of page (overflow area) is sensed during a printer operation, one of the following must occur:

1. Automatic skipping is specified when an overflow indicator is not entered in columns 33-34 of the File Description Specifications. Since the indicator is not defined, it cannot be used to condition calculations or output.

   Automatic skipping moves the paper from the overflow area to the top of the next page before resuming print operations.

   No overflow bookkeeping is available to the user.

   For RPG I, the paper is moved from the overflow area to channel 1.

   For RPG II, the paper is moved from the overflow area to line 6.

2. Continuous printing is specified when an overflow indicator is entered in columns 33-34 of the File Description Specifications but is not used to condition printer record output. The overflow indicator is defined and can be used in Calculation Specifications. If used on the Output-Format Specifications, the overflow indicator must condition only field descriptions of printer files.
3. Overflow operations are specified when an overflow indicator is entered in columns 33-34 of the File Description Specifications and that indicator is used to condition record output in the print file. Using the overflow indicator to condition printer file record output defines those lines of print that are to be written during overflow output. It is the responsibility of the programmer to initiate skipping based on overflow, as this is not done automatically.

When overflow operations are specified, normal overflow output can be printed when the overflow output routine in the RPG cycle is reached. If fetching was specified (F in column 16 of the record description line on the Output-Format Specifications), then overflow output can be printed immediately, before the record described on the line containing the letter F in column 16 is printed. This is fetched overflow output.

Overflow Indicators

Overflow indicators must be used only with printers and are not assigned by default. The assignment of overflow indicators determines the type of paper motion and overflow handling that occurs.

1. Overflow indicators not assigned and not used result in automatic skipping. In RPG II this is a skip to line 6. In RPG I this is a skip to channel 1.
2. Overflow indicators assigned but not used result in printing over the perforation.
3. Overflow indicators assigned and used result in normal overflow handling.

RPG I Dialect

An overflow indicator turns ON:

1. When a line is actually printed in the overflow area, or
2. When, as the result of using the overflow indicator as a calculation resulting indicator, the overflow indicator is set ON.

An overflow indicator turns OFF when:

1. The overflow indicator is used as a calculation resulting indicator, and the operation turns it OFF.
2. The conditions of the overflow turn OFF routine are satisfied. Refer to figure 12-21.

RPG II Dialect

An overflow indicator turns ON when:

1. The destination of a skip before or a skip after is in the overflow area.
2. The destination of space before or a space after is in the overflow area.
3. A line is actually printed in the overflow area.
4. The use of the overflow indicator as a calculation resulting indicator turns it ON.

The overflow indicator is not set OFF if the new page is reached by spacing operations. The overflow indicator is set OFF when:

1. The destination of a skip before or a skip after is a new page, and the record is not conditioned by the overflow indicator.
2. The overflow indicator is used as a calculation resulting indicator, and the operation turns it OFF.
3. The conditions of the overflow turn-off routine are satisfied. Refer to figure 12-21.

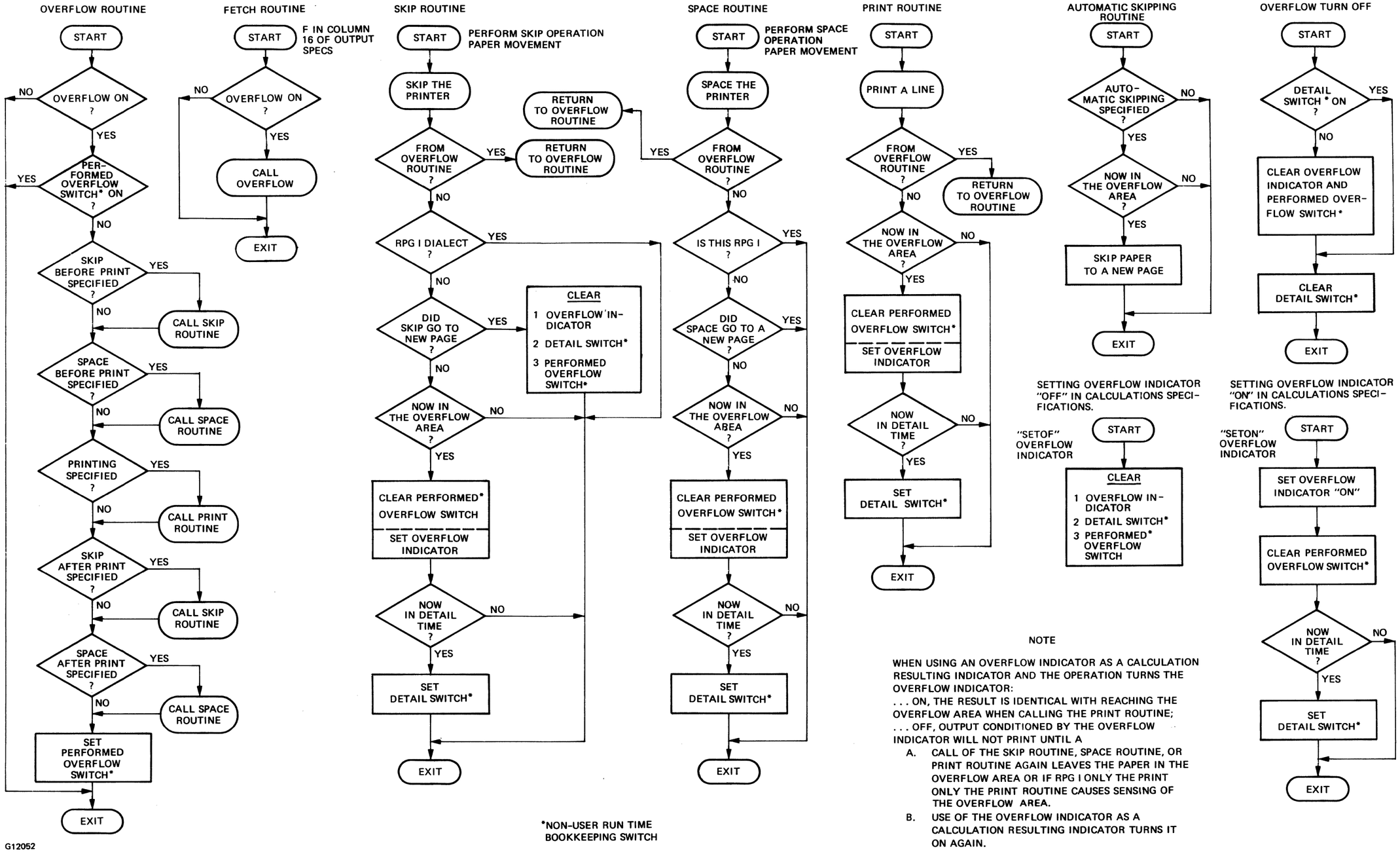


Figure 12-21. Printer Overflow Operations

Overflow

The principles used for handling overflow are:

1. Overflow is performed only once for any single physical overflow, whether fetched or performed during the normal part of the cycle when overflow processing is performed (for example, after total output).
2. The RPG program distinguishes between overflow indicators which are set during the detail-time part of the cycle and those which are set during the total-time part of the cycle. Refer to appendix H for a description of when detail-time and total-time routines occur in the RPG cycle. If set during detail time, overflow indicators do not get turned off after detail-time output but are maintained for one full cycle until the next time around. If an overflow indicator went on during detail and overflow was fetched, the reason overflow does not get performed twice (once for the fetch and once at normal overflow processing) is because overflow is performed only once for each physical overflow.

   If the program overflow specifications leave the current line position in the line printer overflow area, the RPG cycle does not erroneously perform overflow processing again but remembers that it has already been performed for the condition that caused overflow. To do overflow processing again, the program must cause another physical overflow by printing again in the overflow area.
3. Overflow indicators can also be controlled in the Calculation Section when used as resulting indicators. When an overflow indicator is set on in the Calculation Section, the effect is exactly the same as if a physical overflow condition had occurred on the printer, with the cycle also distinguishing whether it has been turned on in detail or total. When an overflow indicator is set off in the Calculation Section, the effect is exactly the same as if overflow processing had just been performed.

A composite flow chart of printer overflow operations is shown in figure 12-21.

The sequence of printed output operations when all conditioning indicators are satisfied follows:

1. If skip before is specified, call SKIP ROUTINE.
2. If space before is specified, call SPACE ROUTINE.
3. Call PRINT ROUTINE.
4. If skip after is specified, call SKIP ROUTINE.
5. If space after is specified, call SPACE ROUTINE.
6. If fetching is specified for this line of print, items 1 through 5 are done once for the fetch (call FETCH OVERFLOW), and again for the line to be printed.
7. After all detail output, call OVERFLOW TURN OFF ROUTINE.
8. After all total output, if the last record indicator (LR) is off, call either:

   1) The OVERFLOW ROUTINE, or
   2) If automatic skipping is specified, call the AUTOMATIC SKIPPING ROUTINE.

Table 12-4. Effect of Printer Operations on Overflow Indicators Destination

	RPG I	RPG II
Overflow Area:		
Skip	No Effect	Turn ON
Space	No Effect	Turn ON
Print	Turn ON	Turn ON
New Page:		
Skip	No Effect	If on a line not conditioned by an overflow indicator, then turn OFF.
Space	No Effect	No Effect
Print	No Effect	No Effect
Overflow Area	Channel 12 to 1 line prior to channel 1.	First line after overflow line to last line prior to line 1.
Automatic Skip	Paper is moved from overflow area to This occurs when an overflow indicator is not assigned in the File Description Specifications for a printer line and is not specified in the rest of the program.	Paper is move from overflow area to line 6.

Normal Overflow Output

When the overflow area is sensed and the indicator comes on, and normal overflow output is specified, then the following events takes place, in sequence, during detail calculation exception output:

1. Complete remaining detail calculations.
2. Perform detail and heading output not conditioned on the overflow indicator.
3. Read the next record.
4. If a control break occurred, perform total calculations.
5. Perform necessary total output not conditioned on the overflow indicator.
6. Perform total output conditioned on the overflow indicator.
7. Perform heading and detail output conditioned on the overflow indicator.
8. Perform detail calculations.
9. Perform heading and detail output not conditioned on the overflow indicator.
10. Turn OFF the overflow indicator.

When the overflow area is sensed during detail output, the sequence is item 2 through item 10.

When the overflow area is sensed during total calculation exception output, the sequence is item 4 through item 10.

Figure 12-22 provides a graphic view of when the overflow indicator is turned ON or OFF and when printing is performed.

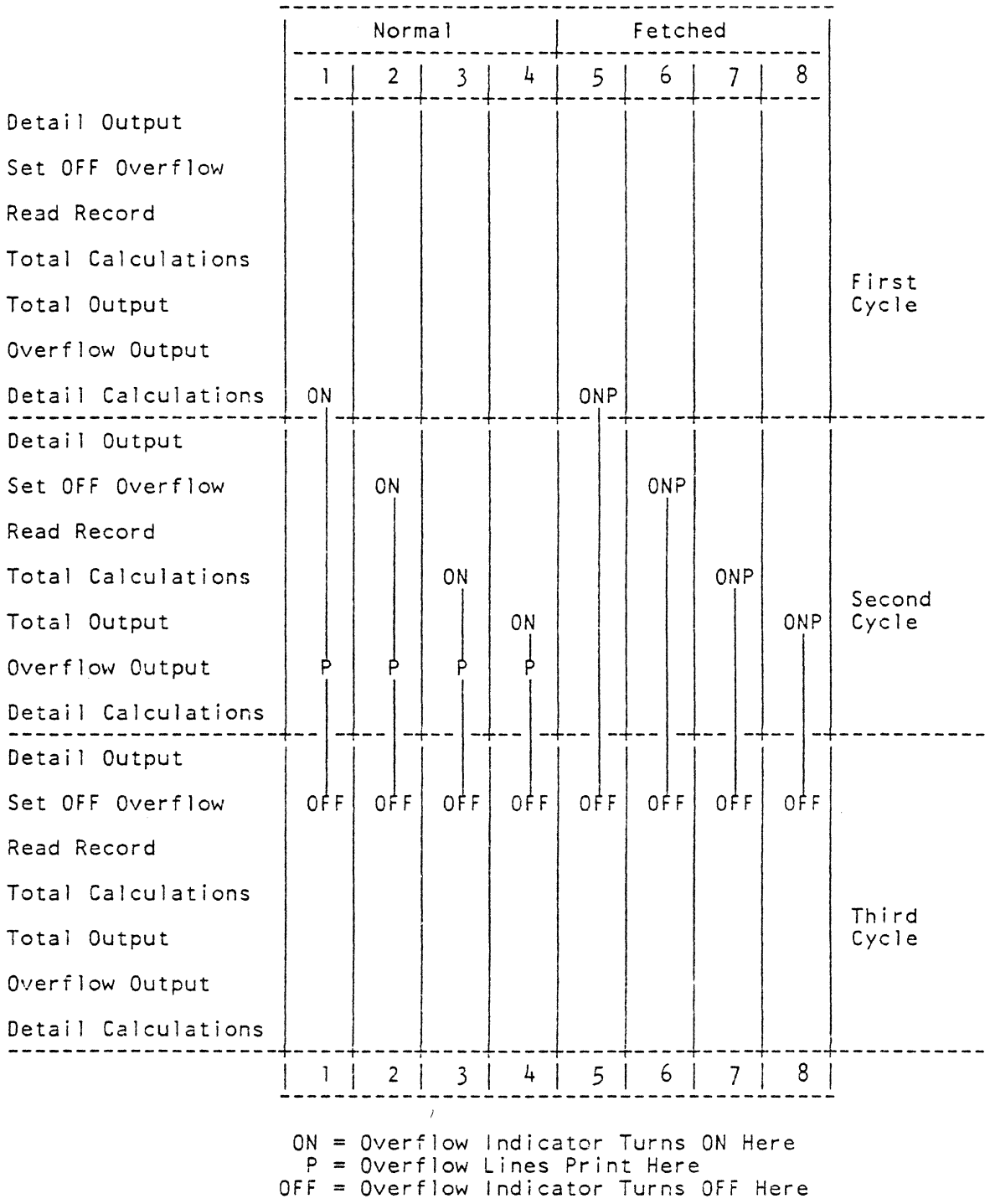


Figure 12-22. Bar Schematic of Printer Overflow Operations

Fetched Overflow Output

When the overflow indicator is sensed and fetched overflow output is specified, the following sequence of events takes place. Before printing the record specified with the letter F in column 16 of the Output-Format Specifications:

1. Perform total output conditioned by the overflow indicator.
2. Perform heading and detail output conditioned by the overflow indicator.
3. Print the record with the letter F in column 16 that caused the overflow routine to be fetched.

If the overflow area was sensed during detail calculation exception output, the remainder of the sequence is:

1. Perform the remainder of the detail calculation exception output, if any.
2. Finish the remainder of the detail calculations.
3. Perform detail output not conditioned on the overflow indicator.
4. Read the next record.
5. If a control break has occurred (item 4), perform total calculations.
6. Perform total output lines not conditioned on the overflow indicator.
7. Do not perform overflow output. For exceptions, refer to the following subsection titled Multiple Output Lines.
8. Perform detail calculations.
9. Perform detail output.
10. Turn OFF the overflow indicator.

If the overflow area was sensed during detail output, the remainder of the sequence is item 3 through 10.

If the overflow area was sensed during total calculation exception output, the remainder of the sequence is item 5 through 10.

If the overflow area was sensed during total output, the remainder of the sequence is item 6 through item 10.

Multiple Output Lines

When multiple output lines have been specified for any printer, it is possible to continue printing, either before or after overflow output, and again arrive in the overflow area. If this happens, all overflow output can occur again.

Examples of overflow coding are shown in figures 12-23 and 12-24.

```
                File Description Specifications

01 FCARDS   IP
02 FLISTING O          132     OF    LPRINTER
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

NOTE

Entry in the OVERFLOW INDICATORS field (columns 33-34) discontinues automatic handling of overflow.

```
                Line Counter Specifications

01 LLISTING  66FL 56OL
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

NOTE

This Line Counter Specifications defines the form length to be 66 lines and the overflow line to be line 56.

```
           Output-Format Specifications

01 OLISTING H  301    1P
02 O        OR        OF
03 O                                      45 'HEADING'
                .
                .  (Detail output lines)
                .
nn O         T 2      LR
nn O        OR        OF
nn O                                      55 'TOTAL'
nn O                           TOTAL 1    62
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

NOTE

The Output-Format Specifications cause HEADING to be printed on the first page and all overflow pages (OF indicator is ON). When the LR (last record) indicator is ON, the Output-Format Specifications cause TOTAL to be printed last on all overflow pages (OF indicator is ON).

Figure 12-23. Overflow Coding Example 1 – RPG II

```
                File Description Specifications
01 FLISTING O         132     OF    LPRINTER
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

NOTE

Entry in the OVERFLOW INDICATORS field (columns 33-34) discontinues automatic handling of overflow.

```
              Line Counter Specifications

01 LLISTING  1001 4512 1302
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

NOTE

Channels 1, 12, and 2 are specified so that channel skipping can be handled in the Output-Format Specifications.

```
          Output-Format Specifications

01 OLISTING H   01   1P
02 O                                    42 'HEADING'
03 O         H  02
04 O                                    62 'DATE TIME PLACE'
             .
             .  (Detail output lines)
             .
nn O         T  12   LR
nn O        OR       OF
nn O                                    45 'TOTAL'
nn O                          TOTAL 1   55
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

NOTE

The Output-Format Specifications cause HEADING to be printed on the first page and totals to be printed on all overflow pages (OF indicator is ON) when the LR (last record) indicator is ON.

Figure 12-24. Overflow Coding Example 2 – RPG I

SECTION 13
DOLLAR CARD SPECIFICATIONS

The Dollar Card Specifications accommodate various extensions to the B 1000 RPG Language, which cannot be handled on the other specification forms. Dollar Card Specifications allow certain compiler-control options to be set or reset during compilation.

Dollar Cards can appear anywhere within the source deck, as required. The standard Dollar Card Specifications form can be used or options can be coded on the other specification sheets at the points where they are to be placed in the source deck.

FIELD DEFINITIONS

Refer to figure 13-1 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 can be left blank or used to contain the form type of the specification in which the option is to be inserted.

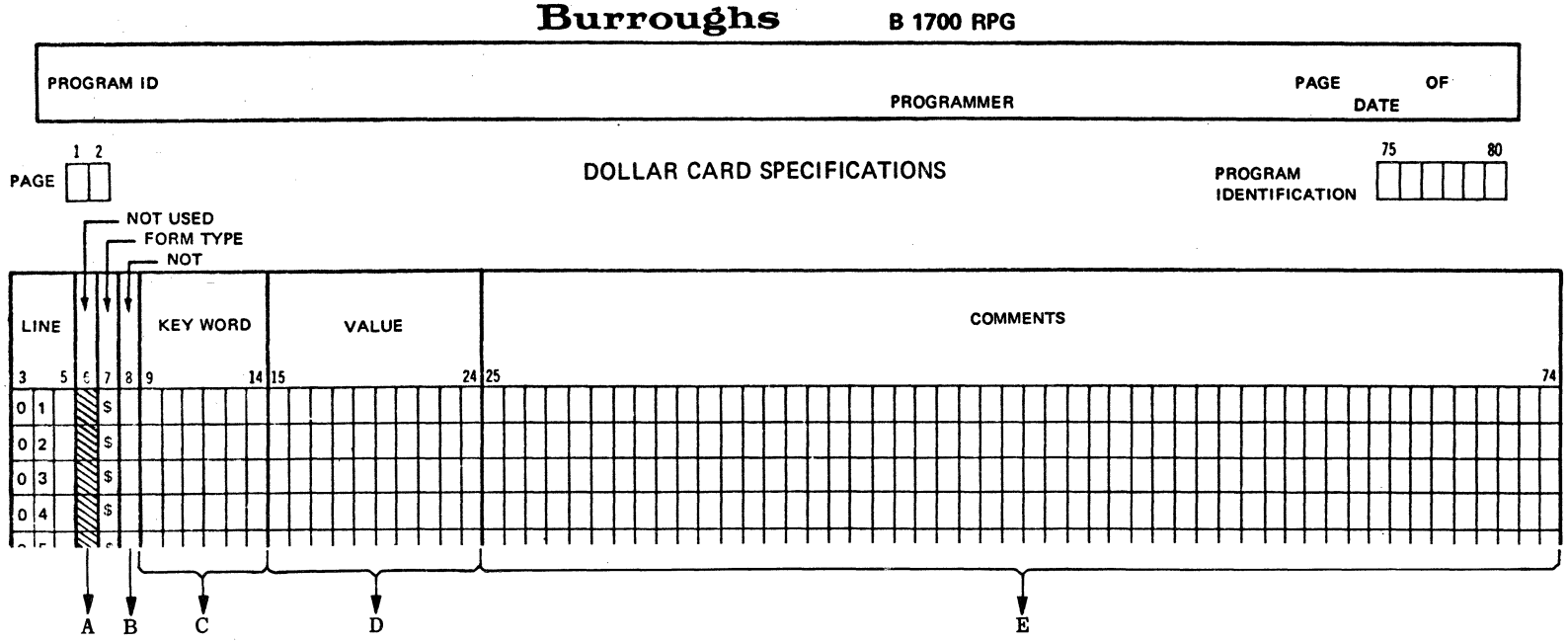
7 $ OPTION

A dollar sign ($) character 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 13-1. Dollar Card Specifications Summary Sheet

9-14 KEY WORD OPTION

The KEY WORD OPTION field contains the compiler option or extension key word. Unrecognized compiler options or extension key words are ignored by the compiler and a warning is emitted. 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.

15-24 VALUE

The VALUE field specifies a value to be associated with the option entered in the KEY WORD field. Not all options require a value; those that do so are 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

The COMMENTS field is available for inclusion of comments and documentary remarks, and can contain any valid EBCDIC characters.

75-80 PROGRAM IDENTIFICATION

Refer to section 2 for a complete description.

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 can be reset.

DISKID Extension

The DISKID extension specifies the pack name of a disk file. If not specified, the disk file defaults to the SYSTEM disk. This entry should be included to ensure correct handling of the file by the operating system (MCPII).

FAMILY Extension

The FAMILY extension specifies the external main directory family name associated with the file. The value field contains the name (one to 10 characters left-justified).

FILEID Extension

The FILEID extension specifies the external file name associated with the file. The value field contains the name (one to 10 characters left-justified).

PACKID Extension

Same as the DISKID extension.

External File Name Format

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

family name
family name/file ID
disk pack ID/family name/file ID
disk pack ID/family name/

The internal name of each file is the filename assigned in the File Description Specifications.

Unless the FAMILY extension is specified, the family name is 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 are the same, for example, the family name. Figure 13-2 illustrates how the insertion of the various file identification extensions before a file called MASTER affects that external name of the file.

```
  Specifications                External Names
  --------------                --------------

01  $ FILEIDMASTERFILE          MASTER/MASTERFILE
02 F MASTER  IP    180
--*----1----*----2----

01  $ FAMILYPAYROLL             PAYROLL
02 F MASTER  IP    180
--*----1----*----2----

01  $ FAMILYPAYROLL
02  $ FILEIDMASTERFILE          PAYROLL/MASTERFILE

01  $ FAMILYPAYROLL
02  $ DISKIDPAYMASTER           PAYMASTER/PAYROLL/
03 F MASTER  IP    180
--*----1----*----2----

01  $ FAMILYPAYROLL
02  $ FILEIDMASTERFILE
03  $ DISKIDPAYMASTER           PAYMASTER/PAYROLL/MASTERFILE
04 F MASTER  IP    180
--*----1----*----2----
```

Figure 13-2. Coding File Identification Extensions – Dollar Sign Options

RPG EXTENSIONS

The following options can appear only within the File Description Specifications, except RSIGN, and must immediately precede the Specifications line describing the file to which they apply. None of the following operations can be reset except RSIGN and ONEPAK.

AAOPEN Extension

The AAOPEN option allocates all the disk areas requested when the file is opened.

AREAS Extension

The AREAS extension 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. The maximum number of areas that can be assigned is 105.

CLOSE extension

The CLOSE extension causes the serial input file to remain open until the program reaches end of job (EOJ). The CLOSE extension must appear before the File Description Specifications for each file that is to remain open until the program goes to end of job. The default is for each serial input file to be explicitly closed upon encountering the end-of-file record.

DIGIT8 Extension

The DIGIT8 extension causes RPG indexed files to use 8-digit relative record numbers in the tag file. The default is six digits. With this option, RPG indexed files are compatible with COBOL indexed files. The DIGIT8 extension must be specified prior to each File Description Specifications of the file for which an 8-digit relative record number tag file is required.

DMSNAM Extension

The DMSNAM extension prints the DMSII library files used for the compile. This option is set on by default.

DNAME Extension

The DNAME extension causes the alpha mnemonic specified in the VALUE field to be used as the physical hardware device for the file.

DRIVE Extension

The DRIVE extension allocates a physical drive to that particular file. Applies only to SYSTEM disks. The VALUE field must be 0-15. This extension cannot be reset and is not related to the DISKID or PACKID extensions.

IXSEQ Extension

The IXSEQ extension specifies that the file is processed as an indexed-sequential file. This option can only be specified for an indexed file. A syntax error is issued if B-style indexed files are declared in the program.

ONEPAK Extension

The ONEPAK extension specifies that this particular file must be contained in one disk.

OPEN Extension

The OPEN extension causes the explicit open of a file at beginning-of-job (BOJ) time. An OPEN extension must appear before the File Description Specifications for each file that is to be explicitly opened at beginning-of-job time of the program.

The default condition is as follows:

At BOJ time, the primary and all secondary files are opened, as well as all input or update indexed files. Demand and output files are implicitly opened upon the first read from, or write to.

REFORM Extension

The REFORM extension causes the record and block sizes stored in the disk file header to be used regardless of those specified in the File Description Specifications. The REFORM extension causes the record and block sizes declared in the File Specifications card to override those in the disk file header. For example, the DEFAULT bit in the File Parameter Block is reset.

The REFORM extension must appear before each input or update disk file where it is necessary to override the record and block sizes specified in the disk file header.

REORG Extension

The REORG extension specifies a specialized method of sorting indexed files and is invoked at end of job (EOJ). This method sorts only the additions and then merges them, in place, into the master file. This method of sorting should decrease the sort time and the temporary disk area required. The VALUE field (columns 15-46) must contain the external file identifier of the indexed file. The VALUE field has the following naming convention:

disk pack ID/family name/file ID

The use of the REORG extension results in the following program execution at EOJ:

INDEXED FILE – RPG/REORD
"B" INDEXED FILE – RPG/REORG

The RPG compiler generates a warning message if the REORG extension is specified for an indexed-sequential file.

RPERA Extension

The RPERA extension specifies the maximum number of logical records that are written in each disk area. The VALUE field contains an integral value (right-justified, leading zeros optional). The default value assigned is the first multiple of the records-per-block that is equal to or greater than 500.

RSIGN Extension

The RSIGN extension 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 (right-most) position of the field; when reset, all signs are assumed to be in the high-order (left-most) position of the field. This option can 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 overrides the sign position specified in the Control Card Specifications.

SECURE Extension

The SECURE extension causes RPG programs to access and create single-named tag files when the program is executed using the USERCODE/PASSWORD pair implemented on the B 1000 file security system. Additionally, the SECURE extension allows an RPG program to use the same file-naming convention for tag files as is used with ANSI 68 COBOL tag files.

STACK Extension

The STACK extension changes the subroutine stack size of the program. This should be used only when a legitimate STACK overflow condition has occurred, as when nested subroutines exist more than eight deep. The default stack size is 313 bits which allows eight entries in the stack. To increase the stack size, add 39 bits to the default size of 313 for each additional stack entry.

TAG Extension

The TAG extension specifies that the file is processed as an indexed (tag) file. This option can be specified only for an indexed file. The RPG compiler generates a syntax error if the TAG extension is specified and B-style indexed files are specified in the program.

Only one of the following four extensions can be used per File Description 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 can appear anywhere within the RPG source program. These options direct the compiler to perform specific functions. All of the following options can be "reset".

CHECK Option

The CHECK option causes all patching records to be sequence checked. Syntax errors are generated for any patch record that is out of sequence.

If the CHECK extension is not specified, then sequence warnings are issued. The warning is denoted by the letter S appearing to the left of the sequence number of the record on the output listing.

CSIGN Option

The CSIGN option forces the sign digit in the output of right-signed, packed, and numeric data to the hexadecimal C value.

FSIGN Option

The FSIGN option forces the sign digit in the output of right-signed, packed, and numeric data to the hexadecimal F value.

LIBR Option

The LIBR option copies source records from a library file located on tape or disk to a new or merging file. This option can reference various library files. Columns 15-24 contain the pack name ; columns 25-34 the multi-file-name; columns 35-44 the File-Id; columns 45-49 the starting sequence number in the library file (optional); and columns 50-54 the ending sequence number in the library file (optional).

The letter L is appended to the record image and is placed to the left of the sequence field of the record in the output listing. The letter L specifies that the record is from a library source file.

LIST Option

The LIST option causes the compiler to 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 does not inhibit the errors or warning messages from printing.

LOGIC

The LOGIC option produces a single-spaced listing of each source specification line, followed immediately by an intermediate code used to generate RPG S-code. The listing is produced after the NAMES listing (if the NAMES option is set) and does not include addresses or bit configurations, but only the RPG S-codes and logical operands of the program.

MAP Option

The MAP option produces 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).

MERGE Option

The MERGE option causes the following records in the patch file to be merged with an existing source file. The source file is from a source other than a card reader. The source file is expected to reside on tape or disk (disk is default). The internal file name is SOURCE and the external file name is SOURCE. The default blocking option is set.

NAMES Option

The NAMES option produces 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.

NEW Option

The NEW option creates new output source files. If the MERGE option is specified, then the patches are included in the new source file. The new source file is created on disk or tape (disk is default). The internal file name is NEWSOURCE and the external file name is NEWSOURCE. If it is desirable to change the external file name or input device from disk to tape, then label equation cards must be used.

NEWID Option

The NEWID option causes the 6-character identifier specified in columns 15-20 of the NEWID option to be placed in columns 75-80 of the subsequent lines in the output listing. If the NEW option is specified, then the 6-character identifier is also placed in columns 75-80 of the subsequent source records in the new source file.

PAGE Option

The PAGE option ejects a page on the output source listing. This option can appear anywhere in the RPG source program and is especially useful for separating subroutines on the output source listing.

PARMAP Option

The PARMAP option produces a single-spaced listing of the compiler-generated paragraph names, source statement numbers, and actual segment displacements of the emitted code. This listing can be used to relate to the LOGIC listing.

SEQ Option

The SEQ option starts sequencing the subsequent source lines on the output listing. If the NEW option is specified, then the subsequent new source file is sequenced. The starting sequence number begins with the number contained in the sequence field (columns 1-5) of the SEQ option. If the sequence field is blank, the starting sequence number begins with zero. The subsequent records have sequence numbers which are incremented by the value contained in columns 22-24 of the SEQ option. If columns 22-24 are blank, then 010 is assigned.

SUPR Option

The SUPR option suppresses all warning messages from the source program listing. All error messages still print.

VOID Option

The VOID option deletes records of the source file on the output listing. If the NEWID option is specified, then the records are deleted in the new source file. Source lines or records are deleted up to and including the 5-digit void limit. If the void limit field (columns 20-24) is blank, then only the source record with sequence number matching the sequence field VOID option is deleted.

XMAP Option

The XMAP option prints 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.

XREF Option

The XREF option causes all the field names in the RPG source file to be listed in alphabetical order. Associated with each field name is the sequence number(s) on which it was referenced.

SECTION 14
COMPILER OPERATION

The Burroughs RPG compiler is an integral part of the B 1000 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 can 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 control cards is as follows:

```
? COMPILE <program-name> WITH RPG [ { TO LIBRARY } ] <control-attributes>
                                  [ { FOR SYNTAX } ]
                                  [ {    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 1000 Systems System Software Operation Guide, Volume 1. The format of the four options of the COMPILE statement are discussed here.

COMPILE System Command

This is a "compile and go" operation. If the compilation is error-free, the operating system (MCPII) schedules the object program for execution. The program is not 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 can be abbreviated as CO.

COMPILE TO LIBRARY System Command

This option leaves the program object file on disk and enters the name of the program into the Disk Directory if no syntax errors occurred during the compilation. The program is not scheduled for execution. LIBRARY can be abbreviated as LI.

COMPILE SAVE Statement

This option combines the execute and library options. The operating system (MCPII) enters name of the program into the Disk Directory and leaves the object program file on disk. The operating system (MCPII) also schedules the program for execution after an error-free compilation. The program remains in the Disk Directory. SAVE can be abbreviated as SA.

COMPILE FOR SYNTAX Statement

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 can be abbreviated as SY.

The following sample deck could be used for compiling to library a source program contained in a punched card deck.

```
? COMPILE PROGRAM/TEST1 WITH RPG TO LIBRARY
? CHARGE 12345
? DATA RPG/CARD
  source specification cards
? END
```

The following sample deck could be used to compile a source program contained on magnetic tape for syntax:

```
? COMPILE AAA/BBB/TEST2 WITH RPG FOR SYNTAX
? FILE CARDS NAME RPG/SOURCE DEFAULT TAPE
? END
```

NOTE

For the effect of the DEFAULT option, refer to the B 1000 Systems System Software Operation Guide, Volume 1.

VECTOR FILE INPUT

The vector input files consist of compile-time and pre-execution-time vector files, both of which are described in the paragraphs that follow.

Compile-Time Vector Files

If the program requires that table or array files be included during compilation, the compiler requires an input card file labeled RPG/VECTOR. This file can be input from a medium other than cards, through use of the FILE statement. Card input can be punched in either the BCD (96 column card) or EBCDIC (80 column card) character set.

Vector files must be entered in the same order as specified in the Extension Specifications, and each file must contain exactly the number of records specified. No separators define the end of one vector and the beginning of the next; records are read into one vector until that vector is full. Filling of the next vector is begun from subsequent records.

The following sample deck could be used to compile a source program (with compile-time vector input) contained on cards to library:

```
? COMPILE AAA/TEST3 WITH RPG TO LIBRARY
? DATA RPG/CARD
  source specification cards
? END
? DATA RPG/VECTOR
  table file cards
? END
```

Pre-Execution-Time Vector Files

If the program requires table or array files to be included at the beginning of object program execution, the program does require card files with the labels as specified by the program in the File Description and Extension Specifications.

LABEL EQUATION CARD

The label equation card is used to change a compiler file name in order to avoid duplication of file names in the disk directory when operating in a multiprogramming environment, or to access a file name other than the default file names in the RPG compiler.

Table 14-1 lists the files that are used by the RPG compiler for source input and compilation output.

Table 14-1. Files Used for Source Input and Compilation Output

Internal File Name	External File Name	Description
CARDS	RPG/CARD	This file is an input card file that contains the source input to the RPG compiler.
DMS_LIBR	DMS_LIBR	This file is a temporary input work file used to read the DMS/DASDL-generated RPG library files.
ERROR_LINE	ERROR_LINE	This file is an output print file that contains the syntax errors and warning messages of this compilation.
LIBRARY	LIBRARY	This file is a temporary input work file that is used to read RPG library files when the LIBR compiler option is specified.
LINE	RPG/LIST	This file is an output print file that contains the source listing generated by the RPG compiler.

Table 14-1. Files Used for Source Input and Compilation Output

Internal File Name	External File Name	Description
NEWSOURCE	NEWSOURCE	This file is an output disk file that contains the source images read as source input to the RPG compiler and is created when the NEW compiler option is specified.
LIBRARY	LIBRARY	This file is a temporary work file that contains the symbolic and absolute code information during the third and fourth passes of the RPG compiler, respectively.
RPGCTL	RPGCTL	This file is a temporary work file used to store the RPG compile-time vector data.
RPGINT	RPGINT	This file is a temporary work file used to store RPG source images.
RPGOBJ	RPGOBJ	This file is an output disk file that contains the RPG object code.
RPGXRF	RPGXRF	This file is a temporary sort file used when the XREF compiler option is specified and contains the records that have been sorted.
RPGXRT	RPGXRT	This file is a temporary sort file used when the XREF compiler option is specified and contains the records to be sorted.
SOURCE	SOURCE	This file is an input disk file that contains the source images to be read by the RPG compiler when the MERGE compiler option is specified.
TABCRD	RPG/VECTOR	This file is an input card file that contains the data for the compile-time vectors.
XRFLINE	XREF/RPG	This file is an output print file that contains the cross-reference information when the XREF compiler option is specified.

Label equation cards use the FILE statement. Refer to the B 1000 Systems System Software Operation Guide, Volume 1, for additional information. The format of the FILE statement is:

? FILE <internal-file-name> NAME = <file-identifier>;

The FILE statement must immediately follow a COMPILE, EXECUTE, DYNAMIC, or MODIFY statement.

The following sample deck could be used to compile a source program which merges a patch file on cards with a SOURCE program located on disk. The SOURCE program's name is PROGRAM1 and the OBJECT program's name is OBJ.PROG1.

```
?  COMPILE OBJ.PROG1 WITH RPG TO LIBRARY
?  FILE SOURCE NAME PROGRAM1
?  DATA RPG/CARD

   $ MERGE card followed by patch card source statements
?  END
?  DATA RPG/VECTOR

   table file cards

?  END
```

LARGE RPG II PROGRAM CODE FILES

Code files on the B 1000 system must be contained in one disk area. The RPG compiler can abort with the following message:

<object-file-name> EXCEEDED ITS FILE SPACE: 1 AREAS, EACH
WITH <integer> BLOCKS, EACH WITH <integer> SECTORS.

To avoid this compiler abort, increase the value of the BLOCKS.PER.AREA file attribute for the RPGOBJ internal file of the RPG compiler as follows:

Compile Time Solution:

COMPILE <code-file-name> RPG LIBRARY
FILE RPGOBJ BLOCKS.PER.AREA = <number;

Permanent Fix:

MODIFY RPG FILE RPGOBJ BLOCKS.PER.AREA = <number>;

<number> represents the number of blocks in one area of a disk file. The RPG compiler has a default value of 800 for the BLOCKS.PER.AREA file attribute, which allows for a code of up to 800, 180-byte records. Note that all code file records are 180 bytes long.

SECTION 15
DATA MANAGEMENT SYSTEM SPECIFICATIONS

RPG DATA MANAGEMENT FACILITY

The RPG data management system provides the ability to process a DMSII data base using conventional RPG syntax and concepts.

A data base is constructed by a DMS/DASDL compilation. The DMS/DASDL compiler, using a description of the data base (DMS/DASDL source statements), produces a data base dictionary file which contains information about each structure described within the data base.

Data sets, sets, and subsets are the structures that make up the data base. A data set description in DMS/DASDL specifies the logical structure of a file. Set and subset descriptions in DMS/DASDL specify the logical structures of indexes or index tables (paths) that are used in storage and retrieval of the data contained in a data set.

Data sets and sets on the outermost level of the description are disjoint data sets and disjoint sets respectively. A data base must contain at least one disjoint data set.

In order to effectively use the RPG data management system, understanding how the data base structures relate to the concepts of RPG files is important.

RPG Data Management Files

The basic structure in a data base is the disjoint data set. A data set is similiar to the RPG concept of a file in that a data set contains the actual records. Unlike RPG, a record can contain not only items of information but other data sets (embedded data sets).

Additionally, records within a data set can contain access paths (or pointers) to the records of another disjoint data set. This access path is called a manual subset.

In order to better understand an embedded data set, consider the data set EMPLOY containing records for each employee. This data set can contain an embedded data set named WRKHIS for each employee's work history. For every employee record in the EMPLOY data set, zero, one, or more work history records can be stored in the WRKHIS data set. Later, when employee records are being read, the DMSII system delivers all the requested WRKHIS records which have been previously stored. If there are no WRKHIS records, the DMSII system informs the RPG program of this condition.

To better understand how a manual subset is used, consider two disjoint data sets, DEPART and EMPLOY. To gain access to all of the employees within each department ordered by the employee's last name, a manual subset can be used. The manual subset provides paths from one disjoint data set to another disjoint data set. That is, within each DEPART record there is a manual subset called DEPEMP.

The following DMS/DASDL source statements pertain to the previous example:

```
DEPART DATA SET
     (
      .
      .
      .
      DEPEMP SUBSET OF EMPLOY KEY IS (LASNAM)
      .
      .
       );
EMPLOY DATA SET
     (
      .
      .
      .
      LASNAM
      .
      .
      .
       );
```

By manual, it is meant that the RPG program must insert the record into the manual subset DEPEMP after creating and storing a record in the data set EMPLOY. Similarly, the RPG program must remove the record in the manual subset DEPEMP after deleting the record in the data set EMPLOY.

A data set can be accessed by way of various paths or, in DMSII terminology, by means of sets or subsets. Sets, automatic subsets, and manual subsets with keys are structures that organize the data set records into logical sequences. In RPG terms, this is known as indexing by way of a key. A set provides access to all of the records of a data set. An automatic subset provides access to a limited collection of records, that is, a condition for membership in the subset exists, and the condition is checked each time a record is to be added to the data set.

Sets and automatic subsets declared as indexed-sequential in the DMS/DASDL source file allow for accessing successive records based on the ordering sequence of the key, or a given record can be accessed based on the value of the key. In DMS, several sets and/or automatic subsets can exist for the same data set. That is, the same data set can be accessed via several different keys.

An indexed-random set is a structure which provides access to a data set record based on the value of a key. Unlike other sets and automatic subsets, ordering is not implied, except in the isolated case where two or more records contain duplicate key values. In this case, access to the next record makes available the record which contains a duplicate of the key. All sets and subsets, other than manual subsets, are maintained by DMSII. When a record is added, updated, or deleted, DMSII adjusts all indexes affected by the various "key" values in the record. No special action is required in the RPG program.

DATA BASE SPECIFICATIONS

There must be only one Data Base Specifications in an RPG program.

The Data Base Specifications is optional but is required when any DMSII files are declared in the program. If used, the Data Base Specifications must immediately follow the Control Specification, if one exists.

Field Definitions

1-2 Page

Refer to section 2 for a complete description.

3-5 LINE

Refer to section 2 for a complete description.

6 FORM TYPE

This field must contain the letter D.

7-16 DATA BASE NAME

This field must contain the name of the physical data base being used and must be the same data base name referenced in the data-base compile. The RPG programmer must specify the exact name of the data base here.

17-26 LOGICAL DATA BASE

This field can contain the name of the logical data base as specified in the data-base compile. The logical data base is a portion of the physical data base which can be accessed by the program. This logical data base name must have been defined in data-base source file. If no entry is made, the entire physical data base is available to the RPG program.

27 ACCESS MODE

This field must contain the letter I or U. If the data base is to be accessed input only, enter the letter I. If the data base is to be updated, enter the letter U.

28-74

These columns must be left blank.

75-80 PROGRAM IDENTIFICATION

Refer to section 2 for a complete description.

DMS/RPG Library Files

If the RPG library files created by the DMS/DASDL compiler reside on a disk other than the system disk, the Data Base Specifications must be preceded by a PACKID or DISKID file identification extension.

Figure 15-1 shows an example of coding a Data Base Specification.

```
                Data Base Specifications

01 DPAYROLL                 U
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

NOTE

This is an example of a data base specification where PAYROLL is the name of the data base. The letter U in the ACCESS MODE field (column 27) allows the program to read and write to the PAYROLL data base.

Figure 15-1. Data Base Specifications

DATA MANAGEMENT FILE DESCRIPTION SPECIFICATIONS

Every disjoint data set to be used by an RPG program must be described to the RPG compiler in the File Description Specifications. The disjoint data set must be specified in the data-base source file. Figure 15-2 shows the various processing methods for DMSII files.

```
          File Description Specifications

01 F*
02 F* SEQUENTIAL PROCESSING WITH NO INDEX
03 F*
04 F         IP                          DMS                           D
05 F         IP                          DMS
06 F         IS                          DMS                           D
07 F         IS                          DMS
08 F         UP                          DMS                           D
09 F         UP                          DMS
10 F         US                          DMS                           D
11 F         US                          DMS
12 F         UP                          DMS                           D
13 F         UP                          DMS
14 F         US                          DMS                           D
15 F         US                          DMS
16 F*
17 F* SEQUENTIAL PROCESSING BY MEANS OF AN INDEX.  COLUMNS 47-52
18 F* MUST CONTAIN THE NAME OF A DMS/DASDL-DEFINED INDEX
19 F*
20 F         IP                          DMS     INDEX                 A
21 F         IP                          DMS     INDEX                 B
22 F         IP                          DMS     INDEX                 D
23 F         IP                          DMS     INDEX
24 F         IS                          DMS     INDEX                 A
25 F         IS                          DMS     INDEX                 B
26 F         IS                          DMS     INDEX                 D
27 F         IS                          DMS     INDEX
28 F         UP                          DMS     INDEX                 A
29 F         UP                          DMS     INDEX                 B
30 F         UP                          DMS     INDEX                 D
31 F         UP                          DMS     INDEX
32 F         US                          DMS     INDEX                 A
33 F         US                          DMS     INDEX                 B
34 F         US                          DMS     INDEX                 D
35 F         US                          DMS     INDEX
36 F*
37 F* DEMAND PROCESSING
38 F*
39 F         ID                          DMS                           A
40 F         ID                          DMS
41 F         UD                          DMS                           A
42 F         UD                          DMS
43 F*
44 F* OUTPUT ONLY PROCESSING
45 F*
46 F         O                           DMS                           A
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 15-2. Processing Methods for DMSII Files

Field Definitions

1-2 PAGE

Refer to section 2 for a complete description.

3-5 LINE

Refer to section 2 for a complete description.

6 FORM TYPE

This field must contain the letter F.

7-14 FILE NAME

The file name specified in the File Description Specifications must be the name of a disjoint data set which is defined in the data-base source file.

15 FILE TYPE

Valid entries for this field are:

Entry	Definition
I	The file specified is an input file.
U	The file specified is an update file.
O	The file specified is an output file.

16 FILE DESIGNATION

This field further describes the use of input and update files. Valid entries for this field are:

Entry	Definition
Blank	Output file.
P	Primary sequential file.
S	Secondary sequential file.
D	Demand file.

Chained, record address, and table files must not be specified as DMSII files.

17 END OF FILE

Refer to section 4 for a complete description.

18 SEQUENCE

This field is used only by primary and secondary files to indicate whether or not the program is to check the sequence of the input records. Columns 61-62 of the Input Specifications must specify the match fields within the input file records, and this file must have an automatic subset or ordered set specified in columns 47-52 of the File Description Specifications. Valid entries for this field are:

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

This column must be blank for unindexed primary and unindexed secondary DMSII files.

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

Console Message	Definition
<program job number>AXGO	Ignore the record out of sequence and read the next record from the same DMSII file.
<program job number>AXSTOP	Ignore the record out of sequence, turn on the LR indicator, and perform all end of job totals.
<program job number>DS	Discontinue the program.

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

19 FILE FORMAT

The only valid entries for this field for DMSII files are blank or the letter F.

20-39

These columns must be blank for DMSII files.

40-46 DEVICE

The only valid entry for this field is DMS.

Specifying DMSII in the DEVICE field indicates that this file is a DMSII file. For all files declared as DMSII files, the compiler obtains RPG/DMSII library files from disk. These library files are created by the DMS/DASDL compiler (refer to $ RPGLIB in the DMS/DASDL Information and Suggestions subsection) for the data base declared in the Data Base Specifications. The compiler uses these library files to determine which field names in the RPG source program are from the DMSII file.

47-52 INDEX NAME

Enter the name of the automatic subset, index random set, or index sequential set which is the index into the specified DMSII file. If this field is blank, then the specified data set is processed in its physical order rather than being processed by index order.

This entry must be blank for demand files, files specified as output, and primary or secondary DMSII files which have no index specified in data-base source file.

53-65

These columns must be left blank for DMSII files.

66 FILE ADDITIONS/UNORDERED/DELETION

The letter U in this column for a DMSII file is invalid. Valid entries for this field are:

Entry	Definition
A	Records can be added to an existing file.
D	Records can be deleted from an existing file.
B	Records can be added to and/or deleted from an existing file.
Blank	Records are not to be added or deleted.

If ADD is specified in columns 16-18 of the Output-Format Specifications for this file, then enter the letter A or B in this field.

NOTE

Adding records to an indexed cycle-driven DMSII file when the file is not the file which has just been read in the current RPG cycle can produce unexpected results. For instance, if a record is added to a secondary file following a read of a record from the primary file, the record work area of the secondary file contains the added record information. At the time the record information for the secondary file is made available, the information reflects the added record. This situation has implications affecting which record the program receives when the file is processed by means of matching records or control level breaks. To avoid this situation, add records at detail time and when the record identification indicator is set ON for this file.

If DEL is specified in columns 16-18 of the Output-Format Specifications for this file, enter the letter D or B in this field.

If the letter O is entered in column 15 of this File Description Specification, only the letter A is allowed in this field.

For primary or secondary DMSII files with no index specified, the only valid entry in this field is blank or the letter D.

The only valid entry for demand DMSII files is blank or the letter A.

Deletion of records from a demand DMSII file is accomplished with the DELET operation code.

67-70

67-70 These columns must be left blank for DMSII files.

71-72 FILE CONDITION

The FILE CONDITION field can be used on input, update, output, and on restart data management files. It indicates whether the file is conditioned by an external indicator. If a DMSII file is conditioned by a U-indicator, the file is used only when that indicator is on.

When the indicator is off, the file is treated on input as if the end-of-file record had been read. In the Calculation Specifications, if any DMSII operations are performed to the conditioned file, the operations should be conditioned with the same external indicator specified for the DMSII file. If the DMSII operations are not conditioned by the same U-indicator, the DMSII operations are ignored and the D1 resulting indicator, if used, is not affected. Likewise, any output performed to the DMSII file should be conditioned by the same external indicator used on the DMSII file.

If the same U-indicator is not used, the output record is built but the record is not written to the file. If column 39 (BLANK AFTER field) of the Output-Format Specifications contains the letter B for any output field, the output field is initialized to blank characters.

It is important to note that DMSII files are accessed by opening the data base. If all DMSII files in the program are conditioned by external indicators that are off, the data base is opened and any exceptions that normally occur at open time can occur. For instance, U-indicators that are off can require a recompile of the DMS/DASDL program to prevent a VERSIONERROR exception at data base open time.

Only disjoint data sets can be specified on File Description Specifications. Therefore, any embedded data sets associated with a disjoint data set conditioned by an external indicator are treated as having been conditioned by the same U-indicator. All Calculation Specifications and Output-Format Specifications affecting the embedded data set should be conditioned by the same external indicator as the parent data set, and are ignored if the U-indicator is off.

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

Refer to FILE CONDITION (columns 71-72) on non-DMSII files in section 4 for more information on external indicators.

73-74

These columns must remain blank for DMSII files.

75-80 PROGRAM IDENTIFICATION

Refer to section 2 for a complete description.

RPG/DMSII Library Files

Since the location of all DMSII files is maintained by DMS, the File Identification options $ PACKID, $ FAMILY, $ FILEID, and $ DISKID must not be specified for any DMSII files defined in the File Description Specifications.

Figure 15-2 shows the various processing methods for DMSII files.

DATA MANAGEMENT EXTENSION SPECIFICATIONS

Extension Specifications describe all tables and arrays used by the RPG program.

Field Definition

The following paragraphs describe the fields in the Extension Specifications.

1-2 PAGE

Refer to section 2 for a complete description.

3-5 LINE

Refer to section 2 for a complete description.

6 FORM TYPE

This field must contain the letter E.

11-26 FROM AND TO FILENAMES

A DMSII file must not be specified for these fields.

27-74

Refer to section 5 for a complete description.

75-80 PROGRAM IDENTIFICATION

Refer to section 2 for a complete description.

Rules for Declaring Vectors

The rules for declaring vectors on Extension Specifications are unchanged. An OCCURS item defined in DASDL must be specified in the Extension Specifications as a vector.

DATA MANAGEMENT INPUT SPECIFICATIONS

Input Specifications describe the records within each disjoint data set or embedded data set to be used as input data for the RPG program.

The entries defined in the Input Specifications for DMSII files are divided into the following two sections:

1. Field Definitions for Record Type Descriptions.
2. Field Definitions for Field Descriptions.

Field Definitions for Record Type Descriptions

The following paragraphs describe the field for record type descriptions:

1-2 PAGE

Refer to section 2 for a complete description.

3-5 LINE

Refer to section 2 for a complete description.

6 FORM TYPE

This field must contain the letter I.

7-14 FILE NAME

Every DMSII file which is described in the File Description Specifications as input or update must be described in the Input Specifications. An Input Specifications is required for an embedded data set which is used with a FIND or LOCK operation.

Embedded data sets can only be accessed on a demand basis (for example, with a FIND operation). All embedded data sets described in the Input Specifications must be embedded in a disjoint data set which is described in the File Description Specifications.

15-18 SEQUENCE

Sequencing can be specified for a DMSII file that is described in the File Description Specifications as primary or secondary and is accessed by means of an index. The File Description Specifications entry for that file, columns 47-52, must be non-blank.

Refer to section 8 for a complete description of sequencing.

19-20 RECORD IDENTIFYING INDICATOR

A DMSII file must not have look-ahead fields or spread records specified in this field.

The DMSII exception indicators D1, DA-DH, and DJ-DS must not be entered in this field.

Valid entries for this field are:

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

21-41 RECORD IDENTIFYING CODES

Entries in these columns can be made if columns 44-51 of the record line contain a DASDL-defined, non-vector field name.

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

An entry in these fields for a DMSII file specifies the position (in digits for numeric items, in bytes for alphanumeric items) within the item specified in columns 44-51 of the record line, not a position in a record.

The item positions are numbered from left to right, with the left-most digit or byte considered to be position one.

These fields must be blank if the letter S is specified in columns 26, 33, or 40 respectively.

26, 33, AND 40 C/Z/D/S

In addition to the character (C), zone (Z), and digit (D) record identification, these fields can contain the letter S. The letter S specifies signed-numeric record identification. If the letter S is specified, columns 21-24, 28-31, and 35-38 must be blank, columns 27, 34, and 41 must contain a positive sign (+) or negative sign (−), and the data-base source file must define the field specified in columns 44-51 as a signed-numeric field.

For DMSII fields, the character, zone, digit, and sign record identification specify a value within the item specified in columns 44-51 of the record line, not a position in the record.

27, 34 AND 41 CHARACTER

If an entry in columns 26, 33, or 40 is the letter S, then the only valid entries for columns 27, 34, and 41 are:

+ Declares that a positive sign be used for comparison with the sign position of the numeric field specified in columns 44-51.

− Declares that a negative sign be used for comparison with the sign position of the numeric field specified in columns 44-51.

42 STACKER SELECT

This field must be blank for DMSII files.

43 PACKED

This field must be blank for DMSII files.

44-51 FIELD LOCATION

When an entry is made in this field on a record description line, it specifies which item in the record is used for comparison in setting the record identifying codes.

When AND or OR lines occur for the record description line using record identification codes, columns 44-51 can contain any one of the following entries:

1. A blank, which defaults to the previous entry in columns 44-51.
2. The same item as previously defined.
3. A different DMS/DASDL-defined item.

When using record identifying codes, the entry in this field in the record description line must not contain a vector name. The only valid entry is a DMS/DASDL-defined item.

52-74

These columns must be blank for record description lines except when using the *ALL option.

52 DECIMAL POSITIONS

This column must remain blank for DMSII files.

53-58 FIELD NAME

The *ALL entry can be placed in these columns. When an entry is made in this field on a record description line, it causes the RPG compiler to generate Input Specifications for the items that have been previously defined in the data-base source file for the specified data set. This allows the user to do record identification of DMSII files.

If the *ALL option is specified on the Input Specifications, any DMS/DASDL-defined name not specified in this record description is made available to the RPG program. Because these DMS/DASDL-defined names are made available, a syntax error is generated at compile time if the programmer attempts to redefine a DMS/DASDL-defined name in the RPG program source file.

59-74

These columns must remain blank for DMSII files.

*ALL Examples for Input Specifications

Figure 15-2a shows an *ALL entry example, using a DMS/DASDL-defined data base. The data base has a data set A for which three items, X, Y, and Z, were declared as NUMBER (4,0) and two items, R and S, were declared as ALPHA (3). Q is an RPG name, not a DMS/DASDL-defined item. ARR is a subscripted array.

```
                    Input Specifications

01 IEMPLOY      01   1 CA                S
02 I                                     R   X
03 I                                     S   Y
04 I                                     Y   Q
05 I            02   1 CB                S   *ALL
06 I       OR   03                           
07 I                                     X   X           L1
08 I                                     X   Y
09 I                                     Y   ARR,Z
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 15-2a. *ALL Entry Example on the Input Specifications

Figure 15-2a shows how the user can influence control level indicators. A control level indicator L1 is assigned to the field X. The figure illustrates that the *ALL entry applies to the OR record description lines that follow the first record line. The Input Specifications in figure 15-2a are the same as the user coding for the following Input Specifications in figure 15-2b.

```
                  Input Specifications

01 IEMPLOY       01    1 CA                S
02 I                                       R   X
03 I                                       S   Y
04 I                                       Y   Q
05 I             02    1 CB                S
06 I        OR   03
07 I                                       R   R
08 I                                       S   S
09 I                                       Y   Y
10 I                                       Z   Z
11 I                                       X   X           L1
12 I                                       X   Y
13 I                                       Y   ARR,Z
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 15-2b. *ALL Entry Example on the Input Specifications

The field description lines coded for the 01 record type define local variables X and Y as ALPHA (3). The field description lines produced by the *ALL entry define local variables X and Y as NUMBER (4,0). This produces the following syntax error:

> RPG FIELD LENGTH AND/OR DEC. POS. DO NOT MATCH DMSII ITEM LENGTH AND/OR DEC. POS.

The Input Specifications can be corrected with the Input Specifications shown in figure 15-2c, where M and N are RPG names, not DMS/DASDL-defined items.

```
                  Input Specifications

01 IEMPLOY       01    1 CA                S
02 I                                       R   M
03 I                                       S   N
04 I                                       Y   Q
05 I             02    1 CB                S
06 I        OR   03
07 I                                       R   R
08 I                                       S   S
09 I                                       Y   Y
10 I                                       Z   Z
11 I                                       X   X           L1
12 I                                       X   Y
13 I                                       Y   ARR,Z
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 15-2c. *ALL Entry Example on Input Specifications

In the resulting compile source listing, the RPG compiler generates field description lines with a number sign (#) character in column 7 of the Input Specifications for the DMS/DASDL items implicitly specified by the *ALL entry. The RPG compiler does not generate field description lines in the following instances:

1. When the DMS/DASDL item name in columns 44-51 matches the RPG name in columns 53-58. A field description line is not generated for X because of this condition.
2. When the DMS/DASDL item name (columns 44-51) does not match the RPG name (columns 53-58), but the RPG name is already a DMS/DASDL item name for the data set. For this reason, the item Y does not have a field description line in the source listing.
3. When the RPG name in column 53 is a subscripted array and the subscript is a DMS/DASDL item name for the named data set. A field description line for Z is not required in this case.

Figure 15-2d shows the resulting source listing.

```
                    Input Specifications

01 IEMPLOY       01    1 CA                 S
02 I                                        R    X
03 I                                        S    Y
04 I                                        Y    Q
05 I             02    1 CB                 S    *ALL
06 I        OR   03
07 I                                        X    X            L1
08 I                                        X    Y
09 I                                        Y    ARR,Z
10 I#                                            R
11 I#                                            S
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 15-2d. Resulting Source Listing

When the *ALL entry is used on data sets with variable format parts, field description lines are not generated for the variable format items.

75-80 PROGRAM IDENTIFICATION

Refer to section 2 for a complete description.

Field Definitions for Field Descriptions

1-2 PAGE

Refer to section 2 for a complete description.

3-5 LINE

Refer to section 2 for a complete description.

6 FORM TYPE

This field must contain the letter I.

7-43

7-43 These fields must be blank for field descriptions.

44-51 FIELD LOCATION

This field specifies the location and data characteristics of the FIELD NAME entered in columns 53-58. A DMS/DASDL-defined item must be specified here which effectively "locates" and "types" the entry in the FIELD NAME columns.

If this field is left blank and the FIELD NAME entry contains a DMS/DASDL-defined name, the FIELD LOCATION defaults to the same entry.

An entry in this field on a field description line must not be a subscripted vector.

52 DECIMAL POSITIONS

This field must be blank.

53-58 FIELD NAME

If the FIELD LOCATION entry is blank, the FIELD NAME entry must be a DMS/DASDL-defined name.

Only those fields which are used in the program need to be specified on the Input Specifications.

Table 15-1 lists the allowable combinations of the FIELD LOCATION and FIELD NAME entries, and the action taken.

59-60 CONTROL LEVEL

These fields must be blank for non-indexed, cycle-driven DMSII files.

For all other primary and secondary files the L1-L9 control level indicators are valid.

61-62 MATCHING FIELDS

These fields must be blank for non-indexed, cycle-driven DMSII files.

For all other primary and secondary files the M1-M9 matching field indicators are valid.

63-64 FIELD RECORD RELATIONS

The DMSII exception indicators, D1, DA-DH, and DJ-DS, must not be entered in this field.

For all other files, the following are valid entries:

Entry	Definition
01-99	Record identifying indicator assigned to a record type.
L1-L9	Control level indicator defined elsewhere.
MR	Matching record indicator.
U1-U8	External indicator defined elsewhere.
H0-H9	Halt indicator defined elsewhere.

65-70 FIELD INDICATORS

The DMSII exception indicators, D1, DA-DH, and DJ-DS must not be entered in this field.

71-74

These columns must be blank.

75-80 PROGRAM IDENTIFICATION

Refer to section 2 for a complete description.

Figure 15-3 illustrates two coding examples for DMSII Input Specifications.

```
                    Input Specifications

01 I*
02 I* EXAMPLE USING RECORD IDENTIFICATION CODES FOR A DMSII
03 I* FILE NAMED DATSET.
04 I*
05 IDATSET  NS  01   1 CB                DMSFLD
06 I                                     DMSFLD   DMSFLD
07 I*
08 I* EXAMPLE USING NO RECORD IDENTIFICATION CODE FOR A
09 I* DMSII FILE NAMED MASTER.
10 I*
11 IMASTER  NS  02
12 I                                     DMSFLD   FIELD1
13 I                                              DMSIT1
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

NOTE

For the file name DATSET, only those records which have the first position of the DMS/DASDL-defined variable DMSFLD equal to the letter A are processed.

Lines 6 and 12 load the RPG-defined DMSFLD and FIELD1 fields from the DMS/DASDL-defined DMSFLD field.

Line 13 loads the RPG-defined DMSIT1 field from the DMS/DASDL-defined DMSIT field. An entry in the FIELD LOCATION field is not required.

Figure 15-3. Input Specifications for DMSII Files

Table 15-1. Combinations of Various DMSII FIELD LOCATION and FIELD NAME Fields in the Input Specifications

FIELD LOCATION Columns 44-51	FIELD NAME Columns 53-58	Program Action
Vector	Array	Load the whole array with the contents of the location specified by the whole DMS/DASDL-defined vector.
Vector	Array,J	Load Array,J with the contents of the location specified by the DMS/DASDL-defined Vector,J.
Vector	Table,J	Load Table,J with the contents of the location specified by the DMS/DASDL-defined Vector,J.
Vector	Array,3	Load Array,3 with the contents of the location specified by the DMS/DASDL-defined Vector,3.
Vector	Table,3	Load Table,3 with the contents of the location specified by the DMS/DASDL-defined Vector,3.
Vector	Table	Not allowed.
Field Name	Array	Not allowed.
Field Name	Array,J	Load Array,J with the contents of the location specified by the DMS/DASDL-defined Field Name.
Field Name	Array,3	Load Array,3 with the contents of the location specified by the DMS/DASDL-defined Field Name.
Field Name	Table,J	Load Table,J with the contents of the location specified by the DMS/DASDL-defined Field Name.
Field Name	Table,3	Load Table,3 with the contents of the location specified by the DMS/DASDL-defined Field Name.
Field Name	Field Name	Load, to the Field Name specified in columns 53-58, the contents from the location of the DMS/DASDL-defined Field Name in columns 44-49.

Table 15-1. Combinations of Various DMSII FIELD LOCATION and FIELD NAME Fields in the Input Specifications (Cont)

FIELD LOCATION Columns 44-51	FIELD NAME Columns 53-58	Program Action
Field Name	Table	Not allowed.
Blank	Array	Load the whole Array with the contents from the location of the DMS/DASDL-defined Array.
Blank	Array,J	Load to Array,J with the contents from the location of the DMS/DASDL-defined Vector,J.
Blank	Array,3	Load to Array,3 with the contents from the location of the DMS/DASDL-defined Vector,3.
Blank	Table	Not allowed.
Blank	Table,J	Load to Table,J with the contents from the location of the DMS/DASDL-defined Vector,J.
Blank	Table,3	Load to Table,3 with the contents from the location of the DMS/DASDL-defined Vector,3.
Blank	Field Name	Load to Field Name with the contents from the location of the DMS/DASDL-defined Field Name.

DATA MANAGEMENT CALCULATION SPECIFICATIONS

The entries defined on the Calculation Specifications for DMSII files are described in the following paragraphs.

Field Definitions

The following paragraphs describe the fields in the Calculation Specifications.

1-2 PAGE

Refer to section 2 for a complete description.

3-5 LINE

Refer to section 2 for a complete description.

6 FORM TYPE

This field must contain the letter C.

7-8 CONTROL LEVEL

AND/OR lines (AN,OR). When several DMKEY lines must be specified to establish the required relationship of the DMKEY operations for the immediately preceding FIND or LOCK using random access, an AN or OR in columns 7-8 must be used on all DMKEY operations after the first DMKEY of a group. Columns 7-8 must be blank for the first DMKEY operation, except for the optional entry SR when the DMKEY operation is in a subroutine. See figure 15-5 for two examples of coding the DMKEY operation.

9-17 INDICATORS

Enter the conditioning indicators in these fields.

In addition to those indicators specified in section 9, the following DMSII exception indicators can also be used:

Entry	Definition
D1	ON EXCEPTION indicator
DA	NOTFOUND indicator
DB	DUPLICATES indicator
DC	DEADLOCK indicator
DD	DATAERROR indicator
DE	NOTLOCKED indicator
DF	KEYCHANGED indicator
DG	SYSTEMERROR indicator
DH	READONLY indicator
DJ	IOERROR indicator
DK	LIMITERROR indicator
DL	OPENERROR indicator
DM	CLOSEERROR indicator
DN	NORECORD indicator
DO	INUSE indicator
DP	AUDITERROR indicator
DQ	ABORT indicator
DR	SECURITYERROR indicator
DS	VERSIONERROR indicator
DT	FATALERROR indicator
DU	INTEGRITYERROR indicator

Refer to the B 1000 Systems Data Management System II (DMSII) Reference Manual for a complete description of the DMSII exception conditions.

All conditioning indicators on a FIND or LOCK operation with random access are applied to the DMKEY operations immediately following the FIND or LOCK operation. Columns 9-17 must be blank for the DMKEY operation(s) which immediately follow the FIND or LOCK statements with random access.

The D1 indicator is set ON for all DMSII exception conditions. In addition, one of the indicators DA-DH, DJ-DU is set ON to indicate the type of DMSII exceptions condition.

18-27 FACTOR 1

When the operation code specified in columns 28-32 is a FIND, LOCK, INSRT, or REMOV operation, the allowable entries for FACTOR 1 are:

1. An index (an index random set, automatic subset, index sequential set, or manual subset which is defined by DASDL) associated with the data set name specified in columns 33-42, except the REMOV operation.
2. For the FIND or LOCK operations, FACTOR 1 can be left blank when the access desired is not by means of an index.

For all other DMSII operations, FACTOR 1 must be blank.

28-32 OPERATION FIELD

Enter the desired operation code in this field. Refer to section 9 for a description of the available non-DMSII operation codes, and refer to the subsection titled Data Management Operation Codes in this section for a description of the DMSII operation codes.

The non-DMSII operation codes which are described in section 9 and cannot be specified for DMSII files are as follows:

CHAIN
DEBUG
DSPLY
READ
RECV
SEND
SETLL

33-42 FACTOR 2

When a DMSII operation code other than DMKEY or REMOV is specified in the OPERATION FIELD field (columns 28-32), the only allowable entry in the FACTOR 2 field is a data set name. This data set name must be one of the following:

1. A disjoint data set previously defined on the File Description Specifications.
2. An embedded data set previously defined in the data-base source file and embedded in a disjoint data set previously defined in the File Description Specifications.

When the DMKEY operation code is specified in the OPERATION FIELD field, the allowable entries in the FACTOR 2 field are a literal or a variable name.

If the operation code is REMOV, the FACTOR 2 field must be blank.

43-48 RESULT FIELD

The RESULT FIELD field specifies the selection expression for FIND and LOCK operation codes, and specifies the DMS/DASDL-defined key name for the DMKEY operation code.

Valid entries for this field are:

Entry	Definition
Blank	Get the current record.
FIRST	Get the first record of the data set. Used with the FIND or LOCK operation codes.
LAST	Get the last record of the data set. Used with the FIND or LOCK operation codes.
NEXT	Get the next record of the data set. Used with the FIND or LOCK operation codes. For index-sequential sets, if there is no current record, get the first record of the data set.
PRIOR	Get the previous record of the data set. Used with the FIND or LOCK operation codes.
Key Name	Key name as defined in data-base source file. Used with the DMKEY operation code.

For all other DMSII operations, the RESULT FIELD field must be blank.

Refer to the B 1000 Systems Data Management System II (DMSII) Reference Manual for a functional description of selection expression.

49-51 FIELD LENGTH

This field must be blank when a DMSII operation is specified in the OPERATION FIELD field.

52 DECIMAL POSITIONS

This field must be blank when a DMSII operation is specified in the OPERATION FIELD field.

53 HALF ADJUST/ACCESS METHOD

This field is used to specify the access method (random or sequential) for the FIND and LOCK operation code.

When FIND or LOCK is specified in the OPERATION FIELD field, the following are the only valid entries:

Entry	Definition
R	Random Access
S	Sequential Access

When the DELET operation code is specified, this field must contain the letter S for sequential deletion.

54-59 RESULTING INDICATORS

When a DMSII operation other than DMKEY is specified in the OPERATION FIELD field, the only valid entries for these columns are the entries D1 or blank. The D1 indicator is the general DMSII exception indicator.

Columns 56-59 must be blank for all DMSII operations other than DMKEY operation code.

If the DMKEY operation code is specified in the OPERATION FIELD field, then columns 54-56 must contain the DMKEY relation. Columns 57-59 must be blank.

If a DMSII exception occurs, then:

1. If the D1 indicator is specified in columns 54-55, D1 is set ON and one of the appropriate exception indicators DA-DH or DJ-DU is set ON.
2. If blanks are specified in columns 54-55, the program branches to the exception handling routine for one of the following:

   1) The data set specifed in the FACTOR 2 field for the FIND, LOCK, FREE, TRBEG, TREND, STORE, and DELET operation codes.
   2) The manual subset specified in the FACTOR 1 field for the INSRT and REMOV operation codes.

3. The program terminates if blanks are specified in columns 54-55 and no exception handling routine exists for either of the following:

   1) The data set specified in the FACTOR 2 field for the FIND, LOCK, FREE, TRBEG, TREND, STORE, and DELET operation codes.
   2) The manual subset specified in the FACTOR 1 field for the INSRT and REMOV operation codes.

60-74

These columns must be blank.

75-80 PROGRAM IDENTIFICATION

Refer to section 2 for a complete description.

Setting Indicators

The D1, DA-DH, and DJ-DU exception condition indicators are set OFF immediately prior to a DMSII operation and are set ON as a result of a DMSII exception condition. The D1 indicator can be used as a resulting indicator of a DMSII operation. The DA-DH and DJ-DU indicators must not be used as resulting indicators for a DMSII operation.

DATA MANAGEMENT OPERATION CODES

The RPG language provides explicit operation codes to interact with DMSII. DMSII structures are manipulated by these operations.

A valid data set name is defined as one of the following:

1. A disjoint data set which has been defined in the File Description Specifications as an input or update demand file.
2. An embedded data set which is embedded in a disjoint data set defined in the File Description Specifications.

Programmed Control of Data Management Input and Output

Within the normal B 1000 RPG cycle, a record is read, calculations are performed (using the data from the input record), and an output record is written. The DELET, FIND, FREE, INSRT, LOCK, REMOV, and STORE operations allow greater control over DMSII input and output providing the capability to read and write records other than those normally available as part of the RPG program cycle.

DELET Operation Code

The DELET operation code provides a method of deleting a record from the specified data set. Figure 15-4 shows one method of coding the DELET operation.

```
                Calculation Specifications

01 C                        DELETDATSET                 D1
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

NOTE
Example of the DELET operation code. DATSET data set is the name of a data set declared in the data-base source file.

Figure 15-4. One Method of Coding the DELET Operation

The DELET operation requires entries in the following fields:

1. The DELET operation code in columns 28-32.
2. A valid data set name in the FACTOR 2 field.
3. The letter S in column 53.
4. The D1 indicator or blanks in columns 54-55.

The DELET operation code prohibits entries in the following fields:

FACTOR 1
RESULT FIELD
RESULT FIELD LENGTH
DECIMAL POSITION FIELD
LOW and EQUAL RESULTING INDICATORS

Refer to the DELETE operation code in the B 1000 Systems Data Management System II (DMSII) Reference Manual for a functional description of the DELET operation code.

DMKEY Operation Code

The DMKEY operation code provides a method of specifying the conditions which must be satisfied by the record accessed in the immediately preceding FIND or LOCK operation code using random access. Figure 15-5 shows two methods of coding the DMKEY operation code.

```
                Calculation Specifications

01 C*
02 C* EXAMPLE 1:
03 C*
04 C   01       SET1       FIND MASTER     NEXT       RD1
05 C                       DMKEY10         KEY1        GTR
06 CAN                     DMKEY20         KEY1        LSS
07 CAN                     DMKEY5          KEY2        NEQ
08 C*
09 C* EXAMPLE 2:
10 C*
11 C  NO1 02   SUB1        LOCK MASTER     NEXT       RD1
12 C                       DMKEYKEY5       KEYA        EQL
13 CAN                     DMKEY10         KEYB        EQL
14 COR                     DMKEYFIELD1     KEYA        GTR
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

NOTE

In example 1, if the 01 indicator is ON, then a find operation on the MASTER data set by way of the SET1 (index path) set is performed. This operation searches for the next record where the KEY1 field is greater than 10 and less than 20 and the KEY2 field is not equal to 5.

In example 2, if the 01 indicator is OFF and the 02 indicator is ON, then a lock operation is performed on the MASTER data set by way of the SUB1 (index patch) set. This operation searches for the next record where the KEYA field is equal to 5 and the KEYB is equal to 10, or the next record where the KEYA field is greater than the FIELD1 field.

Figure 15-5. Two Coding Examples of the DMKEY Operation

The DMKEY operation code requires entries in the following fields:

1. The DMKEY operation code in columns 28-32.
2. A literal or variable name in the FACTOR 2 field. The literal or variable provides a value for the key condition.
3. The RESULT FIELD field contains a key item name. This item name, defined in the database source file, must be a key item of the set or subset name specified in the FACTOR 1 field of the immediately preceding FIND or LOCK operation code.
4. The key relationship desired in columns 54-56. Valid entries for this field are:

Entry	Definition
GTR	The item in the RESULT FIELD field is greater than the value of the field or literal specified in the FACTOR 2 field.
LSS	The item in the RESULT FIELD field is less than the value of the field or literal specified in the FACTOR 2 field.
EQL	The item in the RESULT FIELD field is equal to the value of the field or literal specified in the FACTOR 2 field.
GEQ	The item in the RESULT FIELD field is greater than or equal to the value of the field or literal specified in the FACTOR 2 field.
LEQ	The item in the RESULT FIELD field is less than or equal to the value of the field or literal specified in the FACTOR 2 field.
NEQ	The item in the RESULT FIELD field is not equal to the value of the field or literal specified in the FACTOR 2 field.

The DMKEY operation code prohibits entries in the following fields:

Any conditioning indicator in columns 9-17
FACTOR 1
RESULT FIELD LENGTH
DECIMAL POSITIONS FIELD
HALF ADJUST/ACCESS MODE FIELD
Columns 57-59

The DMKEY operation code describes the key conditions for the immediately preceding FIND or LOCK operation code. Random access must be specified for the FIND or LOCK operation code by entering the letter R in column 53 of the Calculation Specifications.

There must be at least one key of a set or subset defined for the DMKEY operation code with a FIND or LOCK operation code when random access is specified. If only some of the keys of a set or subset are defined, then any key relationship for the undefined key(s) is considered acceptable.

FIND Operation Code

The FIND operation code provides a method of locating a record in a data set and transferring the record to the record area as defined in the Input Specifications. Figure 15-6 shows methods of coding the FIND operation code.

```
          Calculation Specifications

01 C*
02 C* "ISSET" IS THE INDEX NAME OF A INDEX-SEQUENTIAL SET DEFINED
03 C* IN THE DATA-BASE SOURCE FILE, "DATSET" IS THE NAME OF A DATA
04 C* SET DEFINED IN THE DATA-BASE SOURCE FILE, AND "KEYS" IS THE
05 C* KEY NAME OF THE "ISSET" INDEX-SEQUENTIAL SET.
06 C*
07 C            ISSET     FIND DATSET               RD1
08 C                      DMKEY10         KEYS       EQL
09 C            ISSET     FIND DATSET     NEXT      RD1
10 C                      DMKEY20         KEYS       EQL
11 C            ISSET     FIND DATSET               SD1
12 C*
13 C* "IRSET" IS THE INDEX NAME OF AN INDEX-RANDOM SET DEFINED IN
14 C* THE DATA-BASE SOURCE FILE, AND "KEYR" IS THE NAME OF THE
15 C* "IRSET" INDEX-RANDOM SET.
16 C*
17 C            IRSET     FIND DATSET               RD1
18 C                      DMKEY10         KEYR       EQL
19 C            IRSET     FIND DATSET     NEXT      RD1
20 C                      DMKEY20         KEYR       EQL
21 C            IRSET     FIND DATSET               SD1
22 C            IRSET     FIND DATSET     FIRST     SD1
23 C            IRSET     FIND DATSET     LAST      SD1
24 C            IRSET     FIND DATSET     NEXT      SD1
25 C            IRSET     FIND DATSET     PRIOR     SD1
26 C*
27 C* "DATSET" IS THE NAME OF A DATA SET DEFINED IN THE DATA-BASE
28 C* SOURCE FILE.
29 C*
30 C                      FIND DATSET               SD1
31 C                      FIND DATSET     FIRST     SD1
32 C                      FIND DATSET     LAST      SD1
33 C                      FIND DATSET     NEXT      SD1
34 C                      FIND DATSET     PRIOR     SD1
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 15-6. Methods of Coding the FIND Operation Code (Sheet 1 of 2)

```
35 C*
36 C* "MANSET" IS THE NAME OF A MANUAL SUBSET DEFINED IN THE
37 C* DATA-BASE SOURCE FILE, "DATSET" IS THE NAME OF A DATA SET
38 C* DEFINED IN THE DATA-BASE SOURCE FILE, AND "KEYM" IS THE
39 C* NAME OF THE KEY TO THE "MANSET" MANUAL SUBSET.
40 C*
41 C            MANSET    FIND DATSET                RD1
42 C                      DMKEY10         KEYM        EQL
43 C            MANSET    FIND DATSET     NEXT       RD1
44 C                      DMKEY20         KEYM        EQL
45 C            MANSET    FIND DATSET                SD1
46 C            MANSET    FIND DATSET     FIRST      SD1
47 C            MANSET    FIND DATSET     LAST       SD1
48 C            MANSET    FIND DATSET     NEXT       SD1
49 C            MANSET    FIND DATSET     PRIOR      SD1
50 C*
51 C* "AUTSET" IS THE NAME OF A MANUAL SUBSET DEFINED IN THE
52 C* DATA-BASE SOURCE FILE, "DATSET" IS THE NAME OF A DATA SET
53 C* DEFINED IN THE DATA-BASE SOURCE FILE, AND "KEYA" IS THE
54 C* NAME OF THE KEY TO THE "AUTSET" MANUAL SUBSET.
55 C*
56 C            AUTSET    FIND DATSET                RD1
57 C                      DMKEY10         KEYA        EQL
58 C            AUTSET    FIND DATSET     NEXT       RD1
59 C                      DMKEY20         KEYA        EQL
60 C            AUTSET    FIND DATSET                SD1
61 C            AUTSET    FIND DATSET     FIRST      SD1
62 C            AUTSET    FIND DATSET     LAST       SD1
63 C            AUTSET    FIND DATSET     NEXT       SD1
64 C            AUTSET    FIND DATSET     PRIOR      SD1
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 15-6. Methods of Coding the FIND Operation Code (Sheet 2 of 2)

The FIND operation requires entries in the following fields:

1. FIND operation code in columns 28-32.
2. A valid data set name in the FACTOR 2 field.
3. The letter R or S in column 53. The letter R specifies random access of the data set. The letter S specifies sequential access of the data set.
4. The D1 indicator or blanks in columns 54-55.

The FIND operation code can also use the following entries:

1. The contents of the FACTOR 1 field can specify the name of an index random set, index sequential set, automatic subset, or manual subset. This index name must have been previously defined in the data-base source file.
2. The RESULT FIELD field can contain a selection expression. This selection expression specifies the record of a data set which is to be processed, relative to the last record accessed.

A FIND operation code with random access (the letter R in column 53 of the Calculation Specifications) must be immediately followed by appropriate DMKEY operation codes to define the key conditions which must be satisfied.

The FIND operation code prohibits entries in the following fields:

RESULT FIELD LENGTH
DECIMAL POSITIONS FIELD
LOW and EQUAL RESULTING INDICATORS

Refer to the FIND operation in the B 1000 Systems Data Management System II (DMSII) Reference Manual for a functional description of the FIND operation code.

FREE Operation Code

The FREE operation code unlocks the current locked record for this program and this data set. Figure 15-7 shows one method of coding the FREE operation code.

```
            Calculation Specifications

01 C                      FREE DATSET                 D1
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

NOTE
This is an example of the FREE operation code. DATSET is the name of a data set declared in the data-base source file.

Figure 15-7. One Method of Coding the FREE Operation

The FREE operation code requires entries in the following fields:

1. FREE entered in the OPERATION CODE field.
2. A valid data set name in the FACTOR 2 field.
3. The D1 exception indicator or blanks in columns 54-55.

The FREE operation code prohibits entries in the following fields:

FACTOR 1
RESULT FIELD
RESULT FIELD LENGTH
DECIMAL POSITIONS FIELD
HALF ADJUST/ACCESS MODE FIELD
LOW and EQUAL RESULTING INDICATORS

Refer to the FREE operation code in the B 1000 Systems Data Management System II (DMSII) Reference Manual for a functional description of the FREE operation code.

INSRT Operation Code

The INSRT operation code specifies the insertion of a record from a data set into a manual subset. Figure 15-8 shows one method of coding the INSRT operation code.

```
                Calculation Specifications

01 C              MANSET     INSRTDATSET                 D1
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

NOTE
This is an example of the INSRT operation code. DATSET is the name of a data set declared in the data-base source file and MANSET is the name of the manual subset declared in the data-base source file.

Figure 15-8. One Method of Coding the INSRT Operation Code

The INSRT operation code requires entries in the following fields:

1. The manual subset name in the FACTOR 1 field. This manual subset must have been previously defined in the data-base source file, and must be one of the following:

   1) An item of a disjoint data set defined on the File Description Specifications.
   2) An item of an embedded data set which is embedded in a disjoint data set defined on the File Description Specifications.

2. INSRT entered in the OPERATION CODE field.
3. A data set name in the FACTOR 2 field. This data set name must be a disjoint data set which has been defined on the File Description Specifications.
4. The D1 exception indicator or blanks in columns 54-55.

The INSRT operation code prohibits entries in the following fields:

RESULT FIELD
RESULT FIELD LENGTH
DECIMAL POSITIONS FIELD
HALF ADJUST/ACCESS MODE FIELD
LOW and EQUAL RESULTING INDICATORS

The data set name must be the declared source of records for a manual subset. For example, the manual subset name S1 must be a manual subset of the data set D, as the following example illustrates.

Example:

```
DMS/DASDL:    S1 SUBSET OF D
RPG:          S1          INSRT          D
          (FACTOR 1)  (OPERATION)   (FACTOR 2)
```

Refer to the INSERT operation in the B 1000 Systems Data Management System II (DMSII) Reference Manual for a functional description of the INSRT operation code.

LOCK Operation Code

The LOCK operation code locates a record in the specified data set, transfers the data to the record area as described in the Input Specifications, and locks the record to prevent concurrent modifications by another user. If the record is not found, a NOTFOUND exception condition results. Figure 15-9 shows methods of coding the LOCK operation code.

```
          Calculation Specifications

01 C*
02 C* "ISSET" IS THE INDEX NAME OF A INDEX-SEQUENTIAL SET DEFINED
03 C* IN THE DATA-BASE SOURCE FILE, "DATSET" IS THE NAME OF A DATA
04 C* SET DEFINED IN THE DATA-BASE SOURCE FILE, AND "KEYS" IS THE
05 C* KEY NAME OF THE "ISSET" INDEX-SEQUENTIAL SET.
06 C*
07 C            ISSET     LOCK DATSET               RD1
08 C                      DMKEY10         KEYS       EQL
09 C            ISSET     LOCK DATSET     NEXT      RD1
10 C                      DMKEY20         KEYS       EQL
11 C            ISSET     LOCK DATSET               SD1
12 C*
13 C* "IRSET" IS THE INDEX NAME OF AN INDEX-RANDOM SET DEFINED IN
14 C* THE DATA-BASE SOURCE FILE, AND "KEYR" IS THE NAME OF THE
15 C* "IRSET" INDEX-RANDOM SET.
16 C*
17 C            IRSET     LOCK DATSET               RD1
18 C                      DMKEY10         KEYR       EQL
19 C            IRSET     LOCK DATSET     NEXT      RD1
20 C                      DMKEY20         KEYR       EQL
21 C            IRSET     LOCK DATSET               SD1
22 C            IRSET     LOCK DATSET     FIRST     SD1
23 C            IRSET     LOCK DATSET     LAST      SD1
24 C            IRSET     LOCK DATSET     NEXT      SD1
25 C            IRSET     LOCK DATSET     PRIOR     SD1
26 C*
27 C* "DATSET" IS THE NAME OF A DATA SET DEFINED IN THE DATA-BASE
28 C* SOURCE FILE.
29 C*
30 C                      LOCK DATSET               SD1
31 C                      LOCK DATSET     FIRST     SD1
32 C                      LOCK DATSET     LAST      SD1
33 C                      LOCK DATSET     NEXT      SD1
34 C                      LOCK DATSET     PRIOR     SD1
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 15-9. Methods of Coding the LOCK Operation Code (Sheet 1 of 2)

```
35 C*
36 C* "MANSET" IS THE NAME OF A MANUAL SUBSET DEFINED IN THE
37 C* DATA-BASE SOURCE FILE, "DATSET" IS THE NAME OF A DATA SET
38 C* DEFINED IN THE DATA-BASE SOURCE FILE, AND "KEYM" IS THE
39 C* NAME OF THE KEY TO THE "MANSET" MANUAL SUBSET.
40 C*
41 C           MANSET    FIND DATSET                RD1
42 C                     DMKEY10         KEYM        EQL
43 C           MANSET    FIND DATSET     NEXT       RD1
44 C                     DMKEY20         KEYM        EQL
45 C           MANSET    FIND DATSET                SD1
46 C           MANSET    FIND DATSET     FIRST      SD1
47 C           MANSET    FIND DATSET     LAST       SD1
48 C           MANSET    FIND DATSET     NEXT       SD1
49 C           MANSET    FIND DATSET     PRIOR      SD1
50 C*
51 C* "AUTSET" IS THE NAME OF A MANUAL SUBSET DEFINED IN THE
52 C* DATA-BASE SOURCE FILE, "DATSET" IS THE NAME OF A DATA SET
53 C* DEFINED IN THE DATA-BASE SOURCE FILE, AND "KEYA" IS THE
54 C* NAME OF THE KEY TO THE "AUTSET" MANUAL SUBSET.
55 C*
56 C           AUTSET    FIND DATSET                RD1
57 C                     DMKEY10         KEYA        EQL
58 C           AUTSET    FIND DATSET     NEXT       RD1
59 C                     DMKEY20         KEYA        EQL
60 C           AUTSET    FIND DATSET                SD1
61 C           AUTSET    FIND DATSET     FIRST      SD1
62 C           AUTSET    FIND DATSET     LAST       SD1
63 C           AUTSET    FIND DATSET     NEXT       SD1
64 C           AUTSET    FIND DATSET     PRIOR      SD1
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 15-9. Methods of Coding the LOCK Operation Code (Sheet 2 of 2)

The LOCK operation requires entries in the following fields:

1. The LOCK word entered in the OPERATION CODE field.
2. A valid data set name in the FACTOR 2 field.
3. The letter R or S in column 53. If the letter R is specified, the data set is processed randomly. If the letter S is specified, the data set is processed sequentially.
4. The D1 exception indicator or blanks in columns 54-55.

In addition to the preceding required entries, the LOCK operation code can use the following optional entries:

1. The FACTOR 1 field to specify the name of an index random set, index sequential set, automatic subset, or manual subset. The index name must have been previously defined in DASDL.
2. The RESULT FIELD field can contain a selection expression. This selection expression specifies the record of the data set which is to be processed, relative to the last record accessed.

A LOCK operation code with random access (the letter R in column 53 of the Calculation Specifications) must be immediately followed by appropriate DMKEY operation codes to define the key conditions which must be satisfied.

The LOCK operation prohibits entries in the following fields:

RESULT FIELD LENGTH
DECIMAL POSITIONS
LOW and EQUAL RESULTING INDICATORS

Refer to the MODIFY operation in the B 1000 Systems Data Management System II (DMSII) Reference Manual for a functional description of the LOCK operation code.

REMOV Operation Code

The REMOV operation code removes the current record from a manual subset. The record is not removed from the data set. Figure 15-10 shows one method of coding the REMOV operation code.

```
                Calculation Specifications

01 C             MANSET    REMOV                        D1
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

NOTE
This is an example of the REMOV operation code. MANSET is the name of a manual set declared in the data-base source file.

Figure 15-10. One Method of Coding the REMOV Operation Code

The REMOV operation code requires entries in the following fields:

1. A manual subset name in the FACTOR 1 field . The manual subset must be one of the following:

   1) An item of a disjoint data set defined on the File Description Specifications.
   2) An item of an embedded data set which is embedded in a disjoint data set defined on the File Description Specifications.

2. REMOV entered in the OPERATION CODE field.
3. The D1 indicator or blanks in columns 54-55.

The REMOV operation code prohibits entries in the following fields:

FACTOR 2
RESULT FIELD
RESULT FIELD LENGTH
DECIMAL POSITIONS
HALF ADJUST/ACCESS MODE FIELD
LOW and EQUAL RESULTING INDICATORS

Refer to the REMOVE operation in the B 1000 Systems Data Management System II (DMSII) Reference Manual for a functional description of the REMOV operation code.

STORE Operation Code

The STORE operation code provides a method of updating a record or adding a new record to a data set. Figure 15-11 shows one method of coding the STORE operation code.

```
                Calculation Specifications

01 C                        STOREDATSET                 D1
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

NOTE
This is an example of the STORE operation code. DATSET data set is the name of a data set declared in the data-base source file.

Figure 15-11. One Method of Coding the STORE Operation Code

The STORE operation code requires entries in the following fields:

1. STORE entered in the OPERATION CODE field.
2. A valid data set name in the FACTOR 2 field.
3. The D1 exception indicator or blanks in columns 54-55.

The STORE operation code prohibits entries in the following fields:

FACTOR 1
RESULT FIELD
RESULT FIELD LENGTH
DECIMAL POSITIONS
HALF ADJUST/ACCESS MODE FIELD
LOW and EQUAL RESULTING INDICATORS

The STORE operation code uses Output-Format Specifications which contains the following:

1. The data set name in FILENAME field (columns 7-14).
2. The letter S in TYPE field (column 15).

Refer to the STORE operation code in the B 1000 Systems Data Management System II (DMSII) Reference Manual for a functional description of the STORE operation code.

DATA MANAGEMENT EXCEPTION HANDLING

An RPG program which uses DMSII operations can encounter any one of many DMSII exception conditions which prevent the operation from being performed as specified. If an exception condition occurs, the RPG program terminates unless one of the following conditions exists:

1. Columns 54-55 of the Calculation Specifications for a DMSII operation other than the DMKEY operation code contain the D1 indicator for the DMSII operation which caused the exception condition.
2. An exception handling routine exists in the Calculation Specifications for the cycle-driven file, demand file, or manual subset for which the exception condition occurred.

Input Exceptions for Cycle-Driven Data Management Files

If a cycle-driven DMSII file is used and one of the following exception conditions occurs, the operator must respond appropriately. The exception conditions listed below result in the following run-time diagnostics for cycle-driven DMSII files:

1. DEADLOCK exception gives the operator the option to enter:

   1) <program job number>AXRETRY. Attempt the same operation again.
   2) <program job number>AXSTOP. Program performs an orderly termination.

2. IOERROR exception gives the operator the option to enter:

   1) <program job number>AXRETRY. Attempt the same operation again.
   2) <program job number>AXSTOP. Program performs an orderly termination.

3. SECURITYERROR exception only allows the operator to enter:

   <program job number>AXSTOP

   Once the STOP message is entered, the program goes to an orderly termination.

4. VERSIONERROR exception only allows the operator to enter:

   <program job number>AXSTOP

   Once the STOP message is entered, the program goes to an orderly termination.

A NOTFOUND exception condition is treated as an end of file for cycle-driven DMSII files.

All Other Exception Conditions

An exception handling routine can be coded for one of the following:

1. Those exceptions which occur as a result of an explicit data management operation other than DMKEY which does not have the indicator D1 entered in columns 54-55 for that operation.
2. Those exceptions which can occur for primary or secondary DMSII files exclusive of the exceptions previously listed under Input Exceptions for Cycle-Driven Data Management Files.

The exception handling routine operation codes are used only to delimit the beginnning and end of an exception handling routine. Calculation Specifications lines within an exception handling routine must contain the entries UR, OR, AN, or blank in columns 7-8, and all exception handling routines must be specified immediately before all Output-Format Specifications. Exception handling routines must not be nested in the RPG program.

Exception Handling Operation Codes

The exception handling operation codes are described in the following paragraphs.

BEGUR Operation Code

The BEGUR operation code indicates the start of an exception handling routine. The FACTOR 2 field must contain the name of the disjoint data set, embedded data set, or manual subset for which the exception handling routine is applicable. For the BEGUR operation code, columns 9-27 and 43-59 must be left blank.

ENDUR Operation Code

The ENDUR operation code indicates the end of an exception handling routine. The FACTOR 1 field can contain a label. This label is used as a tag, thus allowing exits from different points within the exception handling routine. Columns 9-27 and 33-59 must be blank for this exception handling operation code.

The following rules must be followed when coding an exception handling routine:

1. Exactly one exception handling routine can be defined for:

   1) Each disjoint data set defined in the File Description Specifications.
   2) Each embedded data set which is embedded within a disjoint data set defined in the File Description Specifications.
   3) Each manual subset which is an item of a disjoint data set or an item of an embedded data set.

2. The following operation codes must not appear within an exception handling routine:

   CHAIN
   DELET
   EXCPT
   EXSR
   FIND
   FORCE
   FREE
   INSRT
   LOCK
   READ
   RECV
   REMOV
   SEND
   STORE
   TRBEG
   TREND

3. Branching into or out of a given exception handling routine is not allowed (for example, a branch such as a GOTO or EXSR operation).

Additionally, the following must be noted:

1. The program aborts if an exception condition occurs and no exception handling routine is defined for at least one of the following:

   1) A cycle-driven DMSII file.
   2) A demand DMSII file which does not have the D1 indicator coded in columns 54-55 of the Calculation Specifications for a DMSII operation code other than DMKEY.
   3) A manual subset which does not have the D1 indicator coded in columns 54-55 of the Calculation Specifications for the INSRT or REMOV operation codes.

2. If the exception condition occurs during detail or total output, the operation continues with the next detail or total output line after exiting the exception handling routine.

   If an exception condition occurs as a result of an EXCPT or STORE operation code, the operation continues with the next output line which contains the letter E (if the operation was an EXCEPT) or the letter S (if the operation was a STORE) in column 15 of the Output-Format Specifications or the next calculation operation in the Calculation Specifications after exiting the exception handling routine.

Figures 15-12 and 15-13 illustrate a method of retrying an output operation when an exception condition is reported for a DMSII operation. Figure 15-12 shows a method of retrying when cycle-driven DMSII files are specified. Figure 15-13 shows a method of retrying the output operation following an exception condition when the EXCPT operation code is specified in the Calculation Specifications.

```
            Data Base Specifications

01 DPAYROLL              U
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

            File Description Specifications

01 FMASTER  UPE F                    DMS    DEPEND              A
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

               Input Descriptions

01 IMASTER  NS  01
02                                          DUPKEY   DUPKEY
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

             Calculation Specifications

01 CUR                   BEGURMASTER
02 CUR DB                MOVE 'NEWVALUE'DUPKEY           DUPLICATE
03 CUR                   ENDUR
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

             Output-Format Specifications


01 O*
02 O* FIRST OUTPUT RECORD
03 O*
04 OMASTER  DADD     01
05 O          DUPKEY          DUPKEY
06 O*
07 O* SECOND OUTPUT RECORD
08 O*
09 OMASTER  DADD     01 DB
10            DUPKEY          DUPKEY
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 15-12. Program Example of Retrying a Write after Exiting Exception Routine with Cycle-Driven Processing

```
            Data Base Specifications

01 DPAYROLL              U
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

            File Description Specifications

01 FMASTER  UPE F                    DMS    DEPEND             A
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

                Input Descriptions

01 IMASTER  NS  01
02                                          DUPKEY   DUPKEY
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

             Calculation Specifications

01 C   01                EXCPT
02 C   01 DB             EXCPT
03 C*
04 CUR                   BEGURMASTER
05 CUR                   MOVE 'NEWVALUE'DUPKEY           DUPLICATE
06 CUR                   ENDUR
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

             Output-Format Specifications

01 OMASTER  EADD     01
02 O          DUPKEY         DUPKEY
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 15-13. Program Example of Retrying a Write after Exiting Exception Routine with Exception Output Handling

In both figures, the data base PAYROLL is opened update. The file MASTER, defined in the data-base source file as a disjoint data set, is an update primary DMSII file with additions specified (the letter A in column 66 of the File Description Specifications), and is accessed by way of the index DEPEND. When a record from MASTER is read, the record identifying indicator 01 is turned on, and the value from the DMS/DASDL-defined location DUPKEY is loaded to the program-defined variable DUPKEY. Also, both examples use the same exception handling routine, delimited by the BEGUR and ENDUR operation codes in the Calculation Specifications.

In figure 15-12, a record is read and an attempt is made to add a record with the same key as was read by means of the RPG cycle. In figure 15-13, the first EXCPT operation code in the Calculation Specifications attempts to add a record which has the same key as an existing record. In both examples, a DUPLICATES exception condition is reported and the exception condition indicators D1 and DB are set ON. Each program then branches to the exception handling routine, denoted by the BEGUR operation code in the Calculation Specifications. Here, the literal NEWVALUE is moved into the key field DUPKEY and processing continues. In figure 15-12, the processing continues at the second output record specified on the Output-Format Specifications, and in figure 15-13, processing continues with the second EXCPT operation code in the Calculation Specifications. Both methods achieve the same results.

It is recommended that writing to DMSII files be done using the EXCPT and STORE operation codes. This insures timely identification and handling of exception conditions.

DATA MANAGEMENT OUTPUT-FORMAT SPECIFICATIONS

The entries defined in the Output-Format Specifications for DMSII files are divided into the following two sections:

1. Record description section.
2. Field description section.

Record Description Section

The following paragraphs describe the fields in the Record Selection Section.

1-2 PAGE

Refer to section 2 for a complete description.

3-5 LINE

Refer to section 2 for a complete description.

6 FORM TYPE

This field must contain the letter O.

7-14 FILE NAME

The file specified must be one of two types:

1. A disjoint data set previously described on the File Description Specifications as:

   Input with ADD
   Input with DELETE
   Input with ADD/DELETE
   Update
   Update with ADD
   Update with DELETE
   Update with ADD/DELETE
   Output with ADD

2. An embedded data set which is embedded in a disjoint data set defined on the File Description Specifications.

15 TYPE

The letter entries H, D, E, and T are all valid. Also valid is the DMSII letter entry S.

Enter the letter S in this field when this record is to be written as a result of a STORE operation in the Calculation Specifications.

16-18 RECORD ADDITON/DELETION

A record can be added to an existing DMSII file if the following conditions are met.

1. The file is defined in a File Description Specification as an input or update primary or secondary file with an index specified, an input or update demand file, or an output file.
2. Column 66 of the File Description Specifications has the letter A or B.
3. DMSII is entered in the DEVICE FIELD on the File Description Specifications.
4. ADD is entered in the Output-Format Specifications in columns 16-18. ADD is implied for files declared as Input with ADD or Output with ADD.
5. ADD must not be specified on AND or OR lines of the Output-Format Specifications.

Also, a record can be added to a DMSII file if the following conditions are met.

1. The file is an embedded data set.
2. ADD is entered in the Output-Format Specifications in columns 16-18.
3. ADD must not be specified in AND or OR lines of the Output-Format Specifications.

A record can be deleted from an existing DMSII file only if:

1. The file is defined in the File Description Specifications as an input or update primary or secondary file.
2. The letter D or B is entered in column 66 on the File Description Specifications.
3. DMSII is entered in the DEVICE FIELD field on the File Description Specifications.
4. DEL is entered in columns 16-18 in the Output-Format Specifications.
5. DEL must not be specified in AND or OR lines of the Output-Format Specifications.
6. No field description lines can appear with a DEL record description line.

Deletion of records in embedded data sets or demand disjoint data sets is accomplished with the DELET operation code. This does not reference Output-Format Specifications.

23-31 OUTPUT INDICATORS

Besides the indicators normally allowed in columns 23-25, 26-28, and 29-31 fields, any of the DMSII exception condition indicators (D1, DA-DH, DJ-DU) can also be used, but only if DMSII files are specified in the File Description Specifications.

32-74

These fields must be blank for record description lines except when using the *ALL entry or for variable-format data sets.

*ALL OPTION

The *ALL entry is placed in columns 40-43. When an entry is made in this field on a record description line, the RPG compiler generates Output-Format Specifications for the items previously defined in the data-base source file for the specified data set.

If the *ALL option is used on the Input Specifications, any DMS/DASDL-defined name not specified in this record description is made available by the RPG compiler. Because these DMS/DASDL-defined names are made available, a syntax error is generated if the programmer attempts to redefine the DMS/DASDL-defined name.

However, the RPG compiler generates a syntax error if any of the DMS/DASDL defined names have not been previously defined. The DMS/DASDL names can be defined either by specifying the *ALL option in the TO FIELD LOCATION field (columns 48-51) of the record description line for the Input Specifications of the data set or specifying the DMS/DASDL name(s) in the RESULT FIELD field (columns 43-48) of the Calculation Specifications.

The RPG compiler generates a warning message when the *ALL option is specified on an Output-Format Specifications record description line where group and elementary items are referenced. This warning message notifies the programmer that a potential overlapping can occur where the group item can overlay an elementary item and not properly update the DMSII record. This can occur when the elementary field item is explicitly referenced after the *ALL option and the group item is not. The RPG compiler automatically generates a field description line for the group item (and any other elementary items and field names not explicitly referenced). Since the group item follows the explicitly defined elementary item, the group item overlays the DMSII record and any information stored in the elementary item is lost.

*ALL examples for Output-Format Specifications

Figure 15-13a shows an *ALL example, using a DMS/DASDL-defined item that has a data set A for which three items, X, Y, and Z, were declared as NUMBER (4,0) and two items, R and S, were declared as ALPHA (3). M, N, and Q are RPG names, not DMS/DASDL-defined items. ARR is a subscripted array. This example shows the interaction between the Input and Output-Format Specifications. Refer to *ALL Examples for Input Specifications in this section for the Input Specifications associated with this example.

```
          Output-Format Specifications

01 OEMPLOY  E                          *ALL
02 O          X                             +2
03 O          Z              Y
04 O          Y              ARR,Z
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 15-13a. *ALL Entry on Output-Format Specifications

Figure 15-13b shows the necessary specifications in order to achieve the same results without specifying the *ALL entry.

```
          Output-Format Specifications

01 OEMPLOY  E
02 O          R              R
03 O          S              S
04 O          X              X
05 O          Y              Y
06 O          Z              Z
07 O          X                             +2
08 O          Z              Y
09 O          Y              ARR,Z
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 15-13b. Coding Example without the *ALL Entry

At input time, if field S has the character A in position 1, the record identifying indicator 01 is set on and none of the DMS/DASDL-defined items have values moved into them. If the character A is not in position 2, record identifying indicators 02 or 03 are set on and all of the fields have values moved into them because of the *ALL entry. All of the fields are written at exception time. If the input record was a 01 type, the values of fields R, S, X, Y, and Z are either the values at input time or the results of calculation operations.

The RPG compiler does not generate field description lines in the following instances:

1. When DMS/DASDL-defined items appear in column 17 of the field description lines (columns 17-22 can be implicitly defined by the user who must enter a DMS/DASDL-defined item in the VARIABLE NAME field and leave columns 17-22 blank). For this reason, the item X does not have a field line.
2. When an item in columns 32-37 of the field description lines is a DMS/DASDL-defined item name for the named data set. A field description line is not generated for Y because of this condition.
3. When the VARIABLE NAME field (column 32) of a field description line is a subscripted array and the subscript is a DMS/DASDL-defined item for the named data set. Z has no field line in this case.

Figure 15-13c shows the resulting source listing.

```
          Output-Format Specifications

01 OEMPLOY  E                          *ALL
02 O          X                             +2
03 O          Z              Y
04 O          Y              ARR,Z
05 O#         R
06 O#         S
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 15-13c. Resulting Source Listing for Output-Format Specifications.

The syntax for Output-Format Specifications allows for variable-format record specifications. The *ALL entry only applies to the fixed-format portion of the records. The variable-format portion of the records must be coded explicity.

Variable Format Records

When Variable Format Records are specified for a data set in the data-base specifications, the RPG compiler requires certain record-type specifications in the Output-Format Specifications for added records to that data set.

The following example shows data-base source coding where variable format records are specified:

Example:

```
DEPT  DATA SET (
     DEPNUM        NUMBER (5);
     DEPNAM        ALPHA (10);
     RECKEY   RECORD TYPE NUMBER (S2);),

                  -1: ( A   ALPHA   (3)  ;
                        B   ALPHA   (4)   ),

                  -2: ( C   ALPHA   (7)  ;
                        D   NUMBER  (6)   );

            MAXRECORDS = 100;
```

The following should be noted about the records in the DEPT data set:

1. Every record has a fixed portion and a variable portion which depends on the value of RECKEY.
2. The fixed portion of the record consists of the items:

   DEPNUM
   DEPNAM
   RECKEY

3. When records are added to the DEPT data set, DMSII uses RECKEY to determine which of two possible formats to assign to the variable portion of the record.

   1) If RECKEY = −1, the variable portion consists of items A and B.
   2) If RECKEY = 2, the variable portion consists of items C and D.
   3) If RECKEY = 0 or the value defined in DMS/DASDL as the FIXEDFORMATVALUE, this record has only the fixed portion.
   4) If RECKEY = any other value, a DATAERROR exception condition occurs when adding the record.

4. When variable format records are updated, the control item for the record type must not be changed. If a change is attempted, a DATAERROR exception condition results.

If a record is added to a data set with variable formatting, the first record line(s) of the Output-Format Specifications for the data set must specify the value to be assigned to the record-type. The following entries are required in the Output-Format Specifications:

1. Columns 32-37 must specify a field name or columns 45-70 must specify a constant. If a constant is specified, the constant must be formatted according to the rules for output of constants to data sets on the Output-Format Specifications. Specifying a field name in columns 32-37 and a constant in columns 45-70 is not allowed. Decimal alignment is performed when the record is written. Data truncation is not allowed.

2. If columns 14-16 contain the entry AND, columns 32-37 and columns 45-70 must be blank.
3. If columns 15-16 contain the entry OR, columns 32-37 and columns 45-70 can be blank. If columns 32-37 and columns 45-70 are blank, the field name or constant from the last specified OR line or the first record line is used. If the field name or constant position is not blank, the specified value as given applies. Figure 15-14 illustrates this possibility.

```
          Output-Format Specifications

01 OEMPLOY  DADD      01       FIELD
02 O        AND       02
03 O        OR        03
04 O        OR        04
05 O        OR        05                    +2
06 O        OR        06
07 O        OR        07                    0
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 15-14. Example of Variable Format Records Coding

If indicators 01 and 02 are ON, FIELD contains the value for RECKEY.

If indicator 03 is ON, FIELD contains the value for RECKEY.

If indicator 04 is ON, FIELD contains the value for RECKEY.

If indicator 05 is ON, +2 is the value for RECKEY.

If indicator 06 is ON, +2 is the value for RECKEY.

If indicator 07 is ON, 0 is the value for RECKEY.

75-80 PROGRAM IDENTIFICATION

Refer to section 2 for a complete description.

Field Definitions for the Field Description Lines

The following paragraphs describe the fields for the Field Description lines.

1-2 PAGE

Refer to section 2 for a complete description.

3-5 LINE

Refer to section 2 for a complete description.

6 FORM TYPE

This field must contain the letter O.

7-16

7-16 These fields must be left blank for field description lines.

17-22 DMSII OUTPUT LOCATION

The DMSII OUTPUT LOCATION field indicates the output location and format of the variable name entered in columns 32-37 or the constant entered in columns 45-70. The entry in this field must be a field name that has been previously defined in the specified data set in the data-base source file or can be blank if the variable name in the VARIABLE NAME field is a DMS/DASDL-defined name. An entry in this field must not be a subscripted vector. This is the only case of a field description line having entries in columns 17-22.

If the entry in this field is defined in the data-base source file as unsigned numeric, all signs are stripped off before the variable or constant is written.

23-31 OUTPUT INDICATORS

Besides the indicators normally allowed in columns 23-25, 26-28, and 29-31, any of the DMSII exception condition indicators (D1, DA-DH, DJ-DU) can also be used, but only if DMSII files are specified in the File Description Specifications.

32-37 VARIABLE NAME

The VARIABLE NAME field specifies an output data field. The identifier used must have been previously defined in the Input Specifications, Extension Specifications, or Calculation Specifications. For DMSII files, the entry can be the same as the entry made in the DMSII output location (columns 17-22 on the Output-Format Specifications). The following conditions must be satisfied:

1. If the variable is numeric, the DMS/DASDL definition must be numeric. Decimal point alignment is automatically performed.
2. If the variable is alphanumeric, the DMS/DASDL definition must be alphanumeric. The data is stored left-justified with blank fill if required.

Truncation of data is not allowed.

Table 15-2 illustrates the various combinations of DMSII OUTPUT LOCATION and VARIABLE NAME fields and the action taken.

Table 15-2. Combinations of DMSII OUTPUT LOCATION and VARIABLE NAME Fields in the Output-Format Specifications

OUTPUT LOCATION Columns	VARIABLE NAME Columns	Program Action
17-22 Vector	32-37 Array	Write the contents of the whole Array to the location of the DMS/DASDL-defined Vector.
Vector	Array,J	Write the contents of Array,J to the location of the DMS/DASDL-defined Vector,J.
Vector	Table,J	Write the contents of Table,J to the location of the DMS/DASDL-defined Vector,J.
Vector	Array,3	Write the contents of Array,3 to the location of the DMS/DASDL-defined Vector,3.
Vector	Table,3	Write the contents of Table,3 to the location of the DMS/DASDL-defined Vector,3.
Vector	Table	Not allowed.
Vector	Field Name	Not allowed.
Field Name	Array	Not allowed.
Field Name	Array,J	Write the contents of Array,J to the location of the DMS/DASDL-defined Field Name.
Field Name	Table,J	Write the contents of Table,J to the location of the DMS/DASDL-defined Field Name.
Field Name	Array,3	Write the contents of Array,3 to the location of the DMS/DASDL-defined Field Name.
Field Name	Table,3	Write the contents of Table,3 to the location of the DMS/DASDL-defined Field Name.
Field Name	Field Name	Write the contents of the Field Name in columns 32-37 to the location of the DMS/DASDL-defined Field Name in columns 17-22.
Field Name	Table	Not allowed.

Table 15-2. Combinations of DMSII OUTPUT LOCATION and VARIABLE NAME Fields in the Output-Format Specifications (Cont)

OUTPUT LOCATION Columns	VARIABLE NAME Columns	Program Action
Blank	Array	Write the contents of the whole Array to the location of the DMS/DASDL-defined Vector. The definition used for columns 17-22 defaults to the DMS/DASDL-defined name in columns 32-37.
Blank	Array,J	Write the contents of Array,J to the location of the DMS/DASDL-defined Vector,J. The definition used for columns 17-22 defaults to the DMS/DASDL-defined name in columns 32-37.
Blank	Array,3	Write the contents of Array,3 to the location of the DMS/DASDL-defined Vector,3. The definition used for columns 17-22 defaults to DMS/DASDL-defined name name in columns 32-37.
Blank	Table	Not allowed.
Blank	Table,J	Write the contents of Table,J to the location of the DMS/DASDL-defined Vector,J. The definition used for columns 17-22 defaults to the DMS/DASDL-defined name in columns 32-37.
Blank	Table,3	Write the contents of Table,3 to the location of the DMS/DASDL-defined Vector,3. The definition used for columns 17-22 defaults to the DMS/DASDL-defined name in columns 32-37.
Blank	Field Name	Write the contents of Field Name to the location of the DMS/DASDL-defined Field Name. The definition used for columns 17-22 defaults to the DMS/DASDL-defined name in columns 32-37.

38 EDIT CODES

This field must be blank for DMSII files.

40-43 END POSITION

This field must be blank for DMSII files.

44 PACKED

This field must be blank for DMSII files.

45-70 CONSTANT

Edit words must not be entered in this field for DMSII files.

The following rules must be observed when forming constants with DMSII files:

1. The VARIABLE NAME field entry must be blank.
2. The value of any constant must be numeric if the DMSII OUTPUT LOCATION entry is numeric and alphanumeric if the DMSII OUTPUT LOCATION entry (columns 17-22) is alphanumeric.
3. If the DMSII OUTPUT LOCATION entry (columns 17-22) is numeric, column 45 must contain:

    1) Plus sign (+) character for positive values.
    2) Negative sign (−) character for negative values.
    3) A digit (0-9) to imply positive.

4. Decimal point alignment is performed on the entry.
5. Sign characters and decimal points are not included in the size of the literal.
6. The DMS/DASDL definition must be at least as long as the constant after decimal point alignment has been performed.

    If the DMS OUTPUT LOCATION entry is numeric, embedded blanks must not appear in the constant field.

7. If the DMSII OUTPUT LOCATION field (columns 17-22) is alphanumeric, then:

    1) The DMS/DASDL definition must be at least as long as the constant.
    2) Data is left-justified with blank fill, if necessary.

71-74

These columns must be blank.

75-80 PROGRAM IDENTIFICATION

Refer to section 2 for a complete description.

Figure 15-15 shows coding examples for data management files in the Output-Format Specifications.

```
          Output-Format Specifications

01 O*
02 O* ADDING A RECORD TO THE "MASTER" DATA SET.
03 O*
04 OMASTER  DADD     01
05 O          DMSIT1          FIELD1
06 O*
07 O* DELETING A RECORD FROM THE "MASTER" DATA SET.
08 O*
09 OMASTER  DDEL     02
10 O*
11 O* UPDATING A RECORD IN THE "MASTER" DATA SET.
12 O*
13 OMASTER  S        03
14 O          DMSFLD          FIELD1
15 O          DMSIT1                      +2
16 O          DMSIT2                      'ALPHA CONSTANT'
17 O                          DMSIT3
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

NOTES

Line 14 writes the FIELD1 field to the location defined by the DMS/DASDL-defined DMSFLD field.

Line 15 writes the numeric constant +2 to the location defined by the DMS/DASDL-defined DMSIT1 numeric field.

Line 16 writes the alphanumeric constant ALPHA CONSTANT to the location defined by the DMS/DASLD-defined DMSIT2 alphanumeric field.

Line 17 writes the DMSIT3 field to the location defined by the DMS/DASDL-defined DMSIT3 field, the DMSII OUTPUT LOCATION field is not required.

Figure 15-15. Output-Format Specifications for DMSII Files

DATA MANAGEMENT AUDIT AND RECOVERY

A processing failure can interrupt a logical update to a data base. For example, a data record could have been inserted into a data set but not into one of its automatic sets. To recover data base integrity after such a failure, it is necessary to preserve the "before" images so that partial operations can be removed from the data base. Storage failures can occur requiring "after" images to restore data base integrity. In order to recover the data base, an audit trail must be maintained by DMSII.

If the AUDIT option is set in the data-base source file, an audit trail is maintained by DMS. This audit trail can be stored on tape or disk. If disk is specified, use a disk pack other than the disk pack that contains the data base. If the audit trail and data base are maintained on the same disk pack, and an irrecoverable disk failure occurs, the data base can only be recovered to the point where the data base was dumped to tape or another disk pack. Maintaining separate disk packs for the audit trail and data base makes recovery of the data base faster.

When a data base is audited (AUDIT option set in the data-base source file) all changes to the data base are written sequentially to the audit file. If a processing failure occurs, this audit file contains each change so that they can be reapplied to the data base by the DMSII recovery routines. All logically incomplete transactions are removed from the data base.

The data-base source file for an audited data base includes a RESTART DATA SET declaration. This restart data set holds the necessary information so that a program can realign itself with the last good transaction prior to the processing failure. When DMSII recovers the data base following a processing failure, the DMSII recovery routines store each program's restart information (based on the contents of the audit trail) into the restart data set. Since the restart data set is part of the data base, RPG programs can conveniently retrieve the restart information from this data set to reinitialize themselves after completion of the recovery process.

All updating to an audited data base must be done within transaction state. There are two operation codes in RPG that delimit transaction state for an RPG program. They are TRBEG and TREND. TRBEG specifies the beginning of transaction state and TREND specifies the end of transaction state. These are explicit operations and are not required for RPG cycle-driven file processing.

Cycle-driven DMSII files generate TRBEG and TREND operation codes implicitly. This occurs when:

1. A restart data set is specified in the File Description Specifications.
2. One or more cycle-driven DMSII files are specified in the File Description Specifications.
3. The data base is opened update.

A TRBEG operation code is generated by the RPG compiler after the record identifying indicator is set ON for the cycle-driven DMSII file. One other TRBEG operation code is generated when end of file (the LR indicator ON) is reached for cycle-driven DMSII files just prior to end-of-job totals. Figure H-1 in appendix H shows the implicit TRBEG and TREND operation codes performed on cycle-driven files in relation to the RPG cycle.

A TREND operation code with audit and no syncpoint is generated by the RPG compiler following detail output except when the 1P indicator is ON. One other TREND operation code with no audit and syncpoint is generated after end-of-job totals and before all files are closed.

The DMSII recovery routines are transaction-oriented and always recover to a time when no partial transactions are reflected in the data base.

A transaction consists of a sequence of DMSII operations grouped together in an RPG program. These DMSII operations constitute a logical change to the data base. The DMSII operations that actually change the data base, (STORE, DELET, INSRT, REMOV) must be performed within a transaction state; otherwise, an AUDITERROR exception condition occurs. Other operations such as the FIND operation code can be performed either in or out of a transaction state since the operation does not change any item in the data base.

Following the recovery of a data base, the RPG program must reprocess any transactions which were interrupted due to the processing failure. To assist the RPG program in restarting, restart information must be stored in the restart data set. If a processing failure occurs, the recovery routines insure that the information corresponding to the last good transaction is in the restart data set. Using this information, the program can restart at the point where the data base was recovered.

If a data base is opened update (the letter U in column 27 of the Data Base Specification) and the data base is being audited, the restart data set defined in the data-base source file must be defined in the File Description Specifications in the RPG program. If a data base is opened input only (the letter I in column 27 of the Data Base Specifications), the RPG program must not specify File Description Specifications for a restart data set.

Restart Data Set File Description Specifications

Every restart data set to be used by an RPG program must be described to the RPG compiler in the File Description Specifications. The restart data set must be specified and the AUDIT option must be set in the data-base source file.

Field Definitions

The following paragraphs describe the field definitions for the restart data set File Description Specifications.

1-2 PAGE

Refer to section 2 for a complete description.

3-5 LINE

Refer to section 2 for a complete description.

6 FORM TYPE

This field must contain the letter F.

7-14 DATA SET NAME

This field must contain the name of the restart data set which is specified in the DMS/DASDL source.

15 FILE TYPE

For a restart data set, this field must contain the letter R.

16-39

These fields must be blank for a restart data set.

40-46 DEVICE

For a restart data set, this field must contain the entry DMS.

47-65

For a restart data set, these fields must be blank..

66 FILE ADDITION/UNORDERED/DELETION

For a restart data set, all entries for this field which are valid for demand files with device type DMSII are also valid for a restart data set. Valid entries for this field are:

Entry	Definition
A	Records can be added to this file.
Blank	Records cannot be added to this file.

67-70

These columns must be left blank for DMSII files.

71-72 FILE CONDITION

The FILE CONDITION field can be used on input, update, output, and restart data management files and indicates whether the file is conditioned by an external indicator. If a DMSII file is conditioned by a U-indicator, the file is used only when that indicator is on. When the U-indicator is off, the file is treated on input as if the end-of-file record was read. In the Calculation Specifications, if any DMSII operations are performed to the conditioned file, the operations should be conditioned with the same external indicator specified for the DMSII file. If the DMSII operations are not conditioned by the same U-indicator, the DMSII operations are ignored and the D1 resulting indicator, if used, is not affected. Likewise, any output performed to the DMSII file must be conditioned by the same external indicator used on the DMSII file. If the same U-indicator is not used, the output record is built, but the record is not written to the file. If column 39 (BLANK AFTER field) of the Output-Format Specifications contains the letter B for any output field, the output field is initialized to blank characters.

It is important to note that DMSII files are accessed by opening the data base. If all DMSII files in the program are conditioned by external indicators that are off, the data base is opened and any exceptions that normally happen at open time can occur. For instance, U-indicators that are off can require a recompile of the data base to prevent the occurrence of a VERSIONERROR exception at data base open time.

Only disjoint data sets can be specified on File Description Specifications. Therefore, any embedded data sets associated with a disjoint data set conditioned by an external indicator are treated as having been conditioned by the same U-indicator. All Calculation Specifications and Output-Format Specifications affecting the embedded data set should be conditioned by the same external indicator as the parent's data set, and are ignored if the U-indicator is off.

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

Refer to FILE CONDITION (columns 71-72) on non-DMSII files in section 4 for more information on external indicators.

73-74

These columns must remain blank for DMSII files.

75-80 PROGRAM IDENTIFICATION

Refer to section 2 for a complete description.

Restart Data Set Input Specifications

The Input Specifications for a restart data set are identical to those specifications for any demand-driven file whose device type is DMS. Input Specifications for a restart data set are required only if the program has FIND or LOCK operations applied to it.

Restart Data Set Calculation Specifications

All DMSII operations which are valid for demand DMSII data sets are also valid for the restart data set. Specifically, restart data sets are essentially update, demand DMSII files. However, no operations which alter the restart data set, for example STORE, are allowed outside of a transaction state. If such an operation is attempted, an AUDITERROR exception condition occurs.

Two additional operation codes are needed to explicitly define a transaction to DMSII for an audited data base. These are the TRBEG and TREND operations. The use of these operation codes is defined as follows.

TRBEG Operation Code

Initiates transaction state for this program. The TRBEG operation code must be specified in the OPERATION CODE field (columns 28-32) of the Calculation Specifications. The D1 exception condition indicator can be entered in columns 54-55. All other fields must be blank. No record is written to the restart data set as a result of execution of the TRBEG operation code.

TREND Operation Code

The TREND operation code ends transaction state for this program. The FACTOR 2 field can contain the entry NOAUDIT, and the RESULT FIELD can contain the entry SYNCPT. The D1 exception condition indicator can be entered in columns 54-55.

If the FACTOR 2 field contains the entry NOAUDIT, no information is written to the restart data set. If the FACTOR 2 field is left blank, a STORE operation is performed on the restart record defined with the letter R in column 15 of the Output-Format Specifications.

If the RESULT FIELD field contains the entry SYNCPT, a syncpoint is forced by DMSII prior to returning control back to this program. The SYNCPT option ensures that all transactions performed prior to this TREND operation code with syncpoint are reflected in the audit trail. If the RESULT FIELD field is left blank, no syncpoint is forced.

It is recommended that the D1 exception condition indicator be entered in columns 54-55 of the Calculation Specifications for the TRBEG and TREND operation codes. If columns 54-55 are left blank for a TRBEG or TREND operation code and an exception condition occurs, the exception handling routine for the restart data set is entered. Refer to the subsection titled Data Management Exception Handling in this section for more information. If no exception handling routine exists, the program aborts.

The ABORT exception condition can occur on a TRBEG operation or on a TREND operation code with syncpoint operation. Any exception condition which can occur on a STORE operation code can also occur on a TREND operation code with audit.

Exception conditions can occur on the implicitly generated transaction operations, just as they can occur when TRBEG and TREND operation codes are specified in the Calculation Specifications. However, the D1 indicator cannot be specified for use on the implicitly generated transaction operations.

If an exception condition occurs on an implicitly generated TRBEG or TREND operation code, the exception handling routine for the restart data set is entered. If no such routine exists and an exception condition occurs, the program aborts.

The implicit TREND operation code generates an end transaction with audit and no syncpoint, except when the LR indicator is ON in which case the implicit TREND operation code is generated with no audit and syncpoint. This guarantees that if this operation is successful, no data base operations for this program are backed out on a program abort or CLEAR/START recovery.

Table 15-3 lists the RPG coding options available for the TRBEG and TREND operation codes.

Table 15-3. TRBEG and TREND Operation Coding Options

Operation Code	Factor 2	Result Field	Columns 54-55	Description
TRBEG	Blank	Blank	D1	Beginning of a transaction state.
TREND	Blank	Blank	D1	A STORE operation is performed by the RPG program to the restart data set. An Output-Format Specifications record is required which contains the letter R in column 15.
TREND	NOAUDIT	Blank	D1	A STORE operation is not to the restart data set.
TREND	Blank	SYNCPT	D1	Syncpoint is forced for auditing prior to returning control back to the program.
TREND	NOAUDIT	SYNCPT	D1	A STORE operation is not performed by the RPG program to the restart data set. A syncpoint is forced for auditing prior to returning control back to the program.

Restart Data Set Output-Format Specifications

All allowable Output-Format Specifications entries for DMSII files are also applicable to the restart data set. Also required are special Output-Format Specifications for the implicit or explicit TREND operation codes. These are defined as follows:

7-14 DATA SET NAME

This field must contain the name of the restart data set which is defined in the File Description Specifications.

15 TYPE

This field must contain the letter R whenever a TREND operation code with audit is generated implicitly or explicitly.

The following rules apply when coding type R Output-Format Specifications for the TREND operation code:

1. Only type R Output-Format Specifications are used for a TREND operation code with audit.
2. If more than one type R output record is eligible for output at the time of the TREND operation code, an AUDITERROR exception condition occurs. This is due to the fact that all writing to a restart data set must be done within a transaction. Once one STORE operation has been performed against the restart data set, the transaction ends until the next TRBEG operation.
3. If no type R records are eligible for output at the time of the TREND operation code, an AUDITERROR exception condition occurs when the next TRBEG operation code is performed. The AUDITERROR exception condition results from consecutive TRBEG operation codes because the first transaction state did not end with an implicit or explicit TREND operation code.

The recovery record(s) contained in the restart data set can contain a programmatically defined key. In the case of two or more copies of the same RPG program running concurrently in the mix, the restart data set can contain unique keys (for example a batch ticket).

The effective restart of a program implies that the RPG program must have saved enough information in the restart data set to be able to commence processing as though no interruption occurred. This includes the ability to reposition all non-DMSII files to a point where integrity is assured. This requires RPG source code in each RPG program which is auditing the data base to read from the restart data set and use this information to realign the RPG program relative to the last good transaction.

Figure 15-16 is an example of one technique illustrating audit and recovery. The function of the RPG program is to read records from a disk file and then apply these records (additions, changes, and deletions) to a DMSII master file. The recovery method utilized needs the following information:

1. A data item, STATUS, which contains a value of C or I. If STATUS is equal to C, this run is complete. If STATUS is equal to I, this run is incomplete.
2. A data item, RRNUM, which contains the value of the last successfully processed relative record number in the input file.
3. Since there is only one restart data set for all programs using a data base, and each program needs its own restart record within the restart data set, there must be a "key" within each record of the restart data set which uniquely identifies that record as belonging to this program. In this example, the "key" is the program name, SAMPLE.

In a non-DMSII environment, this program would read one record per cycle, increment a counter according to the record type, "chain" to the "master" file, and perform detail or exception output to the "master" file.

Figure 15-16 follows a similar pattern when a data set named MASTER and index named MSTKEY are substituted for the non-DMSII indexed disk file in the situation described above.

```
                Data Base Specifications

01 DDEMAND                U
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

                File Description Specifications

01 FDISKIN  IPE F  80  80            DISK
02 FMASTER  UD  F                    DMS                           A
03 FRESTART R   F                    DMS                           A
04 FODT     D   F  50  50            CONSOLE
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

                  Input Specifications

01 IDISKIN  NS  01    1 CA
02 I        OR  02    1 CC
03 I        OR  03    1 CD
04 I                                      1  80 KEYNUM
05 I                                      1  80 RECORD
06 IMASTER  NS
07 IRESTART NS  04    1 CC                STATUS
08 I        OR  05    1 CI                STATUS
09 I                                      STATUS   STATUS
10 I                                      RRNUM    RRNUM
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

                  Calculation Specifications

01 C*
02 C* CHECK RESTART STATUS FIRST TIME ONLY.
03 C*
04 C  N10                 EXSR RESTRT
05 C*
06 C* ALWAYS COUNT THE NUMBER OF RECORDS READ.
07 C*
08 C   01       1         ADD  NUMADD    NUMADD  30
09 C   02       1         ADD  NUMADD    NUMADD  30
10 C   03       1         ADD  NUMADD    NUMADD  30
11 C   01
12 COR 02
13 COR 03       1         ADD  NUMREC    NUMREC  40
14 C   05      NUMREC     COMP RRNUM                11
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 15-16. Program Example of Restart with a Demand DMSII Files (Sheet 1 of 4)

Calculation Specifications
(continued)

```
15 C*
16 C* READY TO PROCESS AGAIN
17 C*
18 C    11                  SETOF                          05
19 C    05                  GOTO END
20 C    11                  MOVE 'COMPLETE'MSG
21 C    11          MSG     DSPLYODT
22 C    11                  SETOF                          11
23 C    01                  EXSR ADD
24 C    02                  EXSR CHANGE
25 C    03                  EXSR DELETE
26 C                END     TAG
27 C*
28 CLR                      TRBEG
29 CLR                      TREND           SYNCPT
30 C*
31 CSR              ADD     BEGSR
32 CSR              MSTKEY  FIND MASTER                    RD1
33 CSR                      DMKEYKEYNUM     KEYMST          EQL
34 CSR D1 DA                TRBEG
35 CSR D1 DA                STOREMASTER
36 CSR                      TREND
37 CSR                      ENDSR
38 C*
39 CSR              CHANGE  BEGSR
40 CSR              MSTKEY  LOCK MASTER                    RD1
41 CSR                      DMKEYKEYNUM     KEYMST          EQL
42 CSRND1                   TRBEG
43 CSRND1                   STOREMASTER
44 CSRND1                   FREE MASTER
45 CSRND1                   TREND
46 CSR                      ENDSR
47 C*
48 CSR              DELETE  BEGSR
49 CSR              MSTKEY  FIND MASTER                    RD1
50 CSR                      DMKEYKEYNUM     KEYMST          EQL
51 CSRND1                   TRBEG
52 CSRND1                   TRBEG
53 CSRND1                   STOREMASTER
54 CSRND1                   FREE MASTER
55 CSRND1                   TREND
55 CSR                      ENDSR
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 15-16. Program Example of Restart with a Demand DMSII Files (Sheet 2 of 4)

Calculation Specifications
(continued)

```
57 C*
58 CSR          RESTRT    BEGSR
59 CSR                    SETON                          10
60 CSR                    MOVE 'SAMPLE'  KEYRES
61 CSR          RESKEY    LOCK RESTART                 RD1
62 CSR                    DMKEY'SAMPLE'  KEYRES         EQL
63 C*
64 C* IF THE RESTART RECORD IS FOUND AND THE LAST RUN IS
65 C* COMPLETE, THEN RESET AND GET OUT.
65 C*
67 CSR 04                 MOVE 'I'       STATUS
68 CSR 04                 GOTO ENDRES
69 C*
70 C* IF THE RESTART RECORD IS NOT FOUND, THEN CREATE IT
71 C* AND GET OUT.
72 C*
73 CSR DA                 MOVE 'I'       STATUS
74 CSR DA                 SETON                          12
75 CSR DA                 TRBEG
75 CSR                    TREND
77 CSR DA                 SETOF                          12
78 CSR DA                 GOTO ENDRES
79 C*
70 C* RESTART IS REQUIRED; INDICATOR 05 REMAINS ON UNTIL
81 C* RECOVERY IS COMPLETED.
82 C*
83 CSR 05                 MOVEL'RECOVERY'MSG  16
84 CSR 05                 MOVE 'STARTED 'MSG
85 CSR 05     MSG         DSPLYODT
85 CSR        ENDRES      ENDSR
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 15-16. Program Example of Restart with a Demand DMSII Files (Sheet 3 of 4)

```
            Output-Format Specifications

01 O*
02 O* ADDITIONS TO MASTER DATA SET
03 O*
04 OMASTER  SADD     01
05 O          MSTREC          RECORD
06 O*
07 O* CHANGES TO MASTER DATA SET
08 O*
09 OMASTER  S        02
10 O          MSTREC          RECORD
11 O*
12 O* CREATES RESTART RECORD IF NOT FOUND
13 O*
14 ORESTART RADD     12
15 O          STATUS          STATUS
16 O          KEYRES          KEYRES
17 O          RRNUM                        +0000
18 O*
19 O* UPDATES RESTART RECORD AFTER EACH TRANSACTION (TREND OP CODE)
20 O*
21 ORESTART R       N12NLR
22 O          STATUS          STATUS
23 O          RRNUM           NUMREC
24 O*
25 O* CHANGES STATUS FIELD TO "C" AT END OF JOB (LR INDICATOR ON)
26 O*
27 ORESTART R        LR
28 O          STATUS                       'C'
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 15-16. Program Example of Restart with a Demand DMSII Files (Sheet 4 of 4)

The actual file maintenance occurs in one of three subroutines (ADD, CHANGE, or DELETE) dependent upon input record types. Prior to each STORE operation, transaction state must be entered using the TRBEG operation. The STORE operation is then performed followed by the TREND operation to end the transaction state. When the TREND operation is used, the program looks for a type R output record which is properly conditioned by indicators. In this case, it is the second output record for the data set RESTART of the Output-Format Specifications. The restart record is updated with the current relative record number and status.

This process continues until the end-of-file (EOF) record has been read from the the primary file. When the LR indicator is on, the restart status field named STATUS must be reset to C to indicate completion of the run. By entering and exiting transaction state (by performing TRBEG and TREND operations respectively) an update is made to the restart record to reflect STATUS = C (the last output record for the data set RESTART of Output-Format Specifications in figure 15-16). By including SYNCPT on the TREND operation, a syncpoint is forced on DMS. Thus, if the TREND operation is successful, the program is assured that all of its previous activity is protected by the recovery mechanism of DMS. That is, if the system fails after the TREND and before end of job, all the update operations of the program are reflected in the audit trail and can be recovered by the DMS/RECOVERB program.

In the event of a system failure, the operator must recover the data base. After the data base is recovered, the operator can re-execute any programs that were running at the time of the failure.

The first thing that the RPG program in figure 15-16 does is branch to the subroutine RESTART. This subroutine attempts to locate its restart record in the restart data set using its key SAMPLE. If the record is found and the STATUS = C designating complete, the STATUS field is updated with I, designating incomplete, for the current run. Processing continues with the next instruction following EXSR RESTART in the Calculation Specifications.

If the restart record is not found, the indicators D1 and DA are set ON indicating that this is the first time this program has been run. In this case, the restart record must be initialized and added to the restart data set. The addition is accomplished by the TRBEG and TREND operations. The record is added by the first RESTART output record of the Output-Format Specifications in figure 15-16. The STATUS then equals I, incomplete, for the current run, and processing begins.

If the restart record is found and STATUS equals I, the program aborted during its last execution and special action must be taken by the program. The field RRNUM contains the number of input records which were successfully processed during the previous run. This means that records must be skipped for this run. As long as indicator 05 is on and STATUS equals I, all updating activity is bypassed. When the number of records read during this execution exceeds the number of successfully processed records during the previous run, indicator 05 is set OFF and normal processing resumes.

Figure 15-17 is an example of a cycle-driven DMSII file and a different method than illustrated in figure 15-16 for audit and recovery. The function of this RPG program is to sequentially read an entire data set and update one of its fields. The field CURR is added to the field OVRDUE and the new value of OVRDUE is stored in the data set record (first ACCTREC output record of the Output-Format Specifications, figure 15-17). In the event of a failure, the program must know which records have been updated and which have not, so that the updating process can continue at the correct point. Updating the same record twice results in an incorrect value for the field OVRDUE.

```
            Data Base Specifications

01 DCYCLE                U
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


            File Description Specifications

01 FACCTREC UPE                          DMS
02 FRESTART R                            DMS                          A
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


                Input Specifications

01 IACCTREC NS  01
02 I                                     CURR      CURR
03 I                                     OVRDUE    OVRDUE
04 IRESTART NS  02
05 I                                     RRNUM     RRNUM
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


               Calculation Specifications

01 C  N10                EXSR RESTRT
02 C   01       1        ADD  NUMREC    NUMREC  40
03 C   02       NUMREC   COMP RRNUM                11
04 C   11                SETOF                     0211
05 C   02                GOTO END
06 C*
07 C* DATA BASE IS RECOVERED AND THE FILES ARE REALIGNED,
08 C* PROCESSING CONTINUES
09 C*
10 C            CURR     ADD  OVRDUE    OVRDUE
11 C            END      TAG
12 C*
13 C* IF THE LR INDICATOR IS ON, DELETE THE RESTART RECORD.
14 C*
15 CLR                   DELETRESTART             S
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 15-17. Program Example of Restart with a Primary DMSII File (Sheet 1 of 2)

```
                    Calculation Specifications
                          (continued)
16 C*
17 C* CHECK FOR RESTART RECORD
18 C*
19 CSR          RESTRT    BEGSR
20 CSR                    SETON                        10
21 CSR          RESKEY    LOCK RESTART                RD1
22 CSR                    DMKEY'SAMPL1'  KEYRES         EQL
23 C*
24 C* IF THE RESTART RECORD IS NOT FOUND, THEN CREATE A
25 C* RESTART RECORD.
26 C*
27 CSR DA                 SETON                        12
28 CSR 12                 STORERESTART
29 CSR 12                 SETOF
30 C*
31 C* IF THE RECORD IS FOUND, THEN THE 02 INDICATOR REMAINS ON
32 C* UNTIL THE RECOVERY IS COMPLETE.
33 CSR                    ENDSR
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


              Output-Format Specifications

01 OACCTREC D         01N02
02 O          OVRDUE          OVRDUE
03 ORESTART SADD      12
04 O          KEYRES                        'SAMPL1'
05 O          RRNUM                         +0000
06 ORESTART R
07 O          RRNUM NO2       NUMREC
08 O          RRNUM  02       RRNUM
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure 15-17. Program Example of Restart with a Primary DMSII File (Sheet 2 of 2)

Instead of a STATUS field, as shown in figure 15-16, the restart data set named RESTART in this example uses a different method for determining whether or not a failure occurred. If the result record is present at the beginning of job, then the previous run aborted. If the restart record is not present at the beginning of job, then the previous run was successful. The program adds the restart record to the restart data set named RESTART which has an initial RRNUM value of zero (first output record of the Output-Format Specifications for the restart data set named RESTART, figure 15-17).

As each record is updated, the value of NUMREC is incremented by one and stored in the RRNUM field of the restart record (third output record of the Output-Format Specifications, for the restart data name RESTART, figure 15-17). At the end of job, the entire restart record is deleted.

At beginning of job (BOJ), the RPG program checks for a restart record. This is performed by the subroutine RESTRT. If the restart record is not found, indicators D1 and DA are set ON and the restart record is initialized and added. If the restart record is found, the previous run has aborted and the restart record is required to realign the files. The record identifying indicator 02 is set ON when the restart record is read. As long as indicator 02 remains ON, the only action taken is to add 1 to the counter for each input record, NUMREC, and compare it to RRNUM in the restart record. If NUMREC exceeds RRNUM, indicator 02 is set OFF and normal processing continues.

The RPG compiler generates an implied TRBEG operation at the beginning of each cycle and an implied TREND operation at the end of each cycle. The implied TREND operation uses the Output-Format Specifications line with the letter R in column 15 for the data set (third output record of the Output-Format Specifications, for the restart data set named RESTART, figure 15-17).

DMS/DASDL COMPILATION INFORMATION AND SUGGESTIONS

The following information and suggestions can be used when compiling the DMS/DASDL source program with the DMS/DASDL compiler.

1. Include a $ RPGLIB DMS/DASDL compiler dollar sign option when compiling the DMS/DASDL source statements. This option informs the DMS/DASDL compiler to create RPG library files to be used by the RPG compiler. These library files reside on disk with the following naming convention:

   &<data base name>/<filename>

   The ampersand (&) character appears before the data base name portion of the two-name filename. The ampersand designates the file as an RPG library file. There is one library file created on disk for each disjoint data set declared in the DASDL source statements.

2. Include a $ RPG DMS/DASDL compiler dollar sign option when compiling the DMS/DASDL source statements. This option informs the DMS/DASDL compiler to check for RPG syntax. For example, all data names declared cannot exceed six characters in length.
3. If a restart data set is specified in the RPG program, the DMS/DASDL source program must specify the same restart data set, and the AUDIT option must be specified.

GLOSSARY OF COMMONLY USED DATA MANAGEMENT TERMS

Refer to the B 1000 Systems Data Management System II (DMSII) Reference Manual for a complete description of the following data management terms.

ABORT Exception

An ABORT exception is returned to a program when another program aborts or goes to end of job while in transaction state. At this point, the DMSII recovery routines may have backed out some transactions of the program. The program must be restarted in order to recover the backed out transactions.

AUDITERROR Exception

An AUDITERROR exception occurs when any of the following occur:

1. A program attempts an invalid sequence of begin or end transaction operations. That is, two begin transactions or two end transactions in a row.
2. A program attempts a begin or end transaction when the AUDIT option is not set in the data base.
3. A program attempts to update a data base outside of transaction state.
4. A program attempts to close the data base while in transaction state.

CLOSEERROR Exception

A CLOSEERROR exception occurs when a program attempts to close a data base which is not open.

Contention Limit

Contention limit is the amount of time that a program attempting to lock a record waits on another program which has that record locked. If that amount of time is exceeded, a DEADLOCK condition results.

DATAERROR Exception

A DATAERROR exception occurs when any of the following occur:

1. A program attempts to store a record with a null key or null required item.
2. A program attempts to store a null record.
3. A program attempts to store a record and a DASDL verify condition is not met.

Data Set

A data set is a collection of related records. Only data sets and embedded data sets have records.

DEADLOCK Exception

A DEADLOCK exception is returned to a program following resolution of a deadly embrace. DMSII automatically performs a FREE operation on all locked records for the program.

Deadly Embrace

A deadly embrace occurs when all of the following occur:

1. Two or more programs are accessing the same structure(s).
2. Each program owns a set of locked records.
3. Each program requests access to a record locked by one of the other programs.

When a deadly embrace occurs, a DEADLOCK exception is returned to the lowest priority program involved in the embrace.

Defined State

A pointer is in a defined state when it refers to a valid record or path.

Disjoint Data Set

A disjoint data set is a data set which is not an item within another data set. Disjoint sets can only refer to disjoint data sets.

DMS/DASDL

DMS/DASDL is an acronym for Data Management System Data And Structure Definition Language. Compilation of a DMS/DASDL source deck with the DMS/DASDL compiler defines the structure and type of data allowed in a data base.

DUPLICATES Exception

A DUPLICATES exception occurs when duplicate records are not allowed in a set or manual subset and one of the following occurs:

1. A program attempts to store a duplicate record in a set.
2. A program attempts to insert a duplicate record in a manual subset.

Embedded Data Set

An embedded data set is a lists of items within a data set. An embedded data set can only be referenced by an embedded set on the same level.

FATALERROR Exception

A FATALERROR exception occurs when a failure in the data base system software is experienced.

Index

An index is a table of pointers to a data set which provides a specified access to a data set.

INTEGRITYERROR Exception

An INTEGRITYERROR exception occurs when inconsistent structure control information is encountered, such as key and data mismatch, an automatic set or subset entry missing during a STORE or DELET operation, or an automatic set or subset pointing to a deleted record.

INUSE Exception

An INUSE exception occurs when a program attempts to delete a record with a non-null embedded structure.

IOERROR Exception

An IOERROR exception occurs when a parity error (I/O error) occurred while trying to read from the data base.

Key

A key is a value used to locate specific records in a data set spanned by a set or referenced by a subset.

KEYCHANGED Exception

A KEYCHANGED exception occurs when a program attempts to store a record when the value of an item used as a key in a set is incorrectly changed (duplicate keys or embedded data sets are not allowed).

LIMITERROR Exception

A LIMITERROR exception occurs when a program attempts to store a record in a structure, and the amount of data in the structure exceeds its physical size.

Master Record

A master record is a data set record which has dependent data sets. A master can itself be a record in an embedded data set. An embedded data set cannot be accessed without accessing the master.

NORECORD Exception

A NORECORD exception occurs with any of the following:

1. The current record pointer is not valid for an INSRT operation.
2. The current record pointer is not valid for a FIND operation on a manual subset.
3. The current record of master record is not valid.

NOTFOUND Exception

A NOTFOUND exception occurs with any of the following:

1. There is no record which satisfies the selection expression for a LOCK or FIND operation.
2. The key value in the record does not match the key of a manual subset.
3. The requested record does not exist in the data set (previously deleted).
4. The current record pointer is undefined.
5. The embedded structure is empty.

NOTLOCKED Exception

A NOTLOCKED exception occurs when a STORE operation is not preceded by a LOCK or another STORE operation.

OPENERROR Exception

An OPENERROR exception occurs with any of the following:

1. A program attempts to open a data base which is not initialized.
2. A program attempts to open a data base which is already open.
3. A program attempts to open a data base, and the data base is not the proper level.
4. The data base is not open prior to the first DMSII operation.

Path

A path is an access to a data set record. A set is an index of paths. The current path pointer associated with every set and subset (but not data set) refers to the last record accessed by way of that set or subset. Each current path pointer retains its reference until explicitly changed or until the record referenced by the current path pointer is deleted from the data base.

READONLY Exception

A READONLY exception occurs when a program opens a data base as input only, and either a STORE, DELET, INSRT, or REMOV operation is attempted.

Record

A record contains all the information that pertains to an entity. A record is considered to be the current record of a data set if the appropriate current record pointer refers to an existing record in the data base. The current record pointer for a data set is changed by any operation which causes a new record to be written to the data base or read from the data base. Changing the current record pointer automatically unlocks any previously locked record and, if required, locks the new one.

SECURITYERROR Exception

A SECURITYERROR exception occurs when a usercode does not satisfy the usercode requirements of the data base.

Set

A set is an index of paths to a data set with a pointer to each record of that data set.

Subset

A subset is an index to records of a data set. If a manual subset exists in a data base, the specified records of the data set to be referenced must be programmatically inserted into the subset. If an automatic subset exists in a data base, the condition is evaluated at each STORE operation, and if satisfied, DMSII does the insertion.

SYSTEMERROR Exception

A SYSTEMERROR exception occurs when too many data bases are open at the same time. The maximum number of data bases which can be open at the same time is six.

VERSIONERROR Exception

A VERSIONERROR exception occurs when a program attempts to open a data base, and the program was compiled against a different version of the data base other than the version currently maintained in the data-base dictionary.

EXAMPLES OF RPG/DMSII PROGRAMS

The following example RPG programs and data bases illustrate the use of the RPG/DMSII interface between the RPG and DMS/DASDL compilers. There are four example data bases and several RPG programs that access each of the data bases.

The data-base source statements which define the data base structures are compiled with the DMS/DASDL compiler by entering the following instruction:

COMPILE <data base name> WITH DMS/DASDL TO LIBRARY

The data-base source statements are in a card file named CARDS.

Example 1a: Data Base Named DB1

Figure 15-18 shows the DMS/DASDL source statements that can be used to compile a data base named DB1. This data base contains all of the following structures:

1. A disjoint data set.
2. An index sequential set.

```
 % THIS DATA BASE ILLUSTRATES A DATA SET
 % WITH AN INDEX SEQUENTIAL SET.

$ LIST$
$ RPG
$ RPGLIB
$ NO COBOL
$ NO COBOLIB
$ LIST

EMPLOY DATA SET
       (EMPNAM                          ALPHA  (20);
        EMPNUM                          NUMBER (S9);
        BIRTH                           NUMBER (S6);
        SALARY                          NUMBER (S8,2);
        JOBTIT                          ALPHA  (20);
      ), MAXRECORDS = 1000;

ISEMP SET OF EMPLOY KEY IS (EMPNAM), INDEX SEQUENTIAL;
----*----1----*----2----*----3----*----4----*----5----*----6--
         0         0         0         0         0         0
```

Figure 15-18. DMS/DASDL Source to Compile DB1 Data Base

The DMS/DASDL compiler dollar ($) options perform the following:

$ LIST$	The $ LIST$ compiler option lists the DMS/DASDL compiler options.
$ RPG	The $ RPG compiler option checks the DMS/DASDL source statements for valid RPG names and lengths.
$ RPGLIB	The $ RPGLIB compiler option causes the DMS/DASDL compiler to build RPG library files on disk for future use by the RPG compiler.
$ NO COBOL	The $ NO COBOL compiler option causes the DMS/DASDL compiler not to check the DMS/DASDL source statement for valid COBOL syntax.
$ NO COBOLIB	The $ NO COBOLIB compiler option causes the DMS/DASDL compiler not to build COBOL library files on disk.
$ LIST	The $ LIST compiler option causes the DMS/DASDL compiler to list the DMS/DASDL source statements.

EMPLOY is the data set name for a group of employee records. The data set contains the item entries EMPNAM, EMPNUM, BIRTH, SALARY, and JOBTIT, which represent the employee's name, number, birth date, salary, and job title, respectively. Each data item has a data type and length associated with it. The ALPHA type represents an alphanumeric field and the NUMBER type represents a numeric field. The data length is enclosed within the parentheses ( ) characters. The letter S within the parentheses for numeric items indicates that this field is signed.

MAXRECORDS = 1000 specifies that the EMPLOY data set can contain a maximum of 1000 records.

ISEMP is the index sequential set which spans the whole EMPLOY data set. The key of ISEMP is EMPNAM.

Example 1b: RPG Program Reading a Data Base

The following RPG source statements comprise an RPG program that reads from the data base named DB1 and generates a printed report of all the employees in DB1.

```
                Control Card Specifications

01 H* THIS PROGRAM ILLUSTRATES AN RPG SOURCE PROGRAM
02 H* WHICH READS FROM THE DATA BASE NAMED DB1.
03 H
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

                Data Base Specifications

04 DDB1                  I
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


            File Description Specifications

05 FEMPLOY  IPE                          DMS    ISEMP
06 FLINE    O   F 132 132                PRINTER
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


               Input Specifications

07 IEMPLOY  NS  01
08 I                                                  EMPNAM
09 I                                                  EMPNUM
10 I                                                  BIRTH
11 I                                                  SALARY
12 I                                                  JOBTIT
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


               Output-Format Specifications

13 OLINE    H  1     1P
14 O                                          47 'EMPLOYEE DATA SET
15 O                               UDATE Y    67
16 O        H  2     1P
17 O                                          21 'NAME'
18 O                                          38 'NUMBER'
19 O                                          50 'BIRTH'
20 O                                          60 'SALARY'
21 O                                          77 'JOB TITLE'
22 O        D  1     01
23 O                               EMPNAM     29
24 O                               EMPNUMZ    40
25 O                               BIRTH Y    51
26 O                               SALARY1    61
27 O                               JOBTIT     84
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Example 1c: RPG Program Updating a Data Base

The following RPG source statements comprise an RPG source program that updates the data base DB1. The program reads from a card file named CARDS and checks the CODE field for the letters A, C, or D. If the CODE field equals the letter A, the program branches to the subroutine called ADD. If the CODE field equals the letter C, the program branches to the subroutine called CHANGE. If the CODE field equals the letter D, the program branches to the subroutine called DELETE. The message BAD-ADD, is displayed on the Operator Display Terminal (ODT) when the entry to be added is present. The messages BAD-CHG and BAD-DEL are displayed on the ODT when the entry to be changed or deleted is not present.

```
            Control Card Specificatons

01 H* THIS PROGRAM ILLUSTRATES AN RPG SOURCE PROGRAM
02 H* WHICH UPDATES THE DATA BASE NAMED DB1.
03 H
--*----1----*----2----*----3----*----4----*----5----*----6----*----7-


           Data Base Specificatons

04 DDB1                   U
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


                File Description Specifications

05 FCARDS   IPE F  80  80            READER
06 FEMPLOY  UD                       DMS
07 FDISPLAY D   F  50  50            CONSOLE
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


            Input Specifications

08 ICARDS   NS  01   1 CA
09 I                                  1   1 CODE
10 I                                  2  21 NAME
11 I                                 22  30ONUMBER
12 I                                 31  36OBIRDAT
13 I                                 37  442SAL
14 I                                 45  64 JOB
15 ICARDS   NS  02   1 CC
16 I                                  1   1 CODE
17 I                                  2  21 NAME
18 I                                 22  30ONUMBER
19 I                                 31  36OBIRDAT
20 I                                 37  442SAL
21 I                                 45  64 JOB
22 ICARDS   NS  03   1 CD
23 I                                  1   1 CODE
24 I                                  2  21 NAME
25 IEMPLOY  NS  04
26 I                                     EMPNAM
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Calculation Specifications

```
27 C   01                EXSR ADD
28 C   02                EXSR CHANGE
29 C   03                EXSR DELETE
30 C* ADD SUBROUTINE
31 CSR          ADD      BEGSR
32 CSR          ISEMP    LOCK EMPLOY              RD1
33 CSR                   DMKEYNAME      EMPNAM     EQL
34 CSR D1NDA    'BAD-ADD' DSPLYDISPLAY
35 CSR D1 DA             STOREEMPLOY               D1
36 CSR                   ENDSR
37 C* CHANGE SUBROUTINE
38 CSR          CHANGE   BEGSR
39 CSR          ISEMP    LOCK EMPLOY              RD1
40 CSR                   DMKEYNAME      EMPNAM     EQL
41 CSR D1       'BAD-CHG' DSPLYDISPLAY
42 CSRND1                STOREEMPLOY               D1
43 CSR                   ENDSR
44 C* DELETE SUBROUTINE
45 CSR          DELETE   BEGSR
46 CSR          ISEMP    LOCK EMPLOY              RD1
47 CSR                   DMKEYNAME      EMPNAM     EQL
48 CSR D1       'BAD-DEL' DSPLYDISPLAY
49 CSRND1                DELETEMPLOY              SD1
50 CSR                   ENDSR
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Output-Format Specifications

```
51 OEMPLOY  SADD     01
52 O          EMPNAM          NAME
53 O          EMPNUM          NUMBER
54 O          BIRTH           BIRDAT
55 O          SALARY          SAL
56 O          JOBTIT          JOB
57 OEMPLOY  S         02
58 O          EMPNAMN10       NAME
59 O          EMPNUMN11       NUMBER
60 O          BIRTH N12       BIRDAT
61 O          SALARYN13       SAL
62 O          JOBTITN14       JOB
--*----1----*----2----*----3----*----4----*----5----*----6--
```

Example 2a: Data Base Named DB2

Figure 15-19 shows the data-base source statements that can be used to compile the data base named DB2. This data base contains all of the following structures:

1. A disjoint data set.
2. An embedded data set.
3. An index sequential set spanning the disjoint data set.
4. An access path to the embedded data set.

```
 % THIS DATA BASE ILLUSTRATES A DATA SET WITH AN EMBEDDED DATA SET

$ LIST$
$ RPG
$ RPGLIB
$ NO COBOL
$ NO COBOLIB
$ LIST

EMPLOY DATA SET
       (EMPNAM                          ALPHA (20);
        EMPNUM                          NUMBER (S9);
        BIRTH                           NUMBER (S6);
        SALARY                          NUMBER (S8,2);
        JOBTIT                          ALPHA (20);
        DEPEND ORDERED DATA SET
               (DEPNAM                  ALPHA (20);
                DEPBIR                  NUMBER (S6);
               ), MAXRECORDS = 4;
        DEPSET ACCESS TO DEPEND KEY IS (DEPBIR) DUPLICATES;
       ), MAXRECORDS = 1000;

ISEMP SET OF EMPLOY KEY IS (EMPNAM), INDEX SEQUENTIAL;

----*----1----*----2----*----3----*----4----*----5----*----6--
         0         0         0         0         0         0
```

Figure 15-19. DMS/DASDL Source to Compile Data Base DB2

Refer to Example 1a for a description of the DMS/DASDL syntax for the EMPLOY data set.

The DEPEND data set is an ordered embedded data set of the EMPLOY data set. The data item DEPNAM represents the name of the employee's dependent and DEPBIR represents the birth date of the employee's dependent.

MAXRECORDS = 4 specifies that the DEPEND data set can contain an average number of four dependents.

DEPSET is an ordered embedded set of the EMPLOY data set that accesses the records of the DEPEND data set. The DEPSET set is ordered by the key DEPBIR, and duplicate birth dates are allowed.

Example 2b: RPG Program Which Loads a Data Base

The following RPG source statements comprise an RPG program that loads records to the EMPLOY data set in the DB2 data base. The EMPLOY data set contains the DEPEND ordered embedded data set. The records reside in a card file named CARDS and are only loaded to the EMPLOY data set.

```
                Control Card Specifications

01 H* THIS PROGRAM LOADS THE DISJOINT DATA SET WHICH HAS AN
02 H* EMBEDDED DATA SET.
03 H
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


                  Data Base Specifications

04 DDB2                   U
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


                File Description Specifications

05 FCARDS   IPE F  80  80            READER
06 FEMPLOY  O                        DMS
---*----1----*----2----*----3----*----4----*----5----*----6--


                  Input Specifications

07 ICARDS   NS  01
08 I                                          1  20 NAME
09 I                                         21  29ONUMBER
10 I                                         30  35OBIRDAT
11 I                                         36  432SAL
12 I                                         44  63 JOB
---*----1----*----2----*----3----*----4----*----5----*----6--


           Output-Format Specifications

13 OEMPLOY  DADD     01
14 O          EMPNAM          NAME
15 O          EMPNUM          NUMBER
16 O          BIRTH           BIRDAT
17 O          SALARY          SAL
18 O          JOBTIT          JOB
---*----1----*----2----*----3----*----4----*----5----*----6--
```

Example 2c: RPG Program Reading a Data Base

The following RPG source statements comprise an RPG program that reads from the data base named DB2 and generates a printed report of all the employee's dependents.

```
                Data Base Specifications

01 DDB2                  I
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

                File Description Specifications

02 FEMPLOY  IPE                          DMS    ISEMP
03 FLINE    O   F 132 132                PRINTER
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

                  Input Specifications

04 IEMPLOY  NS  01
05 I                                                  EMPNAM
06 IDEPEND  NS  02
07 I                                                  DEPNAM
08 I                                                  DEPBIR
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

               Calculation Specifications

09 C            NEXT      TAG
10 C                      FIND DEPEND    NEXT      SD1
11 C  ND1                 EXCPT
12 C  ND1                 GOTO NEXT
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

              Output-Format Specifications

13 OLINE    H  1     1P
14 O                                      48 'DEPENDENTS DATA S
15 O                           UDATE Y    67
16 O        H  2     1P
17 O                                      25 'EMPLOYEE NAME'
18 O                                      48 'DEPENDENT NAME'
19 O                                      61 'BIRTH'
20 O        E  1     02
21 O                           EMPNAM     29
22 O                           DEPNAM     51
23 O                           DEPBIRY    62
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Example 2d: RPG Program Updating a Data Base

The following RPG source statements comprise an RPG source program that updates the data base DB2. The program only updates the DEPEND data set records. Cards are read from a card file named CARDS, and each card is checked in the CODE field for the letters A, C, or D. The message BAD-TRAN is displayed on the ODT if the input record does not have a master employee record in the EMPLOY data set. If the CODE field equals the letter A, the program branches to the subroutine called ADD. If the CODE field equals the letter C, the program branches to the subroutine called CHANGE. If the CODE field equals the letter D, the program branches to the subroutine called DELETE. The message BAD-ADD is displayed on the ODT when the entry to be added is present. The messages BAD-CHG and BAD-DEL are displayed on the ODT when the entry to be changed or deleted is not present.

```
          Control Card Specifications

01 H* THIS PROGRAM ILLUSTRATES AN RPG SOURCE PROGRAM
02 H* WHICH UPDATES AN EMBEDDED DATA SET WITHIN A
03 H* DISJOINT DATA SET.
04 H
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


            Data Base Specifications

05 DDB2                  U
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


              File Description Specifications

06 FCARDS   IPE F  80  80            READER
07 FEMPLOY  UD                       DMS
08 FDISPLAY D   F  50  50            CONSOLE
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


              Input Specifications

09 ICARDS   NS  01   1 CA
10 I                                   1   1 CODE
11 I                                   2  21 NAME
12 I                                  22  27OBIRDAT
13 I                                  28  47 ENAME
14 ICARDS   NS  02   1 CC
15 I                                   1   1 CODE
16 I                                   2  21 NAME              10
17 I                                  22  27OBIRDAT            11
18 I                                  28  47 ENAME
19 ICARDS   NS  03   1 CD
20 I                                   1   1 CODE
21 I                                  22  27OBIRDAT
22 I                                  28  47 ENAME
23 IEMPLOY  NS  04
24 I                                      EMPNAM
25 IDEPEND  NS  05
26 I                                      DEPNAM
27 I                                      DEPBIR
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Calculation Specifications

```
28 C            ISEMP     LOCK EMPLOY                RD1
29 C                      DMKEYENAME      EMPNAM      EQL
30 C   D1       'BAD-TRAN'DSPLYDISPLAY
31 C   O1ND1               EXSR ADD
32 C   O2ND1               EXSR CHANGE
33 C   O3ND1               EXSR DELETE
34 C* ADD SUBROUTINE
35 CSR          ADD       BEGSR
36 CSR          DEPSET    LOCK DEPEND                RD1
37 CSR                    DMKEYBIRDAT     DEPBIR      EQL
38 CSRNDA       'BAD-ADD' DSPLYDISPLAY
39 CSR D1 DA              STOREDEPEND                 D1
40 CSR                    ENDSR
41 C* CHANGE SUBROUTINE
42 CSR          CHANGE    BEGSR
43 CSR          DEPSET    LOCK DEPEND                RD1
44 CSR                    DMKEYBIRDAT     DEPBIR      EQL
45 CSR D1       'BAD-CHG' DSPLYDISPLAY
46 CSRND1                 STOREDEPEND                 D1
47 CSR                    ENDSR
48 C* DELETE SUBROUTINE
49 CSR          DELETE    BEGSR
50 CSR          DEPSET    LOCK DEPEND                RD1
51 CSR                    DMKEYBIRDAT     DEPBIR      EQL
52 CSR D1       'BAD-DEL' DSPLYDISPLAY
53 CSRND1                 DELETDEPEND                SD1
54 CSR                    ENDSR
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Output-Format Specifications

```
55 ODEPEND  SADD     01
56 O          DEPNAM          NAME
57 O          DEPBIR          BIRDAT
58 ODEPEND  S        02
59 O          DEPNAMN10       NAME
60 O          DEPBIRN11       BIRDAT
--*----1----*----2----*----3----*----4----*----5----*----6--
```

Example 3a: Data Base Named DB3

Figure 15-20 shows the DMS/DASDL source statements that can be used to compile the data base named DB3. This data base contains all of the following structures:

1. Two disjoint data sets.
2. An index sequential set for each of the disjoint data sets.
3. An embedded manual set in the first disjoint data set.

```
% THIS DATA BASE ILLUSTRATES A DATA SET WITH A MANUAL SET.

$ LIST$
$ RPG
$ RPGLIB
$ NO COBOL
$ NO COBOLIB
$ LIST

EMPLOY DATA SET
       (EMPNAM                          ALPHA (20);
        EMPNUM                          NUMBER (S9);
        BIRTH                           NUMBER (S6);
        SALARY                          NUMBER (S8,2);
        JOBTIT                          ALPHA (20);
        EMPDEP SUBSET OF DEPEND KEY IS (DEPBIR);
       ), MAXRECORDS = 1000;

DEPEND DATA SET
       (DEPNAM                          ALPHA (20);
        DEPBIR                          NUMBER (S6);
       ), MAXRECORDS = 4000;

ISEMP SET OF EMPLOY KEY IS (EMPNAM), INDEX SEQUENTIAL;
ISDEP SET OF DEPEND KEY IS (DEPBIR), INDEX SEQUENTIAL;

----*----1----*----2----*----3----*----4----*----5----*----6--
         0         0         0         0         0         0
```

Figure 15-20. DMS/DASDL Source for Data Base DB3

Refer to examples 1a and 2a for a description of the DMS/DASDL syntax for the data sets EMPLOY and DEPEND, and for the index sequential sets ISEMP and ISDEP.

EMPDEP is a manual set of the EMPLOY data set which contains records that point to records in the DEPEND data set. DEPBIR is the key used in the EMPDEP manual set.

Example 3b: RPG Program Which Loads a Data Base

The following RPG source statements comprise an RPG program that loads records to the EMPLOY data set in the data base named DB3. The records reside in a card file named CARDS and are only added to the EMPLOY data set.

```
                Control Card Specifications

01 H* THIS PROGRAM LOADS THE DISJOINT DATA SET WHICH HAS
02 H* A MANUAL SET.
03 H
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

              Data Base Specifications

04 DDB3                  U
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

                File Description Specifications

05 FCARDS   IPE F  80  80            READER
06 FEMPLOY  O                        DMS
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

                Input Specifications

07 ICARDS   NS  01
08 I                                          1  20 NAME
09 I                                         21  29 0NUMBER
10 I                                         30  35 0BIRDAT
11 I                                         36  43 2SAL
12 I                                         44  63 JOB
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

               Output-Format Specifications

13 OEMPLOY  DADD     01
14 O          EMPNAM          NAME
15 O          EMPNUM          NUMBER
16 O          BIRTH           BIRDAT
17 O          SALARY          SAL
18 O          JOBTIT          JOB
--*----1----*----2----*----3----*----4----*----5----*----6--
```

Example 3c: RPG Program Reading a Data Base

The following RPG source statements comprise an RPG program that reads records from the EMPLOY data set, finds all the dependent records of the DEPEND data set by means of the EMPDEP manual set, and generates a printed report of the employee's dependents.

```
                Control Card Specifications

01 H* THIS PROGRAM ILLUSTRATES AN RPG SOURCE PROGRAM WHICH R
02 H* ANOTHER DISJOINT DATA SET BY MEANS OF A MANUAL SET.
03 H
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


                Data Base Specifications

04 DDB3                  I
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


                File Description Specifications

05 FEMPLOY  IPE                          DMS     ISEMP
06 FDEPEND  ID                           DMS
07 FLINE    O   F 132 132                PRINTER
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


              Input Specifications

08 IEMPLOY  NS  01
09 I                                                    EMPNAM
10 IDEPEND  NS  02
11 I                                                    DEPNAM
12 I                                                    DEPBIR
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


                Calculation Specifications

13 C            NEXT        TAG
14 C            EMPDEP      FIND DEPEND     NEXT        SD1
15 C  ND1                   EXCPT
16 C  ND1                   GOTO NEXT
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


              Output-Format Specifications

17 OLINE    H  2     1P
18 O                                        48 'DEPENDENTS DATA S
19 O                            UDATE Y     67
20 O        H  2     1P
21 O                                        25 'EMPLOYEE NAME'
22 O                                        48 'DEPENDENT NAME'
23 O                                        59 'BIRTH'
24 O        E  1     02
25 O                            EMPNAM      29
26 O                            DEPNAM      51
27 O                            DEPBIRY     62
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Example 3d: RPG Program Updating a Data Base

The following RPG source statements comprise an RPG source program that updates the data base DB3. The program updates the DEPEND data set records and the EMPDEP manual set. Cards are read from a card file named CARDS and a check is made to determine if the CODE field is equal to the letters A, C, or D. The message BAD-TRAN is displayed on the ODT if the input record does not have an employee record in the EMPLOY data set. If the CODE field equals the letter A, the program branches to the subroutine called ADD. If the CODE field equals the letter C, the program branches to the subroutine called CHANGE. If the CODE field equals the letter D, the program branches to the subroutine called DELETE. The message BAD-ADD is displayed on the ODT when the entry to be added is present. The messages BAD-CHG and BAD-DEL are displayed on the ODT when the entry to be changed or deleted is not present. The messages BAD-INST and BAD-REMV are displayed on the ODT when an exception condition occurs during the insertion or removal of records in the EMPDEP set.

```
                    Control Card Specifications

01 H* THIS PROGRAM ILLUSTRATES AN RPG SOURCE PROGRAM WHICH
02 H* UPDATES A DISJOINT DATA SET AND A MANUAL SET.
03 H
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


                      Data Base Specifications

04 DDB3                    U
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


                  File Description Specifications

05 FCARDS   IPE F  80  80            READER
06 FEMPLOY  UD                       DMS
07 FDEPEND  UD                       DMS
08 FDISPLAY D   F  50  50            CONSOLE
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


                  Input Specifications

09 ICARDS   NS  01   1 CA
10 I                                   1   1 CODE
11 I                                   2  21 NAME
12 I                                  22  27OBIRDAT
13 I                                  28  47 ENAME
14 ICARDS   NS  02   1 CC
15 I                                   1   1 CODE
16 I                                   2  21 NAME
17 I                                  22  27OBIRDAT
18 I                                  28  47 ENAME
19 ICARDS   NS  03   1 CD
20 I                                   1   1 CODE
21 I                                  22  27OBIRDAT
22 I                                  28  47 ENAME
23 IEMPLOY  NS  04
24 I                                      EMPNAM
25 IDEPEND  NS  05
26 I                                      DEPNAM
27 I                                      DEPBIR
--*----1----*----2----*----3----*----4----*----5----*----6--
```

Calculation Specifications

```
28 C            ISEMP     LOCK EMPLOY                RD1
29 C                      DMKEYENAME     EMPNAM       EQL
30 C   D1       'BAD-TRAN'DSPLYDISPLAY
31 C   01ND1               EXSR ADD
32 C   02ND1               EXSR CHANGE
33 C   03ND1               EXSR DELETE
34 C* ADD SUBROUTINE
35 CSR          ADD       BEGSR
36 CSR          ISDEP     LOCK DEPEND                RD1
37 CSR                    DMKEYBIRDAT    DEPBIR       EQL
38 CSRNDA       'BAD-ADD' DSPLYDISPLAY
39 CSR D1 DA               SETON                       80
40 CSR 80                  STOREDEPEND                 D1
41 CSR 80ND1               SETON                       81
42 CSR 81       EMPDEP    INSRTDEPEND                 D1
43 CSR D1       'BAD-INST'DSPLYDISPLAY
44 CSR                     SETOF                       8081
45 CSR                     ENDSR
46 C* CHANGE SUBROUTINE
47 CSR          CHANGE    BEGSR
48 CSR          ISDEP     LOCK DEPEND                RD1
49 CSR                    DMKEYBIRDAT    DEPBIR       EQL
50 CSR D1       'BAD-CHG' DSPLYDISPLAY
51 CSRND1                  STOREDEPEND                 D1
52 CSR                     ENDSR
53 C* DELETE SUBROUTINE
54 CSR          DELETE    BEGSR
55 CSR          EMPDEP    FIND DEPEND                RD1
56 CSR                    DMKEYBIRDAT    DEPBIR       EQL
57 CSR D1       'BAD-DEL' DSPLYDISPLAY
58 CSRND1                  SETON                       80
59 CSR 80       EMPDEP    REMOV                       D1
60 CSR 80 D1    'BAD-REMV'DSPLYDISPLAY
61 CSR 80ND1               DELETDEPEND                SD1
62 CSR                     SETOF                       80
63 CSR                     ENDSR
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Output-Format Specifications

```
64 ODEPEND  SADD     01
65 O           DEPNAM          NAME
66 O           DEPBIR          BIRDAT
67 ODEPEND  S        02
68 O           DEPNAMN10       NAME
69 O           DEPBIRN11       BIRDAT
---*----1----*----2----*----3----*----4----*----5----*----6--
```

Example 4a: Data Base Named DB4

Figure 15-21 shows the DMS/DASDL source statements that can be used to compile the example data base named DB4. This data base is to be audited and contains all of the following structures:

1. A disjoint data set.
2. An index sequential set.
3. A restart data set.

```
 % THIS DATA BASE ILLUSTRATES A DATA SET WITH THE AUDIT OPTION

$ LIST$
$ RPG
$ RPGLIB
$ NO COBOL
$ NO COBOLIB
$ LIST

OPTIONS (AUDIT SET);

PARAMETERS (SYNCPOINT=5, CONTROLPOINT=5);

RSTART RESTART DATA SET
       (RRNUM                          NUMBER (S6);
        PROGID                         ALPHA  (10);
       );

EMPLOY DATA SET
      (EMPNAM                          ALPHA (20);
       EMPNUM                          NUMBER (S9);
       BIRTH                           NUMBER (S6);
       SALARY                          NUMBER (S8,2);
       JOBTIT                          ALPHA (20);
      ), MAXRECORDS = 1000;

ISEMP SET OF EMPLOY KEY IS (EMPNAM), INDEX SEQUENTIAL;

----*----1----*----2----*----3----*----4----*----5----*----6--
         0         0         0         0         0         0
```

Figure 15-21. Data-Base Source to Compile Data Base DB4

Refer to Example 1a for a description of the DMS/DASDL syntax for the data set EMPLOY and the index sequential set ISEMP.

The OPTIONS (AUDIT SET) selection sets the audit option for the DB4 data base.

The PARAMETERS (SYNCPOINT = 5, CONTROLPOINT = 5) option sets the syncpoint to every five transactions and the controlpoint to every five syncpoints.

RSTART is referred to as the restart data set for the DB4 data base. The RSTART restart data set contains the data items RRNUM and PROGID, which represent the relative record number and program identification, respectively.

Example 4b: RPG Program Reading a Data Base

The following RPG source statements comprise an RPG program that reads records from the EMPLOY data set and generates a printed report of the data in the EMPLOY data set.

```
                Control Card Specifications

01 H* THIS PROGRAM ILLUSTRATES AN RPG SOURCE PROGRAM
02 H* WHICH READS A DATA BASE
03 H
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


                Data Base Specifications

04 DDB4                  I
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


                File Description Specifications

05 FEMPLOY  IPE                          DMS    ISEMP
06 FLINE    O   F 132 132                PRINTER
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


                Input Specifications

07 IEMPLOY  NS  01
08 I                                                  EMPNAM
09 I                                                  EMPNUM
10 I                                                  BIRTH
11 I                                                  SALARY
12 I                                                  JOBTIT
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


                Output-Format Specifications

13 OLINE    H  1     1P
14 O                                      47 'EMPLOYEE DATA SET
15 O                           UDATE Y    67
16 O        H  2     1P
17 O                                      21 'NAME'
18 O                                      38 'NUMBER'
19 O                                      50 'BIRTH'
20 O                                      60 'SALARY'
21 O                                      77 'JOB TITLE'
22 O        D  1     01
23 O                           EMPNAM     29
24 O                           EMPNUMZ    40
25 O                           BIRTH Y    51
26 O                           SALARY1    61
27 O                           JOBTIT     84
---*----1----*----2----*----3----*----4----*----5----*----6--
```

Example 4c: RPG Program Updating a Data Base

The following RPG source statements comprise an RPG source program that updates the data base DB4. The program updates the EMPLOY data set records and uses the AUDIT/RECOVERY mechanism in DMS. At beginning of job the program checks for a restart record by branching to the subroutine called RESTAR. If a RSTART restart data set record is present, the last run was incomplete and the program must realign itself with the last good transaction designated by the relative record number RRNUM. To accomplish realignment, the program reads from the card file named CARDS until the number of cards (TRANCT) read is greater than the value stored in the RRNUM field. No processing of the cards occurs. When the value of the TRANCT field is greater than the value of the RRNUM field, normal processing continues. If there is no record present in the RSTART restart data set, the last run was complete and normal processing begins. The message RS-ERROR is displayed on the ODT when a DMSII exception condition other than a NOTFOUND occurs when performing the LOCK operation on the RSTART restart data set.

Cards are read from a card file named CARDS and the CODE field of each card is checked for the letters A, C, or D. If the CODE field equals the letter A, the program branches to the subroutine called ADD. If the CODE field equals the letter C, the program branches to the subroutine called CHANGE. If the CODE field equals the letter D, the program branches to the subroutine called DELETE. The message BAD-ADD is displayed on the ODT when the entry to be added is present. The message BAD-CHG and BAD-DEL are displayed on the ODT when the entry to be changed or deleted is not present.

For each transaction card the record in the RSTART restart data set is updated by the TREND transaction code. If an exception condition occurs on the TREND operation, the program branches to the user routine, called RSTART, which displays on the ODT the DMSII exception indicator that is on.

If the end-of-file record is read for the CARDS file and the LR indicator is on, the program deletes the restart data set record.

```
                Control Card Specifications

01 H* THIS PROGRAM ILLUSTRATES AN RPG SOURCE PROGRAM
02 H* WHICH UPDATES AN AUDITED DATA BASE.
03 H
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


                   Data Base Specifications

04 DDB4                    U
--*----1----*----2----*----3----*----4----*----5----*----6----*----7


                  File Description Specifications

05 FCARDS   IPE F  80  80            READER
06 FEMPLOY  UD                       DMS
07 FRSTART  R                        DMS
08 FDISPLAY D   F  50  50            CONSOLE
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Input Specifications

```
09 ICARDS   NS  01   1 CA
10 I                                          1   1 CODE
11 I                                          2  21 NAME
12 I                                         22  30ONUMBER
13 I                                         31  36OBIRDAT
14 I                                         37  442SAL
15 I                                         45  64 JOB
16 ICARDS   NS  02   1 CC
17 I                                          1   1 CODE
18 I                                          2  21 NAME
19 I                                         22  30ONUMBER
20 I                                         31  36OBIRDAT
21 I                                         37  442SAL
22 I                                         45  64 JOB
23 ICARDS   NS  03   1 CD
24 I                                          1   1 CODE
25 I                                          2  21 NAME
26 IEMPLOY  NS  04
27 I                                             EMPNAM
28 IRSTART  NS  05
29 I                                             RRNUM
30 I                                             PROGID
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Calculation Specifications

```
31 C  N99                EXSR RESTAR
32 C  N99                Z-ADDO          TRANCT
33 C  N99                SETON                      99
34 C   70                SETOF                      70
35 C   01
36 COR 02
37 COR 03                SETON                      70
38 C   70      TRANCT    ADD  1          TRANCT
39 C   70 81   RRNUM     COMP TRANCT                82
40 C   82                GOTO END
41 C   01                EXSR ADD
42 C   02                EXSR CHANGE
43 C   03                EXSR DELETE
44 C             END      TAG
--*----1----*----2----*----3----*----4----*----5----*----6--
```

Calculation Specifications
(continued)

```
45 C* DELETE RECORD IN RSTART RESTART DATA SET
46 CLR                     TRBEG
47 CLR                     DELETRSTART
48 CLR                     TRENDNOAUDIT
49 C* ADD SUBROUTINE
50 CSR           ADD       BEGSR
51 CSR                     TRBEG
52 CSR           ISEMP     LOCK EMPLOY              RD1
53 CSR                     DMKEYNAME      EMPNAM     EQL
54 CSR D1NDA   'BAD-ADD'   DSPLYDISPLAY
55 CSR D1 DA               STOREEMPLOY               D1
56 CSR                     TREND
57 CSR                     ENDSR
58 C* CHANGE SUBROUTINE
59 CSR           CHANGE    BEGSR
60 CSR                     TRBEG
61 CSR           ISEMP     LOCK EMPLOY              RD1
62 CSR                     DMKEYNAME      EMPNAM     EQL
63 CSR D1      'BAD-CHG'   DSPLYDISPLAY
64 CSRND1                  STOREEMPLOY               D1
65 CSR                     TREND
66 CSR                     ENDSR
67 C* DELETE SUBROUTINE
68 CSR           DELETE    BEGSR
69 CSR                     TRBEG
70 CSR           ISEMP     LOCK EMPLOY              RD1
71 CSR                     DMKEYNAME      EMPNAM     EQL
72 CSR D1      'BAD-DEL'   DSPLYDISPLAY
73 CSRND1                  DELETEMPLOY              SD1
74 CSR                     TREND
75 CSR                     ENDSR
76 C* RESTART SUBROUTINE
77 CSR           RESTAR    BEGSR
78 CSR                     LOCK RSTART    FIRST     SD1
79 CSR D1 DA               SETON                     80
80 CSR D1NDA   'RS-ERROR'  DSPLYDISPLAY
81 CSRND1                  SETON                     81
82 CSR 80                  TRBEG
83 CSR 80                  TREND
84 CSR 80                  SETOF                     80
85 CSR                     ENDSR
--*----1----*----2----*----3----*----4----*----5----*----6--
```

Calculation Specifications
(continued)

```
 86 C* RSTART USER ROUTINE - DISPLAY DMS EXCEPTION INDICATOR
 87 CUR                    BEGURRSTART
 88 CUR DA       'DA'      DSPLYDISPLAY
 89 CUR DB       'DB'      DSPLYDISPLAY
 90 CUR DC       'DC'      DSPLYDISPLAY
 91 CUR DD       'DD'      DSPLYDISPLAY
 92 CUR DE       'DE'      DSPLYDISPLAY
 93 CUR DF       'DF'      DSPLYDISPLAY
 94 CUR DG       'DG'      DSPLYDISPLAY
 95 CUR DH       'DH'      DSPLYDISPLAY
 96 CUR DJ       'DJ'      DSPLYDISPLAY
 97 CUR DK       'DK'      DSPLYDISPLAY
 98 CUR DL       'DL'      DSPLYDISPLAY
 99 CUR DM       'DM'      DSPLYDISPLAY
100 CUR DN       'DN'      DSPLYDISPLAY
101 CUR DO       'DO'      DSPLYDISPLAY
102 CUR DP       'DP'      DSPLYDISPLAY
103 CUR DQ       'DQ'      DSPLYDISPLAY
104 CUR DR       'DR'      DSPLYDISPLAY
105 CUR DS       'DS'      DSPLYDISPLAY
106 CUR                    ENDUR
 --*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Output-Format Specifications

```
107 OEMPLOY  SADD     01
108 O          EMPNAM          NAME
109 O          EMPNUM          NUMBER
110 O          BIRTH           BIRDAT
111 O          SALARY          SAL
112 O          JOBTIT          JOB
113 OEMPLOY  S        02
114 O          EMPNAMN10       NAME
115 O          EMPNUMN11       NUMBER
116 O          BIRTH N12       BIRDAT
117 O          SALARYN13       SAL
118 O          JOBTITN14       JOB
119 ORSTART  RADD     80N81
120 O          RRNUM                        +000000
121 O          PROGID                       'DB4UPO'
122 ORSTART  R        70
123 O          RRNUM           TRANCT
---*----1----*----2----*----3----*----4----*----5----*----6--
```

DMSDBUG (RPG/DMSII DEBUGGING AID)

If an RPG/DMSII program contains a printer file named DMSDBUG, all of the internal DMSII messages passed from the program to the operating system (MCPII) during program execution are written to this file. This information can be a useful debugging aid for RPG/DMSII programs.

Each record in the DMSDBUG printer file is divided into two parts: the actual internal DMSII message and the English equivalent to the internal DMSII message. The internal DMSII message is contained in columns 1-87 of the DMSDBUG printer file. These columns are for internal use only. The English equivalent to each internal DMSII message is contained in columns 88-132 of the DMSDBUG printer file and can be used to follow the logic of an RPG/DMSII program.

The format of the English part (columns 88-132) is as follows:

CREATE <data set name>

The CREATE word appears when the next operation to be performed is a store with additions for the data set specified. When the STORE word appears without a previous CREATE word, the STORE word represents an update operation to the data base.

STORE <data set name>

The STORE word appears when a store operation is performed for the data set specified in the FACTOR 2 field.

DELETE <data set name>

The DELETE word appears when an implicit delete operation is performed for the data set.

DELETE <data set name>

The DELETE word appears when a delete operation is explicitly performed for the data set specified in the FACTOR 2 field.

FREE <data set name>

The FREE word appears when a free operation is performed for the data set specified in the FACTOR 2 field.

INS <data set name> <manual subset name>

The INS word appears when an insert operation is performed for the data set specified in the FACTOR 2 field and for the manual subset specified in the FACTOR 1 field.

REM <manual subset name>

The REM word appears when a remove operation is performed for the manual subset specified in the FACTOR 1 field.

TRBEG <restart data set name>

The TRBEG word appears when an explicit begin transaction operation is performed for the restart data set.

TRBEG <restart data set name> CYCLE

The TRBEG with CYCLE words appear when an implicit begin transaction operation is performed for the restart data set.

TREND <restart data set name>

The TREND word appears when an explicit end transaction operation is performed for the restart data set and NOAUDIT is specified in the FACTOR 2 field.

TREND <restart data set name> SY

The TREND with SY words appear when an explicit end transaction operation is performed for the restart data set with NOAUDIT specified in the FACTOR 2 field and SYNCPT specified in the RESULT FIELD field.

TREND <restart data set name> CYCLE

The TREND with CYCLE words appear when an implicit end transaction operation with syncpoint and no audit is performed for the restart data set.

NOTE

All end transaction operations with audit appear as store operations to the restart data set.

FND <data set name> S

The FND with S symbols appear when an explicit sequential find operation is performed for the data set specified in the FACTOR 2 field. No index and no selection expression is specified.

LOK <data set name> S

The LOK with S symbols appear when an explicit sequential lock operation is performed for the data set specified in the FACTOR 2 field. No index and no selection expression is specified.

FND <data set name> <index name> S

The FND with <index name> and S symbols appear when an explicit sequential find operation is performed for the data set specified in the FACTOR 2 field by means of the index specified in the FACTOR 1 field.

LOK <data set name> <index name> S

The LOK with <index name> and S symbols appear when an explicit sequential lock operation is performed for the data set specified in the FACTOR 2 field by means of the index specified in the FACTOR 1 field.

FIND NEXT <data set name>

The FIND NEXT words appear when an implicit find operation is performed for the data set on an input-only DMSII file.

LOCK NEXT <data set name>

The LOCK NEXT words appear when an explicit lock operation is performed for the data set on an update DMSII file.

Random Operations

If an explicit random find or lock operation is performed, an R (random) access indication appears instead of the S (sequential) access indication for the FND <data-set-name> <index-name> and LOK <data-set-name> <index-name> symbols.

The letters F, L, N, or P appear for explicit find or lock operations and specify a FIRST, LAST, NEXT, or PRIOR selection expression, respectively. This selection expression indication appears immediately before the R or S access indication for FND <data-set-name> and LOK <data-set-name> symbols.

DMSII Exceptions

If a DMSII exception condition occurs as the result of a DMSII operation, the type of exception is written to the print file beginning in column 110.

The following are the DMSII exceptions printed:

NOTFOUND
DUPLICATE
DEADLOCK
DATAERROR
NOTLOCKED
KEYCHANGED
SYSTEMERROR
READONLY
IOERROR
LIMITERROR
OPENERROR
CLOSEERROR
NORECORD
INUSE
AUDITERROR
ABORT
SECURITYERROR
VERSIONERROR
FATALERROR
INTEGRITYERROR

APPENDIX A
B 1000 CARD CODES

Figure A-1 shows the zone and digit portions of an 80-column card. The zone for character A is 12, and the digit is 1. The zone for character R is 11, and the digit 9.

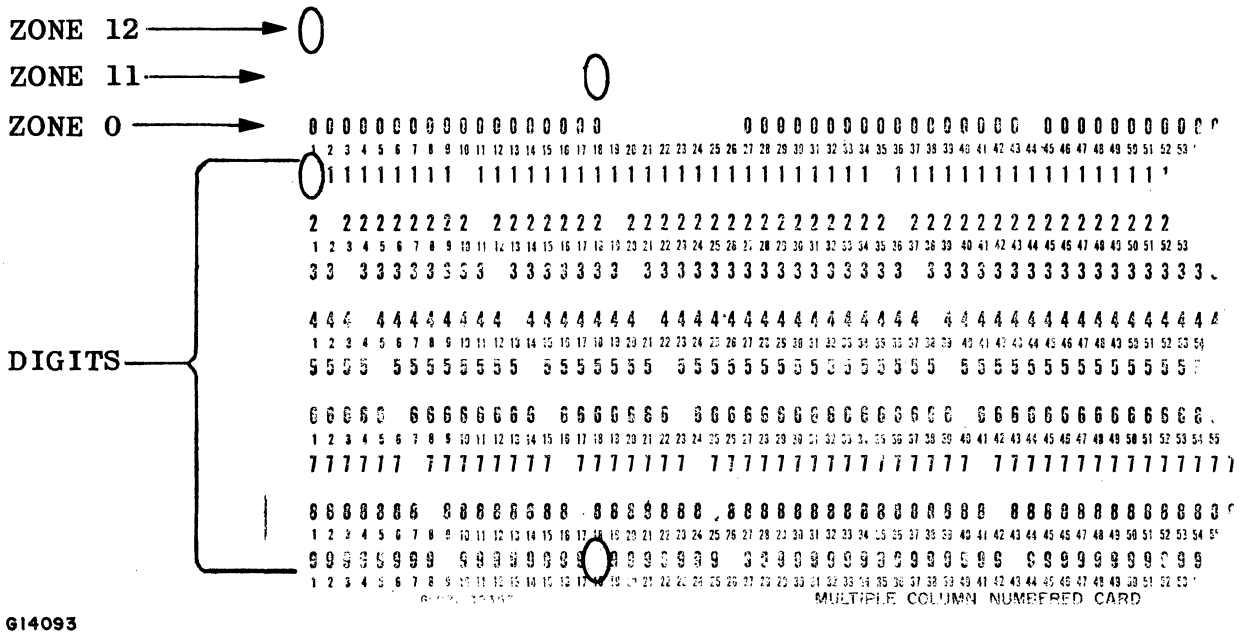


Figure A-1. Zone and Digit Portions of an 80-Column Card

Figure A-2 shows the zone and digit portions of a 96-column card. The zones 12, 11, and 0 on a 96-column card are BA, B, and A, respectively.

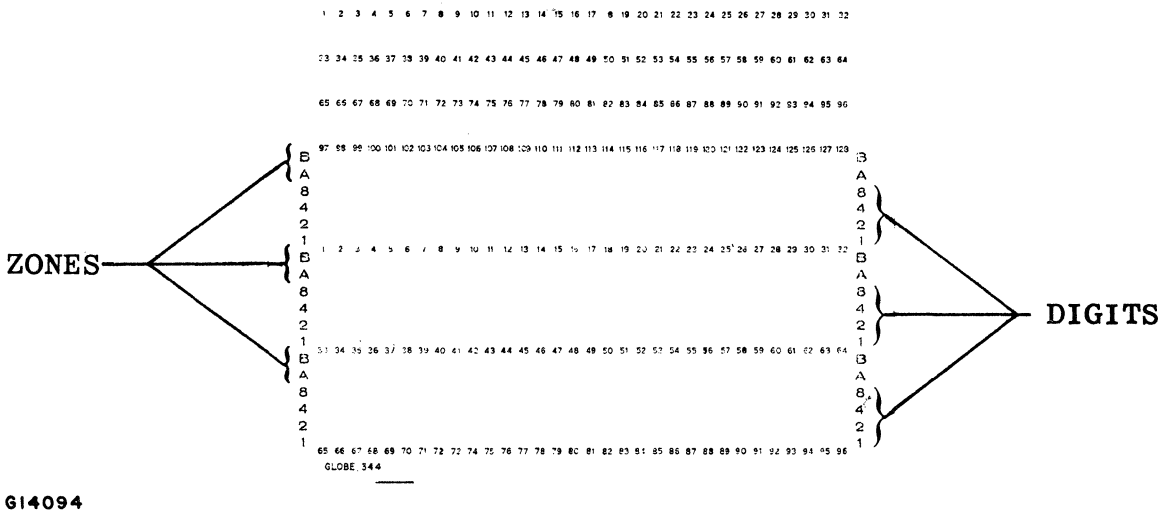


Figure A-2. Zone and Digit Portions of a 96-Column Card

Table A-1 lists the 80-column and 96-column card codes for the complete B 1000 character set.

Any characters that are not included in table A-1 are not punched. No punches are placed in columns for which non-punchable characters are specified. For example, the packed decimal format for a positive zero is @CO@ (the right brace (}) character). This character is not in the character set for the punch codes, and therefore, the column in which the right brace (}) character should be punched is left blank.

Table A-1. Punch Card Codes for the B 1000 Character Set

Character	EBCDIC 80-Column Card Code	BCL 96-Column Card Code	Character	EBCDIC 80-Column Card Code	BCL 96-Column Card Code
blank	no punches	no punches	F	12-6	B-A-4-2
[	12-8-2	B-A-8-2	G	12-7	B-A-4-2-1
.	12-8-3	B-A-8-2-1	H	12-8	B-A-8
<	12-8-4	B-A-8-4	I	12-9	B-A-8-1
(	12-8-5	B-A-8-4-1	J	11-1	B-1
+	12-8-6	B-A-8-4-2	K	ll-2	B-2
¦	12-8-7	B-A-8-4-2-1	L	11-3	B-2-1
&	12	B-A	M	11-4	B-4
]	11-8-2	B-8-2	N	11-5	B-4-1
$	11-8-3	B-8-2-1	O	11-6	B-4-2
*	11-8-4	B-8-4	P	11-7	B-4-2-1
)	11-8-5	B-8-4-1	Q	11-8	B-8
;	11-8-6	B-8-4-2	R	11-9	B-8-1
¬	11-8-7	B-8-4-2-1	S	0-2	A-2
-	11	B	T	0-3	A-2-1
/	0-1	A-1	U	0-4	A-4
,	0-8-3	A-8-2-1	V	0-5	A-4-1
%	0-8-4	A-8-4	W	0-6	A-4-2
_	0-8-5	A-8-4-1	X	0-7	A-4-2-1
>	0-8-6	A-8-4-2	Y	0-8	A-8
?	0-8-7	A-8-4-2-1	Z	0-9	A-8-1
:	8-2	8-2	1	1	1
#	8-3	8-2-1	2	2	2
@	8-4	8-4	3	3	2-1
'	8-5	8-4-1	4	4	4
=	8-6	8-4-2	5	5	4-1
"	8-7	8-4-2-1	6	6	4-2
A	12-1	B-A-1	7	7	4-2-1
B	12-2	B-A-2	8	8	8
C	12-3	B-A-2-1	9	9	8-1
D	12-4	B-A-4	0	0	A
E	12-5	B-A-4-1			

APPENDIX B
HEXADECIMAL VALUES FOR THE B 1000 CHARACTER SET

Table B-1 presents the RPG collating sequence, and table B-2 can 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 1000 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

Ascending Order ↓

Character	Hexadecimal Equivalent
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
U	E4
V	E5
W	E6
X	E7
Y	E8
Z	E9
0	F0
1	F1
2	F2
3	F3
4	F4
5	F5
6	F6
7	F7
8	F8
9	F9

Ascending Order ↓

Table B-2. Hex to Binary Conversion Table

Hex Digit	Binary Equivalent
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

APPENDIX C
BURROUGHS INDICATOR SUMMARY FORM

The RPG Indicator Summary Form illustrated in figure C-1 is used strictly for documentation purposes, to provide an accurate record of the indicators that are used and the function of those indicators in the RPG Program.

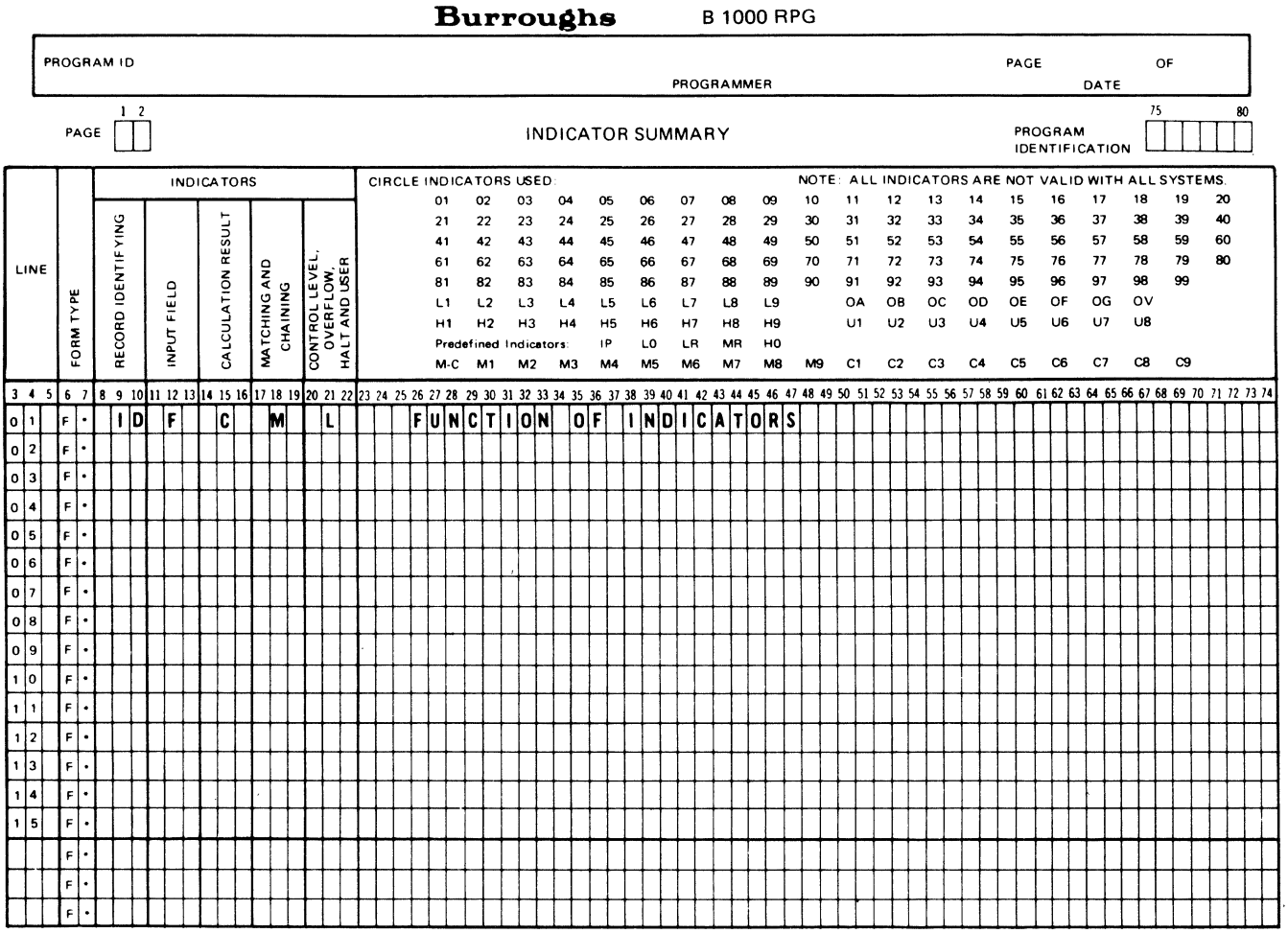


Figure C-1. Burroughs Indicator Summary Form

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.
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
RPG DEBUGGING AIDS

This appendix describes the debugging aids available with the RPG compiler. These include the following:

- Using the DEBUG operation code.
- Using the compiler-directing options.
- Reading an RPG memory dump.
- Getting around compiler aborts.

Each is described in the following paragraphs.

USING THE DEBUG OPERATION CODE

The DEBUG operation code is a special-purpose function which simplifies the location and correction of errors in an RPG program.

This operation causes records to be written during the calculation-processing time. These records contain specific information which can be helpful in locating errors in the program. Each time the DEBUG operation code is performed, one or more fixed-format records are written to an output device. One record contains a list of all indicators which are ON at the time the DEBUG operation code is specified; the other record(s) shows the contents of any one field.

The DEBUG operation code is compiled into the object program only if the DEBUG field (column 15) of the Control Card Specifications contains the number 1. All DEBUG operation code specifications are checked for the correct syntax during the compilation process.

The FACTOR 2 field of the DEBUG operation code must contain a sequential output file. All DEBUG operation code output must go to the same file.

The FACTOR 1 field is optional and can contain a literal or field name to identify the particular DEBUG operation code being performed.

The 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 record.

The DEBUG operation code can be conditioned by indicators in the INDICATORS field (columns 7 through 17) of the Calculation Specifications. Columns 49 through 59 must be blank. The RESULT FIELD field is optional, but if specified, it must be a field vector, or indexed vector whose contents are written on the second and subsequent output records.

DEBUG Operation Output

One or more records are written as output from every DEBUG operation code. The first record is always written; the second and subsequent records are written only if the RESULT FIELD field contains an entry.

Figure D-1 shows the source program, the IN data file, and the DEBUG output listing.

Source Program

```
            Control Card Specifications
01 H          1
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

                Data Base Specifications

02 FIN       IPE  1800  90             DISK
03 FLINE     O     132 132             PRINTER
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

                Extension Specifications

04 E                      ARY          5  6 2
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

                Input Specifications

05 IIN       NS  01
06 I                                              1  90 RECORD
07 I                                              1   101X
08 I                                              2   7 ARDATA
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

                Calculation Specifications
09 C                      MOVE ARDATA     ARY,IX
10 C            '1'       DEBUGLINE       ARY
11 C                      SETON                          102030
12 C            '2'       DEBUGLINE       ARY
13 C                      SETOF                          30
14 C            '3'       DEBUGLINE
15 C                      SETOF                          1020
16 C            '4'       DEBUGLINE
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

                Output-Format Specifications

17 OLINE     D         01
18 O                             RECORD    90
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

IN Data File

1AAAAAA
2BBBBBB
3CCCCCC
4DDDDDD
5EEEEEE
----*----1----*----2----*----3----*----4----*----5----*----6----*----7-
```

Figure D-1. DEBUG Operation Output (Sheet 1 of 3)

DEBUG Output Listing

```
 DEBUG- 1         INDICATORS ON-01
 ARY    ,0001  111111
 ARY    ,0002  000000
 ARY    ,0003  000000
 ARY    ,0004  000000
 ARY    ,0005  000000
 DEBUG- 2         INDICATORS ON-01 10 20 30
 ARY    ,0001  111111
 ARY    ,0002  000000
 ARY    ,0003  000000
 ARY    ,0004  000000
 ARY    ,0005  000000
 DEBUG- 3         INDICATORS ON-01 10 20
 DEBUG- 4         INDICATORS ON-01
1AAAAAA
 DEBUG- 1         INDICATORS ON-01
 ARY    ,0001  111111
 ARY    ,0002  222222
 ARY    ,0003  000000
 ARY    ,0004  000000
 ARY    ,0005  000000
 DEBUG- 2         INDICATORS ON-01 10 20 30
 ARY    ,0001  111111
 ARY    ,0002  222222
 ARY    ,0003  000000
 ARY    ,0004  000000
 ARY    ,0005  000000
 DEBUG- 3         INDICATORS ON-01 10 20
 DEBUG- 4         INDICATORS ON-01
2BBBBBB
 DEBUG- 1         INDICATORS ON-01
 ARY    ,0001  111111
 ARY    ,0002  222222
 ARY    ,0003  333333
 ARY    ,0004  000000
 ARY    ,0005  000000
 DEBUG- 2         INDICATORS ON-01 10 20 30
 ARY    ,0001  111111
 ARY    ,0002  222222
 ARY    ,0003  333333
 ARY    ,0004  000000
 ARY    ,0005  000000
 DEBUG- 3         INDICATORS ON-01 10 20
 DEBUG- 4         INDICATORS ON-01
```

Figure D-1. DEBUG Operation Output (Sheet 2 of 3)

```
3CCCCCC
   DEBUG- 1 INDICATORS ON-01
   ARY ,0001 111111
   ARY ,0002 222222
   ARY ,0003 333333
   ARY ,0004 444444
   ARY ,0005 000000
   DEBUG- 2 INDICATORS ON-01 10 20 30
   ARY ,0001 111111
   ARY ,0002 222222
   ARY ,0003 333333
   ARY ,0004 444444
   ARY ,0005 000000
   DEBUG- 3 INDICATORS ON-01 10 20
   DEBUG- 4 INDICATORS ON-01 4DDDDDD
   DEBUG- 1 INDICATORS ON-01
   ARY ,0001 111111
   ARY ,0002 222222
   ARY ,0003 333333
   ARY ,0004 444444
   ARY ,0005 555555
   DEBUG- 2 INDICATORS ON-01 10 20 30
   ARY ,0001 111111
   ARY ,0002 222222
   ARY ,0003 333333
   ARY ,0004 444444
   ARY ,0005 555555
   DEBUG- 3 INDICATORS ON-01 10 20
   DEBUG- 4 INDICATORS ON-01 5EEEEEE
```

Figure D-1. DEBUG Operation Output (Sheet 3 of 3)

USING THE COMPILER-DIRECTING OPTIONS

The following paragraphs describe the use of the NAME, XREF, MAP, PARMAP, LOGIC, and XMAP compiler-directing options for debugging.

RPG Source Program

Figure D-2 shows an RPG source program has specified the NAMES, XREF, MAP, PARMAP, LOGIC, and XMAP compiler-directing options used in the discussion of each compiler-directing option and the INVALID SUBCRIPT program abort,

```
01  $ NAMES
02  $ XREF
03  $ MAP
04  $ PARMAP
05  $ LOGIC
06  $ XMAP

07 H
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

               File Description Specifications

08 FIN      IPE  1800  90            DISK         U
09 FLINE    O         132            PRINTER
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

               Extension Specifications

10 E                    ARY         5  6 2
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

               Input Specifications

11 IIN      NS  01
12 I                                          1  90 RECORD
13 I                                          1   101X
14 I                                          2   7 ARDATA
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

               Calculation Specifications

15 C                    MOVE ARDATA    ARY,IX
16 C                    SETON                      102030
17 C                    SETOF                      30
--*----1----*----2----*----3----*----4----*----5----*----6----*----7

               Output-Format Specifications

18 OLINE    D        01
19 O                          RECORD    90
20 O                          ARY      100
21 O                          IX    X  110
--*----1----*----2----*----3----*----4----*----5----*----6----*----7
```

Figure D-2. Source Program with Compiler-Directing Options

NAMES Compiler-Directing Option

The NAMES compiler-directing option generates additional information for the programmer. This information includes the following:

- The indicators that are defined and referenced.
- The indicators that are defined and not referenced.
- Information about the files specified.
- Information about the fields specified.

Figure D-3 shows an example RPG source program with the NAMES compiler-directing option specified and the portion of the information generated by the RPG compiler output listing.

```
INDICATORS   -DEFINED AND REFERENCED-

01

INDICATORS   -DEFINED NOT REFERENCED-

10 20 30

FILES

IN      INPUT    PRIMARY   DISK    SERIAL            RECORD LENGTH 0090
LINE    OUTPUT             PRINTER SERIAL            RECORD LENGTH 0132

FIELDS

FIELD ENTRIES USED      16

 .DMSW TYPE IS USER DEFINED   LENGTH 001  NUMERIC  0 DECIMALS REFERENCED
 .DMIX TYPE IS USER DEFINED   LENGTH 004  NUMERIC  0 DECIMALS REFERENCED
ARY    TYPE IS ARRAY/VECTOR   LENGTH 006  NUMERIC  2 DECIMALS REFERENCED
     *** VECTOR IS UNORDERED  WITH 0005 ENTRIES
RECORD TYPE IS USER DEFINED   LENGTH 090  ALPHANUMERIC REFERENCED
IX     TYPE IS USER DEFINED   LENGTH 001  NUMERIC  0 DECIMALS REFERENCED
ARDATA TYPE IS USER DEFINED   LENGTH 006  ALPHANUMERIC REFERENCED
```

Figure D-3. Output from NAMES Compiler-Directing Option

XREF Compiler-Directing Option

The XREF compiler-directing option lists all the specified indicators and field names, and the source record line that they are referenced. Figure D-4 illustrates the output from the XREF compiler-directing option.

```
SYMBOL    TYPE     CROSS REFERENCE STATEMENT NUMBER

          B = BRANCH POINT
          F = FILE
          I = INDICATOR
          V = VARIABLE

ARDATA    V        8     9
ARY       V        4     9    14
IN        F        2     5
IX        V        7     9    15
LINE      F        3    12
RECORD    V        6    13
01        I        5    12
10        I       10
20        I       10
30        I       10    11
```

Figure D-4. Output from XREF Compiler-Directing Option

MAP Compiler-Directing Option

The MAP compiler-directing option lists following information:

- Container sizes used internally in the RPG program.
- File information generated by the RPG compiler.
- COP (Current Operand) Table. This is very useful when reading RPG memory dumps.
- Static Memory generated by the RPG compiler.
- Code Segment generated by the RPG compiler.

Figure D-5 shows the listing generated by the RPG compiler when the MAP compiler-directing option is specified.

```
CONTAINER SIZES -

LENB -OPERAND- EQUALS 8
DISPB -DATA ADDR- EQUALS 11
BDISPB -CODE ADDR- EQUALS 11
COPB -COP WIDTH- EQUALS 23
COPXB -COP INDEX- EQUALS 6


F I L E   I N F O R M A T I O N
------------------------------


FILE NUMBER  1  INTERNAL NAME - IN
                FAMILY NAME -   IN
                FILE ID -
                DEVICE -        DISK
                RECORD SIZE -   90 BYTES
                BLOCK SIZE -    1800 BYTES
                # OF AREAS -    25
                # OF BUFFERS -  1

FILE NUMBER  2  INTERNAL NAME - LINE
                FAMILY NAME -   LINE
                FILE ID -
                DEVICE -        PRINTER
                RECORD SIZE -   132 BYTES
                BLOCK SIZE -    132 BYTES
                # OF BUFFERS -  1

FILE SPACE FOR EACH FILE COMPUTED AS THE SUM OF

          FILE BLOCK SIZE -IN BITS- * # OF BUFFERS
          DESC.SIZE OF 248 BITS PLUS 24 TIMES # OF BUFFERS
          FILE INFORMATION BLOCK OF 930 BITS
          DISK FILE HEADER -VARIES WITH # OF AREAS- FOR DISK FILES

          FILE  1 -  17042 BITS
          FILE  2 -   2258 BITS
                     -----
** TOTAL FILE SPACE   2413 BYTES **
```

Figure D-5. Output from the MAP Compiler-Directing Option (Sheet 1 of 3)

```
C O P   T A B L E
-----------------

LABEL       COP INDEX   BASE-REL ADDRESS   DSEGO ADDR (DIGITS)   BASE-REL ADDRESS   DATA LENGTH
-----       ---------   ----------------   -------------------   ----------------   -----------

DUMMY           0             41
FX              1             64                  128                 1312          3 DIGIT
SK              2             87                  131                 1324          3 DIGIT
SP              3            110                  134                 1336          3 DIGIT
FN              4            133                  137                 1348          2 DIGIT
OS              5            156                  139                 1356          1 DIGIT
1C              6            179                  140                 1360          1 DIGIT
CM              7            202                  141                 1364         15 BYTES
TRASH AREA -COMMUNICATE AREA- AT DADDR 141, DIGIT LENGTH IS 180
OFV IN          8            225                   30                  920          2 DIGIT
SAVED           9            248                   41                  964          2 DIGIT
SKIP-L         10            271                  321                 2084          3 DIGIT
FORCE.         11            294                  324                 2096          2 DIGIT
SAVE.F         12            317                  326                 2104          2 DIGIT
SUBSCR         13            340                  328                 2112          8 DIGIT
SUBSCR         14            363                  336                 2144          8 DIGIT
DMS.EX         15            386                   64                 1056          6 DIGIT
IN             16            409                  344                 2176         90 BYTES
 RN            17            432                  524                 2896          3 DIGIT
LINE           18            455                  527                 2908        132 BYTES
 RN            19            478                  791                 3964          3 DIGIT
 SK            20            501                  794                 3976          3 DIGIT
 ..DMSW        21            524                  797                 3988          1 DIGIT
 ..DMIX        22            547                  799                 3996          4 DIGIT
ARY            23            570                  804                 4016          6 DIGIT
ARY            24            593                  811                 4044          6 DIGIT
          OCCURS 0005
-V.INX         26            639                  846                 4184          4 DIGIT
-V.ARG         27            662                  850                 4200          6 DIGIT
-V.STR         28            685                  857                 4228          4 DIGIT
-V.END         29            708                  861                 4244          4 DIGIT
RECORD         30            731                  865                 4260         90 BYTES
IX             31            754                 1045                 4980          1 DIGIT
ARDATA         32            777                 1047                 4988          6 BYTES

* PLUS 1 DIGIT FOR SIGN

S T A T I C   M E M O R Y
-------------------------

                DATA SPACE -     4236 BITS
                STACK SIZE -      313 BITS
                                 -----
          DATA SEGMENT ZERO -    4549 BITS
            EDIT TABLE SIZE -      41 BITS
             COP TABLE SIZE -     759 BITS
                                 -----
       BASE TO LIMIT SPACE -      669 BYTES
     RUN STRUCTURE NUCLEUS -      263 BYTES
           DATA DICTIONARY -       40 BYTES
                                 -----
**NON-OVERLAYABLE MEMORY -        972 BYTES **
```

Figure D-5. Output from the MAP Compiler-Directing Option (Sheet 2 of 3)

```
C O D E   S P A C E
-------------------

        CODE SEG   0 SIZE -   179 BYTES
        CODE SEG   1 SIZE -   115 BYTES
        CODE SEG   2 SIZE -    43 BYTES
        CODE SEG   3 SIZE -    41 BYTES
        CODE SEG   4 SIZE -    25 BYTES
        CODE SEG   5 SIZE -    16 BYTES
        CODE SEG   6 SIZE -    13 BYTES
        CODE SEG   7 SIZE -    12 BYTES
        CODE SEG   8 SIZE -     0 BYTES
        CODE SEG   9 SIZE -   179 BYTES
        CODE SEG  10 SIZE -   229 BYTES
        CODE SEG  11 SIZE -    34 BYTES
        CODE SEG  12 SIZE -     8 BYTES
        CODE SEG  13 SIZE -    77 BYTES
        CODE SEG  14 SIZE -    16 BYTES
                           -----
CUMULATIVE CODE SEG SIZE -   987 BYTES

                LARGEST CODE SEGMENT -   229 BYTES
                     CODE DICTIONARY -   150 BYTES
                                        -----
** MINIMUM CODE SPACE REQUIRED TO RUN -   379 BYTES **
```

Figure D-5. Output from the MAP Compiler-Directing Option (Sheet 3 of 3)

PARMAP Compiler-Directing Option

The PARMAP compiler-directing option causes the RPG compiler to list the intermediate code that is generated for the RPG program. It lists logical paragraphs, the RPG statement that invokes the logical paragraph, and the code segment and displacement of the logical paragraph.

Figure D-6 shows an example listing of the information generated by the PARMAP compiler-directing option.

```
P A R A G R A P H   M A P
-------------------------

LOGICAL                    RPG   FIRST        LAST
PARAGRAPH                  STMT  SEG DISP     SEG DISP

                                 00  00000 -  00  00005
SETUP-CYCLE
                                 00  00006 -  00  00030
READ-VECTORS
                                 00  00031 -  00  00049
OPEN-FILES
                                 00  00050 -  00  00074
CLOSE-FILES
                                 00  00075 -  00  00366
IN-OUT
                                 00  00367 -  00  00486
                           0003
                                 00  00487 -  00  00766
                           0004
                                 00  00767 -  00  00812
```

Figure D-6. Output from PARMAP Compiler-Directing Option (Sheet 1 of 4)

LOGICAL PARAGRAPH	RPG STMT	FIRST SEG	FIRST DISP		LAST SEG	LAST DISP
LAST-FILE		00	00813	-	00	00852
NEXT-FILE		00	00853	-	00	00889
OUTPUT-DETAIL		00	00890	-	00	00908
READ-VECTORS		04	00000	-	04	00130
	0005	04	00131	-	04	00152
CLOSE-FILES		04	00153	-	04	00177
OPEN-FILES		04	00178	-	04	00196
INPUT-MOVE		02	00000	-	02	00130
	0006	02	00131	-	02	00172
	0007	02	00173	-	02	00214
	0008	02	00215	-	02	00256
	0009	02	00257	-	02	00299
BACK-FROM-IM		02	00300	-	02	00318
	0010					
SETUP-CYCLE		00	00909	-	00	01068
PSS.2		00	01069	-	00	01093
BACK-FROM-OD						
BACK.FROM.OD.BEYOND.ENDTR		01	00000	-	01	00029
FORCE-EOJ-SOON		01	00030	-	01	00067
IN-OUT		01	00068	-	01	00120
LAST-FILE		01	00121	-	01	00145
NEXT-FILE		01	00146	-	01	00164
CHECK-HALTS		01	00165	-	01	00170
TURNOFF-INDS		01	00171	-	01	00189
SPO-OUT		03	00000	-	03	00161
SPO-IN		03	00162	-	03	00323
TURNOFF-INDS		01	00190	-	01	00373
INPUT-CHECK		01	00374	-	01	00392
MAYBE-TOT-CALCS		01	00393	-	01	00407
MAYBE-OFL-OUTPUT						

Figure D-6. Output from PARMAP Compiler-Directing Option (Sheet 2 of 4)

LOGICAL PARAGRAPH	RPG STMT	FIRST SEG	FIRST DISP		LAST SEG	LAST DISP
		01	00408	-	01	00444
OUTPUT-TOTAL		01	00445	-	01	00463
MAYBE-OFL-OUTPUT		01	00464	-	01	00549
CLOSE-FILES		01	00550	-	01	00656
PSS.2		01	00657	-	01	00675
MAKE-DATA-AVAIL		01	00676	-	01	00694
INPUT-MOVE		01	00695	-	01	00713
BACK-FROM-IM		02	00319	-	02	00324
CALC-DETAIL		02	00325	-	02	00343
FORCE-EOJ-SOON		00	01094	-	00	01175
OUTPUT-DETAIL		00	01176	-	00	01407
MAYBE-TOT-CALCS		00	01408	-	00	01426
	0002					
IN-OUT		05	00000	-	05	00062
PIO.1		09	00000	-	09	00599
SPO-OUT		09	00600	-	09	00624
SPO-IN		09	00625	-	09	00719
FORCE-EOJ-SOON		09	00720	-	09	00798
IN-OUT		09	00799	-	09	01425
	0003					
PIO.2		05	00063	-	05	00125
PSS.2		10	00000	-	10	00415
		10	00416	-	10	00440
	0004					
LAST-FILE		01	00714	-	01	00753
FORCE-EOJ-SOON		01	00754	-	01	00778
NEXT-FILE		01	00779	-	01	00918
	0005					
PSS.2		10	00441	-	10	01825
INPUT-CHECK		06	00000	-	06	00037
	0006					
	0007	06	00038	-	06	00052
		06	00053	-	06	00080

Figure D-6. Output from PARMAP Compiler-Directing Option (Sheet 3 of 4)

```
LOGICAL                  RPG   FIRST        LAST
PARAGRAPH                STMT  SEG DISP     SEG DISP

MAYBE-TOT-CALCS
                               06  00081 -  06  00099
CALC-DETAIL
                               11  00000 -  11  00185
                         0010
                               11  00186 -  11  00218
                         0011
                               11  00219 -  11  00233
                         0012
                               11  00234 -  11  00251
OUTPUT-DETAIL
                               11  00252 -  11  00270
CALC-TOTAL
                               12  00000 -  12  00017
OUTPUT-TOTAL
                               12  00018 -  12  00036
                               13  00000 -  13  00024
OUTPUT-DETAIL
                               13  00025 -  13  00058
                         0013
                               13  00059 -  13  00140
                         0014
                               13  00141 -  13  00284
                         0015
                               13  00285 -  13  00566
PIO.2
                               13  00567 -  13  00610
                               14  00000 -  14  00077
OUTPUT-DETAIL
                               14  00078 -  14  00102
MAYBE-OFL-OUTPUT
                               14  00103 -  14  00121
                               07  00000 -  07  00008
MAKE-DATA-AVAIL
                               07  00009 -  07  00073
BACK-FROM-OD
                               07  00074 -  07  00092
```

Figure D-6. Output from PARMAP Compiler-Directing Option (Sheet 4 of 4)

LOGIC Compiler-Directing Option

The LOGIC compiler-directing option lists the intermediate code generated by the RPG compiler. It can be useful in identifying RPG compiler aborts because RPG source records are included in the listing. If the RPG compiler aborts as it is generating the LOGIC information, visually scanning the LOGIC information generated can identify which RPG source record was being compiled. In general, the RPG source record is in error, and should be corrected prior to recompiling. In any case, the Burroughs Systems Representative should be contacted in order to report this error.

Figure D-7 shows the output generated by the LOGIC compiler-directing option. Note that the RPG source records are embedded in this listing.

```
                              00200H
                              00300FIN       IPE  1800  90              D 0
            NTR    PAR SETUP-CYCLE
            BUN    PAR READ-VECTORS
PAR OPEN-FILES     DEFINED
            BUN    BADDR 3-0001
PAR CLOSE-FILES     DEFINED
            CZOS    RELATION NE    BADDR 0-0001
            MVA    AL 06(??)    INTRINSIC CM00
            COMM    INTRINSIC CM00
BADDR 0-0001    DEFINED
            BUN    BADDR 3-0002
BADDR 3-0001    DEFINED
            BUN    BADDR 7-0001
BADDR 8-0001    DEFINED
            SET1    IND=UNUSED
            MVA    AL 06(??????)    INTRINSIC CM00
            COMM    INTRINSIC CM00
            XIT
BADDR 7-0001    DEFINED
            MNFX    NL 01(1)
            SET1    IND=LOADING-PRIMARY-OR-SECONDARY
            NTR    PAR IN-OUT
            BIBT    IND=EOF-1    BADDR 6-0001
            SET1    IND=SECONDARY= 1(RECORD ALREADY IN WORK AREA)
BADDR 6-0001    DEFINED
            CLR1    IND=LOADING-PRIMARY-OR-SECONDARY
            MVZ    INTRINSIC FX00
            BUN    BADDR 3-0003
BADDR 3-0002    DEFINED
                              00400FLINE     0          132             P 0
            CZOS    RELATION NE    BADDR 0-0002
            MVA    AL 06(???)    INTRINSIC CM00
            COMM    INTRINSIC CM00
 BADDR 0-0002    DEFINED
             BUN    BADDR 3-0004
BADDR 3-0003    DEFINED
            BUN    BADDR 7-0002
BADDR 8-0002    DEFINED
            SET1    IND=STATUS-REG-FORMAT-IS-BINARY
            MVA    AL 06(??????)    INTRINSIC CM00
             COMM    INTRINSIC CM00
             XIT
 BADDR 7-0002    DEFINED
             BUN    BADDR 3-0005
 BADDR 3-0004    DEFINED
                              00500E                      ARY          5
             BUN    BADDR 3-0006
 BADDR 3-0005    DEFINED
```

Figure D-7. Output from LOGIC Compiler-Directing Option (Sheet 1 of 8)

```
            SET1    IND=LOADING-PRIMARY-OR-SECONDARY
            NTR     PAR LAST-FILE
            CLR1    IND=LOADING-PRIMARY-OR-SECONDARY
            NTR     PAR NEXT-FILE
            MVZ     INTRINSIC 0S00
            BUN     PAR OUTPUT-DETAIL
 OVERLAY 04    DEFINED
 PAR READ-VECTORS    DEFINED
            BUN     BADDR 3-0007
BADDR 3-0006    DEFINED
           CZOS     RELATION EQ    BADDR 0-0003
           MVA    AL 06(??)    INTRINSIC CM00
           COMM     INTRINSIC CM00
BADDR 0-0003    DEFINED


                                00600111N       NS  01
           MNOS     NL 01(5)
           BUN     PAR CLOSE-FILES
BADDR 3-0007    DEFINED
           BUN     PAR OPEN-FILES


OVERLAY 02    DEFINED
PAR INPUT-MOVE    DEFINED
           CNFX     NL 01(1)    RELATION NE    BADDR 3-0008
           CMPZ     INTRINSIC RN01    RELATION EQ    BADDR 4-0001
           CLR1     IND=SECONDARY= 1(RECORD ALREADY IN WORK AREA)
BADDR 4-0001    DEFINED
           CMPN     NL 01(1)    INTRINSIC RN01    RELATION NE    BADDR 2-00 0
                                007001
           MVA     FILE 0001    REDEFINED UA(00000000005A)    FIELD 0014    0
                                008001
           MVN     FILE 0001    REDEFINED SA(000000000001)    FIELD 0015    0
                                009001
           MVA     FILE 0001    REDEFINED UA(000002000006)    FIELD 0016    0
                                01000C                        MOVE ARDATA
           BUN     BADDR 6-0002
BADDR 2-0001    DEFINED
BADDR 3-0008    DEFINED
BADDR 6-0002    DEFINED
           MVZ     INTRINSIC 1C00
           BUN     PAR BACK-FROM-IM


------040004070016 0F0001000000000060700120F0000000008000006070015 0F00 0
------040004070016 0F0000000000A0000010700120F0000000007000001070015 0F00 0
                                01100C                        SETON
------040028 0B000A 0B00140B001E
                                01200C                        SETOF
------040021 0B001E
                                013000LINE     D         01
                                014000                            RECORD     0
                                015000                            ARY
                                016000                            IX     X    0
```

Figure D-7. Output from LOGIC Compiler-Directing Option (Sheet 2 of 8)

```
OVERLAY 00   DEFINED
PAR SETUP-CYCLE   DEFINED
                              00200H
            MVN   NL 01(1)    INTRINSIC 1C00
            SET1   IND=LOADING-PRIMARY-OR-SECONDARY
            MVN   NL 01(1)    INTRINSIC SP00
            MVZ   INTRINSIC SKIP-LINE00
            MVZ   INTRINSIC SK00
            SET1   IND=1P
            MVN   NL 01(1)    INTRINSIC RN02
            MVN   NL 01(6)    INTRINSIC SK00
            MVZ   INTRINSIC SP00
            NTR   PAR PSS.2
            XIT
OVERLAY 01   DEFINED
PAR BACK-FROM-OD   DEFINED
PAR BACK.FROM.OD.BEYOND.ENDTR   DEFINED
            CLR1   IND=1P
            BIBT   IND=LR   PAR FORCE-EOJ-SOON
            MNFX   INTRINSIC FN00
            NTR   PAR IN-OUT
            CZFX   RELATION NE   BADDR 0-0004
            NTR   PAR LAST-FILE
            NTR   PAR NEXT-FILE
BADDR 0-0004   DEFINED
PAR CHECK-HALTS   DEFINED
            BUN   PAR TURNOFF-INDS
OVERLAY 03   DEFINED
PAR SPO-OUT   DEFINED
            MVA   AL 09( X 0)    INTRINSIC CM00
            MAKP   TRASH0000   REDEFINED UA(00001E00004B)   TRASH0000   0
            COMM   INTRINSIC CM00
            XIT
OVERLAY 03   DEFINED
PAR SPO-IN   DEFINED
            MVA   AL 09( Q 0)    INTRINSIC CM00
            MAKP   TRASH0000   REDEFINED UA(00001E00004B)   TRASH0000   0
            COMM   INTRINSIC CM00
            XIT
OVERLAY 01   DEFINED
PAR TURNOFF-INDS   DEFINED
            SET1   IND=L0
            CBUM    BIT STRING=144 BITS(BFFFFFFFFFFFFFFFFFFFFFFFF803FFF 0
            MNFX   INTRINSIC FN00
            BUN   PAR INPUT-CHECK
PAR MAYBE-TOT-CALCS   DEFINED
            CMPZ   INTRINSIC 1C00   RELATION NE   PAR MAYBE-OFL-OUTPUT   0
            MVZ   INTRINSIC 0S00
            BUN   PAR OUTPUT-TOTAL
PAR MAYBE-OFL-OUTPUT   DEFINED
            BIBT   IND=********** ILLEGAL   BADDR 0-0005
            BIBF   IND=LR   BADDR 1-0001
BADDR 0-0005   DEFINED
            MVZ   INTRINSIC 0S00
            BUN   PAR CLOSE-FILES
BADDR 1-0001   DEFINED
            CMPN   NL 02(60)    INTRINSIC RN02   RELATION GE   BADDR 0-0 0
            MVZ   INTRINSIC SP00
            MVN   NL 01(6)    INTRINSIC SK00
            NTR   PAR PSS.2
```

Figure D-7. Output from LOGIC Compiler-Directing Option (Sheet 3 of 8)

```
BADDR 0-0006   DEFINED
PAR MAKE-DATA-AVAIL   DEFINED
            MNFX    INTRINSIC FN00
            BUN    PAR INPUT-MOVE
OVERLAY 02   DEFINED
PAR BACK-FROM-IM   DEFINED
            BUN    PAR CALC-DETAIL
OVERLAY 00   DEFINED
PAR FORCE-EOJ-SOON   DEFINED
            SET1    IND=********** ILLEGAL
            BIBF    IND=LOADING-PRIMARY-OR-SECONDARY   BADDR 6-0003
            CLR1    IND=LR
            MVZ    INTRINSIC 0S00
            NTR    PAR OUTPUT-DETAIL
            CLR1    IND=LOADING-PRIMARY-OR-SECONDARY
BADDR 6-0003   DEFINED
            SET1    IND=LR
            MVZ    INTRINSIC 1C00
            CBUM     BIT STRING=144 BITS(BFFFFFFFFFFFFFFFFFFFFFFFFFFFF 0
            SET1    IND=LR
            BUN    PAR MAYBE-TOT-CALCS
                              00300FIN       IPE  1800  90            D 0
OVERLAY 05   DEFINED
PAR IN-OUT   DEFINED


            CNFX    NL 01(1)    RELATION NE    BADDR 3-0009
            BUN    OVERLAY 09
OVERLAY 09   DEFINED
PAR P10.1   DEFINED
            BIBF    IND=SECONDARY= 1(RECORD ALREADY IN WORK AREA)    BADD
            XIT
 BADDR 5-0001   DEFINED
            BIBF    IND=EOF-1    BADDR 5-0002
            MVZ    INTRINSIC FX00
            XIT
 BADDR 5-0002   DEFINED
            MVA    AL 09(        })    INTRINSIC CM00
           MAKP    FILE 0001    TRASH0000    REDEFINED 24(000012000003)    0
           COMM    INTRINSIC CM00
           LDCR    TRASH0000    REDEFINED 24(000000000003)
           CMPN    NL 01(2)    TRASH0000    REDEFINED UN(000005000001)     0
           MVA    AL 19(READ ERROR FILE#  :)    TRASH0000    REDEFINED UA
           MVN    INTRINSIC FX00    TRASH0000    REDEFINED UA(00003E00000 0
           MVA    FILE 0001    TRASH0000    REDEFINED UA(000044000038)     0
BADDR 6-0004   DEFINED
           NTR    PAR SP0-OUT
           NTR    PAR SP0-IN
           CMPA    AL 04(STOP)    TRASH0000    REDEFINED UA(00001E00004B) 0
           CMPA    AL 02(GO)    TRASH0000    REDEFINED UA(00001E00004B)    0
           MVA    AL 45(ONLY VALID ENTRIES ARE STOP OR GO. TRY AGAIN.)  0
           BUN    BADDR 6-0004
BADDR 0-0007   DEFINED
           CMPZ    TRASH0000    REDEFINED UN(000005000001)    RELATION EQ 0
           SET1    IND=EOF-1
           MVZ    INTRINSIC FX00
           MVA    AL 06(??)    INTRINSIC CM00
           COMM    INTRINSIC CM00
           XIT
BADDR 0-0008   DEFINED
           XIT
```

Figure D-7. Output from LOGIC Compiler-Directing Option (Sheet 4 of 8)

```
OVERLAY 05    DEFINED
BADDR 3-0009    DEFINED
                              00400FLINE     0          132              P 0
            CNFX    NL 01(2)    RELATION NE    BADDR 3-0010
            BUN    OVERLAY 0A
OVERLAY 0A    DEFINED
PAR P10.2    DEFINED
            MVA    AL 09(??)    INTRINSIC CM00
            MAKP    FILE 0002    TRASH0000    REDEFINED 24(000012000003)    0
            CMPZ    INTRINSIC SK00    RELATION NE    BADDR 1-0002
            CMPN    NL 01(2)    INTRINSIC SP00    RELATION LS    BADDR 1-00 0
            INC    INTRINSIC SP00    INTRINSIC RN02
            CMPZ    INTRINSIC SP00    RELATION EQ    BADDR 0-0009
            MVN    NL 01(?)    TRASH0000    REDEFINED UN(00000B000001)      0
            CMPN    NL 01(1)    INTRINSIC SP00    RELATION EQ    BADDR 0-00 0
            MVN    NL 01(?)    TRASH0000    REDEFINED UN(00000B000001)
BADDR 0-0009    DEFINED
            MVZ    INTRINSIC SP00
BADDR 1-0002    DEFINED
            COMM    INTRINSIC CM00
            NTR    PAR PSS.2
            XIT
OVERLAY 05    DEFINED
BADDR 3-0010    DEFINED
                              00500E                        ARY          5  0
OVERLAY 01    DEFINED
PAR LAST-FILE    DEFINED
            BIBF    IND=EOF-1    BADDR 1-0003
            BUN    PAR FORCE-EOJ-SOON
BADDR 1-0003    DEFINED
            XIT
OVERLAY 01    DEFINED
PAR NEXT-FILE    DEFINED
            INC1    INTRINSIC FN00
            CMPN    NL 01(1)    INTRINSIC FN00    RELATION NE    BADDR 0-00 0
            BIBF    IND=EOF-1    BADDR 3-0011
BADDR 0-0010    DEFINED
            CMPN    NL 01(2)    INTRINSIC FN00    RELATION GE    PAR NEXT-F 0
BADDR 3-0011    DEFINED
            XIT


                              00600IIN       NS  01


OVERLAY 0A    DEFINED
PAR PSS.2    DEFINED
            CMPZ    INTRINSIC SK00    RELATION EQ    BADDR 6-0005
            CMPN    INTRINSIC SK00    INTRINSIC RN02    RELATION EQ    BADD 0
            CMPN    INTRINSIC RN02    INTRINSIC SK00    RELATION LS    BADD
            INC    NL 02(66)    INTRINSIC SK00
            DEC    INTRINSIC RN02    INTRINSIC SK00
            INC    INTRINSIC SK00    INTRINSIC RN02
            BUN    BADDR 4-0002
BADDR 5-0003    DEFINED
            DEC    INTRINSIC RN02    INTRINSIC SK00
            INC    INTRINSIC SK00    INTRINSIC RN02
```

Figure D-7. Output from LOGIC Compiler-Directing Option (Sheet 5 of 8)

```
BADDR 4-0002   DEFINED
           CMPZ   INTRINSIC SKIP-LINE00   RELATION EQ   BADDR 0-0011   0
           MVA   AL 06(???)   INTRINSIC CM00
           CONV   INTRINSIC SKIP-LINE00   TRASH0000   REDEFINED 24(000 0
           COMM   INTRINSIC CM00
           BUN   BADDR 1-0004
BADDR 0-0011   DEFINED
           MVA   AL 09(??)   INTRINSIC CM00
           MAKP   FILE 0002   TRASH0000   REDEFINED 24(000012000003)   0
           MVS   FILE 0002
BADDR 5-0004   DEFINED
           CMPZ   INTRINSIC SK00   RELATION EQ   BADDR 4-0003
           MVN   NL 01(?)   TRASH0000   REDEFINED UN(00000B000001)   0
           DEC1   INTRINSIC SK00
           CMPZ   INTRINSIC SK00   RELATION EQ   BADDR 3-0012
           MVN   NL 01(?)   TRASH0000   REDEFINED UN(00000B000001)   0
           DEC1   INTRINSIC SK00
BADDR 3-0012   DEFINED
           COMM   INTRINSIC CM00
           BUN   BADDR 5-0004
BADDR 4-0003   DEFINED
BADDR 1-0004   DEFINED
           CMPN   NL 02(66)   INTRINSIC RN02   RELATION GE   BADDR 6-0 0
           DEC   NL 02(66)   INTRINSIC RN02
BADDR 6-0005   DEFINED
           CMPZ   INTRINSIC SP00   RELATION EQ   BADDR 4-0004
           INC   INTRINSIC SP00   INTRINSIC RN02
           MVA   AL 09(??)   INTRINSIC CM00
           MAKP   FILE 0002   TRASH0000   REDEFINED 24(000012000003)   0
           MVS   FILE 0002
BADDR 5-0005   DEFINED
           CMPZ   INTRINSIC SP00   RELATION EQ   BADDR 4-0004
           MVN   NL 01(¢)   TRASH0000   REDEFINED UN(00000B000001)   0
           DEC1   INTRINSIC SP00
           CMPZ   INTRINSIC SP00   RELATION EQ   BADDR 3-0013
           MVN   NL 01(?)   TRASH0000   REDEFINED UN(00000B000001)   0
           DEC1   INTRINSIC SP00
BADDR 3-0013   DEFINED
           COMM   INTRINSIC CM00
           BUN   BADDR 5-0005
BADDR 4-0004   DEFINED
           CMPN   NL 02(66)   INTRINSIC RN02   RELATION GE   BADDR 0-0 0
           DEC   NL 02(66)   INTRINSIC RN02
           BUN   BADDR 6-0006
BADDR 0-0012   DEFINED
BADDR 6-0006   DEFINED
           MVZ   INTRINSIC SKIP-LINE00
           MVZ   INTRINSIC SK00
           MVN   NL 01(1)   INTRINSIC SP00
           XIT


OVERLAY 06   DEFINED
PAR INPUT-CHECK   DEFINED
           CNFX   NL 01(1)   RELATION NE   BADDR 3-0014
                             007001
           SET1   IND=01
                             008001
                             009001
                             ??????????????????????????????????????????
           MVN   NL 01(1)   INTRINSIC RN01
           BUN   PAR MAYBE-TOT-CALCS
```

Figure D-7. Output from LOGIC Compiler-Directing Option (Sheet 6 of 8)

```
BADDR 2-0002    DEFINED
BADDR 3-0014    DEFINED


OVERLAY 0B    DEFINED
PAR CALC-DETAIL    DEFINED
            MVN    FIELD 0016    REDEFINED UA(000000000006)    FIELD 0012   0
            REDEFINED SI(000007000005)                  MVN    FIELD 0016    0
            REDEFINED UN(000007000001)    FIELD 0015    REDEFINED SI(0000 0

            SET3    IND=10    IND=20    IND=30
                                ????                        SETOF
            CLR1    IND=30
                                013000LINE     D          01
            MVZ    INTRINSIC 0S00
            BUN    PAR OUTPUT-DETAIL
OVERLAY 0C    DEFINED
PAR CALC-TOTAL    DEFINED
            MVZ    INTRINSIC 0S00
            BUN    PAR OUTPUT-TOTAL
OVERLAY 02    DEFINED


OVERLAY 0D    DEFINED

PAR OUTPUT-TOTAL    DEFINED
            BUN    BADDR 3-0015
PAR OUTPUT-DETAIL    DEFINED
            BIBF    IND=01    BADDR 2-0003
                                14000                                RECORD    0
            CZOS    RELATION NE    BADDR 2-0003
            MVS    FILE 0002
            MVA    FIELD 0014    FILE 0002    REDEFINED UA(00000000005A)    0
                                15000                                ARY
            MVZ    INTRINSIC PX???
BADDR 1-0005    DEFINED
            INC1    INTRINSIC PX???
            MVN     INTRINSIC EX???     INTRINSIC PX???    FILE 0002    REDEF
            REDEFINED SI(00000C000005)
            CMPN    NL 01(5)    INTRINSIC PX???  RELATION GT    BADDR 1-0
                                16000                                IX     X    0
            MVN    FIELD 0015    TRASH0000    REDEFINED SN(000000000001)    0
            CMPZ    FIELD 0015    RELATION GT    BADDR 9-0001
            MVN    NL 01( )    TRASH0000    REDEFINED UN(000000000001)      0
            MVA    TRASH0000    REDEFINED SN(000000000001)    FILE 0002     0
            BUN    BADDR 8-0003
BADDR 9-0001    DEFINED
            MVN    TRASH0000    REDEFINED SN(000000000001)    FILE 0002     0
BADDR 8-0003    DEFINED

            NTR    PAR P10.2
            BUN    OVERLAY 0E
OVERLAY 0E    DEFINED
BADDR 2-0003    DEFINED
            BUN    BADDR 3-0016
BADDR 3-0015    DEFINED
            CNOS    NL 01(2)    RELATION NE    BADDR 0-0013
            XIT
BADDR 0-0013    DEFINED
            CZOS    RELATION NE    PAR OUTPUT-DETAIL
            BUN    PAR MAYBE-OFL-OUTPUT
```

Figure D-7. Output from LOGIC Compiler-Directing Option (Sheet 7 of 8)

```
OVERLAY 07    DEFINED
BADDR 3-0016    DEFINED
           CZOS    RELATION NE    PAR MAKE-DATA-AVAIL
           BIBF    IND=LOADING-PRIMARY-OR-SECONDARY    BADDR 6-0007
           XIT
BADDR 6-0007    DEFINED
           BUN    PAR BACK-FROM-OD
```

Figure D-7. Output from LOGIC Compiler-Directing Option (Sheet 8 of 8)

XMAP Compiler-Directing Option

The XMAP compiler-directing option lists the same information as is included with the MAP compiler-directing option and also lists the actual object code that is generated. If the RPG compiler aborts while generating the XMAP listing, scanning the listing in conjunction with the LOGIC information (specified by the LOGIC compiler-directing option) can help the programmer identify which RPG source record caused the RPG compiler to abort. The XMAP information lists paragraph names which are also listed in the LOGIC information. By comparing the last paragraph name listed in the XMAP information with the paragraph name in the LOGIC information, and then examining the RPG source record listed in the LOGIC information, the programmer can identify the RPG source record that caused the RPG compiler abort. In general, correcting the RPG source record gets around the problem. In any case, the problem should be reported to your Burroughs Representative.

Figure D-8 shows the output from the XMAP compiler-directing option which is generated when the previous RPG example program is compiled.

```
Segment,
Offset   Code Paragraph
-------- ---- ---------
                PAR SETUP-CYCLE SEG 0 DISP 909      0111000110110000000
0,25     BUN
                PAR READ-VECTORS SEG 4 DISP 0      0000000000010000100
0,50     BUN
                BADDR SEG 0 DISP 206      0001100111010000000
0,75     CZOS
                RELATION NEQ
                BADDR SEG 0 DISP 181      0001011010110000000
0,103     MVA
                ALPHA LIT
                COPX FOR INTRINSIC CM      000111
0,169     COMM
                COPX FOR INTRINSIC CM      000111
0,181     BUN
                BADDR SEG 0 DISP 487      0011110011110000000
0,206     BUN
                BADDR SEG 0 DISP 330      0010100101010000000
0,231     SETI
                9 BITS FOR INDICATORS ILLEGAL SPECIFICATION      100100000
0,246     MVA
                ALPHA LIT
                COPX FOR INTRINSIC CM      000111
0,312     COMM
                COPX FOR INTRINSIC CM      000111
0,324     XIT
0,330     MNFX
                NUM LIT
```

Figure D-8. Output from XMAP Compiler-Directing Option (Sheet 1 of 13)

```
Segment,
Offset    Code Paragraph
--------  ---- ---------

0,346      SETI
               9 BITS FOR INDICATORS ILLEGAL SPECIFICATION      011000101
0,361      NTR
               PAR IN-OUT SEG 5 DISP 0      0000000000010000101
0,386      BIBT
               9 BITS FOR INDICATORS ILLEGAL SPECIFICATION      010001011
               BADDR SEG 0 DISP 435      0011011001110000000
0,420      SETI
               9 BITS FOR INDICATORS ILLEGAL SPECIFICATION      010110110
0,435      CLRI
               9 BITS FOR INDICATORS ILLEGAL SPECIFICATION      011000101
0,450      MVZ
               COPX FOR INTRINSIC FX      000001
0,462      BUN
               BADDR SEG 0 DISP 618      0100110101010000000
0,487      CZOS
               RELATION NEQ
               BADDR SEG 0 DISP 593      0100101000110000000
0,515      MVA
               ALPHA LIT
               COPX FOR INTRINSIC CM      000111
0,581      COMM
               COPX FOR INTRINSIC CM      000111
0,593      BUN
               BADDR SEG 0 DISP 767      0101111111110000000
0,618      BUN
               BADDR SEG 0 DISP 742      0101110011010000000
0,643      SETI
               9 BITS FOR INDICATORS ILLEGAL SPECIFICATION      100100001
0,658      MVA
               ALPHA LIT
               COPX FOR INTRINSIC CM      000111
0,724      COMM
               COPX FOR INTRINSIC CM      000111
0,736      XIT
0,742      BUN
               BADDR SEG 0 DISP 792      0110001100010000000
0,767      BUN
               BADDR SEG 4 DISP 25      0000001100110000100
0,792      SETI
               9 BITS FOR INDICATORS ILLEGAL SPECIFICATION      011000101
0,807      NTR
               PAR LAST-FILE SEG 1 DISP 714      0101100101010000001
0,832      CLRI
               9 BITS FOR INDICATORS ILLEGAL SPECIFICATION      011000101
0,847      NTR
               PAR NEXT-FILE SEG 1 DISP 779      0110000101110000001
0,872      MVZ
               COPX FOR INTRINSIC OS      000101
0,884      BUN
               PAR OUTPUT-DETAIL SEG 13 DISP 25      0000001100110001101
4,0      BUN
               BADDR SEG 4 DISP 172      0001010110010000100
4,25      CZOS
               RELATION EQL
               BADDR SEG 4 DISP 131      0001000001110000100
```

Figure D-8. Output from XMAP Compiler-Directing Option (Sheet 2 of 13)

```
Segment,
Offset   Code Paragraph
-------- ---- ---------

4,53      MVA
               ALPHA LIT
               COPX FOR INTRINSIC CM      000111
4,119      COMM
               COPX FOR INTRINSIC CM      000111
4,131      MNOS
               NUM LIT
4,147      BUN
               PAR CLOSE-FILES SEG 0 DISP 75      0000100101110000000
4,172      BUN
               PAR OPEN-FILES SEG 0 DISP 50      0000011001010000000
2,0       CNFX
               NUM LIT
               RELATION NEQ
               BADDR SEG 2 DISP 282      0010001101010000010
2,38       CMPZ
               COPX FOR INTRINSIC       010001
               RELATION EQL
               BADDR SEG 2 DISP 87      0000101011110000010
2,72       CLR1
               9 BITS FOR INDICATORS ILLEGAL SPECIFICATION      010110110
2,87       CMPN
               NUM LIT
               COPX FOR INTRINSIC       010001
               RELATION NEQ
               BADDR SEG 2 DISP 282      0010001101010000010
2,131      MVA
               INLINE DESC FOR FILE      0000000010101100001011010010
               COPX FOR FIELD     011110
2,173      MVN
               INLINE DESC FOR FILE      0000000010101100000000000010110
               COPX FOR FIELD     011111
2,215      MVA
               INLINE DESC FOR FILE      0000000010101101000000110010
               COPX FOR FIELD     100000
2,257      BUN
               BADDR SEG 2 DISP 282      0010001101010000010
2,282      MVZ
               COPX FOR INTRINSIC 1C      000110
2,294      BUN
               PAR BACK-FROM-IM SEG 2 DISP 319      0010011111111000010
0,909      MVN
               NUM LIT
               COPX FOR INTRINSIC 1C      000110
0,931      SET1
               9 BITS FOR INDICATORS ILLEGAL SPECIFICATION      011000101
0,946      MVN
               NUM LIT
               COPX FOR INTRINSIC SP      000011
0,968      MVZ
               COPX FOR INTRINSIC SKIP-LINE      001010
0,980      MVZ
               COPX FOR INTRINSIC SK      000010
0,992      SET1
               9 BITS FOR INDICATORS IP      010000000
```

Figure D-8. Output from XMAP Compiler-Directing Option (Sheet 3 of 13)

```
Segment,
Offset   Code Paragraph
-------- ---- ---------

0,1007     MVN
              NUM LIT
              COPX FOR INTRINSIC      010011
0,1029     MVN
              NUM LIT
              COPX FOR INTRINSIC SK     000010
0,1051     MVZ
              COPX FOR INTRINSIC SP     000011
0,1063     NTR
              PAR PSS.2 SEG 10 DISP 441     0011011100110001010
0,1088     XIT
1,0      CLR1
              9 BITS FOR INDICATORS 1P     010000000
1,15      BIBT
              9 BITS FOR INDICATORS LR     010000010
              PAR FORCE-EOJ-SOON SEG 0 DISP 1094     1000100011010000000
1,49      MNFX
              COPX FOR INTRINSIC FN     000100
1,62      NTR
              PAR IN-OUT SEG 5 DISP 0     0000000000010000101
1,87      CZFX
              RELATION NEQ
              BADDR SEG 1 DISP 165     0001010010110000001
1,115      NTR
              PAR LAST-FILE SEG 1 DISP 714     0101100101010000001
1,140      NTR
              PAR NEXT-FILE SEG 1 DISP 779     0110000101110000001
1,165      BUN
              PAR TURNOFF-INDS SEG 1 DISP 190     0001011111010000001
3,0      MVA
              ALPHA LIT
              COPX FOR INTRINSIC CM     000111
3,98      MAKP
              INLINE DESC FOR TRASH     000000000101010110100101100 10
              INLINE DESC FOR TRASH     00010011111
3,144      COMM
              COPX FOR INTRINSIC CM     000111
3,156      XIT
3,162      MVA
              ALPHA LIT
              COPX FOR INTRINSIC CM     000111
3,260      MAKP
              INLINE DESC FOR TRASH     000000000101010110100101100 10
              INLINE DESC FOR TRASH     00010011111
3,306      COMM
              COPX FOR INTRINSIC CM     000111
3,318      XIT
1,190      SET1
              9 BITS FOR INDICATORS LO     001100100
1,205      CBUM
              BIT STRING     10111111111111111111111111111111111111111
1,355      MNFX
              COPX FOR INTRINSIC FN     000100
1,368      BUN
```

Figure D-8. Output from XMAP Compiler-Directing Option (Sheet 4 of 13)

```
Segment,
Offset   Code Paragraph
-------- ---- ---------

                PAR INPUT-CHECK SEG 6 DISP 0      0000000000010000110
1,393      CMPZ
                COPX FOR INTRINSIC 1C      000110
                RELATION NEQ
                PAR MAYBE-OFL-OUTPUT SEG 1 DISP 464      0011110100001000000
1,427      MVZ
                COPX FOR INTRINSIC OS      000101
1,439      BUN
                PAR OUTPUT-TOTAL SEG 13 DISP 0      0000000000010001101
1,464      BIBT
                9 BITS FOR INDICATORS ILLEGAL SPECIFICATION      010011011
                BADDR SEG 1 DISP 532      0100001010010000001
1,498      BIBF
                9 BITS FOR INDICATORS LR      010000010
                BADDR SEG 1 DISP 569      0100011100110000001
1,532      MVZ
                COPX FOR INTRINSIC OS      000101
1,544      BUN
                PAR CLOSE-FILES SEG 0 DISP 75      0000100101110000000
1,569      CMPN
                NUM LIT
                COPX FOR INTRINSIC       010011
                RELATION GEQ
                BADDR SEG 1 DISP 676      0101010010010000001
1,617      MVZ
                COPX FOR INTRINSIC SP      000011
1,629      MVN
                NUM LIT
                COPX FOR INTRINSIC SK      000010
1,651      NTR
                PAR PSS.2 SEG 10 DISP 441      0011011100110001010
1,676      MNFX
                COPX FOR INTRINSIC FN      000100
1,689      BUN
                PAR INPUT-MOVE SEG 2 DISP 0      0000000000010000010
2,319      BUN
                PAR CALC-DETAIL SEG 11 DISP 0      0000000000010001011
0,1094      SET1
                9 BITS FOR INDICATORS ILLEGAL SPECIFICATION      010011011
0,1109      BIBF
                9 BITS FOR INDICATORS ILLEGAL SPECIFICATION      011000101
                BADDR SEG 0 DISP 1210      1001011101010000000
0,1143      CLR1
                9 BITS FOR INDICATORS LR      010000010
0,1158      MVZ
                COPX FOR INTRINSIC OS      000101
0,1170      NTR
                PAR OUTPUT-DETAIL SEG 13 DISP 25      0000001100110001101
0,1195      CLR1
                9 BITS FOR INDICATORS ILLEGAL SPECIFICATION      011000101
0,1210      SET1
                9 BITS FOR INDICATORS LR      010000010
0,1225      MVZ
                COPX FOR INTRINSIC 1C      000110
```

Figure D-8. Output from XMAP Compiler-Directing Option (Sheet 5 of 13)

```
Segment,
Offset   Code Paragraph
-------- ---- ---------

0,1237     CBUM
               BIT STRING     10111111111111111111111111111111111111111111
0,1387     SET1
               9 BITS FOR INDICATORS LR     010000010
0,1402     BUN
               PAR MAYBE-TOT-CALCS SEG 1 DISP 393     0011000100110000001
5,0      CNFX
               NUM LIT
               RELATION NEQ
               BADDR SEG 5 DISP 63     0000011111110000101
5,38      BUN
               OVLY 09     0000000000010001001
9,0      BIBF
               9 BITS FOR INDICATORS ILLEGAL SPECIFICATION     010110110
               BADDR SEG 9 DISP 40     0000010100010001001
9,34      XIT
9,40      BIBF
               9 BITS FOR INDICATORS ILLEGAL SPECIFICATION     010001011
               BADDR SEG 9 DISP 92     0000101110010001001
9,74      MVZ
               COPX FOR INTRINSIC FX     000001
9,86      XIT
9,92      MVA
               ALPHA LIT
               COPX FOR INTRINSIC CM     000111
9,190      MAKP
               COPX FOR FILE     010000
               INLINE DESC FOR TRASH     00010011111
9,213      COMM
               COPX FOR INTRINSIC CM     000111
9,225      LDCR
               INLINE DESC FOR TRASH     00010001101
9,242      CMPN
               NUM LIT
               INLINE DESC FOR TRASH     0000000001001001000000010000
               RELATION NEQ
               BADDR SEG 9 DISP 1252     1001110010010001001
9,309      MVA
               ALPHA LIT
               INLINE DESC FOR TRASH     0000000001010101101001011010
9,510      MVN
               COPX FOR INTRINSIC FX     000001
               INLINE DESC FOR TRASH     0000000001100101100000010010
9,552      MVA
               COPX FOR FILE     010000
               INLINE DESC FOR TRASH     0000000001101000100111000010
9,594      NTR
               PAR SPO-OUT SEG 3 DISP 0     0000000000010000011
9,619      NTR
               PAR SPO-IN SEG 3 DISP 162     0001010001010000011
9,644      CMPA
               ALPHA LIT
               INLINE DESC FOR TRASH     0000000001010101101001011010
               RELATION EQL
               PAR FORCE-EOJ-SOON SEG 0 DISP 1094     1000100011010000000
```

Figure D-8. Output from XMAP Compiler-Directing Option (Sheet 6 of 13)

```
Segment,
Offset   Code Paragraph
-------- ---- ---------

9,739     CMPA
              ALPHA LIT
              INLINE DESC FOR TRASH     0000000001010101101001011001O
              RELATION EQL
              PAR IN-OUT SEG 5 DISP 0     0000000000010000101
9,818     MVA
              ALPHA LIT
              INLINE DESC FOR TRASH     0000000001010101101001011001O
9,1227     BUN
              BADDR SEG 9 DISP 594     0100101001010001001
9,1252     CMPZ
              INLINE DESC FOR TRASH     0000000001001001000000001OOOO
              RELATION EQL
              BADDR SEG 9 DISP 1420     1011000110010001001
9,1309     SETI
              9 BITS FOR INDICATORS ILLEGAL SPECIFICATION     010001011
9,1324     MVZ
              COPX FOR INTRINSIC FX     000001
9,1336     MVA
              ALPHA LIT
              COPX FOR INTRINSIC CM     000111
9,1402     COMM
              COPX FOR INTRINSIC CM     000111
9,1414     XIT
9,1420     XIT
5,63      CNFX
              NUM LIT
              RELATION NEQ
              BADDR SEG 5 DISP 126     0000111111010000101
5,101      BUN
              OVLY OA     0000000000010001010
10,0      MVA
              ALPHA LIT
              COPX FOR INTRINSIC CM     000111
10,98      MAKP
              COPX FOR FILE     010010
              INLINE DESC FOR TRASH     00010011111
10,121     CMPZ
              COPX FOR INTRINSIC SK     000010
              RELATION NEQ
              BADDR SEG 10 DISP 398     0011000111010001010
10,155     CMPN
              NUM LIT
              COPX FOR INTRINSIC SP     000011
              RELATION LSS
              BADDR SEG 10 DISP 398     0011000111010001010
10,199     INC
              COPX FOR INTRINSIC SP     000011
              COPX FOR INTRINSIC      010011
10,218     CMPZ
              COPX FOR INTRINSIC SP     000011
              RELATION EQL
              BADDR SEG 10 DISP 386     0011000001010001010
10,252     MVN
              NUM LIT
              INLINE DESC FOR TRASH     0000000001001100000000001OOOO
```

Figure D-8. Output from XMAP Compiler-Directing Option (Sheet 7 of 13)

```
Segment,
Offset   Code Paragraph
-------- ---- ---------

10,297     CMPN
              NUM LIT
              COPX FOR INTRINSIC SP      000011
              RELATION EQL
              BADDR SEG 10 DISP 386      0011000001010001010
10,341     MVN
              NUM LIT
              INLINE DESC FOR TRASH      00000000010011000000000010000
10,386     MVZ
              COPX FOR INTRINSIC SP      000011
10,398     COMM
              COPX FOR INTRINSIC CM      000111
10,410     NTR
              PAR PSS.2 SEG 10 DISP 441      0011011100110001010
10,435     XIT
1,714      BIBF
              9 BITS FOR INDICATORS ILLEGAL SPECIFICATION      010001011
              BADDR SEG 1 DISP 773      0110000010110000001
1,748      BUN
              PAR FORCE-EOJ-SOON SEG 0 DISP 1094      1000100011010000000
1,773      XIT
1,779      INC1
              COPX FOR INTRINSIC FN      000100
1,791      CMPN
              NUM LIT
              COPX FOR INTRINSIC FN      000100
              RELATION NEQ
              BADDR SEG 1 DISP 869      0110110010110000001
1,835      BIBF
              9 BITS FOR INDICATORS ILLEGAL SPECIFICATION      010001011
              BADDR SEG 1 DISP 913      0111001000110000001
1,869      CMPN
              NUM LIT
              COPX FOR INTRINSIC FN      000100
              RELATION GEQ
              PAR NEXT-FILE SEG 1 DISP 779      0110000101110000001
1,913      XIT
10,441      CMPZ
              COPX FOR INTRINSIC SK      000010
              RELATION EQL
              BADDR SEG 10 DISP 1270      1001111011010001010
10,475      CMPN
              COPX FOR INTRINSIC SK      000010
              COPX FOR INTRINSIC       010011
              RELATION EQL
              BADDR SEG 10 DISP 1270      1001111011010001010
10,516      CMPN
              COPX FOR INTRINSIC       010011
              COPX FOR INTRINSIC SK      000010
              RELATION LSS
              BADDR SEG 10 DISP 646      0101000011010001010
10,557      INC
              NUM LIT
              COPX FOR INTRINSIC SK      000010
10,583      DEC
              COPX FOR INTRINSIC       010011
```

Figure D-8. Output from XMAP Compiler-Directing Option (Sheet 8 of 13)

```
Segment,
Offset   Code Paragraph
-------- ---- ---------

                COPX FOR INTRINSIC SK      000010
10,602      INC
                COPX FOR INTRINSIC SK      000010
                COPX FOR INTRINSIC       010011
10,621      BUN
                BADDR SEG 10 DISP 684      0101010110010001010
10,646      DEC
                COPX FOR INTRINSIC       010011
                COPX FOR INTRINSIC SK      000010
10,665      INC
                COPX FOR INTRINSIC SK      000010
                COPX FOR INTRINSIC       010011
10,684      CMPZ
                COPX FOR INTRINSIC SKIP-LINE      001010
                RELATION EQL
                BADDR SEG 10 DISP 844      0110100110010001010
10,718      MVA
                ALPHA LIT
                COPX FOR INTRINSIC CM      000111
10,784      CONV
                COPX FOR INTRINSIC SKIP-LINE      001010
                INLINE DESC FOR TRASH      00010011001
10,807      COMM
                COPX FOR INTRINSIC CM      000111
10,819      BUN
                BADDR SEG 10 DISP 1196      1001010110010001010
10,844      MVA
                ALPHA LIT
                COPX FOR INTRINSIC CM      000111
10,942      MAKP
                COPX FOR FILE      010010
                INLINE DESC FOR TRASH      00010011111
10,965      MVS
                COPX FOR FILE      010010
10,977      CMPZ
                COPX FOR INTRINSIC SK      000010
                RELATION EQL
                BADDR SEG 10 DISP 1196      1001010110010001010
10,1011      MVN
                NUM LIT
                INLINE DESC FOR TRASH      00000000010011000000000010000
10,1056      DEC1
                COPX FOR INTRINSIC SK      000010
10,1068      CMPZ
                COPX FOR INTRINSIC SK      000010
                RELATION EQL
                BADDR SEG 10 DISP 1159      1001000011110001010
10,1102      MVN
                NUM LIT
                INLINE DESC FOR TRASH      00000000010011000000000010000
10,1147      DEC1
                COPX FOR INTRINSIC SK      000010
10,1159      COMM
                COPX FOR INTRINSIC CM      000111
```

Figure D-8. Output from XMAP Compiler-Directing Option (Sheet 9 of 13)

```
Segment,
Offset   Code Paragraph
-------- ---- ---------

10,1171      BUN
               BADDR SEG 10 DISP 977      0111110100011000 1010
10,1196      CMPN
               NUM LIT
               COPX FOR INTRINSIC       010011
               RELATION GEQ
               BADDR SEG 10 DISP 1270      1001111011010001010
10,1244      DEC
               NUM LIT
               COPX FOR INTRINSIC       010011
10,1270      CMPZ
               COPX FOR INTRINSIC SP      000011
               RELATION EQL
               BADDR SEG 10 DISP 1675      1101000101110001010
10,1304      INC
               COPX FOR INTRINSIC SP      000011
               COPX FOR INTRINSIC       010011
10,1323      MVA
               ALPHA LIT
               COPX FOR INTRINSIC CM      000111
10,1421      MAKP
               COPX FOR FILE      010010
               INLINE DESC FOR TRASH      00010011111
10,1444      MVS
               COPX FOR FILE      010010
10,1456      CMPZ
               COPX FOR INTRINSIC SP      000011
               RELATION EQL
               BADDR SEG 10 DISP 1675      1101000101110001010
10,1490      MVN
               NUM LIT
               INLINE DESC FOR TRASH      00000000010011000000000010000
10,1535      DEC1
               COPX FOR INTRINSIC SP      000011
10,1547      CMPZ
               COPX FOR INTRINSIC SP      000011
               RELATION EQL
               BADDR SEG 10 DISP 1638      1100110011010001010
10,1581      MVN
               NUM LIT
               INLINE DESC FOR TRASH      00000000010011000000000010000
10,1626      DEC1
               COPX FOR INTRINSIC SP      000011
10,1638      COMM
               COPX FOR INTRINSIC CM      000111
10,1650      BUN
               BADDR SEG 10 DISP 1456      1011011000010001010
10,1675      CMPN
               NUM LIT
               COPX FOR INTRINSIC       010011
               RELATION GEQ
               BADDR SEG 10 DISP 1774      1101110111010001010
10,1723      DEC
               NUM LIT
```

Figure D-8. Output from XMAP Compiler-Directing Option (Sheet 10 of 13)

```
Segment,
Offset    Code Paragraph
--------  ---- ---------

                COPX FOR INTRINSIC      010011
10,1749       BUN
                BADDR SEG 10 DISP 1774     1101110111010001010
10,1774       MVZ
                COPX FOR INTRINSIC SKIP-LINE     001010
10,1786       MVZ
                COPX FOR INTRINSIC SK     000010
10,1798       MVN
                NUM LIT
                COPX FOR INTRINSIC SP     000011
10,1820       XIT
6,0       CNFX
                NUM LIT
                RELATION NEQ
                BADDR SEG 6 DISP 100     0000110010010000110
6,38       SET1
                9 BITS FOR INDICATORS 01     000000001
6,53       MVN
                NUM LIT
                COPX FOR INTRINSIC      010001
6,75       BUN
                PAR MAYBE-TOT-CALCS SEG 1 DISP 393     0011000100110000001
11,0       MVN
                INLINE DESC FOR FIELD     00000010000010111000001100010
                INLINE DESC FOR FIELD     00000001100101100000001100000
                INLINE DESC FOR FIELD     1
                INLINE DESC FOR FIELD     001
                INLINE DESC FOR FIELD     011111
                INLINE DESC FOR FIELD     00000111
                INLINE DESC FOR FIELD     00000011100
11,93       MVN
                INLINE DESC FOR FIELD     00000010000100001000000010000
                INLINE DESC FOR FIELD     00000001100101011000000010000
                INLINE DESC FOR FIELD     1
                INLINE DESC FOR FIELD     001
                INLINE DESC FOR FIELD     011111
                INLINE DESC FOR FIELD     00000111
                INLINE DESC FOR FIELD     00000011100
11,186       SET3
                9 BITS FOR INDICATORS 10     000001010
                9 BITS FOR INDICATORS 20     000010100
                9 BITS FOR INDICATORS 30     000011110
11,219       CLR1
                9 BITS FOR INDICATORS 30     000011110
11,234       MVZ
                COPX FOR INTRINSIC OS     000101
11,246       BUN
                PAR OUTPUT-DETAIL SEG 13 DISP 25     0000001100110001101
12,0       MVZ
                COPX FOR INTRINSIC OS     000101
12,12       BUN
                PAR OUTPUT-TOTAL SEG 13 DISP 0     0000000000010001101
13,0       BUN
                BADDR SEG 14 DISP 25     0000001100110001110
```

Figure D-8. Output from XMAP Compiler-Directing Option (Sheet 11 of 13)

```
Segment,
Offset   Code Paragraph
-------- ---- ---------

13,25     BIBF
              9 BITS FOR INDICATORS 01      000000001
              BADDR SEG 14 DISP 0      0000000000010001110
13,59     CZOS
              RELATION NEQ
              BADDR SEG 14 DISP 0      0000000000010001110
13,87     MVS
              COPX FOR FILE      010010
13,99     MVA
              COPX FOR FIELD      011110
              INLINE DESC FOR FILE      00000001000001111010110100010
13,141     MVZ
              COPX FOR INTRINSIC       011010
13,153     INC1
              COPX FOR INTRINSIC       011010
13,165     MVN
              COPX FOR INTRINSIC       011000
              COPX FOR INTRINSIC       011010
              INLINE DESC FOR FILE      00000001010011011000001100110
              COPX FOR INTRINSIC       1
              COPX FOR INTRINSIC       001
              COPX FOR INTRINSIC       011010
              COPX FOR INTRINSIC       00001100
              COPX FOR INTRINSIC       00000110000
13,241     CMPN
              NUM LIT
              COPX FOR INTRINSIC       011010
              RELATION GTR
              BADDR SEG 13 DISP 153      0001001100110001101
13,285     MVN
              COPX FOR FIELD      011111
              INLINE DESC FOR TRASH      0000000001000110100000010100
13,327     CMPZ
              COPX FOR FIELD      011111
              RELATION GTR
              BADDR SEG 13 DISP 496      0011111000010001101
13,361     MVN
              NUM LIT
              INLINE DESC FOR TRASH      0000000001000110100000010000
13,406     MVA
              INLINE DESC FOR TRASH      0000000001000110100000010100
              INLINE DESC FOR FILE      0000000101110100100000010010
13,471     BUN
              BADDR SEG 13 DISP 561      0100011000110001101
13,496     MVN
              INLINE DESC FOR TRASH      0000000001000110100000010100
              INLINE DESC FOR FILE      0000000101110100100000010110
13,561     NTR
              PAR P10.2 SEG 10 DISP 0      0000000000010001010
13,586     BUN
              OVLY 0E      0000000000010001110
14,0      BUN
              BADDR SEG 7 DISP 0      0000000000010000111
14,25      CNOS
```

Figure D-8. Output from XMAP Compiler-Directing Option (Sheet 12 of 13)

```
Segment,
Offset   Code Paragraph
-------- ---- ---------

                NUM LIT
                RELATION NEQ
                BADDR SEG 14 DISP 69      0000100010110001110
14,63       XIT
14,69       CZOS
                RELATION NEQ
                PAR OUTPUT-DETAIL SEG 13 DISP 25      0000001100110001101
14,97       BUN
                PAR MAYBE-OFL-OUTPUT SEG 1 DISP 464      0011101000010000000
7,0       CZOS
                RELATION NEQ
                PAR MAKE-DATA-AVAIL SEG 1 DISP 676      0101010010010000001
7,28        BIBF
                9 BITS FOR INDICATORS ILLEGAL SPECIFICATION      011000101
                BADDR SEG 7 DISP 68      0000100010010000111
7,62        XIT
7,68        BUN
                PAR BACK-FROM-OD SEG 1 DISP 0      0000000000010000001
```

Figure D-8. Output from XMAP Compiler-Directing Option (Sheet 13 of 13)

READING AN RPG PROGRAM MEMORY DUMP

The following paragraphs describe how to obtain an RPG program memory dump, how to read the useful information contained in an RPG program memory dump, and shows two example programs which illustrate two different RPG program aborts. The first is an INVALID SUBSCRIPT program abort and the second is a STACK OVERFLOW program abort.

How to Obtain a RPG Program Memory Dump

The RPG program memory dump is generated by entering either of the following system commands:

<job-number>DM
<job-number>DP

The DM command causes a memory dump to be created and allows the program to continue executing.

The DP command causes a memory dump to be created and discontinues the program.

The memory dump created is a file with the name DUMPFILE/<integer>, where <integer> is a system-generated number. Once created, enter the following system command to obtain a listing of the memory-dump file.

PM <integer>;

The PM command causes the DUMP/ANALYZER program to analyze the memory-dump file and produce a hard-copy listing.

RPG DATA AREA DUMP Information

The portion of the analyzed dump file begins with the following heading:

```
R P G   D A T A   A R E A   D U M P
----------------------------------
```

The following paragraph describes this portion of the analyzed dump file that can be useful to the RPG programmer.

NEXT INSTRUCTION POINTER Information

The NEXT INSTRUCTION POINTER information has the following format:

NEXT INSTRUCTION POINTER: S = , D = <offset> (hexadecimal>

This information identifies the next instruction to be performed. The LOGIC and XMAP compiler-directing option must be specified in order to locate which RPG source record is being processed. The XMAP information associates the segment-displacement information with a paragraph name and the LOGIC information associates the paragraph name with the RPG source record.

CONTAINER SIZES Information

The CONTAINER SIZES information has the following format:

```
C O N T A I N E R   S I Z E S
-----------------------------
      COP TABLE ENTRY: <integer>
    DATA DISPLACEMENT: <integer>
          DATA LENGTH: <integer>
            COP INDEX: <integer>
  BRANCH DISPLACEMENT: <integer>
```

This information is used in the development of the RPG compiler.

INDICATORS SET Information

The INDICATORS SET information shows which indicators in the RPG program are ON when the memory dump was generated. The following is the format of the INDICATORS information:

```
I N D I C A T O R S   S E T
---------------------------
      <indicator>
      <indicator>
           .
           .
           .
      <indicator>
```

```
Segment,
Offset    Code Paragraph
--------  ---- ---------

                   NUM LIT
                   RELATION NEQ
                   BADDR SEG 14 DISP 69      0000100010110001110
14,63       XIT
14,69       CZOS
                   RELATION NEQ
                   PAR OUTPUT-DETAIL SEG 13 DISP 25      0000001100110001101
14,97       BUN
                   PAR MAYBE-OFL-OUTPUT SEG 1 DISP 464      0011101000010000000
7,0       CZOS
                   RELATION NEQ
                   PAR MAKE-DATA-AVAIL SEG 1 DISP 676      0101010010010000001
7,28       BIBF
                   9 BITS FOR INDICATORS ILLEGAL SPECIFICATION      011000101
                   BADDR SEG 7 DISP 68      0000100010010000111
7,62        XIT
7,68        BUN
                   PAR BACK-FROM-OD SEG 1 DISP 0      0000000000010000001
```

Figure D-8. Output from XMAP Compiler-Directing Option (Sheet 13 of 13)

READING AN RPG PROGRAM MEMORY DUMP

Please refer to the B 1000 Systems Memory Dump Analysis Functional Description Manual for instructions on how to read an RPG program memory dump.

Information deleted by PCN 1152063-001

Pages D-34 thru D-37 deleted by PCN 1152063-001

APPENDIX E
EXECUTION-TIME ERROR MESSAGES

Certain conditions can arise during program execution which require operator notification and, in most cases, acknowledgement. Some program errors are recoverable, and some are not. For more information, refer to B 1000 Systems System Software Operation Guide, Volume 1.

INPUT ERROR MESSAGES

The following messages all denote error conditions arising during input, and all are recoverable. For each error condition, the program requests a reply from the operator. A GO response, <program job number>AXGO, causes the program to ignore the erroneous record and read the next record from the same file. A STOP response, <program job number>AXSTOP, causes the program to ignore the erroneous record, set the LR indicator on, and perform all final detail output, total calculations, and total output.

IDENT Halt

The occurrence of the following message results from reading an unidentifiable input record, that is, none of the designated Record Identification Codes for the file could be found in the record read:

<program-name> = <program job number>: IDENT

MSEQ Halt

The occurrence of the following message results from reading a record that is out of sequence when matching records are specified (the indicators M1-M9 in columns 61-62 of the Input Specifications). All records in matching files must be in either ascending or descending sequence:

<program-name> = <program job number>: MSEQ

SEQ Halt

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

<program-name> = <program job number>: SEQ

REOF Halt

The occurrence of the following message results from reading a demand file which is at the end of the file and no indicator is specified in columns 58-59 of the Calculation Specifications:

<program-name> = <program job number>: REOF

READ ERROR Halt

The occurrence of the following message results from reading a file that contains an error, for example, a parity error:

<program-name> = <program job number>: READ ERROR

PROGRAMMED HALTS

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

<program-name> = <program job number>: HALT nnnnnnnnnn

If the H0, H3, H4, H7, and H9 indicators are set, the following message appears:

<program-name> = <program job number>: HALT 0..34..7.9

Following this message, the program requests a reply from the operator. A GO response, <program job number>AXGO, turns off all halt indicators and the program continues processing. A STOP response, <program job number>AXSTOP, sets the LR indicator on and the program performs all total calculations and output. If the LR indicator is on when a halt occurs, the program displays the halt message and continues without waiting for an operator response.

SPECIAL FORMS POSITIONING

The following message is displayed after all output lines conditioned by the indicator are printed and FORMS POSITIONING is specified (the number 1 in column 41 of the Control Specification):

<program-name> = <program job number>: AGAIN?

After the message is displayed, the program requests a reply from the operator. A YES response, <program job number>AXYES, causes all output lines conditioned by the 1P indicator to be printed again, and the message is repeated.

A NO response, <program job number>AXNO, causes normal processing to continue.

Printing of all output lines conditioned by the 1P indicator can be requested as many times as necessary in order to align the forms properly. Requests for forms positioning occurs only when the file writes directly to the line printer.

ARITHMETIC ERRORS

The following paragraphs describe the arithmetic errors than can cause the RPG program to halt.

Divide by Zero Halt

The following message occurs when an attempt to perform a DIV (divide) operation using a divisor of zero. The operating system automatically discontinues (DS-ed) the program.

<program-name> = <program job number>: DIVIDE BY ZERO

SQRT Halt

The following message occurs when an attempt to perform a SQRT (square root) operation on a negative argument:

<program-name> = <program job number>: SQRT

Following the message, the program requests a reply from the operator. A GO response, <program job number>AXGO, sets the value contained in the RESULT FIELD to zero, and the program continues processing. A STOP response, <program job number>AXSTOP, sets the LR indicator on, and the program performs all final detail, total calculation, and total output.

VECTOR ELEMENT ERRORS – INVALID SUBSCRIPT HALT

The following message occurs when an attempt to reference a vector element that is out of bounds, that is, an index value less than or equal to zero, or greater than the maximum size of the array as specified in columns 36-39 of the Extension Specifications. The operating system automatically discontinues (DS-ed) the program.

<program-name> = <program job number>: INVALID SUBSCRIPT

PRE-EXECUTION TIME VECTOR – VSEQ HALT

The following message occurs when reading a pre-execution time vector record which is out of sequence as specified by the contents of the SEQUENCE fields in the Extension Specifications. Any response causes the program to terminate.

<program-name> = <program job number>: VSEQ

SEQUENCE ERROR – KEYSEQ HALT

The following message indicates that the indexed data file has a key which is out of sequence. Any response forces end of job. The data file of the program requires resorting or the program loading or updating the file requires reworking.

<program-name> = <program job number>: KEYSEQ FOR FILE = <filename>

LIMITS FILE – LIMITS PAIR ERROR

When the low record key is greater in value than the high record key and a file is processed within limits by means of a limits file, the following message occurs:

<program-name> = <program job number>: LIMITS PAIR

MUST BE IGNORED FOR FILE = <filename>

LOW KEY GREATER THAN HIGH KEY

If the limits pairs are entered through the ODT, then the program waits for the operator to re-enter the correct limits pair before processing continues. If the limits pairs are not entered through the console, the program ignores the erroneous limits pair and reads the limits file for the next limits pair.

INVALID TAG FILE/DATA FILE MESSAGES

The following messages occur when a tag file contains a record which is out-of-bounds:

<program-name> = <program job number> FILE = <filename>, TAGFILE

POINTS TO INVALID DATA RECORD (KEY = 0 OR KEY GTR EOF)

The program terminates with the following response:

<program job number>AX

INDEX-SEQUENTIAL FILE HALTS – DELERR HALT

The following message is displayed when a program attempts to delete a non-existent record from an index-sequential chain file.

<program-name> = <program job number>: DELERR

This error message can occur with either of the following conditions:

1. The DELET operation code was specified, columns 54-55 of the RESULTING INDICATORS field was left blank, and there was no record to delete.
2. The letters DEL were specified in the AND/OR field (columns 14-16) of the Output-Format Specifications for the index-sequential chain file and there was no record to delete.

Following the message, the program requests a reply from the operator. The following response causes the program to continue processing:

<program job number>AXGO

The following response causes the program to be terminated:

<program job number>AXSTOP

APPENDIX F
RPG/BTF PROGRAM

The RPG/BTF (Build Tag File) utility program creates a tag file from an existing indexed data disk file.

RPG/BTF PROGRAM RULES

The following rules apply to the RPG/BTF program:

1. The data file is not required to be in key sequence order. The tag file is built in the proper sequence.
2. When a duplicate key is encountered in a data file, an error message on the line printer specifies the relative record numbers of both records. In addition, the tag file is not entered in the disk directory.
3. The RPG/BTF program can handle key fields up to 23 bytes in length for indexed ($ TAG) files. If the RECORD ADDRESS TYPE field (column 31) of the File Description Specifications is left blank, then alphanumeric keys are assumed.
4. The letter I (index) must appear in the FILE ORGANIZATION TYPE field (column 32) of the File Description Specifications.

RPG/BTF PROGRAM INPUT

Input to the RPG/BTF program can consist of the entire RPG source program; however, the RPG/BTF program ignores all but the following cards:

1. The Control Card Specifications with an "R" or "L" (blank) in column 17. The default is "L" when the Control Card Specifications are not included in the input to the RPG/BTF program.
2. The $ RSIGN dollar option card. This dollar option must appear prior to each File Designation Specifications when the indexed data file is right-signed and "R" is not specified in column 17 of the Control Card Specifications. $ RSIGN/$NRSIGN can be used when selected files are right-signed and others are left-signed.
3. The $ PACKID, $ FAMILY, and $ FILEID dollar option card. These dollar options must be correctly related to each indexed file's File Description Specifications when the external file name is different from the internal file name.
4. File Description Specifications for each tag file to be built. Tag files are created for File Description Specifications that are declared as input, update, or output with additions. Each File Description Specifications describes the indexed data file to the RPG/BTF program. This gives the RPG/BTF program the following information:

    1) Key length in bytes.
    2) Key type (alphanumeric, numeric, or packed).
    3) Key field starting location of each record in the data file.

All other source cards are ignored.

RPG/BTF PROGRAM FUNCTIONS

The following functions are performed by the RPG/BTF program:

1. First, the RPG/BTF program reads from a card file called RPG/CARD. This card file can contain the entire RPG source program. The information required to build the tag file is obtained from each File Description Specifications in the RPG source program that defines an indexed file. Files must be declared input, update, or output with additions.
2. Next, the RPG/BTF program locates each indexed data file on disk in the order in which they appear in the File Description Specifications section of the RPG program. If a data file is not on disk, the RPG/BTF program continues with the next indexed data file described in the RPG source program; the message FILE NOT ON DISK appears in the printer error file called TAGFILE/INFO.
3. A tag file is then created for each indexed data file located on disk. Each tag file contains the keys and relative record numbers of each record in its corresponding indexed data file. The keys in the new tag file remain unchanged from the keys in the data file. The relative record number, which is the position number of the record in the data file, also remains unchanged.
4. After the tag file is created, RPG/BTF invokes the SORT/VSORT utility program which sorts the tag file records by key in ascending order.
5. After sorting, the RPG/BTF program locks the tag file in the Disk Directory and returns to step 1. When the input card file, RPG/CARD, is exhausted, the RPG/BTF program goes to end of job.

Table F-1 illustrates the internal and external file names of the tag file.

Table F-1. Internal and External File Names of the Tag File and Data File

Data File Internal File Name	Data File External File Name	Tag File Internal File Name	Tag File External File Name
DISK	A/ABCDEFGHIJ	TAGDISK	A/TAGABCDEFG
DISKIN		TAGDISKIN	DISKIN/TAG
DISK	CCC/A/B	TAGDISK	CCC/A/TAGB
DISK	A/ABCDEFGHIJ	TAGDISK	A/TAGABCDEFG
DISK	A/B/	TAGDISK	A/B/TAG

Table F-2 shows the internal and external file names, associated devices, and functions of the files used by the RPG/BTF program.

Table F-2. Internal and External File Names of RPG/BTF

Internal File Name	Device	External File Name	Function or Description
LINE	Printer	TAGFILE/INFO	Error listing for RPG/BTF.
CARDS	Card	RPG/CARD	Input file for RPG/BTF.
DATA.FILE	Disk	BOBB/DATA.FILE	Workfile used to input data file.
TAG.FILE	Disk	BOBB/TAG.FILE	Workfile used to build and sort tag file.

The following example illustrates the cards necessary to build tag files for indexed files INDEX and USER/RPG/TEST.

Example:

```
? EXECUTE RPG/BTF
? DATA RPG/CARD

     H (optional)

     FINDEX   IPE F 180  60  5PI    10 DISK

      $ PACKIDUSER
      $ FAMILYRPG
      $ FILEIDTEST

     FTEST     IS F 180 180 30AI   125 DISK

       (Remainder of RPG source deck is not required)

? END
```

APPENDIX G
TAGSORT WITH RPG

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

The method for creating a tag file is discussed in the B 1000 Systems SORT Reference Manual.

Basically, the tag file is created by a TAGSORT program which goes through the input file, extracting the key field(s) from the record, and adding the position of the record, relative to the first record in the file. The tag file is then sorted, using the regular sort concept, and leaving a file of relative record numbers (4 bytes in length) sorted in the specified sequence. The original input file remains unchanged.

The record address file (TAGFILE) created by the TAGSORT program can be used in an RPG program to perform record address (ADDROUT) processing. For detailed information, refer to the subsection titled Random Processing – Addrout Files in section 5.

In the example shown in table G-1, the input file DISKFI is used to build a tag file called TAGFILE. Refer to the B 1000 Systems SORT Reference Manual for additional information.

Table G-1. Building a Tag File

Contents of the DISKFI file	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)	00000005
00003 879 1111	KEY (1 3 A A)	00000008
00004 653 3333	TAGSORT	00000001
00005 258 4444		00000007
00006 785 5555		00000004
00007 551 6666		00000006
00008 327 7777		00000003
00009 965 8888		00000009

The contents of DISKFI consists of nine records. The first three digits comprise the key field, the last four comprise the data field.

Since DISKFI was sorted by key in ascending order and record number 2 has the lowest key value, record 2 is placed in the first position of the tag file. Record 5 has the second smallest key (258), so it is placed in the second position of the tag file, etc.

APPENDIX H
RPG PROGRAM CYCLE

For each record from a primary or secondary file that is processed, the RPG object program goes through the same general cycle of operations. After a record is read, there are two different instances in time when calculation operations are performed and records written out. First, all total calculation operations (those conditioned by control level indicators in columns 7-8) and all total output operations are done. Second, all detail calculation operations and all detail output operations are done.

Total calculations are performed before the information on the record selected for processing is made available. Detail calculations are performed after the information on the selected record is made available. The following discussion describes this concept in more detail.

Whenever a record is read, a check is made to determine if information in a control field (when one has been specified) is different from the control field information on the previous card. A change in the control field information indicates that all records from a particular control group have been read and a new group is starting. When all records from a group have been read (indicated by control level indicators being set ON), operations may be done using information accumulated from all records in that group. It is at this time that all calculations conditioned by control level indicators in columns 7-8 are done. Total output operations are also performed immediately after all total calculation operations are completed. Information on the record read at the beginning of the program cycle is not used in these operations; only information from records in the previous control group is used.

Detail calculations occur after the information on the selected record has been made available. Detail calculations are used to calculate values needed each time a record is processed. They are also used to calculate totals for the current control group (if control fields are specified). Immediately after detail calculation operations are completed, detail output operations are performed.

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

1. Initialization:

   1) Data fields and indicators may have been preset to their initial values by the compiler; if not they must be initialized now:

      a. All data fields and vectors are initialized to zero (if numeric) or blank (if alphanumeric).
      b. All match fields are initialized to high (if descending sequence) or low (ascending) collating values.
      c. All control field storage areas are initialized to hexadecimal zeroes.
      d. Indicators LO and 1P are set ON, all other indicators are set OFF. Under the RPG1 dialect any indicators from 01-99 used as zero-blank indicators are set ON, unless also used as record identifying indicators.

   2) UPDATE, UDAY, UMONTH, UYEAR fields are set.
   3) External indicators are set.
   4) Files are opened (unless files are to opened automatically when first accessed).
   5) 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 does 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 controls the heading output required on the first cycle by conditioning it on the indicator lP (which is 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 two options:

   1) STOP. Program branches to step 13. After performing "last record," total calculations, and total output, the program will terminate.
   2) GO. The program continues at step 7.

7. All record identifying indicators are set OFF. The 1P, L1-L9 and H0-H9 indicators are set OFF. The L0 indicator is set ON. Under the RPG1 dialect the LR indicator is also set OFF.
8. Under the RPG1 dialect, the program continues at step 9; otherwise, the LR indicator is tested. If the LR indicator 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, that is, 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. The LR and L1-L9 indicators 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 that 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.

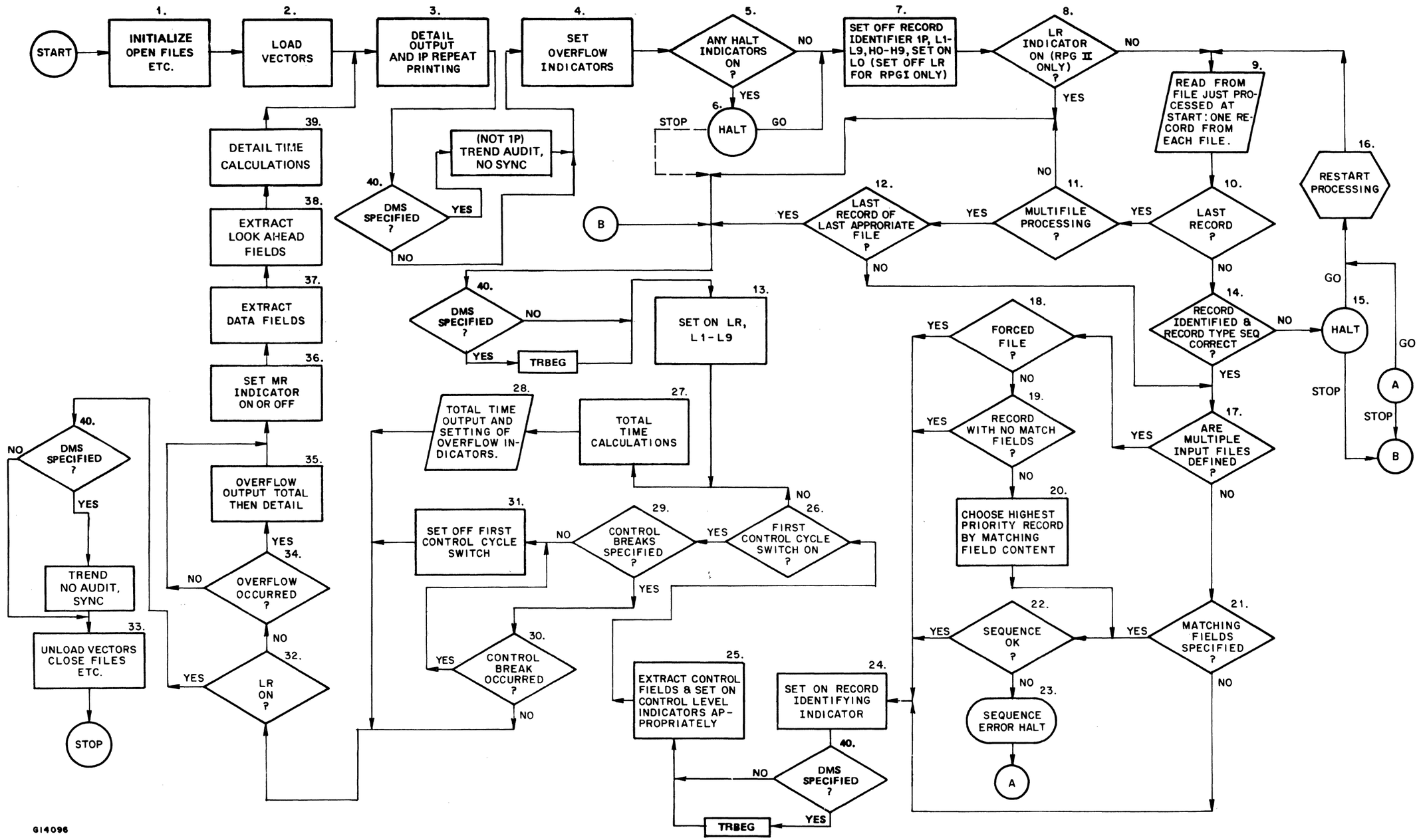


Figure H-1. RPG Program Cycle

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 Description 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 zeroes.
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 operations are performed on all subsequent cycles (if no control break occurs, only the total calculation and total output operations conditioned by the L0 indicator are performed).
32. If LR indicator is ON, the program branches to step 33.
33. End-of-job routine.

    1) Output any vectors specified with a TO FILENAME on the Extensions Specifications.
    2) Perform any file maintenance. For example, sorting an indexed file if unordered or if additions have been made.
    3) 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 is 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 MR (matching record) indicator is set ON or OFF. The MR indicator is set ON if a match has occurred, that is, 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.
39. Detail time calculations are performed. Processing continues with step 3.
40. The "DMS specified?" decision tests for the following conditions:

1) A restart data set is specified on the File Description Specifications.
2) One or more cycle-driven DMSII files are specified on the File Description Specifications.
3) The data base is opened update (the letter U in column 23 of the Data Base Specifications).

INDEX

INDEX (CONT)

INDEX (CONT)

INDEX (CONT)

INDEX (CONT)

INDEX (CONT)

INDEX (CONT)

INDEX (CONT)

INDEX (CONT)

INDEX (CONT)

INDEX (CONT)

INDEX (CONT)

INDEX (CONT)

INDEX (CONT)

INDEX (CONT)

INDEX (CONT)

Documentation Evaluation Form

Title: B 1000 Systems Report Program Generator (RPG) Language Manual

Form No: 1152063

Date: March 1984

Burroughs Corporation is interested in receiving your comments and suggestions regarding this manual. Comments will be utilized in ensuing revisions to improve this manual.

Please check type of Suggestion:

☐ Addition ☐ Deletion ☐ Revision ☐ Error

Comments:

From:

Name _____

Title _____

Company _____

Address _____

_____

Phone Number _____ Date _____

Remove form and mail to:

Burroughs Corporation
Corporate Documentation – West
1300 John Reed Court
City of Industry, CA 91745
U.S.A.