

**Burroughs**  
**B 1700**  
**SYSTEMS**

**Report  
Program  
Generator**  
**REFERENCE MANUAL**

**\$4.00**

**Burroughs**  
**B 1700 Systems**  
**Report Program Generator**

**REFERENCE MANUAL**



**Burroughs Corporation**  
Detroit, Michigan 48232

\$4.00

COPYRIGHT © 1972, 1974, 1975 BURROUGHS CORPORATION  
AA349022 AA575727

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

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

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

# TABLE OF CONTENTS

<u>Section</u>		<u>Page No.</u>
	INTRODUCTION .....	xv
1	RPG OPERATION .....	1-1
	RPG Source Program .....	1-1
	Control Card Specifications .....	1-1
	File Description Specifications .....	1-1
	Extension Specifications .....	1-1
	Line Counter Specifications .....	1-2
	Telecommunications Card Specifications .....	1-2
	Input Specifications .....	1-2
	Calculation Specifications .....	1-2
	Output-Format Specifications .....	1-2
	Dollar Card Specifications .....	1-2
	RPG Source Program Deck .....	1-2
	RPG Format .....	1-3
2	RPG LANGUAGE ELEMENTS .....	2-1
	Character Set .....	2-1
	Characters Used For Names .....	2-1
	Characters Used For Editing .....	2-2
	Definition of Names .....	2-2
	Filenames .....	2-2
	Vector Names .....	2-2
	Field Names .....	2-2
	Labels .....	2-2
	Definition of Literals .....	2-3
	Numeric Literal .....	2-3
	Alphanumeric Literal .....	2-3
	Definition of Reserved Words .....	2-4
	Special Words .....	2-4
	Operation Codes .....	2-4
	Common Field Definitions .....	2-4
	PAGE .....	2-5
	LINE .....	2-5
	FORM TYPE .....	2-5
	COMMENTS/DOLLAR CARD .....	2-6
	PROGRAM IDENTIFICATION .....	2-6
3	CONTROL CARD SPECIFICATIONS .....	3-1
	Field Definitions .....	3-1
	PAGE .....	3-1
	LINE .....	3-1

TABLE OF CONTENTS (Cont)

<u>Section</u>		<u>Page No.</u>
3	FORM TYPE .....	3-1
	DEBUG .....	3-1
	SIGN POSITION .....	3-3
	INVERTED PRINT .....	3-3
	FORMS POSITIONING .....	3-4
	SOURCE INPUT DIALECT .....	3-4
	PROGRAM IDENTIFICATION .....	3-4
4	FILE DESCRIPTION SPECIFICATIONS .....	4-1
	Field Definitions .....	4-1
	PAGE .....	4-1
	LINE .....	4-1
	FORM TYPE .....	4-1
	FILENAME .....	4-1
	FILE TYPE .....	4-3
	Input Files .....	4-3
	Output Files .....	4-3
	Update Files .....	4-3
	Combined Files .....	4-3
	Display Files .....	4-3
	FILE DESIGNATION .....	4-4
	Primary File .....	4-4
	Secondary Files .....	4-4
	Chained File .....	4-4
	Table File .....	4-4
	Demand File .....	4-5
	END OF FILE .....	4-5
	SEQUENCE .....	4-6
	FILE FORMAT .....	4-6
	BLOCK AND RECORD LENGTH .....	4-7
	PROCESSING MODE .....	4-8
	RECORD ADDRESS FIELD LENGTH .....	4-8
	RECORD ADDRESS TYPE .....	4-8
	FILE ORGANIZATION TYPE .....	4-9
	File Organization .....	4-9
	Sequential .....	4-9
	Direct .....	4-10
	File Accessing Methods .....	4-10
	Sequential .....	4-10
	Direct .....	4-11
	Indexed-Sequential .....	4-11
	Additional I/O Areas .....	4-12
	OVERFLOW INDICATOR .....	4-12
	KEY FIELD STARTING LOCATION .....	4-13
	EXTENSION CODE .....	4-13
	DEVICE .....	4-13
	DATACOM .....	4-17
	LABELS .....	4-17
	CORE INDEX .....	4-18
	FILE ADDITION/UNORDERED .....	4-19
	TAPE REWIND .....	4-20
	Card Input .....	4-20
	Card Output .....	4-20

TABLE OF CONTENTS (Cont)

<u>Section</u>		<u>Page No.</u>
4	Tape Input .....	4-23
	Tape Output .....	4-23
	Printer Output .....	4-24
	Disk Files .....	4-24
	FILE CONDITION .....	4-24
	PROGRAM IDENTIFICATION .....	4-25
5	VECTORS AND EXTENSION SPECIFICATIONS .....	5-1
	Vectors .....	5-1
	Compile-Time Vectors .....	5-1
	Pre-Execution-Time Vectors .....	5-2
	Dynamic/Execution-Time Vectors .....	5-2
	Extension Specifications .....	5-2
	Field Definitions .....	5-3
	PAGE .....	5-3
	LINE .....	5-3
	FORM TYPE .....	5-3
	FROM FILENAME .....	5-3
	TO FILENAME .....	5-3
	VECTOR NAME .....	5-5
	ENTRIES PER RECORD .....	5-5
	ENTRIES PER VECTOR .....	5-6
	LENGTH OF ENTRY .....	5-6
	PACKED .....	5-6
	DECIMAL POSITIONS .....	5-7
	SEQUENCE .....	5-7
	VECTOR NAME, LENGTH OF ENTRY, PACKED, DECIMAL POSITIONS, SEQUENCE .....	5-7
	COMMENTS .....	5-8
	PROGRAM IDENTIFICATION .....	5-8
	Vector Loading .....	5-8
	Compile-Time Vector Load .....	5-9
	Pre-Execution Time Vector Load .....	5-10
	Dynamic/Execution Time Vector Load .....	5-10
	Input Specifications Load .....	5-11
	Calculation Specifications Load .....	5-12
	Vector Output .....	5-13
6	LINE COUNTER SPECIFICATIONS .....	6-1
	Field Definitions .....	6-1
	PAGE .....	6-1
	LINE .....	6-1
	FORM TYPE .....	6-1
	FILENAME .....	6-1
	LINE NUMBER, FL OR CHANNEL NUMBER .....	6-1
	LINE NUMBER, OL OR CHANNEL NUMBER .....	6-5
	LINE NUMBER AND CHANNEL NUMBER .....	6-5
	PROGRAM IDENTIFICATION .....	6-6

TABLE OF CONTENTS (Cont)

<u>Section</u>		<u>Page No.</u>
7	TELECOMMUNICATIONS CARD SPECIFICATIONS .....	7-1
	Field Definitions .....	7-1
	PAGE .....	7-1
	LINE .....	7-1
	FORM TYPE .....	7-1
	FILENAME .....	7-1
	CONFIGURATION .....	7-1
	TRANSMIT/RECEIVE .....	7-2
	TYPE OF LINE CONTROL .....	7-2
	TRANSMISSION CODE .....	7-2
	TRANSPARENCY .....	7-3
	TYPE OF CONNECTION .....	7-3
	TELEPHONE NUMBER .....	7-3
	LOCATION OF IDENTIFICATION-THIS STATION .....	7-3
	IDENTIFICATION-THIS STATION .....	7-3
	LOCATION OF IDENTIFICATION-REMOTE STATION .....	7-3
	IDENTIFICATION-REMOTE STATION .....	7-3
	REMOTE TERMINAL .....	7-3
	ITB .....	7-3
	PERMANENT ERROR INDICATOR .....	7-3
	TIMEOUT VALUE .....	7-3
	RECORD AVAILABLE INDICATOR .....	7-3
	LAST FILE .....	7-4
	TRANSMIT ADDRESS .....	7-4
	RECEIVE ADDRESS .....	7-4
	PROGRAM IDENTIFICATION .....	7-4
8	INPUT SPECIFICATIONS .....	8-1
	Field Definitions .....	8-1
	PAGE .....	8-1
	LINE .....	8-1
	FORM TYPE .....	8-1
	FILENAME .....	8-1
	AND/OR LINES .....	8-3
	AND Line .....	8-3
	OR Line .....	8-3
	SEQUENCE .....	8-4
	NUMBER .....	8-4
	OPTION .....	8-5
	RECORD IDENTIFYING INDICATOR .....	8-6
	Record Identifying Indicator 01-99 .....	8-6
	Control Level Indicator L1-L9 .....	8-7
	Last Record Indicator LR .....	8-7
	Halt Indicator H0-H9 .....	8-7
	Look-Ahead Field ** .....	8-7
	RECORD IDENTIFICATION CODES .....	8-10
	POSITION .....	8-10
	NOT .....	8-10
	C/Z/D .....	8-11
	CHARACTER .....	8-13
	STACKER SELECT .....	8-13
	PACKED .....	8-14

TABLE OF CONTENTS (Cont)

<u>Section</u>		<u>Page No.</u>
8	FIELD LOCATION .....	8-15
	DECIMAL POSITIONS .....	8-16
	FIELD NAME (VARIABLE NAME) .....	8-16
	OR Relationship .....	8-16
	Special Words .....	8-18
	CONTROL LEVEL .....	8-19
	Split Control Fields .....	8-21
	MATCHING FIELDS .....	8-22
	Sequence Checking .....	8-22
	Matching Fields .....	8-23
	FIELD RECORD RELATION .....	8-29
	Record Identifying Indicators .....	8-29
	Control Level or Matching Record Indicator .....	8-30
	External Indicators .....	8-31
	Halt Indicators .....	8-31
	FIELD INDICATORS .....	8-31
	PLUS .....	8-31
	MINUS .....	8-31
	ZERO OR BLANK .....	8-31
	PROGRAM IDENTIFICATION .....	8-32
9	CALCULATION SPECIFICATIONS AND OPERATION CODES .....	9-1
	Field Definitions .....	9-1
	PAGE .....	9-1
	LINE .....	9-1
	FORM TYPE .....	9-1
	CONTROL LEVEL .....	9-1
	LO-L9 .....	9-3
	LR .....	9-4
	SR .....	9-4
	AN-OR .....	9-4
	INDICATORS .....	9-5
	01-99 .....	9-6
	LO-L9 .....	9-6
	LR .....	9-6
	MR .....	9-6
	HO-H9 .....	9-7
	U1-U8 .....	9-7
	OA-OG, OV .....	9-7
	FACTOR 1 .....	9-7
	OPERATION .....	9-8
	FACTOR 2 .....	9-8
	RESULT FIELD .....	9-9
	FIELD LENGTH .....	9-9
	DECIMAL POSITIONS .....	9-9
	HALF ADJUST .....	9-10
	RESULTING INDICATORS .....	9-10
	COMMENTS .....	9-10
	PROGRAM IDENTIFICATION .....	9-10
	Operation Codes .....	9-12
	Arithmetic Operations .....	9-12
	ADD .....	9-14
	SUB .....	9-14



TABLE OF CONTENTS (Cont)

<u>Section</u>		<u>Page No.</u>
9	MULT .....	9-14
	DIV .....	9-14
	MVR .....	9-14
	SQRT .....	9-16
	XFOOT .....	9-17
	Z-ADD .....	9-17
	Z-SUB .....	9-17
	Move Operations .....	9-17
	MOVE .....	9-19
	MOVEL .....	9-20
	Move Zone Operations .....	9-20
	MHZO .....	9-21
	MHLZO .....	9-21
	MLHZO .....	9-22
	MLLZO .....	9-22
	Compare Operations .....	9-23
	COMP .....	9-23
	Comparison of Numeric Fields .....	9-23
	Comparison of Alphanumeric Fields .....	9-24
	TESTZ .....	9-24
	Binary Field Operations .....	9-24
	BITON .....	9-25
	BITOF .....	9-26
	TESTB .....	9-26
	Columns 54-55 .....	9-26
	Columns 56-57 .....	9-26
	Columns 58-59 .....	9-26
	Setting Indicators .....	9-27
	SETON .....	9-27
	SETOF .....	9-27
	Program Branching Operations .....	9-27
	GOTO .....	9-27
	TAG .....	9-27
	Transfer Control Function .....	9-28
	ZIP .....	9-28
	Lookup Operations .....	9-28
	LOKUP .....	9-28
	Single-Vector LOKUP .....	9-31
	Two-Vector LOKUP .....	9-32
	Subroutines .....	9-33
	BEGSR .....	9-34
	ENDSR .....	9-34
	EXSR .....	9-34
	Programmed Control of Input and Output .....	9-35
	CHAIN .....	9-35
	Direct Files .....	9-36
	Indexed Sequential Files .....	9-36
	Chained Input Files .....	9-36
	Chained Output Files .....	9-36
	Chained Update Files .....	9-37
	DSPLY .....	9-37
	EXCPT .....	9-38
	FORCE .....	9-38
	READ .....	9-39

TABLE OF CONTENTS (Cont)

<u>Section</u>		<u>Page No.</u>
9	DEBUG Operation .....	9-40
	DEBUG .....	9-40
	DEBUG Operation Output .....	9-40
10	OUTPUT-FORMAT SPECIFICATIONS .....	10-1
	Field Definitions .....	10-1
	PAGE .....	10-1
	LINE .....	10-1
	FORM TYPE .....	10-1
	FILENAME .....	10-1
	TYPE .....	10-3
	RECORD ADDITION .....	10-3
	STACKER SELECT/FETCH OVERFLOW .....	10-4
	STACKER SELECT .....	10-4
	FETCH OVERFLOW .....	10-4
	Exception Lines .....	10-6
	SPACE .....	10-7
	SKIP (RPG II Dialect) .....	10-7
	SKIP (RPG I Dialect) .....	10-8
	OUTPUT INDICATORS .....	10-9
	01-99 .....	10-10
	L0-L9 .....	10-10
	LR .....	10-11
	MR .....	10-11
	H0-H9 .....	10-11
	U1-U8 .....	10-11
	0A-0G, 0V .....	10-11
	Control Level Indicators With Overflow Indicators .....	10-13
	1P .....	10-14
	AND/OR .....	10-14
	FIELD NAME (Variable Name) .....	10-15
	Special Words .....	10-15
	Page Fields (PAGE, PAGE1, and PAGE2) .....	10-15
	Date Fields (UPDATE, UMONTH, UDAY, and UYEAR) .....	10-16
	*PLACE Specification .....	10-17
	*PRINT Specification .....	10-18
	EDIT CODES .....	10-19
	BLANK AFTER .....	10-20
	END POSITION .....	10-21
	PACKED .....	10-21
	CONSTANT OR EDIT WORD .....	10-22
	Constants .....	10-22
	Edit Words .....	10-23
	Special Characters Used in Edit Words .....	10-24
	No Editing .....	10-26
	Zero Suppression .....	10-26
	Dollar Sign .....	10-26
	Asterisk Fill .....	10-28
	Negative Signs .....	10-28
	Ampersand .....	10-28
	Constants in the Edit Word .....	10-28
	PROGRAM IDENTIFICATION .....	10-30

TABLE OF CONTENTS (Cont)

<u>Section</u>		<u>Page No.</u>
11	DOLLAR CARD SPECIFICATIONS .....	11-1
	Field Definitions .....	11-1
	PAGE .....	11-1
	LINE .....	11-1
	FORM TYPE .....	11-1
	\$ OPTION .....	11-1
	NOT .....	11-1
	KEY WORD .....	11-3
	File Identification Extensions .....	11-3
	\$ PACKID .....	11-3
	\$ FAMILY .....	11-3
	\$ FILEID .....	11-3
	RPG Extensions .....	11-4
	\$ AREAS .....	11-4
	\$ RPERA .....	11-5
	\$ OPEN .....	11-5
	\$ CLOSE .....	11-5
	\$ AAOPEN .....	11-5
	\$ ONEPAK .....	11-5
	\$ CYL .....	11-5
	\$ DRIVE .....	11-5
	\$ REFORM .....	11-5
	\$ REORG .....	11-5
	\$ RESIGN .....	11-6
	\$ CHAIN .....	11-6
	\$ DNAME .....	11-6
	\$ PTAPE .....	11-6
	\$ TAPE7 .....	11-6
	\$ TAPE9 .....	11-6
	\$ CASSET .....	11-6
	Compiler-Directing Options .....	11-6
	\$ LIST .....	11-6
	\$ LOGIC .....	11-6
	\$ MAP .....	11-6
	\$ NAMES .....	11-7
	\$ BAZBON .....	11-7
	\$ ZBINIT .....	11-7
	\$ STACK .....	11-7
	\$ XREF .....	11-7
	\$ PARMAP .....	11-7
	\$ SEG .....	11-7
	\$ SUPR .....	11-8
	\$ XMAP .....	11-8
	VALUE .....	11-8
	COMMENTS .....	11-8
	PROGRAM IDENTIFICATION .....	11-8
12	COMPILER OPERATION .....	12-1
	Source Input .....	12-1
	Control Card Syntax .....	12-1
	Vector File Input .....	12-3

TABLE OF CONTENTS (Cont)

<u>Section</u>	<u>Page No.</u>
12	Compile-Time Vector Files ..... 12-3
	Execution-Time Vector Files ..... 12-3
	Compiler File Names ..... 12-3
<u>Appendix</u>	<u>Page No.</u>
A	B 1700 CARD CODES ..... A-1
B	HEXADECIMAL VALUES FOR THE B 1700 CHARACTER SET ..... B-1
C	BURROUGHS INDICATOR SUMMARY FORM ..... C-1
D	COMPILE-TIME ERROR AND DIAGNOSTIC MESSAGES ..... D-1
	General ..... D-1
	File Description Specifications ..... D-1
	Extension Specifications ..... D-3
	Line Counter Specifications ..... D-4
	Input Specifications ..... D-5
	Calculation Specifications ..... D-7
	Output-Format Specifications ..... D-9
	Dollar Card Specifications ..... D-10
E	EXECUTION-TIME ERROR MESSAGES ..... E-1
	Input Error Messages ..... E-1
	Programmed Halts ..... E-2
	Forms Positioning ..... E-2
	Arithmetic Errors ..... E-2
	Subscript Errors ..... E-3
	Sequence Errors ..... E-3
F	B 1700 RPG TO COBOL TRANSLATOR (COFIRS II) ..... F-1
	COFIRS II Operation ..... F-1
G	TAGSORT WITH RPG ..... G-1
H	RPG PROGRAM CYCLE ..... H-1
	INDEX ..... Index-1

LIST OF ILLUSTRATIONS

<u>Figure</u>	<u>Page No.</u>
1-1	RPG Source Program Deck ..... 1-3
2-1	Insertion of Coding Lines ..... 2-5
2-2	Comment Line Coding ..... 2-6
3-1	Control Card Specifications Summary Sheet ..... 3-2
4-1	File Description Specifications Summary Sheet ..... 4-2
4-2	A Sequentially Organized Card File on Disk ..... 4-9
4-3	Random Processing of a Direct File ..... 4-11

LIST OF ILLUSTRATIONS (Cont)

<u>Figure</u>		<u>Page No.</u>
4-4	Device Coding Example - MFCU Files .....	4-15
4-5	Device Coding Example - Console Files (SPO) .....	4-15
4-6	Device Coding Example - Printer Files .....	4-16
4-7	Device Coding Example - Tape Files .....	4-16
4-8	Core Index Selection of Data Keys .....	4-18
4-9	Processing Methods of Disk Files - Loading .....	4-21
4-10	Processing Methods of Disk Files - ADD Records only .....	4-21
4-11	Processing Methods of Disk Files - By CHAIN .....	4-22
4-12	Processing Methods of Disk Files - Sequential .....	4-23
4-13	Indexed and Direct Files - File Update .....	4-26
4-14	Indexed and Direct Files - File Creation .....	4-27
5-1	Entries Necessary to Describe a Vector .....	5-2
5-2	Extension Specifications Summary Sheet .....	5-4
5-3	Vectors in Alternating Format .....	5-8
5-4	Setting Up a Data Deck for a Compile-Time Vector .....	5-9
5-5	Compile-Time Vector Load .....	5-9
5-6	Pre-Execution Time Vector Load .....	5-10
5-7	Dynamic Vector Load (Sheet 1 of 2) .....	5-11
	Dynamic Vector Load (Sheet 2 of 2) .....	5-12
6-1	Line Counter Specifications Summary Sheet .....	6-2
6-2	Overflow Coding Example 1 .....	6-3
6-3	Overflow Coding Example 2 .....	6-4
6-4	Line Counter Specifications Code Example .....	6-6
7-1	Telecommunications Card Specifications Summary Sheet ....	7-2
8-1	Input Specifications Summary Sheet .....	8-2
8-2	AND/OR Relationships - Record Identification Codes .....	8-3
8-3	Input Sequence Checking .....	8-5
8-4	Records Available for Look-Ahead: Two Input Files .....	8-8
8-5	Records Available for Look-Ahead: One Update File, One Input File .....	8-9
8-6	Coding Look-Ahead Fields .....	8-9
8-7	C/Z/D Coding Example 1 .....	8-12
8-8	C/Z/D Coding Example 2 .....	8-12
8-9	C/Z/D Coding Example 3 .....	8-13
8-10	FIELD LOCATION Coding Example .....	8-15
8-11	OR Relationship - Identical Fields within Different Record Types .....	8-17
8-12	PAGE Field Coding .....	8-18
8-13	Control Fields in Two Record Types .....	8-20
8-14	Split Control Fields .....	8-21
8-15	Coding for Sequence Checking .....	8-23
8-16	Coding Matching Fields .....	8-26
8-17	Record Selection from Two Matching Files .....	8-27
8-18	Record Selection from Three Matching Files .....	8-28
8-19	Field Record Relations - Using the OR Relationship .....	8-30
9-1	Calculation Specifications Summary Sheet .....	9-2
9-2	LO Indicator Coding Example .....	9-4
9-3	AND/OR Relationship Coding Example .....	9-5
9-4	Resulting Indicators Coding Example .....	9-11
9-5	DIV and MVR Coding Example .....	9-16
9-6	Z-ADD and Z-SUB Coding Example .....	9-19
9-7	MHHZO Move Zone Operation .....	9-21
9-8	MHLZO Move Zone Operations .....	9-22

## LIST OF ILLUSTRATIONS (Cont)

<u>Figure</u>		<u>Page No.</u>
9-9	MLHZO Move Zone Operations .....	9-22
9-10	MLLZO Move Zone Operations .....	9-23
9-11	Binary Field Operations Coding Example .....	9-25
9-12	The ZIP Operation Using a Field Name .....	9-29
9-13	The ZIP Operation Using a Literal .....	9-29
9-14	LOKUP Coding Example - Using an Index .....	9-31
9-15	Using a Table in an Arithmetic Operation After LOKUP .....	9-32
9-16	LOKUP Coding Example - Two Vectors .....	9-33
9-17	Subroutine Coding Examples .....	9-34
9-18	DSPLY Operation Coding Examples .....	9-38
9-19	DEBUG Operation Output .....	9-41
10-1	Output-Format Specifications Summary Sheet .....	10-2
10-2	FETCH OVERFLOW Coding Example .....	10-5
10-3	Overflow Indicator with Skip Specified .....	10-6
10-4	FETCH OVERFLOW with Exception Output Coding Example .....	10-7
10-5	Output Indicators Coding Example .....	10-10
10-6	Using Control Level Indicators with Overflow Indicators ..	10-13
10-7	Coding for the Printing of Certain Fields .....	10-13
10-8	1P Indicator Coding Example .....	10-14
10-9	PAGE Specification Coding Example .....	10-16
10-10	*PLACE Specification Coding Example .....	10-17
10-11	*PRINT Specification Coding Example .....	10-19
10-12	Coding of Output Constants .....	10-23
10-13	No Editing Coding Examples .....	10-26
10-14	Zero Suppression Zero Coding Examples .....	10-27
10-15	Dollar Sign Coding Examples .....	10-27
10-16	Asterisk Fill Coding Examples .....	10-28
10-17	Negative Signs Coding Examples .....	10-29
10-18	Ampersand Coding Examples (RPG II Dialect) .....	10-29
10-19	Ampersand Coding Examples (RPG I Dialect) .....	10-29
10-20	Examples of Constants in the Edit Word .....	10-30
11-1	Dollar Card Specifications Summary Sheet .....	11-2
11-2	Coding File Identification Extensions - Dollar Sign Options	11-4
A-1	Zone and Digit Portions of an 80-Column Card .....	A-1
A-2	Zone and Digit Portions of a 96-Column Card .....	A-2
C-1	Burroughs Indicator Summary Form .....	C-1
H-1	RPG Program Cycle .....	H-2

## LIST OF TABLES

<u>Table</u>		<u>Page No.</u>
3-1	Inverted Print Specifications .....	3-3
4-1	The Maximum and Minimum Values Allowed for Block and Record Sizes .....	4-7
4-2	Resulting Action of BLOCK LENGTH and RECORD LENGTH .....	4-7
4-3	Default Block and Record Lengths .....	4-8
4-4	Device Assignment for Files .....	4-14
5-1	Guide for Determining Vector Type .....	5-6
9-1	Summary of Operation Codes .....	9-13
9-2	RESULT FIELD Contents for Various Field Lengths and Decimal Positions - MULT Operation .....	9-15

LIST OF TABLES (Cont)

<u>Table</u>		<u>Page No.</u>
9-3	RESULT FIELD Contents for Various Field Lengths and Decimal Positions - SQRT Operation .....	9-18
9-4	MOVE Operations .....	9-20
9-5	MOVEL Operations .....	9-21
10-1	Edit Codes Table .....	10-20
10-2	Effect of Edit Codes on End Position .....	10-22
12-1	Files Used for Source Input and Compilation Output .....	12-3
A-1	Punch Card Codes for the B 1700 Character Set .....	A-3
B-1	B 1700 RPG Collating Sequence .....	B-1
B-2	Hex to Binary Conversion Table .....	B-2
G-1	Building a Tagfile .....	G-1
G-2	Accessing a Tagfile .....	G-2

LIST OF APPLICABLE B 1700 PUBLICATIONS

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

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

# INTRODUCTION

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

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

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

- a. Simple, generative syntax for ease of program implementation.
- b. Accelerated programmer training and simplified retraining requirements.
- c. Ease of conversion through standard implementation.
- d. Ease of program modification.
- e. Standardized documentation.
- f. Facilities for program conversion to COBOL (refer to appendix F).

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

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

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

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



# RPG OPERATION

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

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

## RPG SOURCE PROGRAM

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

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

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

### CONTROL CARD SPECIFICATIONS

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

### FILE DESCRIPTION SPECIFICATIONS

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

### EXTENSION SPECIFICATIONS

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

## LINE COUNTER SPECIFICATIONS

These specifications provide information about the number of lines to be printed on each page of the output forms that are used.

## TELECOMMUNICATIONS CARD SPECIFICATIONS

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

## INPUT SPECIFICATIONS

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

## CALCULATION SPECIFICATIONS

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

## OUTPUT-FORMAT SPECIFICATIONS

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

## DOLLAR CARD SPECIFICATIONS

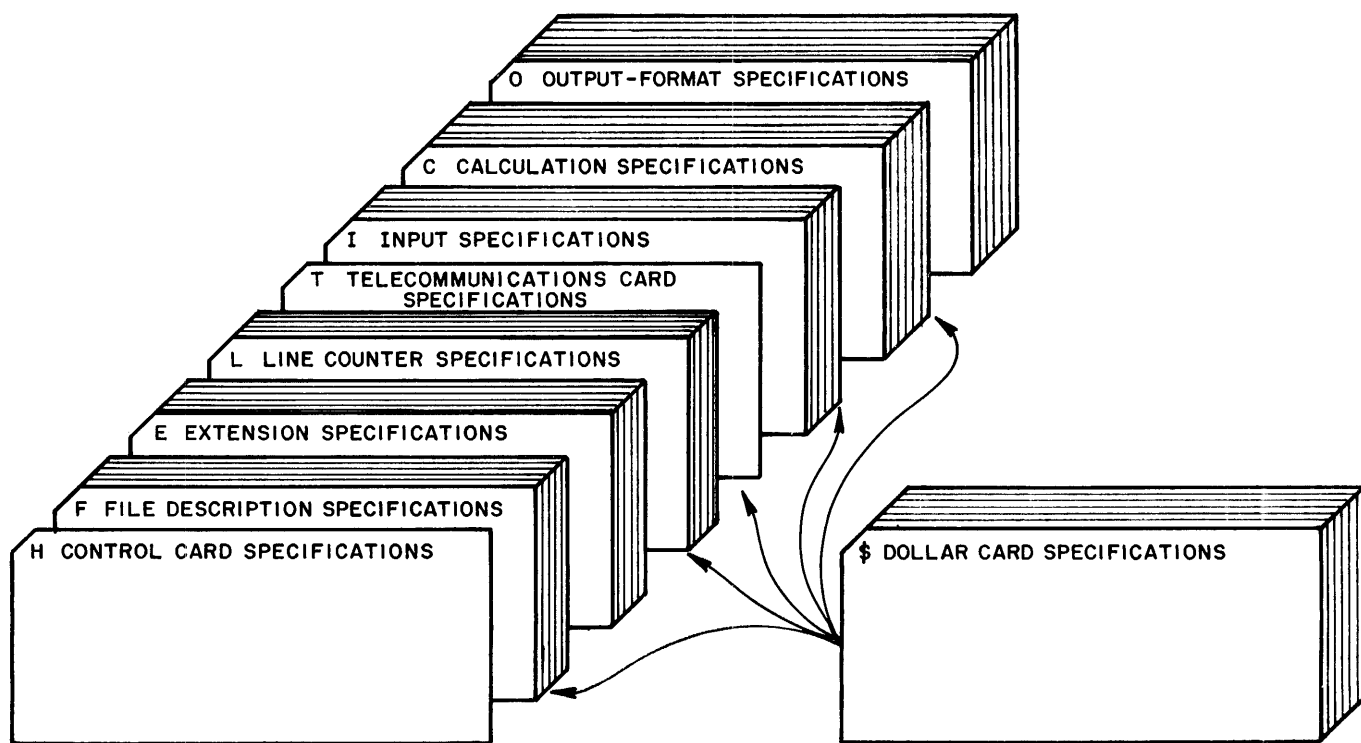
The Dollar Card (\$ Card) Specifications sheet is also defined for Burroughs RPG. These specifications are used to accommodate machine- and system-dependent features, and may appear anywhere in a RPG Source Program, unless otherwise specified (see Section 11).

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

- a. Control Card Specifications/File Description Specifications  
(Form No. 1055837)
- b. Extension Specifications/Line Counter Specifications  
(Form No. 1055852)
- c. Input Specifications (Form No. 1055860)
- d. Calculation Specifications (Form No. 1055878)
- e. Output-Format Specifications (Form No. 1055886)
- f. Dollar Card Specifications (Form No. 1055845)

## RPG SOURCE PROGRAM DECK

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



614003

Figure 1-1. RPG Source Program Deck

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

#### RPG FORMAT

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

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

- a. The Control Card requires simplified recoding before compiling programs from systems other than B 1700.
- b. B 1700 RPG allows 511 characters for alpha fields and 31 for numeric.
- c. IBM's RPG uses standard right signed numeric fields, whereas Burroughs B 1700 RPG uses standard left sign. IBM's System 3 standard plus sign is 'F' right sign, the 360/20 standard plus sign is 'C' right sign, whereas the Burroughs standard plus sign is 'C' left sign. B 1700 RPG allows the right sign option for system compatibility. (Refer to \$RSIGN option in Section 11.)

- d. Numeric fields must be edited if the C zone punch is not wanted on output. Otherwise, a positive unedited numeric field will contain an alpha character in the left-most position of the field. For example, C1F0F1F3 would appear as A013 on the printed output. However, if this field is edited, it will assign a F to the left-most character, changing the output to appear as 1013.
- e. 96-column devices will not allow packed decimal format.
- f. The following features and language constructs are not supported in Burroughs RPG:
  - 1. Binary data format.
  - 2. Sterling data format.
  - 3. Alternate collating sequence.
  - 4. Record address files.
  - 5. Inquiry programs.
  - 6. Spread card or TR format.
  - 7. Printer keyboard output files.
  - 8. External assembler subroutines.
  - 9. Blank after literals.
  - 10. Redefined field lengths.
  - 11. Factor 1 and Factor 2 both literals.
  - 12. Card print feature of IBM's 360/20.
  - 13. File translation.
  - 14. Processing sequential files within limits.

# SECTION 2

## RPG LANGUAGE ELEMENTS

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

### CHARACTER SET

The minimum RPG character set consists of the following characters:

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

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

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

### CHARACTERS USED FOR NAMES

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

## CHARACTERS USED FOR EDITING

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

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

## DEFINITION OF NAMES

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

RPG defines the following four types of names:

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

## FILENAME

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

## VECTOR NAMES

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

## FIELD NAMES

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

## LABELS

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

## DEFINITION OF LITERALS

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

### NUMERIC LITERAL

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

- a. There must be at least one digit in a numeric literal.
- b. The sign of a numeric literal must appear as the left-most character. If no sign is present, the literal is defined as a positive value.
- c. Only one sign character and/or one decimal point may be contained in a numeric literal.
- d. The maximum total length of a numeric literal is 10 characters, including sign and decimal point.
- e. Decimal commas must be used in place of decimal points if the Inverted Print Options I or J are used (Control Card, Column 21).
- f. Embedded blanks are not allowed.

The following are examples of numeric literals:

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

### ALPHANUMERIC LITERAL

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

- a. There must be at least one character in an alphanumeric literal.
- b. The maximum length of an alphanumeric literal used in the Calculation Specifications is eight characters, and in the Output-Format Specification is 24 characters.
- c. Alphanumeric literals may not be used for arithmetic operations.

The following are examples of alphanumeric literals:

Literal on Source Program Level

Literal Stored by Compiler

'ACTUAL'	ACTUAL
'-1234.56'	-1234.56
'WEEK'S'	WEEK'S
'TODAY'S DATE'	TODAY'S DATE
''''	'
'A'B'	A'B
'A''B'	A''B

DEFINITION OF RESERVED WORDS

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

SPECIAL WORDS

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

The following special words are reserved for use as variables:

PAGE  
PAGE1  
PAGE2  
UPDATE  
UMONTH  
UDAY  
UYEAR  
\*PRINT  
\*PLACE

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

NOTE

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

OPERATION CODES

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

COMMON FIELD DEFINITIONS

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





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

7 COMMENTS/DOLLAR CARD

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

3	5	6	7	14	15	16	17	18	19	20	23	24	27	28	29	30	31	32	33	34	35	38	39	40	46	47											
0	2	F	*	T	H	I	S	I	S	A	N	E	X	A	M	P	L	E	O	F	H	O	W	C	O	M	M	E	N	T	L	I	N	E	S		
0	3	F	*	A	R	E	C	O	D	E	D	.	A	L	L	L	I	N	E	S	W	I	T	H	A	N	A	S	T	E	R	I	S	K			
0	4	F	*	I	N	C	O	L	U	M	N	7	A	R	E	T	R	E	A	T	E	D	A	S	D	O	C	U	M	E	N	T	A	R	Y		
0	5	F	*	A	N	D	A	R	E	I	G	N	O	R	E	D	B	Y	T	H	E	C	O	M	P	I	L	E	R	.							
0	6	F	*																																		
0	7	F	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	
		F																																			
		F																																			

014005

Figure 2-2. Comment Line Coding

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

75-80 PROGRAM IDENTIFICATION

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

## CONTROL CARD SPECIFICATIONS

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

FIELD DEFINITIONS

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

1-2 PAGE

Refer to Section 2 for a complete description.

3-5 LINE

Refer to Section 2 for a complete description.

6 FORM TYPE

This field must be coded with the letter H.

15 DEBUG

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

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

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

**Burroughs**      B 1700 RPG

PROGRAM ID	PROGRAMMER	PAGE	DATE	OF
------------	------------	------	------	----

PAGE	1 2	CONTROL CARD SPECIFICATIONS	PROGRAM IDENTIFICATION	75 80
------	-----	-----------------------------	------------------------	-------

LINE	FORM TYPE		74
3 5 6 7	0 1 H	<div style="display: flex; justify-content: space-around; margin-top: 20px;"> <div style="text-align: center;">A</div> <div style="text-align: center;">B</div> <div style="text-align: center;">C</div> <div style="text-align: center;">D</div> <div style="text-align: center;">E</div> </div>	

- A. 15 Specifies whether the DEBUG operation is to be used during compilation. Entries: Blank or 1.
- B. 17 Specifies the location of the sign in all numeric data items. Entries: Blank or L, or R.
- C. 21 Specifies the punctuation (INVERTED PRINT) to be used for numeric literals. Entries: Blank, I, J, or D.
- D. 41 Causes output lines to be conditioned by the LP indicator. Entries: Blank or 1.
- E. 51 Specifies which RPG dialect to use, RPG I or RPG II. Entries: Blank or 1.

G14006

Figure 3-1. Control Card Specifications Summary Sheet

17 SIGN POSITION

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

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

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

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	Mon/Day/Year
D	5678.90	7,654.32	.60	Day/Mon/Year
I	5678,90	7.654,32	.60	Day.Mon.Year
J	5678,90	7.654,32	0.60	Day.Mon.Year

41 FORMS POSITIONING

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

51 SOURCE INPUT DIALECT

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

75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a complete description.

## FILE DESCRIPTION SPECIFICATIONS

Every file to be used by a RPG Program, except compile-time vector files, must be described to the compiler through the File Description Specifications. Information, such as file-names, hardware devices, record and block lengths, and how the file is to be used, is provided by these specifications.

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

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

FIELD DEFINITIONS

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

1-2 PAGE

Refer to Section 2 for a complete description.

3-5 LINE

Refer to Section 2 for a complete description.

6 FORM TYPE

This field must contain a F.

7-14 FILENAME

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





15 FILE TYPE

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

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

I Input Files

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

O Output Files

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

U Update Files

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

C Combined Files

Combined files are card files which can be used for both input and output. These files consist of cards that are read by the program and are subsequently punched and/or printed as output. The punching will be into the same cards that have been read and output may occur only once per cycle. Combined files must be further described on the Input Specifications and should (not required) be described on the Output-Format Specifications form.

D Display Files

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

16 FILE DESIGNATION

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

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

P Primary File

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

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

S Secondary Files

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

C Chained File

A chained file may be an input, output, or update file assigned to a disk that uses the CHAIN operation code to read or write records randomly. Chained indexed files may only be input or update.

T Table File

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

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

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

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

D Demand File

A demand file may be an update, input, or combined file from which records are read through use of the READ operation code in the Calculation Specifications. Demand files may only be read sequentially.

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, and is used to indicate whether or not the program may end before all of the records from the file are processed. Valid entries for this field are:

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

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

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

18 SEQUENCE

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

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

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

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

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

19 FILE FORMAT

This field specifies whether the file contains fixed- or variable-length records. Variable-length records may not be specified for files other than those assigned to tape. The acceptable entries are:

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

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 1700 in table 4-1.

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

Device	Block Length		Record Length	
	Minimum	Maximum	Minimum	Maximum
CARD (80 col)	Same as record length	Same as record length	1	80
CARD (96 col)	Same as record length	Same as record length	1	96
PRINTER	Same as record length	Same as record length	1	132
DISK	1	9999	1	9999
TAPE	16	9999	1	9999

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

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

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

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

Table 4-3. Default Block and Record Lengths

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

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

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

28 PROCESSING MODE

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

29-30 RECORD ADDRESS FIELD LENGTH

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

31 RECORD ADDRESS TYPE

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

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

32 FILE ORGANIZATION TYPE

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

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

NOTE

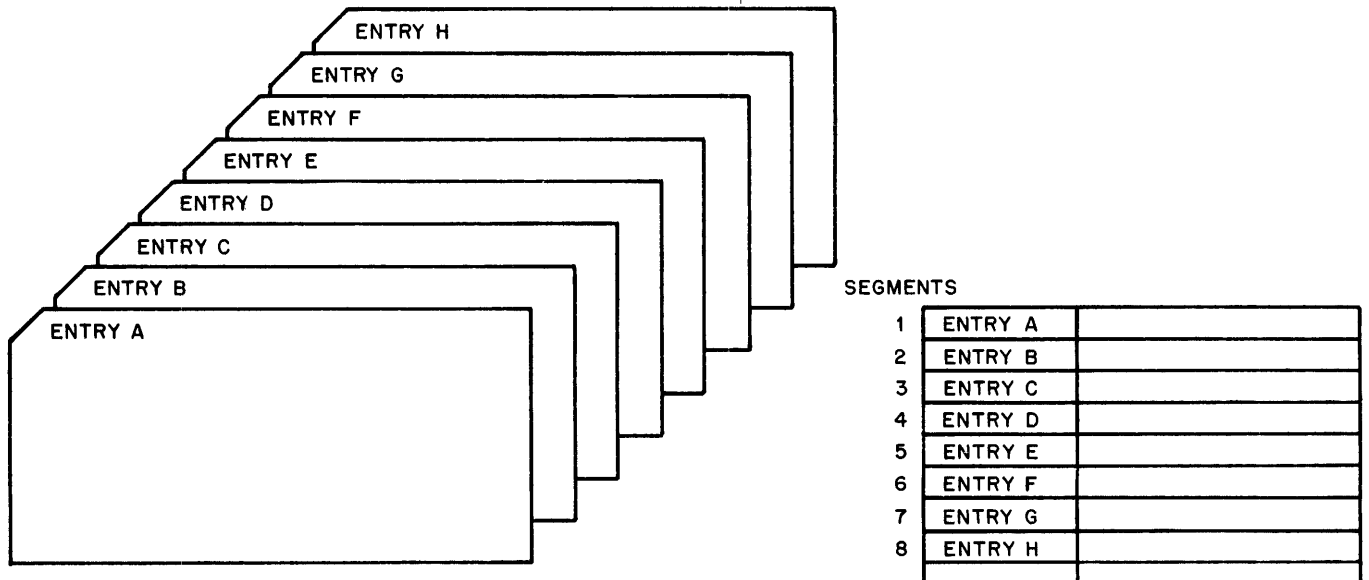
B 1700 RPG organizational techniques are such that file accessing and file organization are independent of each other. This is not so for other RPG implementations.

File Organization

File organization for the B 1700 is discussed in the following paragraphs.

Sequential

A sequentially organized file is one whose records follow one after the other, in ascending addresses. A sequentially organized file usually implies that the records either are in some logical sequence or are going to be sorted into some desired logical sequence (see figure 4-2).



614008

Figure 4-2. A Sequentially Organized Card File On Disk

## Direct

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

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

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

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

## File Accessing Methods

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

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

### Sequential

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

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

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



## Direct

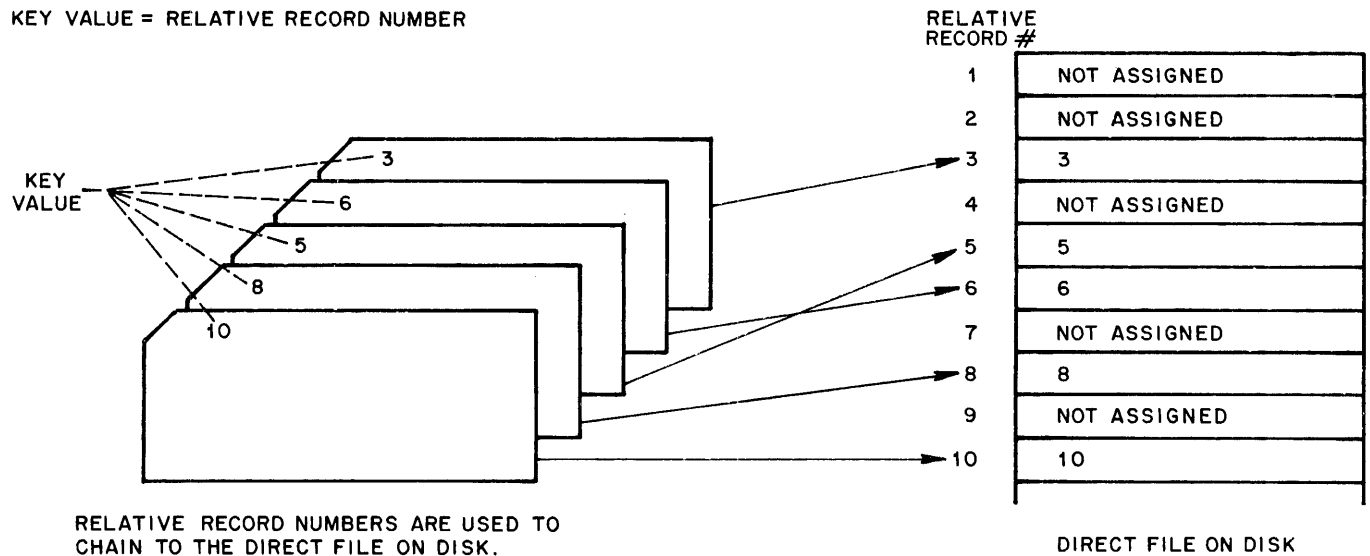
A file created either sequential or direct may be accessed randomly.

Direct accessing of a sequentially organized file will access the records as specified by the value of the key, regardless of the fact that the records were not created that way. The current End-of-File pointer (which points to the last sequential record created) is the maximum permissible value of the key.

Attempting to access a record with a key value greater than the End-of-File pointer will result in a run-time error.

Direct accessing of a randomly organized file will access the records as specified by the value of the key. Records which have never been created can be accessed, and whatever information happens to be in that record area from the previous time that space was used, will be read as the record contents (see figure 4-3).

KEY VALUE = RELATIVE RECORD NUMBER



614009

Figure 4-3. Random Processing of a Direct File

## Indexed-Sequential

An indexed-sequential file is a file whose records are accessed through the use of a core index. Each record contains a data key whose length and location have been assigned by the user (columns 29-39 and 35-38). The core index is a table in memory which contains entries from the key fields of the data file on disk. The use of the core index significantly reduces the time necessary 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. (For more information see CORE INDEX in this section and figure 4-8).

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

#### Additional I/O Areas

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

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

#### 33-34 OVERFLOW INDICATOR

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

Entry	Definition
OA-OG, OV	Specified indicator is used to condition records in the file.
Blank	Default to OF or OV according to entry in columns 40-46 (DEVICE).

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

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

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

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

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 may be omitted. The RECORD ADDRESS FIELD LENGTH (column 29-30) added to the KEY FIELD STARTING LOCATION must not exceed the record length.

39 EXTENSION CODE

This field applies to printer output files and table or pre-execution-time arrays. It indicates whether the file is further described on the Extension Specifications (table or array input files) or Line Counter Specifications (printer files) form. Valid entries for this field are:

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

40-46 DEVICE

This field is used to identify the input/output device to which the file will be assigned. All entries must be left-justified. Valid entries for this field are:

Entry	Definition
READER	80-column card reader.
MFCU1	96-column MFCU - primary hopper.
MFCU2	96-column MFCU - secondary hopper.
PUNCH	80-column card punch.
PRINTER or PRINTR2	Line printer. (A maximum of 5 printer files may be specified).
TAPE	Magnetic tape.
DISK	Disk file.
CONSOLE*	Console printer.
BSCA	Telecommunications file.
DATAKOM	Telecommunications file.
*Only one CONSOLE file may be specified and must be specified as a display file (see DSPLY in Section 9).	

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

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
	Disk	DISK
	Tape	TAPE
Table Files	Card	MFCU1 or MFCU2 READER
	Disk	DISK
	Tape	TAPE
Chained Input Files	Disk	DISK
Update Files	Disk	DISK
Combined Files	Card	MFCU1 or MFCU2
Output Files	Card	MFCU1 or MFCU2 PUNCH
	Disk	DISK
	Listing	PRINTER, PRINTR2
Display File	Listing	CONSOLE

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

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

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

FILE DESCRIPTION SPECIFICATIONS

LINE	FORM TYPE	FILENAME	FILE TYPE	FILE DESIGNATION	FILE FORMAT	SEQUENCE	END OF FILE	PROCESSING MODE	BLOCK LENGTH	RECORD LENGTH	RECORD ADDRESS TYPE	RECORD ADDRESS FIELD LENGTH	KEY FIELD STARTING LOCATION	FILE ORGANIZATION TYPE	OVERFLOW INDICATOR	EXTENSION CODE	DEVICES	NOT USED	FI										
3	5	6	7	14	15	16	17	18	19	20	23	24	27	28	29	30	31	32	33	34	35	38	39	40	46	47	52	53	54
02	F	CARD1		IP	F				96	96														MFCU1					
03	F	CARD2		IS	F				96	96														MFCU2					
04	F	CARD3		IT	F				96	96														EMFCU2					
05	F	CARD4		ID	F				96	96														MFCU2					
06	F	CARD5		Ø	F				96	96														MFCU2					
07	F	CARD6		CP	F				96	96														MFCU1					
08	F	CARD7		CS	F				96	96														MFCU2					
09	F	CARD8		CD	F				96	96														MFCU2					

614010

Figure 4-4. Device Coding Example - MFCU Files

FILE DESCRIPTION SPECIFICATIONS

LINE	FORM TYPE	FILENAME	FILE TYPE	FILE DESIGNATION	FILE FORMAT	SEQUENCE	END OF FILE	PROCESSING MODE	BLOCK LENGTH	RECORD LENGTH	RECORD ADDRESS TYPE	RECORD ADDRESS FIELD LENGTH	KEY FIELD STARTING LOCATION	FILE ORGANIZATION TYPE	OVERFLOW INDICATOR	EXTENSION CODE	DEVICES	NOT USED	FI										
3	5	6	7	14	15	16	17	18	19	20	23	24	27	28	29	30	31	32	33	34	35	38	39	40	46	47	52	53	54
02	F	MESSAGE		Ø	F					10														CONSOLE					
03	F																												
04	F																												
05	F																												
06	F																												
07	F																												
	F																												
	F																												

614011

Figure 4-5. Device Coding Example - Console Files (SPO)

FILE DESCRIPTION SPECIFICATIONS

LINE	FILENAME	FILE TYPE	FILE DESIGNATION	FILE FORMAT	SEQUENCE	END OF FILE	RECORD ADDRESS TYPE	RECORD ADDRESS FIELD LENGTH	PROCESSING MODE	FILE ORGANIZATION TYPE	FILE ORGANIZATION TYPE OVERFLOW INDICATOR	KEY FIELD STARTING LOCATION	EXTENSION CODE	DEVICES	NOT USED	FI														
3	5	6	7	14	15	16	17	18	19	20	23	24	27	28	29	30	31	32	33	34	35	38	39	40	46	47	52	53	54	
02	FLINEOUT								F	132	132			L	PRINTER															
03	VRTOUT								F	132	132			R	PRINTR2															
04	F																													
05	F																													
06	F																													
07	F																													
	F																													
	F																													

614012

Figure 4-6. Device Coding Example - Printer Files

FILE DESCRIPTION SPECIFICATIONS

LINE	FILENAME	FILE TYPE	FILE DESIGNATION	FILE FORMAT	SEQUENCE	END OF FILE	RECORD ADDRESS TYPE	RECORD ADDRESS FIELD LENGTH	PROCESSING MODE	FILE ORGANIZATION TYPE	FILE ORGANIZATION TYPE OVERFLOW INDICATOR	KEY FIELD STARTING LOCATION	EXTENSION CODE	DEVICES	NOT USED	FI															
3	5	6	7	14	15	16	17	18	19	20	23	24	27	28	29	30	31	32	33	34	35	38	39	40	46	47	52	53	54		
02	TAPE1								F	400	100			T	TAPE																
03	TAPE2								F	400	100			T	TAPE																
04	TAPE3								F	400	100			E	TAPE																
05	TAPE4								F	400	100			T	TAPE																
06	TAPE5								F	400	100			T	TAPE																
07	F																														
	F																														
	F																														

614013

Figure 4-7. Device Coding Example - Tape Files

## DATAKOM

If communication with a remote device is desired, the entry DATAKOM or BSCA is used. The DATAKOM file can be an input, output, or combined file. If DATAKOM or BSCA is specified, a Telecommunications Card must be declared preceding the Input Specifications (see Section 7).

DATAKOM files are used to reference NDL (Network Definition Language) files in the Network Control. The filename used in columns 7-14 must be the same as the NDL file it is referencing or the names should be file equated with a FILE statement. For information on the NDL Language refer to the B 1700 Systems Network Definition Language, Form No. 1073715. For information on the FILE statement, see the B 1700 System Software Operational Guide, Form No. 1068731.

53

## LABELS

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

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

The B 1700 Standard Label consists of one 80 character record, formatted as follows:

Position	Field Description
1	blank
2- 8	"LABEL "
9	zero
10-16	"multi-file-id" or zeros
17	zeros
18-24	"file-id"
25-27	reel number within a tape file
28-32	date written (creation date, YYDDD)
33-34	cycle
35-39	purge date (YYDDD)
40	sentinel
41-45	block count
46-52	record count
53	memory dump key
54-58	serial number
59-63	creating system
64-66	binary buffer size
67-69	binary record size
70-80	reserved

60-65 CORE INDEX

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

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

Once the core index is built, the program will begin processing. As an access occurs, the code emitted will do a binary search of the core index to determine in which partition the record should be found. It calculates the lowest actual address and one greater than the highest to be used as the area to be searched for the record. At this point two techniques are possible. The program will always activate the disk SCAN operator first, then will revert to a binary search technique when necessary (see figure 4-8).

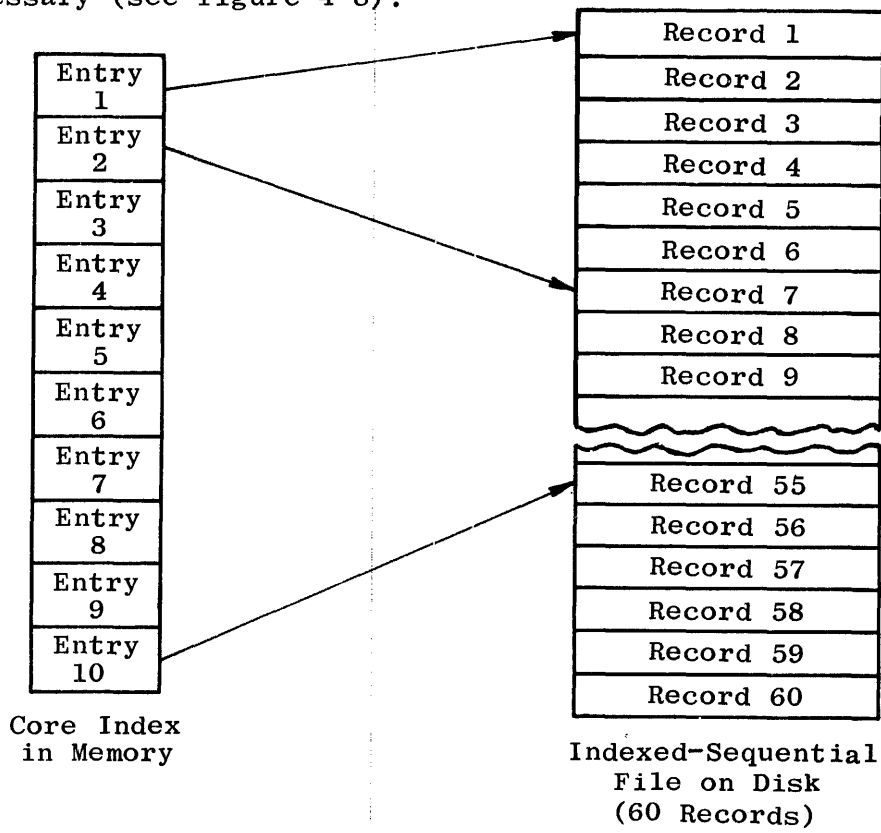


Figure 4-8. Core Index Selection of Data Keys



Several considerations should be made in space/time trade-offs when processing indexed-sequential files. The core index should reflect the size of the data file to be accessed and the number of accesses to be performed. When a significant number of accesses are to be performed, the core index should be as large as practical; however, the core index should be moderate-to-small if only a few accesses are desired. When input is in sequential order, many disk accesses could be avoided, especially in the case of the binary search technique, when a large blocking factor is used.

66 FILE ADDITION/UNORDERED

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

- a. New records are to be added to an existing file; or
- b. Unordered records are to be loaded into an output file.

Valid entries for this field are listed below:

Entry	Definition
A	Records are to be added to the file.
U	Records are to be loaded for an indexed file in unordered sequence.
Blank	Output file (must be ordered).

Records may be added to an input file. Records added to a sequential or indexed file are added at the end of the file. After all records have been added to an indexed file, the file is sorted so that the records are in ascending order. Records added to an indexed file need not be in ascending order.

An A entry in this field is required if there is an ADD entry in columns 16-18 of the Output-Format Specifications record description for the file. If A is in this entry for any file other than an update file, all output records must have ADD specified on the Output Specifications (columns 16-18). However, Output Specifications are optional.

Columns 15 and 66 coding options are discussed in the following paragraphs.

If column 15 contains a 0 and column 66 is blank, the records are loaded in ascending key sequence to an indexed file.

If column 15 contains a 0 and column 66 contains a U, the records are loaded in unordered key sequence to an indexed file.

If column 15 contains a 0 and column 66 contains an A, the records are added to an existing indexed file. An ADD entry is required in columns 16-18 of the Output Specifications.

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

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

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

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

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

## 70 TAPE REWIND

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

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

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

### Card Input

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

### Card Output

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

FILE DESCRIPTION SPECIFICATIONS

FORM TYPE	LINE	FILENAME	FILE TYPE	FILE FORMAT SEQUENCE	END OF FILE	FILE DESIGNATION	FILE TYPE	PROCESSING MODE	BLOCK LENGTH	RECORD LENGTH	RECORD ADDRESS TYPE	RECORD ADDRESS FIELD LENGTH	FILE ORGANIZATION TYPE	OVERFLOW INDICATOR	KEY FIELD STARTING LOCATION	EXTENSION CODE	DEVICES	NOT USED	NOT USED	CORE INDEX	FILE ADDITION/UNORDERED	LABELS	TAPE REWIND	FILE CONDITION	NOT USED																
																										3	5	6	7	14	15	16	17	18	19	20	23	24	27	28	29
SEQUENTIAL	02	DSKFI1						F	180	180						DISK																									
	03	F																																							
INDEXED	04	DISKFI2						F	180	180			AI			DISK																									
	05	DISKFI3						F	180	180			AI			DISK																									
	06	F																																							
DIRECT	07	DISKFI4						F	180	180						DISK																									
		F																																							
		F																																							

G14015

Figure 4-9. Processing Methods of Disk Files - Loading

FILE DESCRIPTION SPECIFICATIONS

FORM TYPE	LINE	FILENAME	FILE TYPE	FILE FORMAT SEQUENCE	END OF FILE	FILE DESIGNATION	FILE TYPE	PROCESSING MODE	BLOCK LENGTH	RECORD LENGTH	RECORD ADDRESS TYPE	RECORD ADDRESS FIELD LENGTH	FILE ORGANIZATION TYPE	OVERFLOW INDICATOR	KEY FIELD STARTING LOCATION	EXTENSION CODE	DEVICES	NOT USED	NOT USED	CORE INDEX	FILE ADDITION/UNORDERED	LABELS	TAPE REWIND	FILE CONDITION	NOT USED																
																										3	5	6	7	14	15	16	17	18	19	20	23	24	27	28	29
SEQUENTIAL	02	DISK1						F	180	180						DISK																									
	03	F																																							
INDEXED - SEQUENTIAL	04	DISK2						F	180	180			AI			DISK																									
	05	F																																							
	06	F																																							
	07	F																																							
		F																																							
		F																																							

G14016

Figure 4-10. Processing Methods of Disk Files - ADD Records Only

FILE DESCRIPTION SPECIFICATIONS

LINE	FORM TYPE	FILE DESIGNATION FILE TYPE	FILE FORMAT SEQUENCE	END OF FILE	RECORD ADDRESS FIELD LENGTH	RECORD ADDRESS TYPE	FILE ORGANIZATION TYPE OVERFLOW INDICATOR	KEY FIELD STARTING LOCATION	EXTENSION CODE	LABELS	FILE ADDITION/UNORDERED	CORE INDEX	FILE CONDITION																																						
													TAPE REWIND	NOT USED	NOT USED	NOT USED	NOT USED	NOT USED	NOT USED	NOT USED	NOT USED	NOT USED	NOT USED	NOT USED																											
3	5	6	7	14	15	16	17	18	19	20	23	24	27	28	29	30	31	32	33	34	35	38	39	40	46	47	52	53	54	59	60	65	66	67	68	69	70	71	72	73	74										
02	F	DIRFI1	IC	F	180	180	AI																																												
03	F	DIRFI2	UC	F	180	180	AI																																												
04	F																																																		
05	F	DIRFI3	IC	F	180	180	AI																																												
06	F	DIRFI4	UC	F	180	180	AI																																												
07	F																																																		
	F																																																		
	F																																																		

FILE DESCRIPTION SPECIFICATIONS

LINE	FORM TYPE	FILE DESIGNATION FILE TYPE	FILE FORMAT SEQUENCE	END OF FILE	RECORD ADDRESS FIELD LENGTH	RECORD ADDRESS TYPE	FILE ORGANIZATION TYPE OVERFLOW INDICATOR	KEY FIELD STARTING LOCATION	EXTENSION CODE	LABELS	FILE ADDITION/UNORDERED	CORE INDEX	FILE CONDITION																																							
													TAPE REWIND	NOT USED	NOT USED	NOT USED	NOT USED	NOT USED	NOT USED	NOT USED	NOT USED	NOT USED	NOT USED	NOT USED																												
3	5	6	7	14	15	16	17	18	19	20	23	24	27	28	29	30	31	32	33	34	35	38	39	40	46	47	52	53	54	59	60	65	66	67	68	69	70	71	72	73	74											
02	F	DISK1	IC	F	180	180																																														
03	F	DISK2	UC	F	180	180																																														
04	F																																																			
05	F																																																			
06	F																																																			
07	F																																																			
	F																																																			
	F																																																			

G14017

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

FILE DESCRIPTION SPECIFICATIONS

LINE	FILENAME	FILE TYPE	FILE FORMAT	SEQUENCE	END OF FILE	FILE DESIGNATION	FILE TYPE	RECORD ADDRESS TYPE	RECORD ADDRESS FIELD LENGTH	PROCESSING MODE	BLOCK LENGTH	RECORD LENGTH	FILE ORGANIZATION TYPE	OVERFLOW INDICATOR	KEY FIELD STARTING LOCATION	EXTENSION CODE	DEVICES	LABELS	FILE ADDITION/UNORDERED	CORE INDEX	TAPE REWIND	FILE CONDITION
02	DISK1	IP	F	180	180											DISK						
03	DISK2	IS	F	180	180											DISK						
04	DISK3	IT	F	180	180											DISK						
08	DISK4	VP	F	180	180											DISK						
06	DISK5	US	F	180	180											DISK						
07	DISK6	ID	F	180	180											DISK						
	DISK7	UD	F	180	180											DISK						

Figure 4-12. Processing Methods of Disk Files - Sequential

Tape Input

The effects of the various options for tape input are:

Entry	Effect
U	Releases the input areas, rewinds the tape, and the MCP marks the unit "not ready".
N	Same as "blank", except the tape is not rewound.
Blank	Releases the input areas, rewinds and returns the unit to the MCP.

Tape Output

The effects of the various options for tape output are:

Entry	Effect
P	Releases the output areas, writes the trailer label, rewinds the tape, and overwrites the label, thus making the tape a scratch tape.
U	Releases the output areas, writes the trailer label, rewinds the tape, and the MCP marks the unit "not ready".
N	Same as "blank", except the tape is not rewound.
Blank	Releases the output areas, writes the trailer label, rewinds the tape and returns the unit to the MCP.

### Printer Output

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

### Disk Files

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

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

### 71-72 FILE CONDITION

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

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

The user is required to set the external indicators at the beginning of program execution. The indicators are entered on the console and once set they cannot be modified.

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

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

#### 75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a complete description.

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

FILE DESCRIPTION SPECIFICATIONS

LINE	FILENAME	BLOCK LENGTH	RECORD LENGTH	KEY FIELD STARTING LOCATION	DEVICE	NOT USED	NOT USED	CORE INDEX	NOT USED	NOT USED	NOT USED
02	INPUT	80	80	2	READER						
03	DIRECT	430	10	2	DISK						
04	INDEXED	170	17	4	DISK			400			

INPUT SPECIFICATIONS

LINE	FILENAME	OPTION	RECORD IDENTIFYING INDICATOR	PACKED	MATCHING FIELDS OR CHAINING FIELDS	FIELD RECORD RELATION
01	INPUT	AA	99			
02					50 DIRKEY	
03					14 INXKEY	
04					10 SHØRT	
05					17 LØNG	

CALCULATION SPECIFICATIONS

LINE	INDICATORS	FACTOR 1	OPERATION	FACTOR 2	RESULT FIELD	FIELD LENGTH	COMMENTS
01		DIRKEY	CHAIN	DIRECT			
02							

OUTPUT - FORMAT SPECIFICATIONS

LINE	FILENAME	TYPE	SKIP	OUTPUT INDICATORS	FIELD NAME (VARIABLE NAME)	CONSTANT OR EDIT WORD	NOT USED
01	DIRECT	D		99			
02					SHØRT	10	
03	INDEXED	D		99			
04					LØNG	17	

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



**FILE DESCRIPTION SPECIFICATIONS**

LINE	FILENAME	BLOCK LENGTH	RECORD LENGTH	KEY FIELD STARTING LOCATION	DEVICE	NOT USED	NOT USED	CORE INDEX
02	F INPUT	80	80	2	READER			
03	F DIRECT	430	10	2	DISK			
04	F INDEXED	170	17	4	DISK			4000

**INPUT SPECIFICATIONS**

LINE	FILENAME	OPTION	RECORD IDENTIFYING INDICATOR	PACKED	MATCHING FIELDS OR CHAINING FIELDS	CONTROL LEVEL	FIELD RECORD RELATION
01	I INPUT	AA	99				
02					1		50 DIRKEY
03					11		14 INXKEY
04					1		10 SHØRT
05					1		17 LØNG
06	I DIRECT	BB	98				
07					1		50 DIRKEY
08	I INDEXED	CC	77				
09					11		14 INXKEY

**CALCULATION SPECIFICATIONS**

LINE	INDICATORS	FACTOR 1	OPERATION	FACTOR 2	RESULT FIELD	FIELD LENGTH	COMMENTS
01	C	INXKEY	CHAIN	INDEXED			
02	C	DIRKEY	CHAIN	INDIRECT			

**OUTPUT - FORMAT SPECIFICATIONS**

LINE	FILENAME	TYPE	SKIP	OUTPUT INDICATORS	FIELD NAME (VARIABLE NAME)	CONSTANT OR EDIT WORD
01	O DIRECT	D	98			
02					SHØRT	10
03	O INDEXED	D	77			
04					LØNG	17

G14020

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

# VECTORS AND EXTENSION SPECIFICATIONS

## VECTORS

Vectors are tables and arrays which are logical configurations of data elements having similar characteristics. Each element of a vector must be of the same length and data type (numeric or alphanumeric). If numeric, each element must have the same number of decimal positions. A maximum of 64 vectors is allowed.

There is very little distinction made between tables and arrays. Where differences do arise in their characteristics, they will be referred to individually as tables or arrays; otherwise they will both be referred to as vectors. The following are the only differences that arise between tables and arrays. These differences exist mainly for compatibility.

- a. Every vector to be used by the program must be given a name. All table names must begin with the letters TAB, whereas array names may begin with any alphabetic character.
- b. Both tables and arrays may use indices to access specific elements within them. If, however, a table name appears without an index assigned, it refers to the last item referenced in the table. If an array name appears without an index assigned, it refers to the entire array. Such a reference specifies that the designated operation be performed repetitively on each element of the array.

Vectors can be distinguished according to the time they are loaded with data, which may occur at three distinct times: during compilation, at the beginning of program execution, or during program execution. Data elements in a vector may, however, be altered at any time during program execution, regardless of the time the vector was loaded.

## COMPILE-TIME VECTORS

For a compile-time vector load, the data to be loaded is read into the program storage area reserved by the entries in the Extension Specifications at the same time that the source program is compiled. Physically, the data is placed in a card file called "RPG/VECTOR", which immediately follows the source deck (see figure 5-4).

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.

**PRE-EXECUTION-TIME VECTORS**

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

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

**DYNAMIC/EXECUTION-TIME VECTORS**

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

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

**EXTENSION SPECIFICATIONS**

All vectors in a program must be described on the Extension Specifications sheet. Certain entries are required for all types of vectors, regardless of the time at which they will be loaded (see figure 5-1). Columns 27-45 must be completed for each vector (columns 46-57 if the vector is loaded in alternating format with another vector). These columns specify the name assigned to identify the vector (VECTOR NAME), the number of vector elements occurring in each input record (ENTRIES PER RECORD), the size of the vector (ENTRIES PER TABLE OR ARRAY and LENGTH OF ENTRY), whether the input data is in packed decimal format (PACKED), the number of decimal positions in each entry (DECIMAL POSITIONS), and the order in which the elements are sequenced (SEQUENCE).

PAGE		EXTENSION SPECIFICATIONS											PROGRAM IDENTIFICATION																						
FORM TYPE		NOT USED											IDENTIFICATION																						
LINE	FORM TYPE	FROM FILENAME	TO FILENAME	TABLE OR ARRAY NAME (VECTOR NAME)	ENTRIES PER RECORD	ENTRIES PER TABLE OR ARRAY	LENGTH OF ENTRY	DECIMAL POSITIONS	SEQUENCE	DECIMAL POSITIONS	SEQUENCE	TABLE OR ARRAY NAME (VECTOR NAME)	LENGTH OF ENTRY	DECIMAL POSITIONS	SEQUENCE	COMMENTS																			
3	5	6	7	8	9	10	11	18	19	26	27	32	33	35	36	39	40	42	43	44	45	46	51	52	54	55	56	57	58	74					
01	E																																		
02	E																																		
03	E																																		
04	E																																		
05	E																																		
06	E																																		
07	E																																		
08	E																																		
09	E																																		
10	E																																		

614021

Figure 5-1. Entries Necessary to Describe a Vector

Additional entries are required if the vector is to be loaded at execution time (FROM FILENAME), or if the vector is to be written to an output file at End-of-Job (TO FILENAME). Any files named in these fields must also be described on the File Description Specifications.

For dynamic vectors, the ENTRIES PER RECORD, FROM FILENAME, and TO FILENAME fields must be left blank (dynamic vectors cannot be output at EOJ). Also, a second vector in alternating format (columns 46-57) cannot be declared.

#### FIELD DEFINITIONS

Figure 5-2 can be used in conjunction with the following field definitions for the Extension Specifications.

1-2 PAGE

Refer to Section 2 for a complete description.

3-5 LINE

Refer to Section 2 for a complete description.

6 FORM TYPE

An E must appear in this field.

11-18 FROM FILENAME

This field is used to name a vector file and must contain the FILENAME of every pre-execution-time vector to be used by the program. The file must have been specified on the File Description Specifications as an input table file. If the vector is loaded at compile time or via Input or Calculation Specifications during 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 deck 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 file deck must follow the source program deck (see figure 5-4).

19-26 TO FILENAME

If it is desirable to write or punch out any vector file, the filename of the proper output file must be entered in this field. The designated output file must have been previously described on the File Description Specifications form as a sequential output file. A vector can be written to only one output device, and is automatically written or punched at End-of-Job after all other records have been processed. More than one vector may be written to the same output file but it is the users responsibility to ensure that this is a meaningful thing to do. Filenames must always be entered in this field left-justified.

## Burroughs

B 1700 RPG

PROGRAM ID											PROGRAMMER											PAGE		DATE		OF							
EXTENSION SPECIFICATIONS											75		80																				
LINE	FORM TYPE	1	2	NOT USED							CHAINING FIELD NUMBER		TABLE OR ARRAY NAME (VECTOR NAME)		ENTRIES PER RECORD	ENTRIES PER TABLE OR ARRAY		LENGTH OF ENTRY	SEQUENCE		DECIMAL POSITIONS PACKED		TABLE OR ARRAY NAME (VECTOR NAME)		LENGTH OF ENTRY	SEQUENCE		DECIMAL POSITIONS PACKED		PROGRAM IDENTIFICATION		COMMENTS	
3	5	6	7	8	9	10	11	18	19	26	27	32	33	35	36	39	40	42	43	44	45	46	51	52	54	55	56	57	58	74			
01	E																																
02	E																																
03	E																																
04	E																																
05	E																																
06	E																																
07	E																																
08	E																																
09	E																																
10	E																																

A
B
C
D
E
F
G
H
I
J

- A. 11-18 Contains the name of a pre-execution time vector file.
- B. 19-26 Contains the name of the file to which vector will be outputted. If blank, vector not to be output at EOJ.
- C. 27-32 Contains the name of the input table or array. Entries: TABXXX (X=any alphanumeric character) or 1-6 alphanumeric characters. Also used to name the first of two alternating vectors.
- D. 33-35 Specifies the exact number of entries contained in each vector input record.
- E. 36-39 Specifies the maximum number of items contained in the first vector named. Entry maximum: 4095, right-justified.
- F. 40-42 Specifies the length (in bytes) of each element in the vector. Maximum numeric entry is 31, maximum alphanumeric entry is 511, right-justified.
- G. 43 Specifies if the external vector elements are in packed or unpacked decimal format. Entries: Blank or P.
- H. 44 Specifies the number of decimal positions contained in each element. Entries: Blank or 0-9.
- I. 45 Specifies the sequence in which elements will be loaded for the first vector. Entries: Blank, A or D.
- J. 46-57 These columns are used to describe the second of two alternating vectors. Entries are of the same type as specified for the first vector. The second vector is loaded in alternating format with the first.

G14022

Figure 5-2. Extension Specifications Summary Sheet

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 must contain a blank or a non-zero integer and specify the exact number of entries contained in each vector input record. Every record, except the last one, must contain the same number of entries as specified in field; the last record may contain fewer entries than specified. Entries in this field must be right-justified; leading zeros may be omitted. Corresponding items from related (alternating) vectors must be on the same record, and in alternating format. Each pair of items is considered one entry. (For more information, see columns 46-57 in this section.)

If ENTRIES PER RECORD are specified in alternating format, the entry must satisfy certain conditions. The following formula may be used to determine if the entry satisfies these conditions:

If	L1 = Length entered in columns 40-42 unless vector 1 is packed, when $L1=1 + (\text{length in columns 40-42})/2$ ;
and	L2 = Length of any alternate vector as entered in columns 52-54 unless packed, when $L2=1 + (\text{length in columns 52-54})/2$ ;
and	N = Number of entries per record;
and	R1 = The record length of any TO file;
and	R2 = The record length of the FROM file, if a FROM file is specified, else 96;
Then,	$R1 \geq (L1 + L2) \times N$ $R2 \geq (L1 + L2) \times N$

The FROM FILENAME and ENTRIES PER RECORD entries are used to determine the type of vector and therefore how it will be loaded. (For more information see VECTOR LOADING in this section.) Table 5-1 is a guide for determining vector type.

Table 5-1. Guide for Determining Vector Type

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

36-39 ENTRIES PER VECTOR

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

40-42 LENGTH OF ENTRY

This field is used to specify the length (in bytes) of each element in the vector named in the first VECTOR NAME field (columns 27-32). For numeric vectors in packed decimal format, enter the number of digits. Entries in this field must be right-justified; leading zeros may be omitted.

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

The maximum length of a numeric vector element is 31 characters; the maximum for an alphanumeric vector element is 511 characters. However, an element must be completely contained on one record; therefore, input record sizes will also limit the maximum element sizes.

43 PACKED

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

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

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

44           DECIMAL POSITIONS

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

The acceptable entries for this field are:

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

45           SEQUENCE

This field is used to specify the sequence in which elements will be loaded for the vector named in the first VECTOR NAME field (columns 27-32). A sequence error during a compile time vector load will not be detected; however, it can cause an error or "not found" condition during program execution. A vector loaded at pre-execution time is checked for the specified sequence. A sequence error causes the program to be discontinued. Note that the sequence check does allow two consecutive elements to be equal. This column must contain an entry if high or low LOKUP is to be used. Alternate vectors need not have the same sequence.

Valid entries for this field are:

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

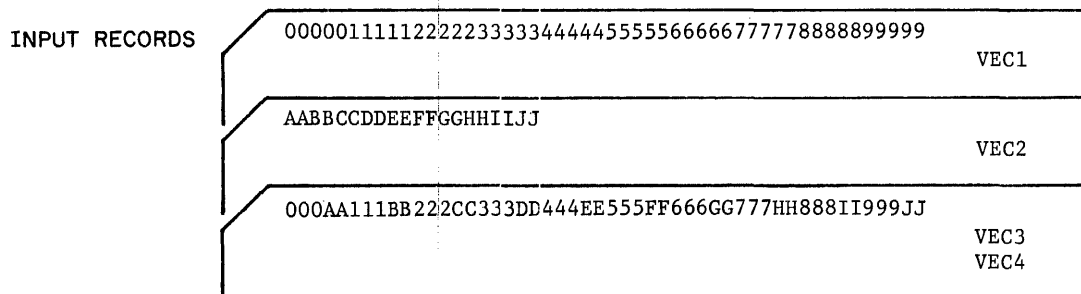
46-57       VECTOR NAME, LENGTH OF ENTRY, PACKED, DECIMAL POSITIONS, SEQUENCE

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



Vectors specified as occurring in alternating format have related elements contained in alternating format on each input record (see figure 5-3). The two vectors need not be of the same size (LENGTH OF ENTRY), type (numeric or alphanumeric), or sequence order, but they must have the same number of elements contained on each input record, and each vector must contain the same number of elements (NUMBER OF ENTRIES PER TABLE OR ARRAY). Each pair of elements on the input record is considered one entry. The first element of each pair belongs to the vector described in columns 27-45; the second element belongs to the vector described in columns 46-57.

PAGE		FORM TYPE		EXTENSION SPECIFICATIONS																									
1 2		NOT USED		CHAINING FIELD NUMBER																									
LINE				FROM FILENAME	TO FILENAME	TABLE OR ARRAY NAME (VECTOR NAME)	ENTRIES PER RECORD	ENTRIES PER TABLE OR ARRAY	LENGTH OF ENTRY	SEQUENCE DECIMAL POSITIONS PACKED	TABLE OR ARRAY NAME (VECTOR NAME)	LENGTH OF ENTRY	SEQUENCE DECIMAL POSITIONS PACKED																
3	5	6	7	8	9	10	11	18	19	26	27	32	33	35	36	39	40	42	43	44	45	46	51	52	54	55	56	57	58
01	E			VECTØR1		VEC1	10	10	5	2																			
02	E			VECTØR2		VEC2	10	10	2																				
03	E					VEC3	10	10	3	2	VEC4												2						



G14023

Figure 5-3. Vectors in Alternating Format

58-74 COMMENTS

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

75-80 PROGRAM IDENTIFICATION

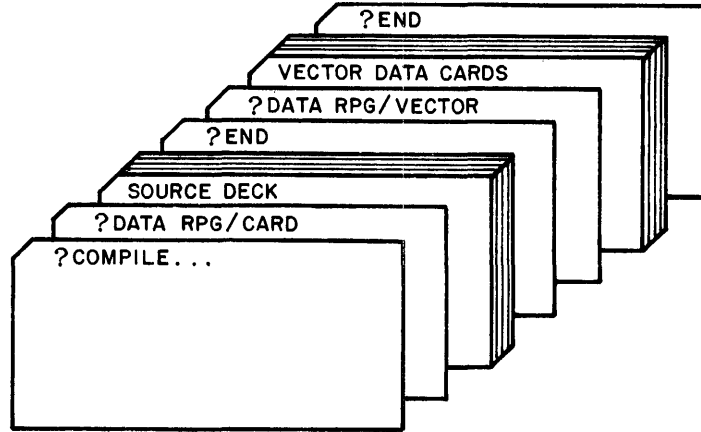
Refer to Section 2 for a complete description.

VECTOR LOADING

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

### COMPILE-TIME VECTOR LOAD

To load a compile-time vector, all that is necessary is to include the data cards for the vector in the proper order in the file "RPG/VECTOR" (figure 5-4). The compiler will automatically read this file after the source program has been read, and will load the data into the vectors declared.



G14024

Figure 5-4. 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 Specification form. The compiler reads data cards and stores the elements in the first vector until it is full, then proceeds to do the same for the second vector, and so forth. If more or fewer records are present than are necessary to exactly fill a vector, data will be placed into the wrong vector. See figure 5-5 for an example of a compile-time vector load declaration.

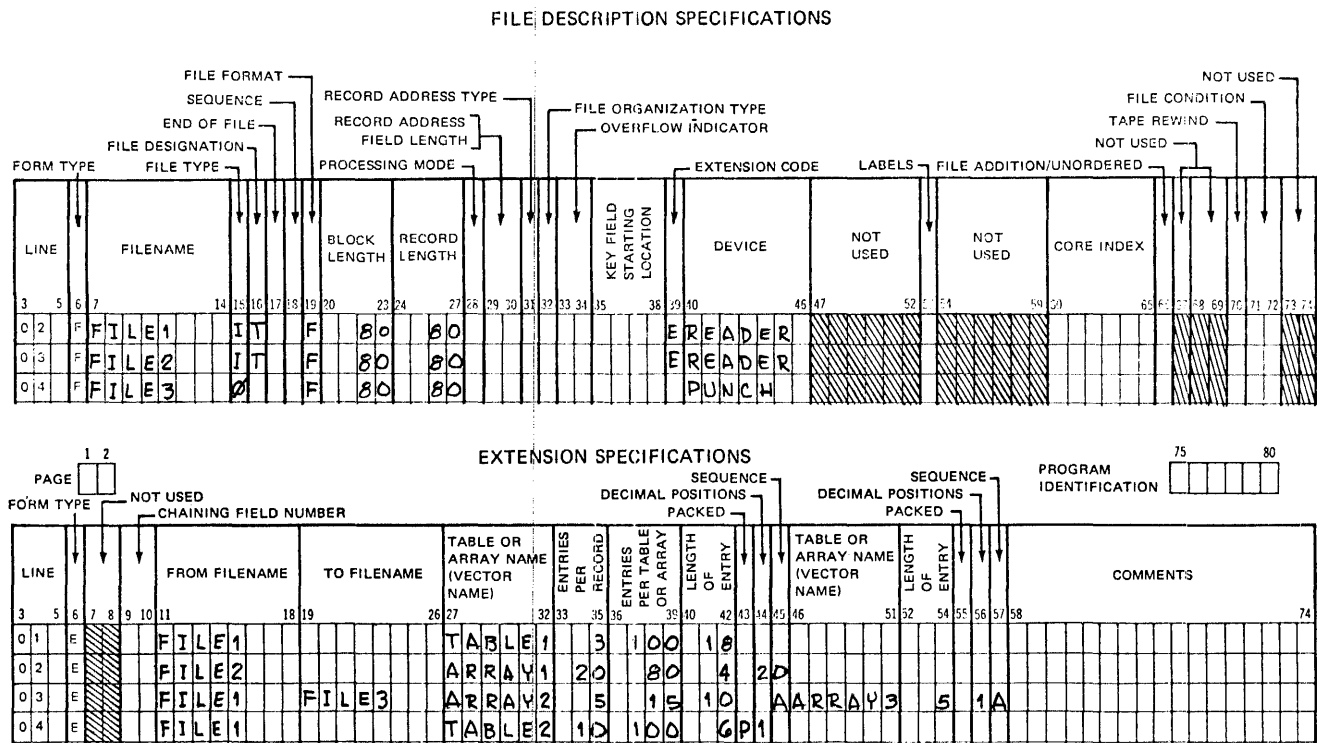
PAGE	FORM TYPE		NOT USED CHAINING FIELD NUMBER	EXTENSION SPECIFICATIONS													PROGRAM IDENTIFICATION														
	1	2		DECIMAL POSITIONS PACKED						DECIMAL POSITIONS PACKED							75	80													
LINE	3	5	6	7	8	9	10	11	18	19	26	27	32	33	35	36	39	40	42	43	44	45	46	51	52	54	55	56	57	58	74
	FROM FILENAME	TO FILENAME	TABLE OR ARRAY NAME (VECTOR NAME)	ENTRIES PER RECORD	ENTRIES PER TABLE OR ARRAY	LENGTH OF ENTRY	SEQUENCE	TABLE OR ARRAY NAME (VECTOR NAME)	LENGTH OF ENTRY	SEQUENCE	COMMENTS																				
01	E										PUNCH1	ARRAY1	25	50	3	A															
02	E											ARRAY2	15	60	4																
03	E											TABLE1	4	100	15	A	TABLE2	5	2A												
04	E																														

G14025

Figure 5-5. Compile-Time Vector Load

## PRE-EXECUTION TIME VECTOR LOAD

To load a pre-execution time vector, the data to be loaded must be placed in the file described in the File Description Specifications to which the vector is assigned (FROM FILENAME). Data is read in at the beginning of program execution and placed in the vector until the vector is full. If more than one vector (not in alternating format) is assigned to a single file, some special considerations must be taken into account (see figure 5-6). Vectors are still loaded in the same order that they are specified in the Extension Specifications, so that files will be opened, read, and closed as necessary to load the designated vectors. If two vectors are assigned to the same table file, and no other pre-execution time vector declaration comes between them, the data for both must be in the same card file. No separators are allowed between the data decks, so that restrictions are imposed the same as those for compile-time vector loads.



G14026

Figure 5-6. Pre-Execution Time Vector Load

## DYNAMIC/EXECUTION TIME VECTOR LOAD

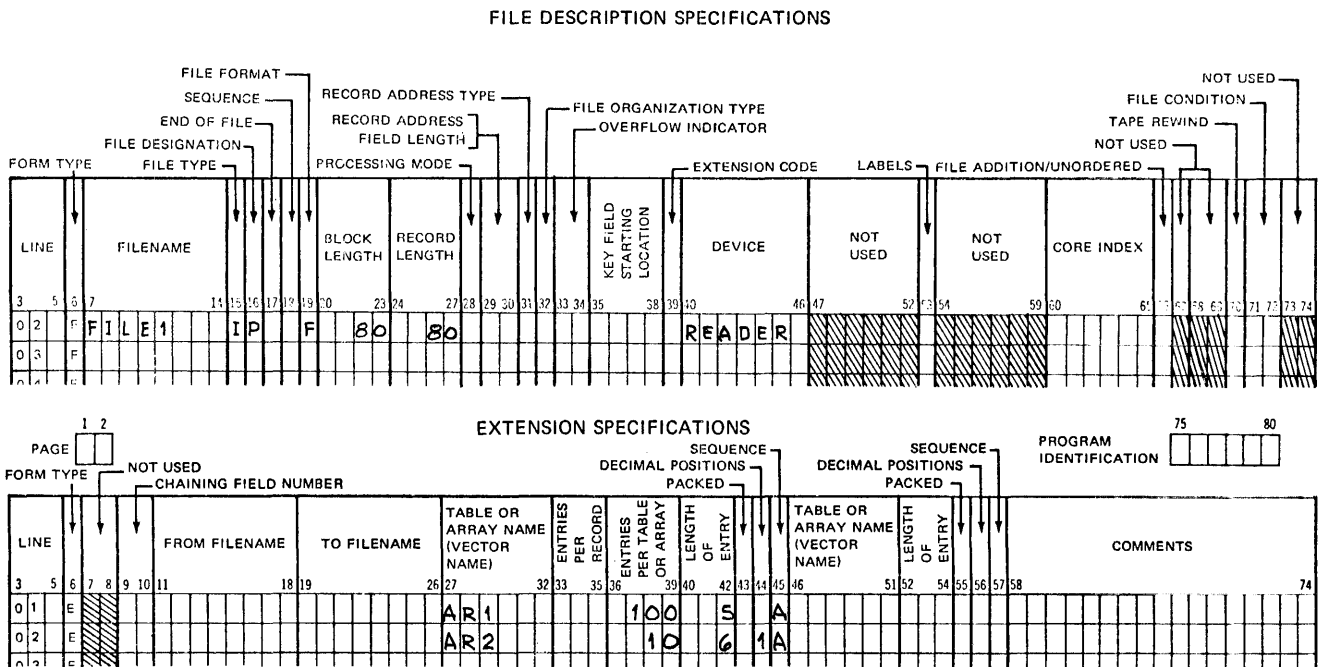
To load a dynamic vector, the data elements may be obtained from fields within input records or from the result of operations in the Calculation Specifications.

Input Specifications Load

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

Calculation Specifications Load

Any operation which specifies a vector (with or without an index) as the RESULT FIELD will cause the designated vector element (or entire array, if no index is specified) to be loaded with the result of the operation. See figure 5-7 for an example of a load via the Calculation Specifications. Refer to Section 9 for the operations which may be specified for vectors.



G14027

Figure 5-7. Dynamic Vector Load (Sheet 1 of 2)

INPUT SPECIFICATIONS

PAGE 1 2  
PROGRAM IDENTIFICATION 75 80

LINE	FORM TYPE	FILENAME	RECORD IDENTIFICATION CODES			FIELD LOCATION		FIELD NAME (VARIABLE NAME)	FIELD RECORD RELATION			NOT USED		
			POSITION	NOT (N) C/Z/D CHARACTER	POSITION	NOT (N) C/Z/D CHARACTER	FROM		TO	PLUS	MINUS		ZERO OR BLANK	
01	I	FILE1	14	AA	18	01	24	1	CT					
02	I						2	6	AR1,1					
03	I						8	67	AR2					
04	I	*												
05	I	*	WHEN A RECORD FROM FILE1 WITH A "T" IN POSITION 1 IS READ,											
06	I	*	THE 01 INDICATOR WILL BE TURNED ON. THE 5 CHARACTERS											
07	I	*	BEGINNING IN POSITION 2 WILL BE LOADED INTO THE FIRST ELEMENT											
08	I	*	OF THE ARRAY AR1. THE 60 CHARACTERS BEGINNING IN POSITION 8											
09	I	*	WILL BE LOADED INTO THE ENTIRE ARRAY AR2.											
10	I													

CALCULATION SPECIFICATIONS

PAGE 1 2  
PROGRAM IDENTIFICATION 75 80

LINE	FORM TYPE	INDICATORS			FACTOR 1	OPERATION	FACTOR 2	RESULT FIELD	FIELD LENGTH	COMMENTS
		AND	AND	AND						
01	C	01			AR1,1	ADD	AR2	AR2		
02	C	*			ADD THE FIRST ELEMENT OF AR1 TO EVERY ELEMENT OF AR2					
03	C	01			M0VE 0		IX	30		
04	C	*			DEFINE THE FIELD "IX" (3 LONG, 0 DECIMALS) & SET IT TO ZERO					
05	C	01			LOOP		TAG			
06	C	*			DEFINE THE LABEL "LOOP"					
07	C	01			1	ADD	IX	IX		
08	C	*			INCREMENT IX BY 1					
09	C	01			10	COMP	IX		02	
10	C	*			IF IX > 10 TURN INDICATOR 02					
11	C	01	NO2		AR2,IX	DIV	3.1415	AR2,IX		
12	C	*			IF THE 02 INDICATOR IS OFF, DIVIDE 3.1415 INTO THE ELEMENT OF					
13	C	*			AR2 POINTED TO BY IX.					
14	C	01	NO2		GOTO LOOP					
15	C	*			IF THE 02 INDICATOR IS OFF, BRANCH TO "LOOP"					
16	C				'E0J'	DSPLY				
17	C	*			DISPLAY "E0J" MESSAGE					

G14028

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

## VECTOR OUTPUT

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

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

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

# LINE COUNTER SPECIFICATIONS

Line Counter Specifications are optional and are used only for printer files to indicate form length, the position of the last line to be printed on each page, and the format of the carriage control tape. If Line Counter Specifications are absent for a printer file, a form length of 66 lines (11 inches long at 6 lines per inch) and an overflow line of 60 (one inch from bottom of form at 6 lines per inch) are assumed.

## FIELD DEFINITIONS

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

1-2 PAGE

Refer to Section 2 for a complete description.

3-5 LINE

Refer to Section 2 for a complete description.

6 FORM TYPE

This field must contain the letter L.

7-14 FILENAME

This field is used to specify the name of the printer file to which these Line Counter Specifications apply. The filename must also be described on the File Description Specifications and must be assigned to a printer. The entry in this field must be left-justified and must not be left blank.

15-19 LINE NUMBER, FL OR CHANNEL NUMBER

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

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

If columns 18-19 contain a numeric entry between one and 12, inclusive, the entry in columns 15-17 specifies the number of lines from the top of the form to be associated with the CHANNEL NUMBER entry designated in columns 18-19. The LINE NUMBER entry must be between one and 112, inclusive. Both entries must be right-justified in their respective fields, and leading zeros are optional (see figure 6-3).

LINE COUNTER SPECIFICATIONS

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

A B C D E F

- A. 7-14 Specifies the name of a printer file.
- B. 15-17 If 18-19 contains FL, this entry specifies the length of the page. If 18-19 are numeric, this entry specifies the number of lines from the top of the page to associate with that channel number. Entries: 1-112, right-justified.
- C. 18-19 Designates the use of the numeric entry in columns 15-17. Entries: FL or 1-12, right-justified.
- D. 20-22 If 23-24 contains OL, this entry specifies the line number of the overflow line. If 23-24 are numeric, this entry specifies the number of lines from the top of the page to associate with that channel number. Entries: 1-N, right-justified (N=entry in 15-17).
- E. 23-24 CHANNEL NUMBER associated with the overflow line designated in columns 20-22. Entries: OL or 1-12, right-justified.
- F. 25-74 CHANNEL NUMBER related to preceding LINE NUMBER entry. Entries: 1-12, right-justified LINE NUMBER designates a particular line on each page. Entries: 1-N, right-justified (N=entry in 15-17 or the default length 66).

614029

Figure 6-1. Line Counter Specifications Summary Sheet





FILE DESCRIPTION SPECIFICATIONS

LINE	FILENAME	BLOCK LENGTH	RECORD LENGTH	KEY FIELD STARTING LOCATION	DEVICE	NOT USED	NOT USED	CORE INDEX
02	F	* ENTRY IN COLUMNS 33-34 DISCONTINUES						
03	F	* AUTOMATIC HANDLING OF OVERFLOW						
04	F							
05	F	LISTING #	132	OF	LPRINTER			
06	F							
07	F							

LINE COUNTER SPECIFICATIONS

LINE	FILENAME	LINE NUMBER	FL OR CHANNEL NUMBER	LINE NUMBER	OL OR CHANNEL NUMBER	LINE NUMBER	CHANNEL NUMBER	LINE NUMBER	CHANNEL NUMBER	LINE NUMBER	CHANNEL NUMBER	LINE NUMBER	CHANNEL NUMBER	LINE NUMBER	CHANNEL NUMBER	LINE NUMBER	CHANNEL NUMBER	LINE NUMBER	CHANNEL NUMBER	LINE NUMBER	CHANNEL NUMBER	
11	L	* CHANNEL 1, 12, AND 2 ARE USED TO SPECIFY LINES 10, 45, AND 13.																				
12	L																					
13	L	LISTING	1001	4512	1302																	

OUTPUT - FORMAT SPECIFICATIONS

LINE	FILENAME	TYPE	SKIP	OUTPUT INDICATORS	FIELD NAME (VARIABLE NAME)	CONSTANT OR EDIT WORD	NOT USED	
01	O	* THESE ENTRIES WILL CAUSE THE HEADINGS TO BE PRINTED ON THE						
02	O	* FIRST PAGE (1P) AND TOTALS TO BE PRINTED WHEN THE LAST RECORD						
03	O	* (LR) IS READ AND ON ALL OVERFLOW PAGES.						
04	O							
05	O	LISTING #	01	1P				
06	O					42 'HEADING'		
07	O	H	02	1P		62 'DATE TIME PLACE'		
08	O							
09	O							
10	O							
11	O	T	12	LR				
12	O	OR		OF				
13	O							
14	O				TOTAL 1	45 'TOTAL'		
15	O					55		

614032

Figure 6-3. Overflow Coding Example 2

20-24 LINE NUMBER, OL OR CHANNEL NUMBER

These fields have two possible meanings, depending upon the entry in columns 23-24.

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

- a. Detail lines still to be printed as part of the current program cycle will be completed.
- b. Total lines will be printed.
- c. Total line conditioned by the overflow indicator for this file will be printed.

Since all these actions will take place after the overflow line is reached, the programmer should be certain that enough space is left between the overflow line and the bottom of the page to allow all the lines to be printed (see figure 6-2).

If either the form length or overflow line is specified then both must be specified. In the absence of both, a form length of 66 and an overflow line of 60 are assumed (6 lines per inch).

If columns 23-24 contain a numeric entry between one and 12, inclusive, the entry in columns 20-22 must specify the number of lines from the top of the form to be associated with the CHANNEL NUMBER entry (columns 23-24). The LINE NUMBER field entry must be between one and 112, inclusive. Both entries must be right-justified in their respective fields, and leading zeros are optional (see figure 6-3).

25-74 LINE NUMBER AND CHANNEL NUMBER

The rest of the form is divided into ten 5-character fields, each consisting of a 3-character LINE NUMBER field and a 2-character CHANNEL NUMBER field. All fields are optional and must be left blank if they are not to be used (see figure 6-4).

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

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

LINE COUNTER SPECIFICATIONS

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

G14030

LIST1 specifies channel 1 as line 10 and channel 12 as line 50.  
 LIST2 specifies the form length (50 lines) and channels 1 and 12.  
 LIST3 specifies the form length (80 lines) and the overflow line (line 60).

Figure 6-4. Line Counter Specifications Code Example

75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a complete description.

# TELECOMMUNICATIONS CARD SPECIFICATIONS

A Telecommunications Card is used when a DATACOM file(s) is specified on the File Description Specifications. DATACOM files are used to transmit and/or receive information contained in corresponding NDL files in the Network Controller.

If used, DATACOM files should be a reflection of what is specified in the NDL files; therefore, before referencing a DATACOM file it is necessary to know the name of the corresponding NDL file and how it is specified.

Due to the timing and other limitations of RPG, only one terminal may be assigned to a DATACOM file. Telecommunications in RPG is designed to function as a slave rather than a master controller.

Only three entries are necessary on a Telecommunications Card. All other entries are used for documentation only and are syntactically checked. These unused areas, if coded, should reflect what is in the NDL file. (See B 1700 System Network Definition Language, Form No. 1073715.)

There are no B 1700 RPG forms especially designed for coding DATACOM files. The Control Card Specifications are used here to illustrate coding positions on a Telecommunications Card.

## FIELD DEFINITIONS

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

1-2 PAGE

Refer to Section 2 for a complete description.

3-5 LINE

Refer to Section 2 for a complete description.

6 FORM TYPE

This field must contain a T.

7-14 FILENAME

This field contains the name of the DATACOM file specified on the File Description Specifications. It must be the same as the NDL filename it is referencing.

15 CONFIGURATION

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

**Burroughs**      B 1700 RPG

PROGRAM ID		PROGRAMMER		PAGE	OF
				DATE	
PAGE	1 2	CONTROL CARD SPECIFICATIONS			PROGRAM IDENTIFICATION
					75 80
LINE	FORM TYPE				
3 5 6 7					
0 1	H				

A
B
C

- A. 6 Change entry to the letter T.
- B. 7-14 Contains the name of a DATACOM file entered on the File Description Specifications.
- C. 16 Specifies if the file is to be transmitted or received. Entries: T or R.

614033

Figure 7-1. Telecommunications Card Specifications Summary Sheet

16      TRANSMIT/RECEIVE

Acceptable entries for this field are:

Entry	Definition
T	Transmit messages.
R	Receive messages.

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

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

17      TYPE OF LINE CONTROL

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

18      TRANSMISSION CODE

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

- 19       TRANSPARENCY
- This field is unused. Any entry other than Y, N, or blank will be given a syntax error.
- 20       TYPE OF CONNECTION
- This field is unused. Any entry other than M, E, S, A, B, or blank will be given a syntax error.
- 21-31   TELEPHONE NUMBER
- This field is unused. Any entry in this field will be accepted; however, it should reflect the numeric, alphanumeric, or blank entry in the corresponding NDL file.
- 32       LOCATION OF IDENTIFICATION-THIS STATION
- This field is unused. Any entry other than S, E, or blank will be given a syntax error.
- 33-39   IDENTIFICATION-THIS STATION
- This field is unused and may contain any alphanumeric or blank entry. The entry should reflect the entry in the corresponding NDL file.
- 40       LOCATION OF IDENTIFICATION-REMOTE STATION
- This field is unused. Any entry other than S, E, or blank will be given a syntax error.
- 41-47   IDENTIFICATION-REMOTE STATION
- This field is unused and may contain any alphanumeric or blank entry. The entry should reflect the entry in the corresponding NDL file.
- 48-51   REMOTE TERMINAL
- This field is unused and should remain blank.
- 52       ITB
- This field is unused. Any entry other than I or blank will be given a syntax error.
- 53-54   PERMANENT ERROR INDICATOR
- This field is unused. Any entry other than 01-99, L1-L9, LR, H1-H9, or blank will be given a syntax error.
- 55-57   TIMEOUT VALUE
- This field is unused. Any entry other than a numeric or blank entry will be given a syntax error.
- 58-59   RECORD AVAILABLE INDICATOR
- This field is unused. Any entry other than 01-99, L1-L9, LR, H1-H9, or blank will be given a syntax error.

60 LAST FILE

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

61-62 TRANSMIT ADDRESS

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

63-63 RECEIVE ADDRESS

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

75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a complete description.



## INPUT SPECIFICATIONS

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

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

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

FIELD DEFINITIONS

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

1-2 PAGE

Refer to 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), 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.

## Burroughs B 1700 RPG

PROGRAM ID		PROGRAMMER				PAGE	OF																			
							DATE																			
INPUT SPECIFICATIONS																										
PAGE		OPTION				PROGRAM IDENTIFICATION				75	80															
1	2																									
FORM TYPE	NUMBER SEQUENCE	RECORD IDENTIFYING INDICATOR				PACKED STACKER SELECT				MATCHING FIELDS OR CHAINING FIELDS																
		RECORD IDENTIFICATION CODES				DECIMAL POSITIONS				CONTROL LEVEL																
LINE	FILENAME	POSITION				FIELD LOCATION				FIELD NAME (VARIABLE NAME)																
		1	2	3	4	FROM	TO					FIELD INDICATORS														
		POSITION	NOT (N) C/Z/D CHARACTER	POSITION	NOT (N) C/Z/D CHARACTER																					NOT USED
3	5 6 7	14	15 16 17 18 19 20 21	24	25 26 27 28	31	32 33 34 35	38	39 40 41	42	43 44	47	48	51	52 53	58	59 60	61	62 63 64	65	66 67 68	69	70	71	74	
0 1	I																									
0 2	1																									

A                      B    C    D    E    F                      G                      H    I                      J    K    L                      M                      N    O    P    Q    R    S

- A. 7-14 Contains a filename specified in the File Description Specifications.
- B. 14-16 Puts the record identification codes in an AND or OR relationship. Entries: AND or OR.
- C. 15-16 Specifies if records are to be processed in a predetermined sequence. Entries: 01-99 or any alphabetic character.
- D. 17 If sequence is specified, the entry indicates the number of records of each type in a sequence group. Entries: 1 or N.
- E. 18 Specifies if records sequenced must be present. Entries: Blank or 0.
- F. 19-20 Contains either a record identifying indicator (01-99), a control level indicator (L1-L9), a halt indicator (H1-H9), or specifies look ahead records(\*\*).
- G. 21-41 Record Identification Codes: 21-24, 28-31, 35-38 (specifies the position within the record that contains a record identification code. Entries: Blank or 1-N where N = record length). 25, 32, 39 (indicates if the character in columns 27, 34, or 41 is or is not present. Entries: Blank or N. 26, 33, 40 (Specifies what part of the code character in columns 27, 34, or 41 is to be read. Entries: C - the entire character, Z - the zone portion, D - the digit portion). 27, 34, 41 (Contains the code character. Entries: Any EBCDIC character).
- H. 42 Specifies which stacker will be used. Entries: Blank or 1-6.
- I. 43 Specifies if numeric input is in packed or unpacked decimal format. Entries: Blank or P.
- J. 44-47 Contains the left-most position of the input field. Entries: numeric, right-justified.
- K. 48-51 Contains the right-most position of the input field. Entries: numeric, right-justified.
- L. 52 Contains the number of decimal positions for numeric fields. Entries: Blank or 0-9.
- M. 53-58 Contains field names (1-6 alphanumeric characters, left-justified) or one of the special field names, PAGE, PAGE1, PAGE2, UDATE, UMONTH, UDAY, UYEAR, TABXXX (X=any alphanumeric character).
- N. 59-60 Contains the control level indicator. Entries: Blank or L1-L9.
- O. 61-62 Specifies sequence checking for a single input or combined file or sequence checking with matching records for two or more input and/or combined files. Entries: Blank or M1-M9.
- P. 63-64 Contains one of the following field record relations indicators or blank: 01-99 (record identifying indicator), L1-L9 (control level indicator previously defined), MR (matching record indicator), U1-U8 (external indicator), or H0-H9 (halt indicator).
- Q. 65-66 Used to indicate if the specified field is greater than blank or positive. Entries: Blank, 01-99, H0-H9.
- R. 67-68 Used to indicate if the specified field is less than blank or negative. Entries: Blank, 01-99, H0-H9.
- S. 69-70 Used to indicate if the specified field is blank or zero. Entries: Blank, 1-99, H0-H9.

614034

Figure 8-1. Input Specifications Summary Sheet

14-16 AND/OR LINES

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

AND Line

If it is necessary to specify more than three record identifying codes to identify a record type, an AND line may be used. The word AND 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 may be identified by two or more different codes. For this condition, the word OR entered in columns 14-15 indicates that only one of the codes specified need be present to identify the record type (see figure 8-2). Record identification codes are not required on OR lines, although this is not necessarily a meaningful thing to do. Other uses of the OR relationship are discussed later in this section (refer to figures 8-10 and 8-18).

PAGE		NUMBER		INPUT SPECIFICATIONS																		PROGRAM IDENTIFICATION	
FORM TYPE		SEQUENCE		RECORD IDENTIFYING INDICATOR			PACKED STACKER SELECT			MATCHING FIELDS OR CHAINING FIELDS			DECIMAL POSITIONS			CONTROL LEVEL			FIELD RECORD RELATION				
LINE	FILENAME	RECORD IDENTIFICATION CODES									FIELD LOCATION		FIELD NAME (VARIABLE NAME)	FIELD INDICATORS			NOT USED						
		1	2	3	FROM	TO	PLUS	MINUS	ZERO OR BLANK														
		POSITION	NOT (N) C/Z/D CHARACTER	POSITION	NOT (N) C/Z/D CHARACTER	POSITION	NOT (N) C/Z/D CHARACTER																
01	I INPUT	01	L1	77	CD	78	CE	79	CP														
02	I AND	80	CT																				
03	I OR	1	D9																				
04	I *																						
05	I *	THE LINES ABOVE DESCRIBE A RECORD TYPE WHICH CAN BE																					
06	I *	IDENTIFIED BY THE CODE "DEPT" APPEARING IN POSITIONS																					
07	I *	77-80 OF THE INPUT RECORD. THE "AND" LINE IS USED IN																					
08	I *	ORDER TO SPECIFY ADDITIONAL CHARACTERS AS PART OF THE																					
09	I *	RECORD IDENTIFICATION CODE.																					
10	I *																						
11	I *	THE "OR" LINE ALLOWS AN ALTERNATE CODE TO BE SPECIFIED.																					
12	I *	THUS, A DIGIT 9 IN THE FIRST POSITION WILL ALSO SERVE																					
13	I *	TO IDENTIFY THE RECORD TYPE, EVEN IF THE FIRST CODE																					
14	I *	DOES NOT APPEAR IN THE INPUT RECORD.																					
15	I *																						
16	I *	FOR THE ABOVE RECORD TYPE, EITHER THE WORD "DEPT" IN																					
17	I *	POSITIONS 77-80 OR A DIGIT 9 IN POSITION 1 (OR BOTH)																					
18	I *	WILL CAUSE THE L1 INDICATOR TO BE TURNED ON.																					

614035

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

## 15-16 SEQUENCE

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

Within each file, all record types having alphabetic entries in the SEQUENCE field must be specified before those with numeric entries. All chained and demand files must have an alphabetic entry in this field.

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

If this field contains a numeric entry, it indicates that sequence checking is to be done. The order of precedence is the sequence in which the records are declared on the Input Specifications. This allows the programmer to specify that one record type must appear before another record type within a sequenced group. The program will automatically check the designated order as the records are read.

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

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

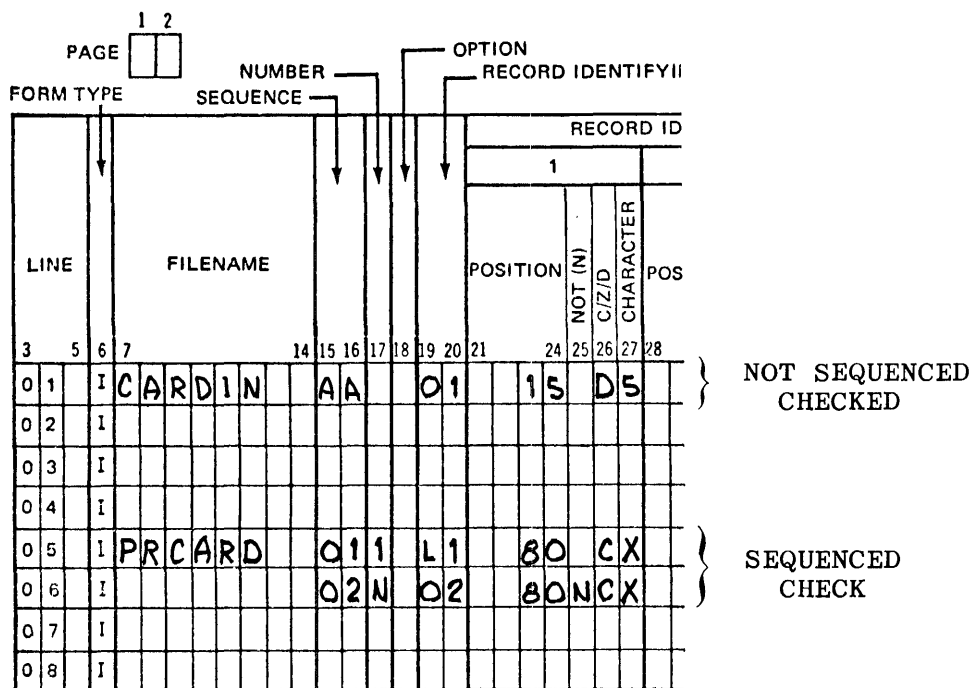
Records in an AND or OR line cannot have a sequence field entry; the entry from the previous line also applies to the line with the AND or OR entry.

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

## 17 NUMBER

This field is used only if sequence checking is to be done (i.e., the SEQUENCE field contains a numeric entry). An entry in this field indicates whether more than one record of the designated type may appear in each group of a sequenced input file (see figure 8-3). Valid entries for this field are:

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



914036

Figure 8-3. Input Sequence Checking

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.

18 OPTION

This field is used only if sequence checking is to be done (i.e., the SEQUENCE field contains a numeric entry). An entry in this field indicates whether certain record types are optional. An alphanumeric "0" entry specifies that a record of this type may or may not be present in each group of a sequenced file. If this field is left blank, each group may contain one or any number of records of this type, depending on the entry in column 17 (see figure 8-3).

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

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

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

19-20 RECORD IDENTIFYING INDICATOR

This field may be used for two purposes:

- a. To assign an indicator to each record type.
- b. To indicate look-ahead fields.

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

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

Record Identifying Indicator 01-99

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

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

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

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. 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.

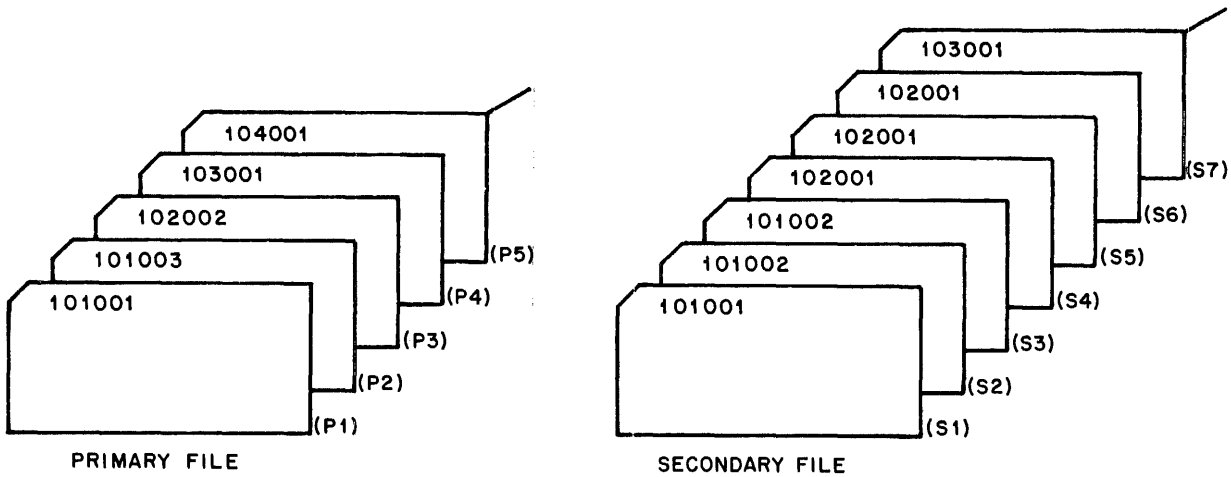
Look-Ahead Field \*\*

A pair of asterisks (\*\*) is used to indicate that fields named in the VARIABLE NAME field following the \*\* line are look-ahead fields. The \*\* line can follow only a file or record type which has an alphabetic sequence entry in columns 15-16. Columns 17-18 and 21-74 are left blank.

A look-ahead field allows the program to access information in a field of the next record that will be available for processing. Thus, the program may use information from the look-ahead field to condition certain operations ahead of the time the record is available for processing. By using a look-ahead field, the program can do jobs that the matching record capability cannot handle or determine when the last card of a control group is being processed.

Figure 8-4 shows processing of records from two input files, one primary and one secondary. All primary records are processed before any secondary records are available; therefore, to use data from the secondary file while processing the primary file record it is necessary to use the look-ahead feature.

<u>Record Processed</u>	<u>Look-Ahead Records Available</u>
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



614037

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

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

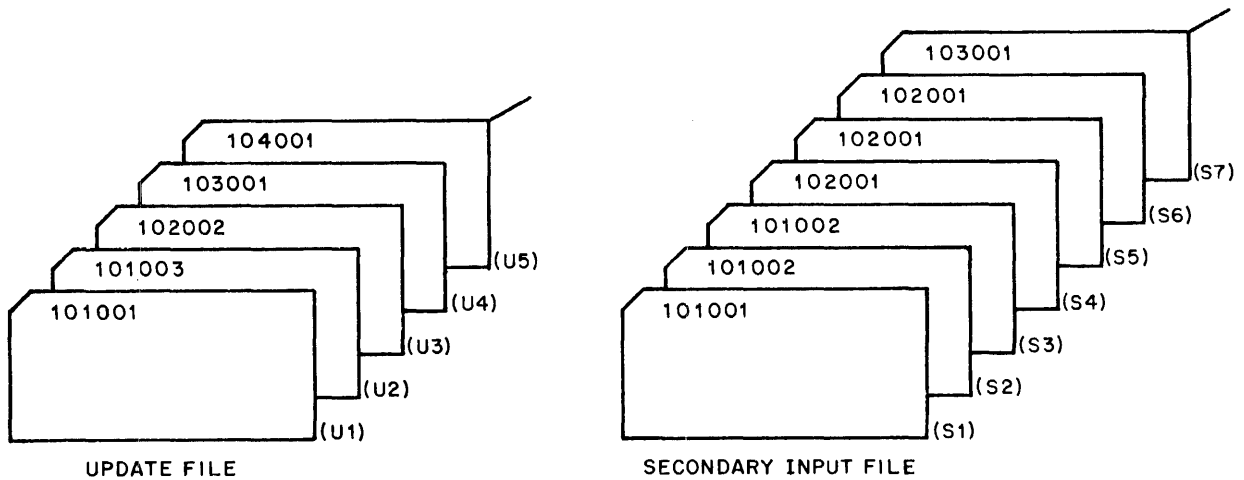
For combined and update files, the look-ahead fields apply to the next record in the file only if the current record was read from some other file. Therefore, in a program which reads from one file only and that file is a combined or update file, look-ahead fields will apply to the current record.

Figure 8-5 shows processing records from an update file and a secondary input file.

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

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





G14038

Figure 8-5. Records Available for Look-Ahead: One Update File, One Input File

Look-ahead fields may not be used as result fields in calculation operations. The name given to a look-ahead field must not occur on any other Input or Extension Specification. If the look-ahead field occurs on an Output Specification, blank after must not be specified.

In the example shown in figure 8-6, the fields are described twice so that information will be accessed both before and after the record is selected for processing.

LINE	FORM TYPE	PAGE	SEQUENCE	NUMBER	OPTION	INPUT SPECIFICATIONS												PRO IDEI												
						RECORD IDENTIFICATION INDICATOR						PACKED			MATCHING FIELDS OR CHAINING FIELD															
						RECORD IDENTIFICATION CODES						STACKER SELECT			DECIMAL POSITIONS				CONTROL LEVEL											
						1		2		3					FIELD LOCATION															
POSITION	NOT (N) C/Z/D	CHARACTER	POSITION	NOT (N) C/Z/D	CHARACTER	POSITION	NOT (N) C/Z/D	CHARACTER	FROM	TO	FIELD NAME (VARIABLE NAME)																			
01	I	1	1	1	1	24	25	26	27	28	31	32	33	34	35	38	39	40	41	42	43	44	47	48	51	52	53	58	59	60
02	I																							1	19					FIL1
03	I																							20	33					FIL2
04	I																							34	40					FIL3
05	I																													
06	I																													
07	I																								1	19				NXFIL1
08	I																								20	33				NXFIL2
09	I																								34	40				NXFIL3
10	I																													
11	I																													
12	I																													

Figure 8-6. Coding Look-Ahead Fields

21-41 RECORD IDENTIFICATION CODES

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

In order to identify the various record types to the program for the purpose of record selection, each record type must have a unique code assigned to it. This code consists of a certain character or combination of characters occurring in certain positions of the record.

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

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

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

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

25,32,39 NOT

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

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

C/Z/D

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

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

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

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

1 1 0 0 0 0 0 1 .

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

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

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

In figure 8-7, only the records of customers whose last names begin with the letters A through I will be processed since the zone portion of A is the same as for the characters B through I. The first letter of each last name begins in column 10.

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

G14040

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

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

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

G14041

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

In figure 8-9, only persons whose names start with C, L, or T will be processed, since the digit portion of the characters L and T is the same as C.

LINE	FILENAME	RECORD IDENTIFICATION CODES																								
		1			2			3																		
		POSITION	NOT (N)	C/Z/D CHARACTER	POSITION	NOT (N)	C/Z/D CHARACTER	POSITION	NOT (N)	C/Z/D CHARACTER																
3	5	6	7	14	15	16	17	18	19	20	21	24	25	26	27	28	31	32	33	34	35	38	39	40	41	
0 1	I	R	E	C	F	I	L	E	A	A	1	2	7	D	C											
0 2	I																									

614042

Figure 8-9. C/Z/D Coding Example 3

NOTE

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

27, 34, 41

CHARACTER

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

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

42

STACKER SELECT

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

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

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

At execution time, any record types specifying a stacker number higher than that available on the device being used will go to the default stacker.

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

43 PACKED

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

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

NOTE

96-column devices will not handle packed decimal format.

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

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

zone	digit	zone	digit	zone	digit	zone	digit
------	-------	------	-------	------	-------	------	-------

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

With packed decimal format, each byte can contain two decimal numbers. The byte is divided into two 4-bit portions, and each 4-bit portion holds one digit, except for the sign position.

Packed data is always character aligned externally (on tape or disk). If a field specifying an even number of digits is to be output, the following example illustrates the rules to be followed:

S	O	dn ...	d <sub>2</sub>	d <sub>1</sub>	left signed output
O	d <sub>n</sub>	...d <sub>2</sub>	d <sub>1</sub>	S	right signed output

- S = sign
- O = digit zero - (even number digits are zero padded)
- dn...d<sub>1</sub> = digits of the field.

Packed data input will always cause an odd field size for the field to contain the data, since even number digits are padded with a zero. All digits are considered data except the sign digit position.

44-51 FIELD LOCATION

This field is used to describe the location of data fields within a record, and is divided into two subfields that specify the beginning (FROM) and ending (TO) positions of the data field. A field of only one character will have the same position number entered in both subfields. Both entries must lie between 1 and the RECORD LENGTH. The TO entry must be greater than or equal to the FROM entry. Entries in the FROM and TO subfields must be right-justified; leading zeros are optional (figure 8-10).

FIELD LOCATION						FIELD NAME (VARIABLE NAME)				
FROM			TO							
43	44	47	48	51	52	53	58			
		1		9	O	E	H	P	N	Ø
		10		23	N	A	M	E	L	
		24		42	O	H	R	S		
		53		55	S	O	C	O	D	E
		56		57	S	E	X			

614043

Figure 8-10. FIELD LOCATION 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 will be read in from element 1 up to as many elements as will fit into the locations specified. The decimal positions must contain the same entry as specified on the Extension Specifications for that array.

## 52 DECIMAL POSITIONS

This field is used to specify the number of positions to the right of the implied decimal point in a numeric field. This entry may not be blank for a numeric field; if the data field contains only integral values, a 0 should be entered to indicate no decimal positions. Valid entries are:

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

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

## 53-58 FIELD NAME (VARIABLE NAME)

This field is used to assign an identifier (name) to an input data field. All fields that will be referenced by the program must be named. Names must be assigned in accordance with the rules for forming field names as described in Section 2. A previously defined vector name may be used, which allows loading of the vector during input. Refer to Section 5 for a complete discussion of this method of vector loading. A separate line must be used for each field description.

All fields within one record type should have unique names; if two or more fields within the same record have identical names, only the last one defined is used. Fields from different record types may have the same name, but all names not uniquely defined must have the same length and data type (decimal position entry). These fields do not have to occur in the same location in each record.

### OR Relationship

To eliminate duplicate coding of identical fields within different record types, the OR relationship may be used. The OR relationship, illustrated in figure 8-11, shows two record types which have identical fields in the same record positions. Refer to columns 14-16 in this section.



INPUT SPECIFICATIONS

PAGE 1 2 PROGRAM IDENTIFICATION 75 80

LINE	FORM TYPE	FILENAME	NUMBER SEQUENCE	OPTION	RECORD IDENTIFICATION INDICATOR									PACKED STACKER SELECT	MATCHING FIELDS OR CHAINING FIELDS		CONTROL LEVEL	FIELD RECORD RELATION							
					RECORD IDENTIFICATION CODES										DECIMAL POSITIONS			FIELD NAME (VARIABLE NAME)	FIELD INDICATORS			NOT USED			
					1	2	3	4	5	6	7	8	9		10	11			12	13	14		15	16	17
POSITION	NOT (N) C/Z/D CHARACTER	POSITION	NOT (N) C/Z/D CHARACTER	POSITION	NOT (N) C/Z/D CHARACTER	POSITION	NOT (N) C/Z/D CHARACTER	POSITION	NOT (N) C/Z/D CHARACTER	POSITION	NOT (N) C/Z/D CHARACTER	FROM	TO	FIELD NAME (VARIABLE NAME)	PLUS	MINUS	ZERO OR BLANK	NOT USED							
01	I	FILENAM	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											
11	I																								
12	I	* THE RECORD DESCRIPTIONS ABOVE BOTH CONTAIN THE SAME FIELDS.																							
13	I	* THE TWO DESCRIPTIONS CAN BE COMBINED USING AN "OR" LINE.																							
14	I	* AS SHOWN BELOW.																							
15	I																								

INPUT SPECIFICATIONS

PAGE 1 2 PROGRAM IDENTIFICATION 75 80

LINE	FORM TYPE	FILENAME	NUMBER SEQUENCE	OPTION	RECORD IDENTIFICATION INDICATOR									PACKED STACKER SELECT	MATCHING FIELDS OR CHAINING FIELDS		CONTROL LEVEL	FIELD RECORD RELATION							
					RECORD IDENTIFICATION CODES										DECIMAL POSITIONS			FIELD NAME (VARIABLE NAME)	FIELD INDICATORS			NOT USED			
					1	2	3	4	5	6	7	8	9		10	11			12	13	14		15	16	17
POSITION	NOT (N) C/Z/D CHARACTER	POSITION	NOT (N) C/Z/D CHARACTER	POSITION	NOT (N) C/Z/D CHARACTER	POSITION	NOT (N) C/Z/D CHARACTER	POSITION	NOT (N) C/Z/D CHARACTER	POSITION	NOT (N) C/Z/D CHARACTER	FROM	TO	FIELD NAME (VARIABLE NAME)	PLUS	MINUS	ZERO OR BLANK	NOT USED							
01	I	FILENAM	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											
07	I																								
08	I																								
09	I																								
10	I																								
11	I																								
12	I																								
13	I																								
14	I																								
15	I																								

Figure 8-11. OR Relationship - Identical Fields Within Different Record Types

## Special Words

The following special words are reserved for use as variable names in columns 53-58 of the Input Specifications:

PAGE  
PAGE1  
PAGE2

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

This feature allows a page number to be entered through a field in an input record, the field called "PAGE" (or "PAGE1" or "PAGE2"). The page number printed will be one greater than the page number contained in the "PAGE" field of the input record. A page field is incremented by 1 each time before it is printed. The field may be defined as any length, but it must contain zero decimal positions. Unless otherwise specified, it is assumed to be four digits in length with zero decimal positions (see figure 8-12). The "PAGE" field may be used in calculations like any other field.

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

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

FORM TYPE		PAGE		NUMBER		OPTION		RECORD IDENTIFYING INDICATOR		PACKED		MATCHING FIELDS OR CHAINING FIELDS		PROGRAM IDENTIFICATION		FIELD RECORD RELATION	
		1 2								STACKEF SELECT		DECIMAL POSITIONS					
LINE	FILENAME	RECORD IDENTIFICATION CODES:			FIELD LOCATION		FIELD NAME (VARIABLE NAME)	FIELD INDICATORS			NOT USED						
		1	2	3	FROM	TO		PLUS	MINUS	ZERO OR BLANK							
		POSITION	NOT (N) C/Z/D CHARACTER	POSITION	NOT (N) C/Z/D CHARACTER	POSITION	NOT (N) C/Z/D CHARACTER										
01	INPUT	PG	91	1	CX												
02						2	40	PAGE									
03	I*																
04	I*	THE SPECIFICATION LINES ABOVE DESCRIBE AN INPUT FILE WHICH															
05	I*	CONTAINS A RECORD WITH A "PAGE" FIELD. THE FIELD IS															
06	I*	DEFINED AS 3 CHARACTERS IN LENGTH (WITH NO DECIMALS),															
07	I*	BEGINNING IN POSITION 2 OF THE INPUT RECORD. THE VALUE															
08	I*	CONTAINED IN THE PAGE FIELD OF THE INPUT RECORD WILL BE															
09	I*	ASSIGNED TO THE PAGE VARIABLE WHEN THE RECORD IS READ.															
10	I*																
11	I*	THE PAGE FIELD MAY BE USED IN CALCULATIONS LIKE ANY OTHER															
12	I*	FIELD. ON OUTPUT, THE PAGE FIELD IS AUTOMATICALLY															
13	I*	INCREMENTED BY 1 BEFORE THE OUTPUT RECORD CONTAINING IT															
14	I*	IS PRINTED.															
15	I																

Figure 8-12. PAGE Field Coding

59-60 CONTROL LEVEL

This field is used to assign control level indicators to primary or secondary files. Any field other than a whole array or a look-ahead field may be assigned a control level indicator, in which case it is known as a Control Field. Control fields are checked each program cycle for a change in information; when data in the field changes, a control break occurs. A group of records with the same information in the control field is known as a Control Group. Valid entries for this field are:

Entry	Definition
L1-L9	Control level field assigned.
Blank	No control level field assigned.

A control break occurs when a record containing a control field is read and the information in that control field is different from the information in the same control field of the previous record. When a control break occurs, the designated control level indicator turns ON, along with all control level indicators lower than it. For example, if control level indicator L5 is turned ON, L4, L3, L2, and L1 are also automatically turned ON. Control level indicator L0 is always ON and cannot be assigned to a control field.

A control level indicator may be turned on or off by SETON or SETOF, or may be used as a record identifying indicator. However, in such cases, control level indicators lower than the one specified are not turned on or off automatically.

Control level indicators are used to condition operations, such as:

- a. Calculations that must be performed when a control group changes (totals, etc.).
- b. Operations that must be performed on the first card of a new control group.
- c. Summary punching or total printing that must be performed for each control group.

The following rules must be observed when assigning control level indicators:

- a. The same control level indicator may be used in different record types or files; however, the control fields associated with that indicator must be of the same length. (See figure 8-13).
- b. Field names have no effect upon the control level indicator assigned; therefore, control fields in different record types may have both the same name and the same indicator assigned.
- c. The maximum size of a control field is 255 characters.
- d. Within one record type, control fields may overlap.





- c. Any one portion of a numeric split control field may not exceed the maximum size for numeric fields as specified in Section 1; however, the total length of all fields assigned to one control level indicator (within each record type) may be as great as 255 characters.
- d. A mixture of packed and unpacked control fields is allowed.
- e. No other specification lines may come between lines describing split control fields.

61-62 MATCHING FIELDS

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

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

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

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

Sequence Checking

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

The following rules must be observed when assigning matching field values for sequence checking:

- a. All fields designated for sequence checking must be in the same order, either ascending or descending.
- b. When more than one field is designated for sequence checking, all fields specified are combined in order by ascending sequence of matching field values and are treated as one contiguous field.



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

The following rules must be observed when assigning matching field values:

- a. All Match Fields must be in the same sequence during input, because sequence checking is automatically done on all fields designated as matching fields. A sequence error in any field will cause a program halt. When the system operator resumes program operation, the record in error is ignored and the next record from the same file is read.
- b. If matching is used, it is not necessary for all primary and secondary files to have Match Fields. Neither must all record types within a file have Match Fields. But at least one record type from two files must have Match Fields specified if the files are ever to be matched.
- c. All fields given the same matching field value (M1-M9) must be of the same length.
- d. Overlapping of different Match Fields within one record type is allowed; however, the length of any individual Match Field must not exceed 255 characters.
- e. All records to be matched should (not required) contain the same Match Fields (M1-M9); otherwise, a match may not be obtained.
- f. When more than one Match Field is designated for a record type, all fields specified are combined in order by ascending sequence of matching field values and are treated as one contiguous Match Field. M9 is high order.
- g. Split Match Fields are not allowed; thus, the same matching field value should not be used more than once in one record type, unless field record relation indicators are used.
- h. Numeric Match Fields are treated as though they had no decimal positions.
- i. For numeric Match Fields, only the digit portion of each character is compared; thus, negative numbers are treated the same as positive numbers.
- j. All Match Fields with the same matching field value are considered numeric, if any one of the fields is numeric.



- k. When more than one matching field value is used for matching records, all Match Fields must match before the MR indicator turns ON.
- l. Field names have no effect upon the matching field values assigned; therefore, Match Fields in different record types may have both the same name and the same matching field value assigned.

Processing of matching fields proceeds as follows:

- a. When a record from the primary file matches a record from the secondary file, the primary file record is processed first.
- b. When records do not match, the record with the lowest (ascending files) or highest (descending files) Match Field value is processed first.
- c. A record type which has no matching field specification is processed immediately after the record it follows, and the MR indicator is not turned ON. If such a record is the first one in a file, it is processed first (even if it is not in the primary file).
- d. Matching records allow the program to enter data from the primary record into the matching secondary record, since the primary record is processed first. Transfer of data from secondary records into matching primary records may be done through the use of look-ahead fields.
- e. When additional secondary files are declared, all matching records are processed in one secondary file before passing control to the next secondary file. The precedence of the secondary files is determined by their order of appearance on the Input Specifications.

In figure 8-16 the two matching fields will be combined in the ascending order of the matching field values, M1 and M2. Notice that an End-of-File has been specified for the secondary file. This will cause the job to end after the last record in the file, TIMECDS, has been read.

FILE DESCRIPTION SPECIFICATIONS

LINE	FILENAME	BLOCK LENGTH	RECORD LENGTH	KEY FIELD STARTING LOCATION	DEVICE	NOT USED	NOT USED	CORE INDEX
02	EMPLYCRDIP A	96	96		MFCU1			
03	TIMECDS CSEA	96	96		MFCU2			
04	PAYROLL	96	96		PRINTER			
05	F							
06	F							
07	F							
	F							
	F							

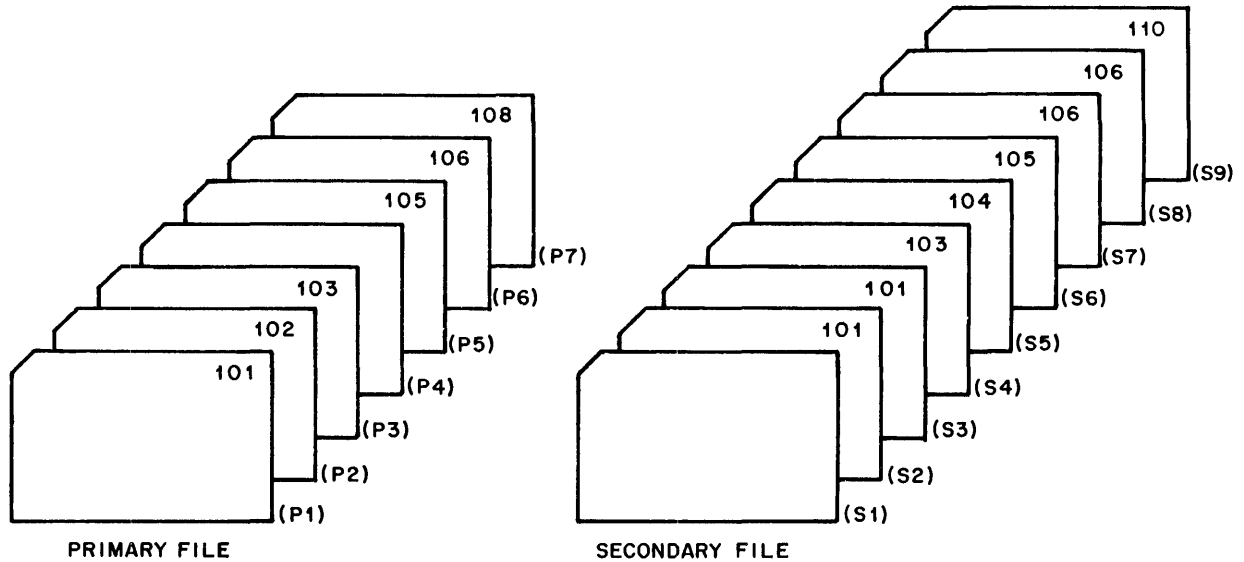
INPUT SPECIFICATIONS

LINE	FILENAME	RECORD IDENTIFICATION CODES	FIELD LOCATION	FIELD NAME (VARIABLE NAME)	FIELD INDICATORS
01	EMPLYCRD	10 1 CN 2 CA	3 15	NAME	
02			16 18	DEPTNO	M2
03			19 22	EMPLNO	M1
04			23 29	PAYRAT	
05					
06	TIMECDS BB	20 1 CR 2 CC	3 15	NAME	
07			16 18	DEPTNO	M2
08			19 22	EMPLNO	M1
09			40 46	HRS WRK	
10					
11					
12					
13					
14					
15					

614048

Figure 8-16. Coding Matching Fields

The example shown in figure 8-17 is used to illustrate the order in which two matching fields will be processed. An End-of-File has been specified for the primary file.



014049

Figure 8-17. Record Selection From Two Matching Files

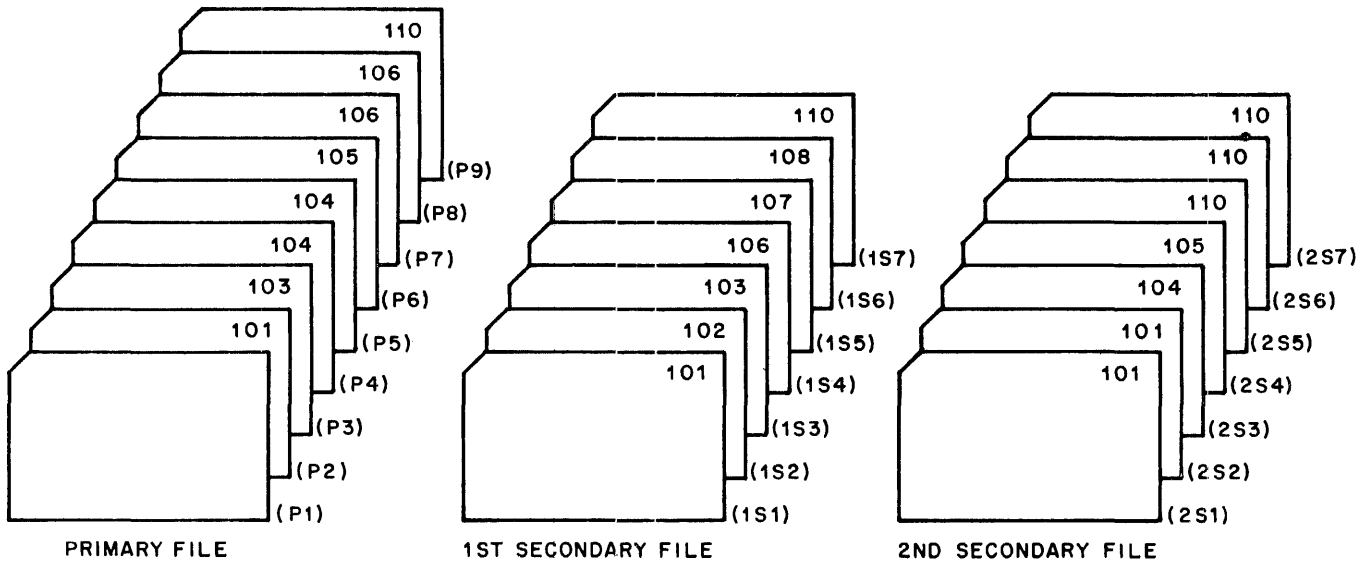
The files (figure 8-17) will be processed in the following order:

- S1       Records with no matching fields are processed before records with matching fields regardless of file type.
- P1 }     The MR indicator is ON. All matching secondary records are processed after the primary record.
- S2 }
- S3 }
- P2
- P3 }     The MR indicator is ON.
- S4 }
- P4
- S5       When no records match, the record with the lowest sequence number is processed, regardless of file type.
- P5 }     The MR indicator is ON.
- S6 }
- P6 }     The MR indicator is ON.
- S7 }
- S8 }
- P7       Last record processed.

---

- S9       When an End-of-File condition is specified for a primary file, secondary records are not processed unless the MR indicator is turned ON.

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



014050

Figure 8-18. Record Selection From Three Matching Files

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

P1

P2

1S1

2S1

2S2

The MR indicator is ON since all Match Fields match.

1S2

When records do not match, the lowest matching field is processed next.

P3

1S3

P4

P5

2S3

P6

2S4

P7

P8

1S4

1S5

1S6

P9

1S7

2S5

2S6

2S7

The MR indicator is ON. 2S7 is the last record processed.

63-64 FIELD RECORD RELATION

The following rules should be observed when assigning field record relation indicators.

- a. A field record relation indicator need not necessarily be the same as any record identifying indicator specified for the file.
- b. Fields within one record type which specify the same field record relation indicator may be entered in any order; however, the most efficient data storage is obtained when they are written as a group on specification lines following one another.
- c. All portions of a Split Control Field must be assigned the same field record relation indicator and must be written as a group on specification lines following one another.

If field record relation indicators are not to be assigned, this field should be left blank (see figure 8-19). The acceptable entries are as follows:

Entry	Definition
Blank	No field record relations.
01-99	Record identifying indicator assigned to a record type.
L1-L9	Control level indicator defined elsewhere.
MR	Matching record indicator.
U1-U8	External indicator defined elsewhere.
H0-H9	Halt indicator defined elsewhere.

Each of the entries is discussed in the following paragraphs.

01-99 Record Identifying Indicators

When several record types have been defined in an OR relationship, all fields defined apply to all record types. In many cases, however, not all of the record types defined have exactly the same fields. The FIELD RECORD RELATION field allows the programmer to specify that some fields apply only to certain record types and not to others. If the FIELD RECORD RELATION field is left blank, the associated field applies to all record types to which it is subordinate. However, by placing the same entry found in the Record Identifying Indicator field of one record type in the FIELD RECORD RELATION field, the field is identified as applying only to the corresponding record type.

Control Fields and Match Fields may also be related to a particular record type in an OR relationship by a FIELD RECORD RELATION entry.



U1-U8      External Indicators

External indicators are used primarily to condition files in the File Description Specifications (see columns 71-72, FILE CONDITION). However, they may also be used to condition fields even though file conditioning is not specified. Data from the associated field will be accepted only when the specified indicator is ON.

H0-H9      Halt Indicators

A halt indicator is used to associate a field with a record type in an OR relationship which has the same halt indicator specified in the RECORD IDENTIFYING INDICATOR field.

65-70      FIELD INDICATORS

This field is divided into three subfields to allow testing of a field for a condition of plus (positive), minus (negative), or zero. The indicator specified turns ON if the condition tested is True, and turns OFF if the condition tested is False. Thus, these indicators may be used to condition certain calculation or output operations. Valid entries 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 will turn ON if the corresponding data field is greater than zero (numeric field only) or greater than blank (alphanumeric field).

67-68      MINUS

Any valid indicator specified in this field will turn ON if the corresponding data field is less than zero (numeric field only) or less than blank (alphanumeric field).

69-70      ZERO OR BLANK

Any valid indicator specified in this field will turn ON if the corresponding data field is zero (numeric field only) or blank (alphanumeric field).

The following rules must be observed when assigning and using field indicators:

- a. All field indicators are OFF at the beginning of the program, and remain OFF until the condition being tested is satisfied on the input record just read. Note that if RPG 1 dialect is specified on the Control Card Specifications, all valid indicators used as zero or blank indicators will be initialized ON at the beginning of the object program execution, unless they are also used as record identifying indicators.
- b. A field may be assigned more than one indicator; however, only the indicator specified for the condition with a true result will be turned ON. All other indicators assigned to the field will be turned OFF.
- c. The state of a field indicator assigned to fields in different record types is always determined by the last record selected.
- d. The state of a field indicator assigned to more than one field within one record type is determined by the last field to which it is assigned.
- e. When different field indicators are assigned to fields in different record types, a field indicator will remain ON (or OFF) until another record of the same type is selected.
- f. If a halt indicator specified in the FIELD INDICATOR field is turned ON as a result of the corresponding condition being true, the program will halt after the input record which caused it to turn ON has been completely processed.

75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a complete description.



# CALCULATION SPECIFICATIONS AND OPERATION CODES

Calculation Specifications are used to describe the operations to be performed on the data and to specify the order in which those operations are to be performed. Calculation Specifications are of three types, which are optional, but if they occur, must appear in the following order:

- a. Detail calculations.
- b. Total calculations.
- c. Subroutines.

Within each grouping, operations are performed in the order in which they are written.

Each specification line describes one operation, and is divided into three functional parts:

- a. Conditions under which the operation is to be performed (columns 7-17).
- b. The kind of operation to be performed and the data which is to be operated upon (columns 18-53).
- c. Tests to be made upon the results of the operation (columns 54-59).

## FIELD DEFINITIONS

Figure 9-1 can be used in conjunction with the following field definitions for the Calculation Specification.

1-2 PAGE

Refer to Section 2 for a complete description.

3-5 LINE

Refer to Section 2 for complete description.

6 FORM TYPE

A C must appear in this field.

7-8 CONTROL LEVEL

The valid entries for this field are listed below, and each entry is described in the paragraphs that follow.

## Burroughs B 1700 RPG

PROGRAM ID		PROGRAMMER		PAGE		DATE		OF																									
CALCULATION SPECIFICATIONS																																	
PROGRAM IDENTIFICATION 75 80																																	
PAGE		CONTROL LEVEL		HALF ADJUST		DECIMAL POSITIONS																											
LINE	INDICATORS	FACTOR 1	OPERATION	FACTOR 2	RESULT FIELD	FIELD LENGTH	Resulting Indicators																										
	AND AND						Arithmetic																										
	NOT NOT NOT						Plus Minus Zero																										
							Compare																										
							High Low Equal																										
							Table(Factor 2) is																										
							High Low Equal																										
3	5	6	7	8	9	10	11	12	13	14	15	16	17	18	27	28	32	33	42	43	48	49	51	52	53	54	55	56	57	58	59	60	74
0	1	C																															
		A	B	C	D	E	F	G	H	I	J	K	L																				

- 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, MLLZ0, MHHZ0, MLHZ0, MHLZ0, COMP, TESTZ, BITON, BITOF, TESTB, SETON, SETOF, GOTO, TAG, ZIP, LOKUP, BEGSR, ENDSR, EXSR, FORCE, EXCPT, DSPLY, READ, CHAIN, or DEBUG. Entry must be left-justified.
- E. 33-42 Contains either the name of any user or compiler defined field, any alphanumeric or numeric literal, a subroutine name, a vector name, any special name, a GOTO operation label, a filename or blank. Entry must be left-justified.
- F. 43-48 Specifies the name of the field, vector, or vector element that will be used to store the results of the operation. Entry must be alphanumeric and left-justified.
- G. 49-51 Specifies the length of the result field. Entries: Blank or any decimal numeric.
- H. 52 Specifies the number of decimal positions. If blank, entry is alphanumeric. Entries: Blank or 0-9.
- I. 53 Specifies if the contents of the result field are to be rounded off. Entries: Blank or H.
- J. 54-55 Entry is turned on if the result field is positive or FACTOR 1 is the highest in a compare operation, or FACTOR 2 is the highest in a lookup operation or a tested zone (TESTZ) is a plus-zone. Entries: 01-99, L1-L9, LR, H0-H9, 0A-0G, OV or blank.
- K. 56-57 Entry is turned on if the result field is negative, or FACTOR 1 is the lowest in a compare operation, or FACTOR 2 is lowest in a lookup operation, or a tested zone (TESTZ) is a minus-zone. Entries: 01-99, L1-L9, LR, H0-H9, 0A-0G, OV or blank.
- L. 58-59 Entry is turned on if the result field is zero, or FACTOR 1 is equal to FACTOR 2 in a compare or lookup operation, or a tested zone (TESTZ) is neither a plus- or minus-zone. Entries: 01-99, L1-L9, LR, H0-H9, 0A-0G, OV or blank.

614052

Figure 9-1. Calculation Specifications Summary Sheet

Entry	Definition
Blank	Calculation operation is part of detail calculations or may be part of a subroutine.
L0-L9	Calculation operation is performed in total calculations when the designated control break occurs or the specified indicator is set ON (L0 is always ON).
LR	Calculation operation is performed after the last record has been processed (or if set ON).
SR	Calculation operation is part of a subroutine (documentational not required for subroutines).
AN,OR	Establishes AND and OR relationship between lines of indicators.

An entry of L0-L9 or LR indicates that the operation is performed during total calculations when the specified indicator is ON. The first entry of L0-L9 or LR in columns 7-8 determines the start of total calculations. All detail calculations (if any are specified) must precede any total calculations. Any operations with blanks in columns 7-8 occurring before any total operation, or any BEGSR operation, are treated as detail operations, and are performed during detail calculations in every cycle depending upon the conditions specified in columns 9-17.

Subroutines, if used, must be specified after all detail and total lines. For a subroutine, column 7-8 may contain SR, OR, AN, or blank. The first BEGSR operation determines the start of subroutines.

**L0-L9** Control level indicators L0-L9 are used to condition operations that are to occur at a control break. If a control level indicator is specified in this field, the operation described on the same specification line is done only when the designated indicator is ON. When a control break for a certain level occurs, all lower control level indicators also turn ON. However, when a control level indicator used as a record identifying indicator turns ON to indicate a specific record type, or when a control level indicator is turned ON by a SETON operation, all lower level indicators remain OFF. Control level indicators need not be specified in any particular order. Operations will be executed in the order in which they are specified.

The L0 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 L0 indicator may be used to condition those operations. L0 indicator coding is illustrated in figure 9-2.

PAGE		CALCULATION SPECIFICATIONS																PROGRAM IDENTIFICATION																
FORM TYPE		CONTROL LEVEL						HALF ADJUST DECIMAL POSITIONS										75 80																
LINE		INDICATORS						FACTOR 1	OPERATION	FACTOR 2	RESULT FIELD	FIELD LENGTH	Resulting Indicators			COMMENTS																		
		AND	AND	AND	AND	AND	AND						Arithmetic	Plus	Minus		Zero																	
		NOT	NOT	NOT	NOT	NOT							High	Low	Equal																			
		9	10	11	12	13	14	15	16	17	18	27	28	32	33	42	43	48	49	51	52	53	54	55	56	57	58	59	60	74				
01	C		40					WAGE	ADD	TØTWAG	TØTWAG	82																						
02	C		50					WAGE	ADD	TØTWAG	TØTWAG																							
03	C								SETØF																									
04	C		40						SETØW																									
05	C	LO						TØTWAG	ADD	TØTAL	TØTAL	82																						
06	C	LR						TØTWAG	ADD	TØTAL	TØTAL																							
07	C	*																																
08	C	*	THE LO INDICATOR MAY BE USED TO CONDITION DETAIL OPERATIONS																															
09	C	*	SO THAT THEY WILL BE PERFORMED AT TOTAL CALCULATION TIME																															
10	C	*	(AHEAD OF NORMAL DETAIL CALCULATIONS).																															
11	C																																	
12	C																																	
13	C																																	
14	C																																	
15	C																																	

G14053

Figure 9-2. LO Indicator Coding Example

- LR**            The LR indicator automatically turns ON when the last record has been read and processed (all control level indicators L1-L9 also turn ON). This indicator is used to condition those operations that are to be performed at End-of-File.
- SR**            The SR entry is not an indicator; rather, it is used to indicate that the specification on the same line is part of a subroutine. All subroutine lines must be specified after all other calculation lines.
- AN-OR**        This field may also be used to specify that lines of indicators are in an AND or OR relationship. There is no limit on the number of AND or OR lines that may be specified; however, it is recommended that the user not exceed seven if compatibility with other Burroughs systems is desired. The last line of a group in an AND or OR relationship contains the Operation Code and all operands. All previous lines in the group must contain blanks in columns 18-59. The first line of a group may contain an L0-L9, LR, or SR entry if the entire group is conditioned by a control level indicator or is part of a subroutine. AND/OR relationship is illustrated in figure 9-3. Each AND/OR line, and the previous line, must contain at least one indicator in columns 9-17.

PAGE		CALCULATION SPECIFICATIONS																PROGRAM IDENTIFICATION																		
FORM TYPE		CONTROL LEVEL			HALF ADJUST DECIMAL POSITIONS																PROGRAM IDENTIFICATION															
LINE		INDICATORS			FACTOR 1	OPERATION	FACTOR 2	RESULT FIELD	FIELD LENGTH	Resulting Indicators			COMMENTS																							
		AND	AND							Plus	Minus	Zero																								
3	5	6	7	8	9	10	11	12	13	14	15	16	17	18	27	28	32	33	42	43	48	49	51	52	53	54	55	56	57	58	59	60	74			
01	C																																			
02	C	AN																																		
03	C	OR																																		
04	C	OR																																		
05	C	AN																																		
06	C	OR																																		
07	C	*																																		
08	C	*	AND/OR LINES ARE USED TO SPECIFY THE CONDITIONS UNDER WHICH																																	
09	C	*	THE ADD OPERATION SPECIFIED WILL BE PERFORMED. THIS ALLOWS																																	
10	C	*	EXTENSION OF THE CONDITIONS WHICH MAY BE SPECIFIED BY THE																																	
11	C	*	THREE PARTS OF THE INDICATORS FIELD.																																	
12	C																																			
13	C																																			
14	C																																			
15	C																																			

614054

Figure 9-3. AND/OR Relationship Coding Example

9-17 INDICATORS

This field is divided into three subfields such that up to three indicators on each line may be specified to condition a calculation operation. Each subfield is divided into two parts, as follows:

- a. NOT (one column).
- b. INDICATOR (two columns).

The NOT portion is used to specify that the associated indicator must be OFF in order for the operation to occur. If this condition is desired, an N must be entered in the NOT position; otherwise, the NOT portion must be left blank.

The INDICATOR portion is used to specify the indicator to be tested for ON (NOT = blank) or OFF (NOT = N). Only the following entries are allowed in this portion of the INDICATOR field:

Entry	Definition
Blank	Operation not conditioned by any indicator.
01-99	Operation conditioned by indicator used elsewhere in the program.
L0-L9	Operation conditioned by control level indicator previously assigned.
LR	Operation conditioned by last record indicator.
MR	Operation conditioned by matching record indicator.
H0-H9	Operation conditioned by halt indicator used elsewhere in the program.
U1-U8	Operation conditioned by external indicator previously defined.
OA-OG, OV	Operation conditioned by overflow indicator previously defined.

All three indicators on one line are in an AND relationship. All indicators on one line (or grouped lines), plus the Control Level Indicator (if used) must be ON or OFF as specified, in order for the associated operation to take place.

Each of the indicators is discussed in the paragraphs that follow.

- 01-99            The indicators 01-99 may be used to condition calculation operations that are to be performed only for specific input record types (record identifying indicators), to condition calculation operations that are to be performed only when an input field meets certain conditions (field indicators), or to condition calculation operations according to the results of previous operations in the Calculation Specifications (resulting indicators).
- L0-L9            The control level indicators, L1-L9, assigned in the CONTROL LEVEL field (Input Specifications) or the RESULTING INDICATORS field (Calculation Specifications), may be used to condition operations that are to be performed only on the first record of a new Control Group. L0 may be specified, but is always ON.
- LR                The LR indicator is used to condition operations that are to be performed at End-of-Job.
- MR                The MR indicator is used to condition operations that are to be performed only when matching input records are found.

- HO-H9            The halt indicators, previously assigned in the FIELD INDICATORS field (Input Specifications) or the RESULTING INDICATORS field (Calculation Specifications), may be used to condition 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 results. By using the halt indicators in conjunction with an N in the NOT portion of the INDICATORS field, an operation may be inhibited when the specified halt indicator is ON.
- UI-U8            The external indicators, UI-U8, may be used to condition operations that are to be performed only when the specified indicator has been set at execution time. See FILE CONDITION field (Section 4) for a complete description of external indicators.
- OA-OG,OV        The overflow indicators, OA-OG, OV, previously assigned in the OVERFLOW INDICATOR field (File Description Specifications), may be used to condition operations that are to be performed when the overflow line on a printer file has been reached.

The following special rules must be observed in specifying indicators:

- a. When a control level indicator (L1-L9) is used in the INDICATORS field and not in the CONTROL LEVEL field, the operation conditioned by the designated indicator is performed only for the record which caused the Control Break or any higher-level Control Break.
- b. All operations conditioned by control level indicators specified in the CONTROL LEVEL field are performed before those conditioned by control level indicators specified in the INDICATORS field.
- c. If a control level indicator is specified in the CONTROL LEVEL field and the matching record indicator (MR) is specified in the INDICATORS field, MR indicates a matching condition in the previous record (not the record just read, which caused the Control Break). After all operations conditioned by control level indicators in the CONTROL LEVEL field are done, MR then indicates the matching condition on the record just read.

18-27    FACTOR 1

This field is used to supply the data to be operated upon by the Operation Code specified in columns 28-32.

Allowable entries are:

- a. The name of any field previously defined or to be defined later.
- b. A literal (alphanumeric or numeric).
- c. The name of a subroutine (BEGSR operation only).

- d. A table or array name (vector name) or an element of a vector. A vector element is composed of the vector name, followed by a comma, followed by the desired index value.
- e. The special words UDATE, UMONTH, UDAY, UYEAR, PAGE, PAGE1, and PAGE2.
- f. A label (TAG or ENDSR operation only).

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

#### 28-32 OPERATION

This field is used to specify the proper Operation Code for the type of operation to be performed using FACTOR 1 and FACTOR 2. The Operation Code must be left-justified in the field.

Operations are performed in the order in which they appear on the Calculation Specifications sheet; however, all operations conditioned by control level indicators specified in the CONTROL LEVEL field (except those which belong to subroutines) must appear after those operations not conditioned by control level indicators.

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

#### 33-42 FACTOR 2

This field is used to supply the data to be operated upon by the Operation Code specified in columns 28-32. Allowable entries are:

- a. The name of any field defined elsewhere in the program.
- b. A literal (alphanumeric or numeric).
- c. The name of a subroutine (EXSR operation only).
- d. A table or array name (vector name) or an element of a vector. A vector element is composed of the vector name, followed by a comma, followed by the desired index value.
- e. The special words UDATE, UMINTH, UDAY, UYEAR, PAGE, PAGE1, and PAGE2.
- f. A label (GOTO operation only).
- g. A filename (DEBUG, DSPLY, CHAIN, READ, or FORCE operation only).

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



43-48 RESULT FIELD

This field is used to name the field, vector, or vector element that will be used to store the results of the operation specified. The use of this field is dependent upon the particular operation being specified. Any vector or vector element previously defined or any field defined here or elsewhere in the Input or Calculation specifications may be used as a RESULT FIELD.

The special words UDATE, UDAY, UMONTH, UYEAR must not be specified as RESULT FIELD.

Definition of a new field is also allowed, and is accomplished by entering the new field name, along with FIELD LENGTH and DECIMAL POSITIONS entries for that field. The field name specified must be entered left-justified, must begin with an alphabetic character, and must contain no imbedded blanks or special characters.

49-51 FIELD LENGTH

This field is used to specify the length of a result field that has not been previously defined. Numeric and alphanumeric fields are limited to the size specified in Section 1.

If this entry is left blank the field must be defined elsewhere. This field must be blank if RESULT FIELD is blank. It is allowable to enter the length of a field that has been previously defined; however, the length and number of decimal positions specified must be the same as that previously defined.

Entries in this field must be right-justified, and leading zeros are optional.

52 DECIMAL POSITIONS

This field is used to specify the number of positions to the right of the implied decimal point in a numeric result field. If the RESULT FIELD is alphanumeric, or the FIELD LENGTH is blank, this field must be left blank. If an entry is made, however, it must agree with the previous entry specified for the field. The valid entries are:

Entry	Definition
Blank	Alphanumeric field (or a field defined elsewhere).
0-9	Number of decimal positions in a numeric field.

Numeric result fields when defined must have the number of decimal positions specified; if the field only contains integral values, a zero must be entered to indicate no decimal positions.

The number of decimal positions specified must not exceed the length of the RESULT FIELD, as specified by the FIELD LENGTH entry.

53 HALF ADJUST

This field is used to indicate whether or not the contents of the RESULT FIELD are to be rounded. Rounding is accomplished by adding 5 (or -5 for negative values) to the digit to the right of the last decimal position of the RESULT FIELD. Then all digits to the right of the last decimal position are dropped. Valid entries for this field are:

Entry	Definition
Blank	Do not half adjust.
H	Half adjust.

Entries in this field are allowable only for certain arithmetic operations.

54-59 RESULTING INDICATORS

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.
0A-0G,0V	Any overflow indicator.

Indicators specified in these columns are set OFF immediately before the operation is performed. Immediately after the operation, indicators are set ON to indicate the result of the operations. The use of the result indicators is dependent on the type of operation specified.

Resulting Indicators must not be specified when the RESULT FIELD is an array name, the only exception being a LOKUP operation.

Figure 9-4 illustrates the use of resulting indicators and various operation codes. The specific uses of the result fields is further described in conjunction with the discussion on the various operation code types in this section.

60-74 COMMENTS

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

75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a complete description.



## OPERATION CODES

Various categories of Operation Codes are provided to allow most types of data manipulation required by the RPG programmer. Operation Codes are entered only in the OPERATION field of the Calculation Specifications, and specify what type of arithmetic or logical operation is to be performed on the associated operands. In the following paragraphs an indexed vector or a table name is valid anywhere that a field name is valid.

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

### ARITHMETIC OPERATIONS

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

FACTOR 1, FACTOR 2 and the RESULT FIELD may all be different fields or any two or all three may be the same field. HALF ADJUST may be specified for all operations except SQRT and MVR (i.e., DIV when followed by MVR). Resulting indicators may be specified for all operations provided the RESULT FIELD is not a whole array.

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

The use of result indicators in connection with arithmetic operations is as shown below:

- a. Plus (Columns 54-55). Any indicator entered in this field will be turned on if the result of an arithmetic operation is positive.
- b. Minus (Columns 56-57). Any indicator entered in this field will be turned on if the result of an arithmetic operation is negative.
- c. Zero (Columns 58-59). Any indicator entered in this field will be turned on if the result of an arithmetic operation is zero.

Each of the arithmetic operations is discussed in the paragraphs that follow.

Table 9-1. Summary of Operation Codes

Operation Type	Entry Columns 28-32	Definition	Control Level	Indicators	Factor 1	Factor 2	Result Field	Field Length	Decimal Position	Halt Adjust	Resulting Indicators
Arithmetic Operations	ADD	Add Factor 2 to Factor 1	0	0	R	R	R	0	0	0	0
	Z-ADD	Clear Result Field and add Factor 2	0	0	B	R	R	0	0	0	0
	SUB	Subtract Factor 2 from Factor 1	0	0	R	R	R	0	0	0	0
	Z-SUB	Clear Result Field and subtract Factor 2	0	0	B	R	R	0	0	0	0
	MULT	Multiply Factor 1 by Factor 2	0	0	R	R	R	0	0	0	0
	DIV	Divide Factor 1 by Factor 2	0	0	R	R	R	0	0	0	0
	MVR	Move remainder of preceding division to a Result Field	0	0	B	B	R	0	0	B	0
	XFOOT	Sum elements of an array and put sum in Result Field	0	0	B	R	R	0	0	0	0
	SQRT	Derive the square root of Factor 2	0	0	B	R	R	0	0	B	B
Move Operation	MOVE	Move Factor 2 into Result Field, right-justified	0	0	B	R	R	0	0	B	0
	MOVEL	Move Factor 2 into Result Field, left-justified	0	0	B	R	R	0	0	B	0
Move Zone Operation	MLLZO	Move zone from low-order position of Factor 2 to low-order position of Result Field	0	0	B	R	R	0	0	B	B
	MHHZO	Move zone from high-order position of alphanumeric Factor 2 to high order of alphanumeric Result Field	0	0	B	R	R	0	B	B	B
	MLHZO	Move zone from low-order position of Factor 2 to high-order position of alphanumeric Result Field	0	0	B	R	R	0	B	B	B
	MHLZO	Move zone from high-order position of alphanumeric Factor 2 to low-order position of Result Field	0	0	B	R	R	0	0	B	B
Compare Operations	COMP	Compare Factor 1 to Factor 2	0	0	R	R	B	B	B	B	R
	TESTZ	Tests the zone portion of the leftmost character of Result Field	0	0	B	B	R	0	B	B	R
Binary Field Operations	BITON	Set on specified bits	0	0	B	R	R	0	B	B	B
	BITOF	Set off specified bits	0	0	B	R	R	0	B	B	B
	TESTB	Test specified bits	0	0	B	R	R	0	B	B	R
Setting Indicators	SETON	Set one, two, or three specific indicators on	0	0	B	B	B	B	B	B	R
	SETOF	Set one, two, or three specific indicators off	0	0	B	B	B	B	B	B	R
Program Branching Operations	GOTO	Branch to another calculation specification line	0	0	B	R	B	B	B	B	B
	TAG	Identify the name in Factor 1 as a destination label to which GOTO may branch	0	B	R	B	B	B	B	B	B
Transfer Control Function	ZIP	Initialize System commands	0	0	B	R	B	B	B	B	B
Lookup Operations	LOKUP	Table or array lookup	0	0	R	R	0	0	0	B	R
Sub-routines	BEGSR	Beginning of the subroutine	0	B	R	B	B	B	B	B	B
	ENDSR	End of the subroutine	0	B	0	B	B	B	B	B	B
	EXSR	Call to execute the subroutine	0	0	B	R	B	B	B	B	B
Program Control of Input and Output	FORCE	Forcing record to be read next	B	0	B	R	B	B	B	B	B
	EXCPT	Causes output	0	0	B	B	B	B	B	B	B
	DSPLY	A field is printed on the console printer and/or data is entered via the console printer into a field	0	0	0	R	0	0	0	B	B
	READ	A record is read from a demand file	0	0	B	R	B	B	B	B	EI
	CHAIN	A record is read from a disk file	0	0	R	R	B	B	B	B	EI
Debug Operation	DEBUG	Aid in finding programming errors	0	0	0	R	0	B	B	B	B

NOTES

- 0 - Optional
- R - Required
- B - Blank
- EI - Only the Equal Indicator columns may be used. The indicator will be turned on if end-of-file occurs on the demand file while reading or attempting to read.

**ADD** This operation adds the contents of FACTOR 2 to the contents of FACTOR 1 and stores the sum in the RESULT FIELD. FACTOR 1 and FACTOR 2 are not affected by this operation, unless one of them is also designated as the RESULT FIELD. HALF ADJUST may be specified.

**SUB** This operation subtracts the contents of FACTOR 2 from the contents of FACTOR 1 and places the difference in the RESULT FIELD. FACTOR 1 and FACTOR 2 are not affected by this operation, unless one of them is also designated as the RESULT FIELD.

Note that subtracting two fields which are the same gives the same result as clearing the RESULT FIELD to zero, and can be used as a method of clearing fields to zero.

**MULT** This operation multiplies the contents of FACTOR 1 by the contents of FACTOR 2 and stores the product in the RESULT FIELD. FACTOR 1 and FACTOR 2 are not affected by this operation, unless one of them is also designated as the RESULT FIELD. HALF ADJUST may be specified.

Table 9-2 provides an example of the contents of the RESULT FIELD for the multiplication operation, showing various field lengths and decimal positions. In the example, all fields that are permitted are shown, although some contain incomplete results. Fields not permitted are blank; HALF ADJUST is not specified. Note that a field length of eight with five decimal positions gives all the significant digits without adding zeros either left or right.

**DIV** This operation divides the contents of FACTOR 1 by the contents of FACTOR 2 and places the quotient in the RESULT FIELD. FACTOR 1 and FACTOR 2 are not affected by this operation, unless one of them is also designated as the RESULT FIELD.

Any remainder resulting from the divide operation will be lost unless the next operation specified is the MOVE REMAINDER operation (MVR); if so, the result of the divide operation cannot be half adjusted.

If FACTOR 2 is equal to zero, the operator is notified and the program discontinues.

**MVR** This operation moves the REMAINDER from a previous DIV operation to the designated RESULT FIELD. The MVR operation must immediately follow the DIV operation, and FACTOR 1 and FACTOR 2 must be left blank. The RESULT FIELD for the MVR operation must be the same length as FACTOR 2 for the DIV operation; if the RESULT FIELD is shorter, truncation may occur. Both the DIV and the MVR may be conditioned on indicators, which need not be the same for both operations. However, the programmer must ensure that the MVR operation is never performed without the DIV, otherwise the result of the MVR is undefined. HALF ADJUST must not be specified.

Table 9-2. RESULT FIELD Contents for Various Field Lengths and Decimal Positions - MULT Operation

Field Length	Decimal Positions									
	0	1	2	3	4	5	6	7	8	9
1	2	.5								
2	92	2.5	.54							
3	192	92.5	2.54	.546						
4	0192	192.5	92.54	2.546	.5467					
5	00192	0192.5	192.54	92.546	2.5467	.54672				
6	000192	00192.5	0192.54	192.546	92.5467	2.54672	.546720			
7	0000192	000192.5	00192.54	0192.546	192.5467	92.54672	2.546720	.5467200		
8	00000192	0000192.5	000192.54	00192.546	0192.5467	192.54672	92.546720	2.5467200	.54672000	
9	000000192	00000192.5	0000192.54	000192.546	00192.5467	0192.54672	192.546720	92.5467200	2.54672000	.546720000
10	0000000192	000000192.5	00000192.54	0000192.546	000192.5467	00192.54672	0192.546720	192.5467200	92.54672000	2.546720000

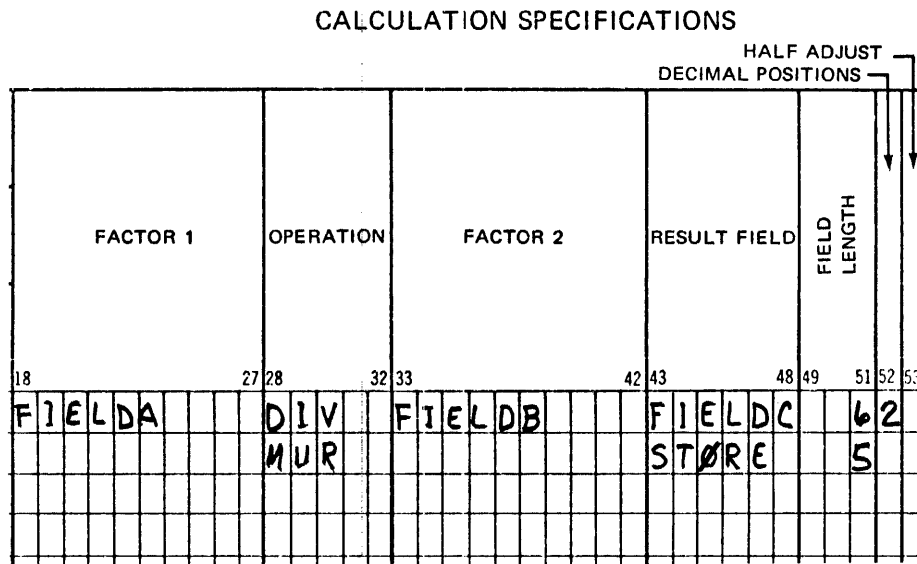
NOTE

RESULT FIELD contents for the multiplication operation: 89.67 MULT 2.148.

The following considerations should be made when specifying the size of the RESULT FIELD:

- a. The number of significant decimal places in the REMAINDER is the larger of:
  - 1) The number of decimal positions in FACTOR 1 of the previous DIV operation.
  - 2) The sum of the decimal positions in FACTOR 2 and the RESULT FIELD of the previous DIV operation.
- b. The maximum integer positions in the REMAINDER equals the integer positions in FACTOR 2 of the previous DIV operation.
- c. The RESULT FIELD must not be a whole array.

Figure 9-5 provides a DIV and MVR coding example.



G14056

Figure 9-5. DIV and MVR Coding Example

**SQRT** The operation derives the SQUARE ROOT of the contents of FACTOR 2 and places it in the RESULT FIELD. FACTOR 1 must be left blank.

To obtain reasonable precision from SQRT, the following points should be observed:

- a. For every digit left of the decimal place in the RESULT FIELD, there should be two digits left of the decimal placed in FACTOR 2.
- b. For every digit right of the decimal point in the RESULT FIELD, there should be two digits right of the decimal point in FACTOR 2.



The result of the SQRT operation is automatically adjusted; therefore, a HALF ADJUST entry for SQRT is not allowed.

If the FACTOR 2 operand is negative, the program will display a "SQRT" error message and halt. The operator may resume processing, in which case the RESULT FIELD will set to zero.

Table 9-3 provides an example of the contents of the RESULT FIELD for the square root operation. In the example, all fields that are permitted are shown, although some contain incomplete results. Fields not permitted are blank.

**XFOOT** This operation is used to crossfoot total the elements of a numeric array. All the elements of the array specified by FACTOR 2 are summed and the total placed in the RESULT FIELD. FACTOR 1 must be left blank. RESULT FIELD must not be a whole array.

Resulting indicators may be specified. If the RESULT FIELD is an element of the array named in FACTOR 2, the value of that element before the XFOOT operation is used in obtaining the total. HALF ADJUST may be specified.

**Z-ADD** This operation sets the RESULT FIELD to zeros, and adds the contents of FACTOR 2 to the RESULT FIELD. FACTOR 1 must be left blank. FACTOR 2 is not affected by this operation. HALF ADJUST may be specified.

Note that this is equivalent to a right-adjust numeric move with the zero-fill and/or truncation.

**Z-SUB** This operation sets the RESULT FIELD to zeros, then subtracts the contents of FACTOR 2 from the RESULT FIELD. FACTOR 1 must be left blank. This operation is used to change the sign of the field designated by FACTOR 2. HALF ADJUST may be specified.

Note that this is the same as Z-ADD, but the sign is changed.

In the example shown in figure 9-6, if the contents of FIELD A equals 678.9321, then the contents of FIELD B will equal 000678.93, and the contents of FIELD C will equal -000678.93.

#### MOVE OPERATIONS

MOVE operations transfer the contents of FACTOR 2 to the RESULT FIELD. FACTOR 2 is not affected by these operations. FACTOR 1 must be left blank. RESULTING INDICATORS may not be specified. If FACTOR 2 and RESULT FIELD are identical (same length), MOVE and MOVE L operate the same. FACTOR 2 may be a field or literal; however, a RESULT FIELD may be a field or a vector.

Table 9-3. RESULT FIELD Contents for Various Field Lengths  
and Decimal Positions - SQRT Operation

Field Length	Decimal Positions								
	0	1	2	3	4	5	6	7	8
1	5	.7							
2	55	4.7	.72						
3	055	54.7	4.72	.721					
4	0055	054.7	54.72	4.721	.7213				
5	00055	0054.7	054.72	54.721	4.7213	.72130			
6	000055	00054.7	0054.72	054.721	54.7213	4.72130			
7	0000055	000054.7	00054.72	0054.721	054.7213	54.72130	.721305		
8	00000055	0000054.7	000054.72	00054.721	0054.7213	054.72130	4.721305	.7213048	
9	000000055	00000054.7	0000054.72	000054.721	00054.7213	0054.72130	54.721305	4.7213048	.72130482
10	0000000055	000000054.7	00000054.72	0000054.721	000054.7213	00054.72130	054.721305	54.7213048	4.72130482
11	00000000055	0000000054.7	000000054.72	00000054.721	0000054.7213	000054.72130	0054.721305	054.7213048	54.72130482
12	000000000055	00000000054.7	0000000054.72	000000054.721	00000054.7213	0000054.72130	00054.721305	0054.7213048	054.72130482
13	0000000000055	000000000054.7	00000000054.72	0000000054.721	000000054.7213	00000054.72130	000054.721305	00054.7213048	0054.72130482
14	00000000000055	0000000000054.7	000000000054.72	00000000054.721	0000000054.7213	000000054.72130	0000054.721305	000054.7213048	00054.72130482
15	000000000000055	00000000000054.7	0000000000054.72	000000000054.721	00000000054.7213	0000000054.72130	00000054.721305	0000054.7213048	000054.72130482

NOTE

RESULT FIELD contents for the square  
root operation: SQRT 2994.42.

### CALCULATION SPECIFICATIONS

HALF ADJUST -															
DECIMAL POSITIONS															
OPERATION	FACTOR 2				RESULT FIELD				FIELD LENGTH						
28	32	33	42	43	48	49	51	52							
Z-ADD	F	I	E	L	D	A			F	I	E	L	D	B	82
Z-SUB	F	I	E	L	D	A			F	I	E	L	D	C	82

G14057

Figure 9-6. Z-ADD and Z-SUB Coding Example

An array name may be entered in FACTOR 2 as long as the RESULT FIELD also specifies an array. In this case, the designated move operation will be performed on each element of the array designated in FACTOR 2, with the result being placed in the corresponding elements of the array designated in the RESULT FIELD. The operation is terminated when the end of the shorter array is reached. If RESULT FIELD is an array and FACTOR 2 is not, then the designated move operation will be performed on FACTOR 2, with the result being placed in all elements of the RESULT FIELD array.

For the purpose of MOVE, MOVEL, the "sign" of the variable is defined as (1) the algebraic sign if the variable is numeric, or (2) the low order zone if the variable is alphanumeric.

Move operations may be specified between fields of different data types. This will convert an alphanumeric to a numeric and vice versa. Decimal points are ignored in all move operations. HALF ADJUST may not be specified with MOVE or MOVEL.

**MOVE** This operation moves characters from FACTOR 2 to the RESULT FIELD, starting with the rightmost character and continuing until either the source field is exhausted, or the destination field is filled. The sign of FACTOR 2 is moved to the sign position of the RESULT FIELD.

When an alphanumeric-to-numeric move is specified, the digit portion of each character is moved to the RESULT FIELD. Blanks are transferred as zeros. The zone portion of the rightmost alphanumeric character is used as the sign of the RESULT FIELD.

When a numeric-to-alphanumeric move is specified, each digit is converted to its corresponding internal character code as it is moved to the RESULT FIELD. The sign of the numeric field is placed in the zone of the rightmost character.

MOVE operations are shown in table 9-4.

Table 9-4. MOVE Operations

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

**MOVE** This operation moves, left-justified, characters from FACTOR 2 to the RESULT FIELD, starting with the leftmost character and continuing until either the source field is exhausted, or the destination field is filled.

When a numeric to alphanumeric move is specified, each digit is converted to its corresponding internal character code as it is moved to the RESULT FIELD.

When an alphanumeric-to-numeric move is specified, only the digit portion of each character is moved to the RESULT FIELD. Blanks are transferred as zeros.

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

MOVE operations are shown in table 9-5.

#### MOVE ZONE OPERATIONS

Move zone operations are used to move the zone portion of only one character. FACTOR 2 is not affected by this operation, and only the zone portion of one character of the RESULT FIELD is affected. FACTOR 1 must be left blank. HALF ADJUST and resulting indicators must not be specified.

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

Table 9-5. MOVE Operations

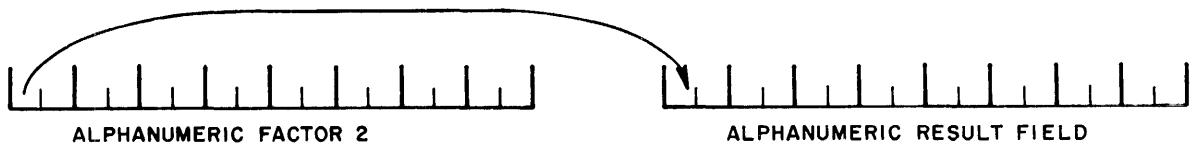
Contents of FACTOR 2	Contents of Numeric RESULT FIELD		Contents of Alphanumeric RESULT FIELD	
	Before MOVE Operation	After MOVE Operation	Before MOVE Operation	After MOVE Operation
-78.425	+567.84	-784.25	ABCDE	7842N
PH4SK	+56789	-78422	ABCDE	PH4SK
-000008425	+5.6789	-0.0000	ABCDE	00000
BRWCXH4SK	+56789	-29637	ABCDE	BRWCX
-78425	+1.30943210	+7.84253210	ABCDEFGHI	7842NFGHI
GEB5N	-730943210	-752553210	ABCDEFGHI	GEB5NFGHI

An array name may be entered in FACTOR 2, as long as the RESULT FIELD also specifies an array. In this case, the designated move zone operation will be performed on each element of the array designated in FACTOR 2, with the result being placed in the corresponding elements of the array designated in the RESULT FIELD. The operation is terminated when the end of the shorter array is reached.

If the RESULT FIELD is an array, but FACTOR 2 is not, the zone from FACTOR 2 is moved into the appropriate position in all elements of the array.

Each of the move zone operations is discussed in the following paragraphs.

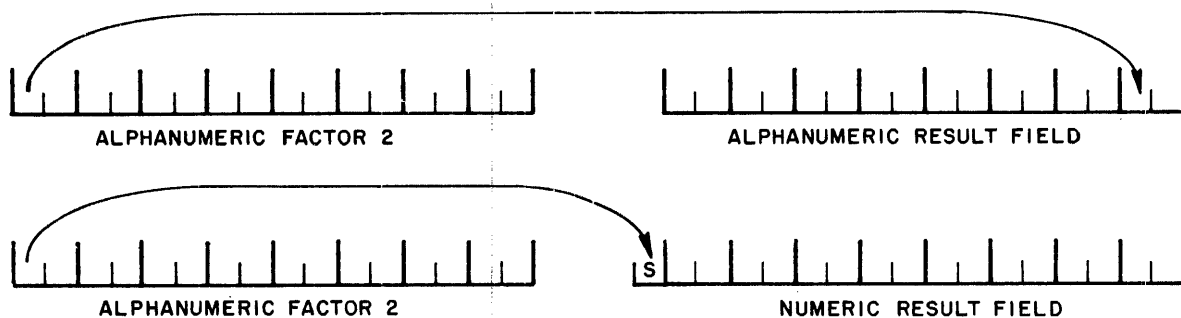
**MHHZO** This operation moves the zone from the high-order (leftmost) position of FACTOR 2 to the high-order (leftmost) position of the RESULT FIELD. Both FACTOR 2 and RESULT FIELD must be alphanumeric (see figure 9-7).



614058

Figure 9-7. MHHZO Move Zone Operation

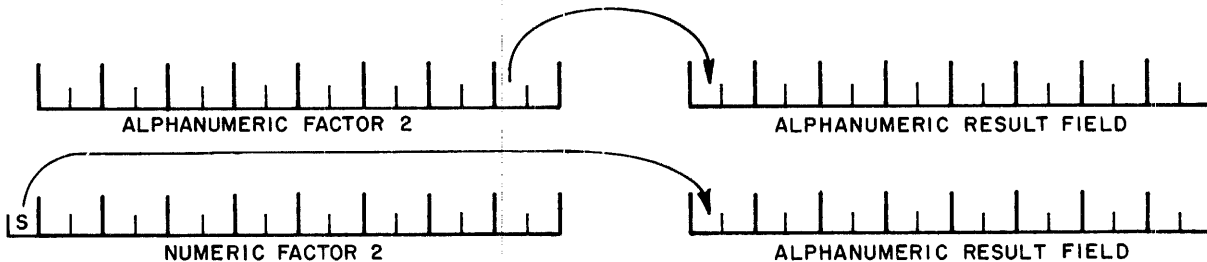
**MHLZO** If FACTOR 2 is alphanumeric and the RESULT FIELD is alphanumeric, this operation moves the zone from the high-order (leftmost) position of FACTOR 2 to the low-order (rightmost) position of the RESULT FIELD. If FACTOR 2 is alphanumeric and the RESULT FIELD is numeric, this operation moves the high-order (leftmost) position of FACTOR 2 to the sign position of the RESULT FIELD. FACTOR 2 must be alphanumeric (see figure 9-8).



G14059

Figure 9-8. MHLZO Move Zone Operations

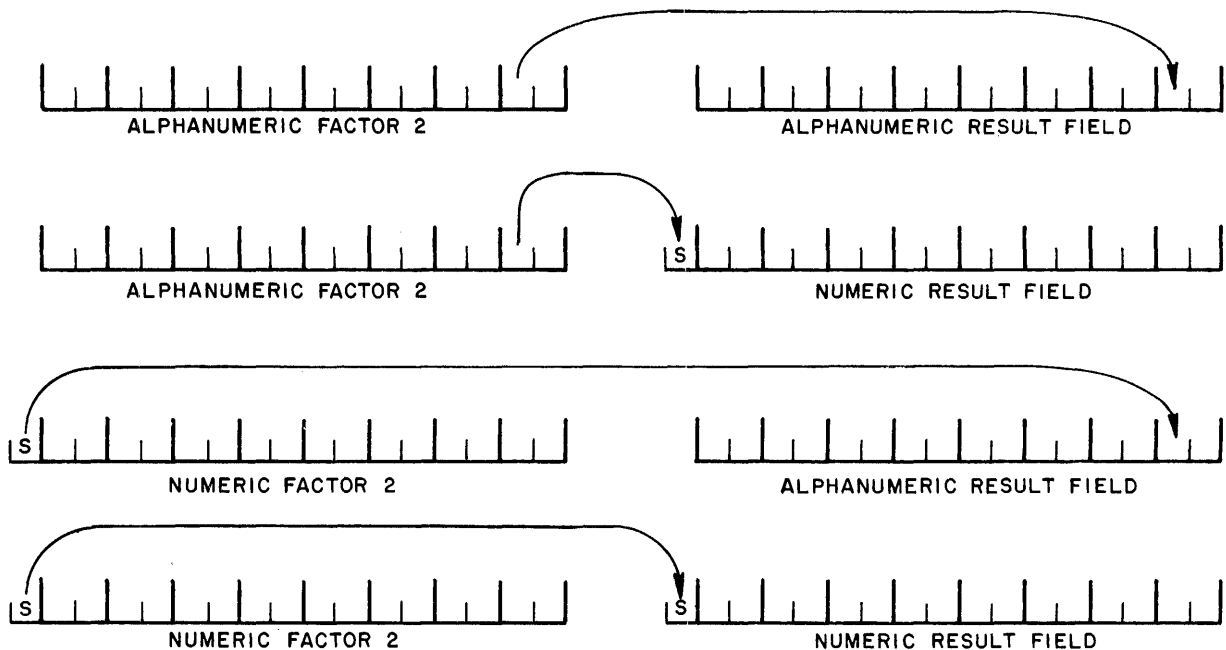
**MLHZO** If FACTOR 2 is alphanumeric and the RESULT FIELD is alphanumeric, this operation moves the zone from the low-order (rightmost) position of FACTOR 2 to the high-order (leftmost) position of the RESULT FIELD. If FACTOR 2 is numeric and the RESULT FIELD is alphanumeric, this operation moves the zone from the sign position of FACTOR 2 to the high-order (leftmost) position of the RESULT FIELD. The RESULT FIELD must be alphanumeric (see figure 9-9).



G14060

Figure 9-9. MLHZO Move Zone Operations

**MLLZO** If FACTOR 2 is alphanumeric and the RESULT FIELD is alphanumeric, this operation moves the zone from the low-order (rightmost) position of FACTOR 2 to the low-order (rightmost) position of the RESULT FIELD. If FACTOR 2 is alphanumeric and the RESULT FIELD is numeric, this operation moves the zone from the low-order (rightmost) position of FACTOR 2 to the sign position of the RESULT FIELD. If FACTOR 2 is numeric and the RESULT FIELD is alphanumeric, this operation moves the zone from the sign position of FACTOR 2 to the low-order (rightmost) position of the RESULT FIELD. If FACTOR 2 is numeric and the RESULT FIELD is numeric, this operation moves the zone from the sign position of FACTOR 2 to the sign position of the RESULT FIELD (see figure 9-10).



614061

Figure 9-10. MLLZO Move Zone Operations

#### COMPARE OPERATIONS

These operations are used to test specified fields for certain conditions. The results of these operations are shown by the setting of the specified RESULTING INDICATOR (01-99, L1-L9, LR, HO-H9, OA-OG, OV). FACTOR 1 and FACTOR 2 are not affected by the operation. At least one RESULTING INDICATOR must be specified. The RESULT FIELD must be blank. HALF ADJUST must not be specified. FACTOR 1 and/or FACTOR 2 must not be a whole array.

COMP This operation compares FACTOR 1 with FACTOR 2, causing the RESULTING INDICATORS to be set as follows:

- a. HIGH (columns 54-55) - FACTOR 1 > FACTOR 2
- b. LOW (columns 56-57) - FACTOR 1 < FACTOR 2
- c. EQUAL (columns 58-59) - FACTOR 1 = FACTOR 2

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

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

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

TESTZ This operation tests the ZONE portion of the leftmost character of the RESULT FIELD, setting the specified RESULTING INDICATOR (01-99, L1-L9, LR, H0-H9, OA-OG, OV) to the results of the test. FACTOR 1 and FACTOR 2 must be blank. HALF ADJUST must not be specified. The RESULT FIELD must not be a whole array.

If the RESULT FIELD is alphanumeric, the character under test will set a specific RESULT INDICATOR according to the following:

<u>Character Under Test</u>	<u>RESULTING INDICATOR Set</u>
A-I, &	PLUS (columns 54-55)
J-R, -	MINUS (columns 56-57)
All Others	ZERO (columns 58-59)

If the RESULT FIELD is numeric, the PLUS (columns 54-55) or MINUS (columns 56-57) indicator will be set according to the sign of the field. A ZERO indicator must not be specified for a numeric field. Note that TESTZ is essentially a zone portion test and will differentiate between +0 and -0. If the programmer wishes to test for greater than, less than, or equal to zero, he should use a compare (COMP opcode) against zero, instead of TESTZ.

### BINARY FIELD OPERATIONS

Three operation codes, BITON, BITOF, and TESTB, are provided to set and test individual bits. The individual bits can be used as switches in a program.

In binary field operations the operation codes BITON, BITOF, or TESTB are used. FACTOR 2 may contain either of the following:



- a. Bit values 0-7. One or more bits (maximum of eight) may be set ON, set OFF, or tested per operation. The bits are numbered from left to right and are enclosed in apostrophes. The order of specification of the bits is not restricted. Bits not specified in FACTOR 2 are not changed.
- b. The name of a one-position, alphanumeric field or table, or array element. In this case, the bits which are on in the field or array element are set ON, set OFF, or tested in the RESULT FIELD; bits which are not ON are not affected.

**BITON** This operation code causes bits identified in FACTOR 2 to turn ON (set to one) in a previously defined field named as the RESULT FIELD. The operation code BITON must appear in columns 28-32. Conditioning indicators can be used in columns 7-17. Any entry under FIELD LENGTH must be 1.

FACTOR 1, DECIMAL POSITIONS, HALF ADJUST, and RESULTING INDICATORS are not used with the BITON operation (see figure 9-11).

**CALCULATION SPECIFICATIONS**

OPERATION	FACTOR 2	RESULT FIELD	FIELD LENGTH	Resulting Indicators			PI	IC												
				Plus	Minus	Zero														
				Compare																
				High 1>2	Low 1<2	Equal 1=2														
				Lookup																
				Table(Factor 2) is																
				High	Low	Equal														
				7	28	32	33	42	43	48	49	51	52	53	54	55	56	57	58	59
BITON	'1234'	BITA																		
BITON	ITEMA	BITB																		
BITOFF	ITEMB	ARR,XY																		
BITOFF	'123'	ARR,XY																		
BITOFF	ARR,XY	TBL,20																		
TESTB	'057'	BITC													10	12	12			

614062

Figure 9-11. Binary Field Operations Coding Examples

**BITOF** This operation code causes bits identified in FACTOR 2 to turn OFF (set to zero) in a previously defined field named as the RESULT FIELD.

The operation code BITOF must appear in columns 28-32. All other specifications are the same as those for the BITON operation (see figure 9-11).

**TESTB** This operation code causes bits identified in FACTOR 2 to be tested for an ON or OFF condition in the previously defined field named as a RESULT FIELD, setting the specified RESULTING INDICATOR to the result of the test. All other specifications are the same as those for BITON and BITOF.

At least one RESULTING INDICATOR must be used with the TESTB operation; as many as three can be named for one operation. Two indicators may be the same for one TESTB operation, but not three. If FACTOR 2 contains bits which are all OFF, no RESULTING INDICATORS are turned ON. RESULTING INDICATORS have the meanings described in the following paragraphs.

#### Columns 54-55

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

#### Columns 56-57

An indicator in these columns is turned ON if two or more bits were tested and found to be of mixed status; that is, some bits ON and other bits OFF. It is important to ensure that the field named in FACTOR 2 contains more than one bit which is ON if an indicator appears in columns 56-57.

#### Columns 58-59

An indicator in these columns is turned ON if each bit specified in FACTOR 2 is ON (1) in the Result Field.

The following explanations refer to figure 9-11.

- a. Bits 1234 are turned ON in the field named BITA.
- b. Bits that are ON in the field named ITEMA will cause the corresponding bits in the field named BITB to be turned ON.
- c. Bits that are OFF in the field named ITEMB will cause the corresponding bits in the array element ARR,XY to be turned OFF. Then bits 1, 2, 3 will be turned off in array element ARR,XY, and finally the bits that are off in ARR,XY will be turned off in the array element TBL,20.
- d. If bits 0, 5, and 7 are OFF in the field named BITC, indicator 10 will be turned ON. If bits 0, 5 and 7 are of mixed status in the field named BITC, indicator 12 will be turned ON. If bits 0, 5, and 7 are ON in the field named BITC, indicator 12 will be turned ON.

## SETTING INDICATORS

Up to three indicators may be set ON or OFF with one operation. These may be entered in the RESULTING INDICATORS field. FACTOR 1, FACTOR 2, HALF ADJUST, and the RESULT FIELD must be left blank. The following rules must be observed when setting indicators:

- a. The following indicators may not be set ON or OFF: 1P, MR, LO, or U1-U8.
- b. Setting a control level indicator (L1-L9) ON or OFF does not affect any other control level indicator.
- c. All control level and record identifying indicators are automatically turned OFF after detail output operations are completed, regardless of any previous SETON or SETOF operations.
- d. If any halt indicators (H0-H9) are set ON and are not turned OFF before the detail output operations finish, the program will halt.
- e. If the LR indicator is turned ON by a SETON operation which is conditioned by a control level indicator (columns 7-8 on Calculations Specifications form), the program will stop after all total output operations are completed. If the LR indicator is turned ON by a SETON operation which is not conditioned by a control level indicator, the program stops after the next total output operation is completed.

**SETON** This operation sets the indicators entered in the RESULTING INDICATORS field ON.

**SETOF** This operation sets the indicators entered in the RESULTING INDICATORS field OFF.

## PROGRAM BRANCHING OPERATIONS

Operations within the Calculation Specifications are normally performed in the order in which they are written. Branching operations allow variation of the order of operation; thus, conditional branching and repetitive operations are possible.

**GOTO** This operation causes the program to branch (GOTO) to some other instruction rather than "falling through" to the next sequential operation. Branching both forward and backward is allowed. Branching to a TOTAL CALCULATION from a DETAIL CALCULATION and vice versa is allowed. Branching into or out of subroutines is not allowed.

FACTOR 2 must contain a label (which must be defined elsewhere in the program as a TAG) which must follow the rules for formation of labels as described in Section 2. Columns 18-27 and 39-59 must be blank.

**TAG** This operation is used to identify the point where a GOTO operation will branch. FACTOR 1 must contain a unique label (TAG) which must follow the rules for formation of labels as described in Section 2.

Conditioning by a control level indicator in columns 7-8 is permissible if the TAG is part of TOTAL CALCULATIONS. The INDICATOR field (columns 9-17) must be left blank. Columns 33-59 and 24-27 must be blank.

#### TRANSFER CONTROL FUNCTION

This operation causes the MCP to execute a control instruction within the operating RPG object program. The MCP control instructions which may be specified are system dependent. Refer to the B 1700 System Software Operational Guide, Form No. 1068731.

**ZIP** The operation code ZIP causes the MCP to execute the control instruction contained in FACTOR 2. FACTOR 2 may be a field name or vector name, previously defined, or a meaningful alphanumeric literal enclosed in apostrophes. The operation may be conditioned by the conditioning indicator. All other fields must be left blank. The information contained in FACTOR 2 must be a valid MCP Control Statement. The contents of FACTOR 2 must not exceed a maximum of 511 alpha characters.

ZIP may be used for programmatic scheduling of object programs contained in the Disk Directory, or ZIP may be used to accomplish any of the MCP control functions performed through the console printer (SPO) or card reader.

In the example shown in figure 9-12, the field DATA contains the alphanumeric information, "EX JOB10". EX JOB10 is a control instruction. When the priority for JOB10 is recognized once memory space becomes available, the MCP will retrieve JOB10 from the Disk Directory and place it in the MIX for subsequent operation.

The program containing the ZIP operation will proceed to the next sequential instruction following the ZIP operation, without waiting for the execution of the program JOB10.

The example of a literal in FACTOR 2, shown in figure 9-13, will cause the same action.

#### LOOKUP OPERATIONS

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

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

LINE	INDICATORS																	FACTOR 1	OPERATION	FACTOR 2	RES
	AND					AND					NOT										
	3	5	6	7	8	9	10	11	12	13	14	15	16	17	18	27	28				
01	C					02	03	04										ZIP	DATA		
02	C																				

Figure 9-12. The ZIP Operation Using a Field Name

LINE	INDICATORS																	FACTOR 1	OPERATION	FACTOR 2	RES
	AND					AND					NOT										
	3	5	6	7	8	9	10	11	12	13	14	15	16	17	18	27	28				
01	C					02	03	04										ZIP	'EX JOB10'		
02	C																				

014064

Figure 9-13. The ZIP Operation Using a Literal

The search word may be an alphanumeric or numeric literal, a field name, an array element, or a table name. If it is a table name, it refers to the element of the table last selected in a LOKUP operation, and not to the whole table.

The word must be of the same length and type as each element of the vector being searched. No decimal alignment is performed.

The RESULT FIELD is used only if two vectors are to be specified in the LOKUP operation.

RESULTING INDICATORS must always be assigned to a LOKUP operation specification, and are used to indicate the type of vector search desired, as follows:

- a. An indicator assigned to EQUAL (columns 58-59) specifies that the vector is to be searched starting from the first entry for an item equal to the search word.

- b. An indicator assigned to HIGH (columns 54-55) specifies that the vector is to be searched for the item that is nearest to but higher in sequence than the search word. The first such entry found causes the indicator assigned to HIGH to turn ON.
- c. An indicator assigned to LOW (columns 56-57) specifies that the vector is to be searched for the item that is nearest to but lower in sequence than the search word. The first such entry found causes the indicator assigned to LOW to turn ON.

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

- a. At least one RESULTING INDICATOR must be assigned, but more than two are not allowed.
- b. If two RESULTING INDICATORS are assigned, one of them must be assigned to EQUAL. The program then searches for an item that satisfies either condition with EQUAL given precedence; that is, if no EQUAL entry is found, then the nearest low or higher entry satisfies the search.
- c. Any search operation for other than an EQUAL condition (LOW, HIGH, LOW and EQUAL, HIGH and EQUAL) is allowed only if the vector was specified as having ascending or descending sequence on the Extension Specifications. The search algorithm assumes that the vector is in the specified sequence; a vector out of sequence will cause undefined results.
- d. If the search is successful, the RESULTING INDICATOR designating the type of search is turned ON. If two indicators are assigned, only the one indicating the comparison of the search word and the vector element will be turned ON.
- e. If no element of the vector being searched satisfies the conditions, no RESULTING INDICATOR will be turned ON.

A LOKUP operation may begin searching a vector at some element other than the first by assigning a numeric literal or data item to the vector as an index. The search will start at the designated element and continue until the desired item is found or the end of the vector is reached. If the search is unsuccessful and the index is a data item rather than literal, the index will be set to one. If the search is successful, the index will contain the number of the element which satisfied the designated conditions. If a literal is used as an index, it will not be changed to indicate the search result. The value of a literal index must be integral, not less than one or greater than the number of elements of the vector. If at run-time, the value of a data item index violates the above rules, the program will be terminated (see figure 9-14).

7	18	27	28	32	33	42	43	48	49	51	52	53	54	55	56	57	58	59	60	Resulting Indicators		
																				High	Low	Equal
FACTOR 1		OPERATION		FACTOR 2		RESULT FIELD				FIELD LENGTH					Arithmetic							
												Plus Minus Zero										
												Compare										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										
												Table(Factor 2) is										
												High Low Equal										
												1 > 2 1 < 2 1 = 2										
												Lookup										

INDICATORS																		FACTOR 1	OPERATION	FACTOR 2	RESULT FIELD	FIELD LENGTH	Resulting Indicators					
AND						AND						Arithmetic																
NOT			NOT			NOT			Plus	Minus	Zero																	
9	10	11	12	13	14	15	16	17	18	27	28	32	33	42	43	48	49						51	52	54	55	56	57
									PAYRATE			LOOKUP	DTABLE													20		
	20								DTABLE			ADD	1.25			DTABLE												
									DTABLE			LOOKUP	CARRAY													30		
DTABLE IS SEARCHED FOR A MATCH. IF THE ELEMENT IS FOUND 1.25 IS ADDED TO IT. THE CONTENTS OF THE HOLD AREA ARE CHANGED AND USED TO SEARCH CARRAY.																												

G14066

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

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

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

Two-Vector LOKUP

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







The EXSR operation may be conditioned by an indicator, allowing the subroutine to be executed (called) only when all the conditions are satisfied. FACTOR 2 must contain the name of the subroutine being called, which must be the same name entered in FACTOR 1 of a BEGSR operation. FACTOR 1 and columns 39-59 must be left blank.

#### PROGRAMMED CONTROL OF INPUT AND OUTPUT

Under the normal B 1700 RPG program cycle, a record is read, calculations are performed (using the data from that input record), and an output record is written. The CHAIN, DSPLY, EXCPT, FORCE, and READ operations allow greater control over input and output, providing the capability to read and write records at times other than those normally available as part of the B 1700 RPG program cycle.

**CHAIN** This operation positions the actual key for a CHAINED disk file during calculations so that a record will be read from or written into the proper location. It is used for random processing of a direct or indexed file.

The RESULT FIELD, FIELD LENGTH, DECIMAL POSITIONS, and HALF ADJUST fields must be left blank. Indicators in columns 7-17 may be assigned. If the chained file is conditioned by an external indicator in the File Description Specifications, the CHAIN operation should be conditioned by the same indicator, otherwise the object program must automatically suppress the CHAIN operation if that external indicator is not set ON. FACTOR 1 must not be a whole array.

The HIGH subfield (columns 54-55) of the RESULTING INDICATORS field may contain an entry. If the designated record is not found on a read, the specified indicator will turn ON. If no indicator is specified and the record is not found, then the program will be terminated. The LOW and EQUAL subfields (columns 56-59) should be left blank. However, the LOW subfield may contain an entry; if so, it must be the same as the entry in the HIGH subfield.

If the filename in FACTOR 2 has packed keys, FACTOR 1 must be numeric. FACTOR 1 of the CHAIN operation must be the same length when packed as the record key in the file named in FACTOR 2.

If column 31 on the File Description Specifications contains either an A or K and FACTOR 1 is a numeric identifier, the key comparison will be done 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 will be numeric.

### Direct Files

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

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

### Indexed-Sequential Files

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

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

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

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

#### Chained Input Files

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

#### Chained Output Files

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

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

Direct output files are handled in the same manner as direct input files, except that no record is read or written. Only the disk address to be used for output is generated. If the designated key points to a record outside the limits of the file, the indicator specified in columns 54-55 will turn ON, and the output operation is inhibited.

#### Chained Update Files

For chained files declared as UPDATE, the CHAIN operation causes a record to be read during calculations, and saves the disk address from where the record was read. This address is used during output, causing the updated record to be written back into the same location from which it was read.

Direct update and indexed update files are handled in the same manner as their respective input files. The disk address is saved for use by the output operation, to cause the record to be written back into the same location in the file.

#### DSPLY

This operation causes data to be displayed on the console printer (SPO) or provides for certain low volume data entries to be entered by the systems operator through the console printer. The file name of the file assigned to the console printer must be entered in FACTOR 2. The HALF ADJUST and RESULTING INDICATORS fields must be left blank. Control level and conditioning indicators may be assigned. FACTOR 1 is optional and may be used to name a data item, which may be a field name or a vector element, or to specify a literal (numeric or alphanumeric). The RESULT FIELD is optional and may be used to specify the name of a data item (field name or vector element). If the RESULT FIELD is used, the systems operator must respond on the console printer. If both FACTOR 1 and the RESULT FIELD are blank, a syntax error is emitted.

The DSPLY operates in the following manner:

- a. If the RESULT FIELD is blank and FACTOR 1 contains a data item, the data item is printed on the console printer and the program proceeds to the next operation. (See figure 9-18, line 2.)
- b. If the RESULT FIELD contains a data item and FACTOR 1 is blank or contains a data item, the data is printed on the console printer and an ACCEPT message will be generated by the MCP, to which the systems operator must respond. The contents of the response will be placed in the field specified in the RESULT FIELD. For the format of the ACCEPT message, refer to the B 1700 System Software Operational Guide, Form No. 1068731. (See figure 9-18, line 5.)



The FORCE operation may apply only to input, update, or combined files designated as PRIMARY or SECONDARY. 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.

If more than one FORCE operation is executed during one program cycle, all but the last are ignored. When End-of-File is encountered on a "forced" file, normal record selection processes will determine the next record to be provided for input.

#### READ

This operation is used to cause a record to be read from a demand file during calculations. This differs from the FORCE operation because FORCE causes input during the next program cycle, whereas the READ operation 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 is used to read records sequentially.

FACTOR 2 must contain the name of the file to be read. The FACTOR 1, RESULT FIELD, FIELD LENGTH, DECIMAL POSITIONS, and HALF ADJUST fields must be left blank. The READ operation may be conditioned by CONTROL LEVEL or conditioning INDICATORS in columns 7-17.

An indicator should be specified in columns 58-59; if not, an End-of-File condition on the READ will cause the program to be discontinued. If End-of-File is reached and an indicator is specified, that indicator will be turned ON. Any subsequent attempts to read from the same file will cause that indicator to be turned ON.

The following rules must be observed when using the READ operation:

- a. Only demand files designated as INPUT, UPDATE, or COMBINED may be read by the READ operation.
- b. Sequence-checking in the Input Specifications is not allowed for demand files.
- c. Control levels, matching fields, and look-ahead fields are not allowed for demand files.
- d. If a demand file is conditioned by an external indicator (U1-U8) which is not set, the READ operation will be ignored (the End-of-File indicator in columns 58-59 will not turn ON).

## DEBUG OPERATION

The DEBUG operation is a special-purpose function which simplifies the location and correction of errors in an RPG Program.

DEBUG This operation causes records to be written during calculations. These records contain specific information which may be helpful in locating errors in the program. DEBUG operations may appear in as many places as needed within the Calculation Specifications. Every time the DEBUG operation is executed, one or more fixed-format records will be written to an output device. One record contains a list of all indicators which are ON at the time the DEBUG operation is specified; the other record(s) shows the contents of any one field.

DEBUG operations will be compiled into the object program only if the DEBUG field of the Control Card contains a 1 (column 15). All DEBUG operations are syntax checked during compilation.

The file named must be defined as a sequential output file on the File Specifications. All DEBUG output must go to the same file. FACTOR 2 must contain the name of the output file on which the records are written.

FACTOR 1 is optional, and may contain a literal or field name to identify the particular DEBUG operation being executed.

FACTOR 1 must not be a whole array, and the size of the field must not exceed eight positions. The literal or the value of the designated field is written as part of the output records.

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

### DEBUG Operation Output

One or more records will be written as output from every DEBUG operation. The first record is always written; the second and subsequent records will be written only if the RESULT FIELD contains an entry. See figure 9-19 for an example of the output produced by the DEBUG operation.





The first record written is in the following format:

Record Position	Entry
2-7	DEBUG
9-17	Blank
18-31	The words INDICATORS ON-
32-any position depending on the number of indicators ON	List of indicators that are ON, separated by blanks.

## OUTPUT-FORMAT SPECIFICATIONS

Output-Format Specifications describe the records within each file and fields within each record to be used as output from the program. There are two types of Output-Format Specifications as follows:

- a. Record type descriptions (columns 7-31) which define the various output records and their relationship to other records in the file. Columns 32-70 must be blank.
- b. Field descriptions (columns 23-70) which define each field within the records. Columns 7-22 must be blank.

The first Output Specification must be a record type description. Field description entries must start one line below the associated record type descriptions. A warning is emitted if a record type description has no associated field descriptions.

FIELD DEFINITIONS

Refer to figure 10-1 in conjunction with the following field definitions for the Output-Format Specifications.

1-2 PAGE

Refer to Section 2 for complete description.

3-5 LINE

Refer to Section 2 for complete description.

6 FORM TYPE

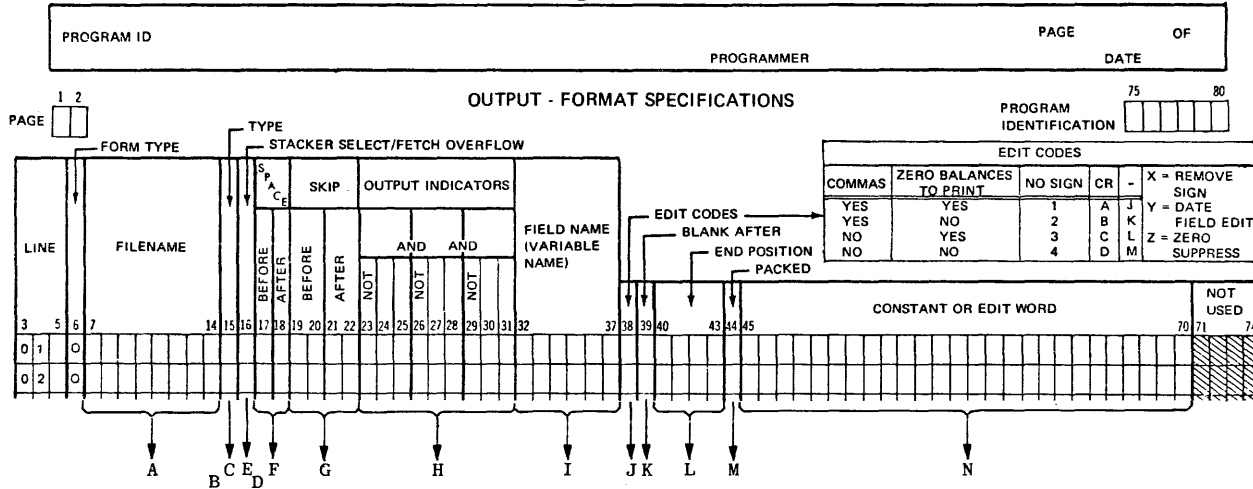
An 0 must appear in this field.

7-14 FILENAME

This field is used to identify the file to which the subsequent record type and field description entries belong. The file specified must have been previously described on the File Description Specifications form as output, update, combined, or input, with an A in column 66 on the File Description Specifications and ADD specified in columns 16-18 of the Output-Format Specifications. Every output file described 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.

Burroughs B 1700 RPG



- A. 7-14 Contains a filename specified in the File Description Specifications.
- B. 14-16 Puts the output indicators in an AND or OR relationship. Entries: AND or OR.
- C. 15 Specifies the type of output record to be written. Entries: H, D, T or E.
- D. 16-18 Specifies if a record is to be added to an indexed sequential file. Entries: Blank or ADD.
- E. 16 Specifies (1) which stacker the output card is to be placed or (2) that the overflow routine is to be invoked. Entries: Blank, F, or 1-N (N=number of stackers).
- F. 17-18 Specifies forms spacing, before or after printing, for printer output files. Entries: 0 or blank, or 1-9.
- G. 19-22 Specifies forms skipping, before or after printing, for printer output files. Entries: 0-99, A0-A9, B0-B2, or blank.
- H. 23-31 Output Indicators:  
 23, 26, 29 Indicates if the output indicator in columns 24-25, 27-28, or 30-31, must be ON or OFF. Entries: Blank or N.  
 24-25, 27-28, 30-31 Contains a previously defined indicator which is to condition output. Entries: 01-99, L0-L9, LR, MR, H0-H9, U1-U8, 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).

614071

Figure 10-1. Output-Format Specifications Summary Sheet

## TYPE

This field is used to specify the type of output record to be written. These records contain such information as the following:

- a. Heading records contain such information as page headings.
- b. Detail records usually contain some type of data obtained directly from input records and calculation operations. Detail records are written once during every cycle, depending upon conditioning indicators.
- c. Total records usually contain totals accumulated from a group of input records. Total records are written only during cycles in which a control break occurs, depending upon conditioning indicators.
- d. Exception records are written during calculation time, through use of the EXCPT operation code.

Valid entries for this field are:

Entry	Definition
H	Heading records.
D	Detail records.
T	Total records.
E	Exception records (written during calculation time).

Heading records are treated in the same manner as detail records. The order of the output will correspond to the order in which they are specified. Records may be specified in any order.

Update files are handled as field updates, not record updates; only the fields specified in an output record are modified. The rest of the record remains intact.

If an update file has total output specified, the record update will be the last record read (i.e., the record that caused the Control Break). Exception records may be specified for a combined file, but this is not recommended.

## 16-18 RECORD ADDITION

If a record is to be added to an input, output, or update file, the following conditions apply:

- a. The file must be a disk file.
- b. The corresponding File Description Specification must have an A in column 66.
- c. The word ADD must be entered in columns 16-18 of the record type description.
- d. ADD must not be specified on an AND/OR line.

**STACKER SELECT/FETCH OVERFLOW**

This field may be used to specify:

- a. The stacker into which the output card is to be placed after being punched, or
- b. That the overflow routine can be invoked at this point for a printer file.

Valid entries for this field are:

Entry	Definition
Blank	Cards automatically go to default stacker.
Numeric Entry	Stacker into which card type is stacked.
F	Fetch overflow (printer files only).

**STACKER SELECT**

Only card output or combined files may be stacker selected. Output files may be stacker selected only in the Output-Format Specifications; combined files may be stacker selected either in the Input or Output-Format Specifications. If a combined file is stacker selected in both the Input and Output-Format Specifications, the Output-Format stacker specification overrides the stacker specified in the Input Specifications.

Stacker selection on the basis of matching records should be specified only for detail output lines, because the MR indicator is on only at detail output time to signal the matching status. Record types identified by OR lines may be stacker selected for a special stacker by an entry in this field; however, if the STACKER SELECT field entry is left blank, the card type selected by the OR line will go to the default stacker. AND lines may not have an entry in STACKER SELECT.

At execution time, any record types specifying a stacker number higher than that available on the device being used will go to the default stacker.

**FETCH OVERFLOW**

If the printing of a line could cause overflow, leaving insufficient space on the page to print the remaining detail, total output lines, or lines conditioned by the overflow indicator, FETCH OVERFLOW should be specified (F in column 16).

When the overflow line is reached, the same sequence of events always takes place. These are described later in this section in the discussion on overflow indicators. Briefly, remaining detail lines, total lines, and overflow lines (lines conditioned by the overflow indicator) are printed on the page after overflow has occurred.

If, however, it is desired to print overflow lines ahead of the usual time, a FETCH OVERFLOW routine may be specified. This may be initiated any time after the overflow line has been reached. When overflow is caused in this manner, the following actions take place:

- a. All total lines conditioned by the overflow indicator are printed. If skipping is specified on these lines, it occurs as defined by the user.
- b. Heading and detail output lines conditioned by the overflow indicator are printed. If skipping is specified on these lines, it occurs as defined by the user.
- c. The line that fetched the overflow routine is printed.
- d. Any detail and/or total lines left to be printed in the current program cycle are printed.

For the printer file, an F in column 16 on the Output Format Specifications specifies that the overflow routine will be fetched. An F can be specified for any total, detail, or exception line that is not conditioned by an overflow indicator.

If a line causes the overflow indicator to turn ON, the next line containing an F in column 16 will cause the execution of the overflow routine. When that is complete, normal printing will resume with the statement that fetched the overflow routine (see figure 10-2).

LINE	FILENAME	SPACE		SKIP		OUTPUT INDICATORS										
		BEFORE	AFTER	BEFORE	AFTER	AND		AND								
						NOT	NOT	NOT	NOT							
3	5 6 7	14	15 16	17	18	19 20	21 22	23	24	25	26	27	28	29	30	31 3
01	O PRINTOUTH			2	0				F							
02	O															
03	O			DF	2				1	0						
04	O															
05	O			T	1				L	1						
06	O															
07	O			TF	1				L	1						
08	O															
09	O			T	1					F						
10	O															
11	O			T	1					F						

014072

Figure 10-2. FETCH OVERFLOW Coding Example







A skip entry must not be greater than the form length as specified in the Line Counter Specifications or as defaulted.

If both skipping and spacing are specified on the same line, the operations are performed in the following order:

- a. SKIP BEFORE.
- b. SPACE BEFORE.
- c. SKIP AFTER.
- d. SPACE AFTER.

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

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

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

Different SPACE and SKIP entries may be specified for OR lines. If all these entries are blank for an OR line, spacing and skipping are done according to the specifications on the preceding line. If any entries are made for an OR line, all desired spacing and skipping must be specified. SPACE and SKIP entries are not permitted on AND lines. Spacing and skipping must not be specified for other than printer files.

#### 19-22 SKIP (RPG 1 DIALECT)

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

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

## OUTPUT INDICATORS

This field is divided into three subfields such that up to three indicators on each line may be specified to condition an output operation. Each subfield is divided into two parts as follows:

- a. NOT (one column).
- b. INDICATOR (two columns).

The NOT portion is used to specify that the associated indicator must be OFF in order for the operation to occur. If this condition is desired, an N must be entered in the NOT portion. Otherwise, the NOT portion must be left blank.

The INDICATOR portion is used to specify the indicator to be tested for ON (NOT = blank) or OFF (NOT = N). The following entries are allowed in this portion of the INDICATORS field:

Entry	Definition
Blank	Operation not conditioned by an indicator.
01-99	Operation conditioned by indicator used elsewhere in the program.
L0-L9	Operation conditioned by control level indicator previously assigned.
LR	Operation conditioned by last record indicator.
MR	Operation conditioned by matching record indicator.
H0-H9	Operation conditioned by halt indicator used elsewhere in the program.
U1-U8	Operation conditioned by external indicator previously set.
OA-OG, OV	Operation conditioned by overflow indicator previously set.
1P	Operation conditioned by first page indicator.

All three indicators on one line are in an AND relationship. All indicators on one line (or grouped lines) must be ON or OFF as specified in order for the associated operation to take place.

An indicator specified on the line describing the record type will condition the entire output record. An indicator used to condition a field within the record is placed on the same line as the field description (see figure 10-5).



- LR The last record indicator is used to condition output operations that are to be performed at End-of-Job.
- MR The matching records indicator is used to condition output operations that are to be performed only when matching input records are found.
- HO-H9 Halt indicators previously assigned in the FIELD INDICATORS field (Input Specifications) or the RESULTING INDICATORS field (calculation Specifications) may be used to condition output operations that are to be performed only when an error condition occurs.

Since the program does not halt until after the record in error has been completely processed, some operations must be prevented in order to avoid erroneous output. By using halt indicators in conjunction with an N in the NOT portion of the INDICATORS field, an operation may be inhibited when the specified halt indicator is ON.

- U1-U8 If an output file is specified as conditioned by an external indicator in the File Description Specifications (EXTERNAL INDICATORS field), every output record described for the file should be conditioned by the same external indicator, otherwise the object program will still build the output record (and perform any blank after operations specified) but must suppress the write operation.

- OA-OG,OV Overflow indicators previously assigned in the OVERFLOW INDICATOR field (File Description Specifications) may be used to condition output operations that are to be performed when the overflow line on a printer file has been reached.

Overflow indicators which have not been previously assigned in the File Description Specifications (except OF and OV, which are compiler defined) may not be used in the Output Format Specifications. Forms advancing at End-of-Page are handled automatically if no overflow indicators are assigned to the file. Any specification line not conditioned by an overflow indicator which designates a skip to the next page turns OFF all overflow indicators before the skip takes place.

Overflow indicators must not be used to condition exception records but may condition fields within exception records.

No more than one overflow indicator may be associated with a group of output indicators in an AND or OR relationship, and it must be the same indicator assigned to the file in the File Description Specifications.

The overflow line can be sensed when printing total or detail time output. If an overflow indicator is being used to condition output lines, the following steps occur when the overflow line is sensed during total time output:

- a. The overflow indicator turns on.

- b. The remaining total lines not conditioned by overflow are printed.
- c. All total lines conditioned by overflow are printed. If skipping is specified on these lines, it occurs as defined by the user.
- d. Heading and detail output lines conditioned by overflow are printed. If skipping is specified on these lines, it occurs as defined by the user.
- e. Heading and detail lines not conditioned by overflow are printed.
- f. The overflow indicator turns off.

If the overflow line is sensed during detail time output, the following steps occur:

- a. The overflow indicator turns on.
- b. The remaining detail lines not conditioned by overflow are printed.
- c. All total lines not conditioned by overflow are printed.
- d. All total lines conditioned by overflow are printed. If skipping is specified on these lines, it occurs as defined by the user.
- e. Heading and detail output lines conditioned by overflow are printed. If skipping is specified on these lines, it occurs as defined by the user.
- f. Heading and detail lines not conditioned by overflow are printed.
- g. The overflow indicator turns off.

When using the overflow indicator to condition overflow printing, remember:

- a. Overflow indicators may be turned on and off by the operation codes SETON and SETOF.
- b. Spacing past the overflow line causes the overflow indicator to turn on.
- c. Spacing or skipping past the overflow line to any line on the new page does not turn the overflow indicator on.
- d. A skip to a new page specified on a line not conditioned by an overflow indicator causes the overflow indicator to turn off.

### Control Level Indicators With Overflow Indicators

If it is desired to have headings identifying the type of information on each page or each page to contain information from only one control group, control level indicators may be used in conjunction with overflow indicators. Together they condition when headings and/or group information are to be printed.

In the example in figure 10-6, the control level indicator L1 is used in conjunction with the overflow indicator in order to print headings on every page. Line 01 allows the headings to be printed at the top of a new page only when overflow occurs. Line 02 allows printing of headings on a new page only at the beginning of a new control group (L1). In this way, duplicate headings caused by both L1 and OF being ON at the same time will not occur.

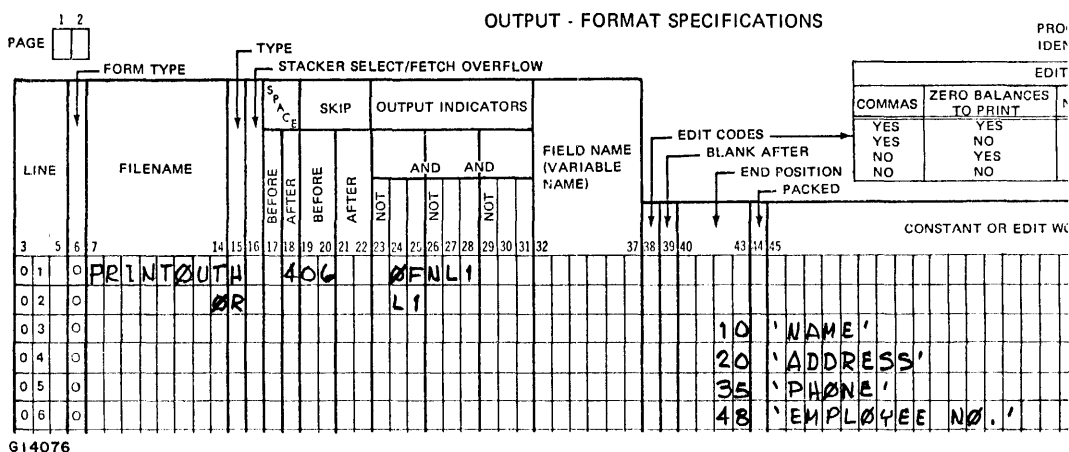


Figure 10-6. Using Control Level Indicators With Overflow Indicators

Figure 10-7 shows the necessary coding for the printing of certain fields on every page.

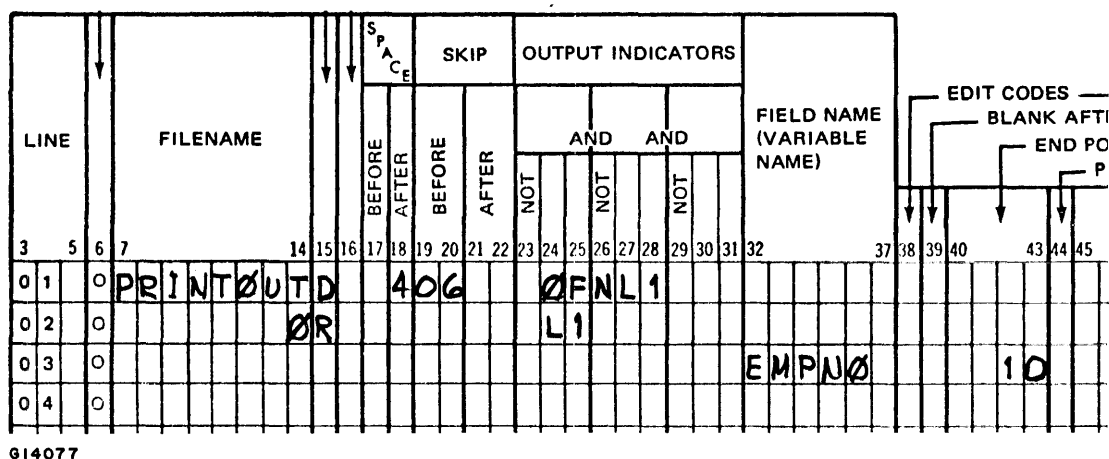


Figure 10-7. Coding for the Printing of Certain Fields

1P

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

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

OUTPUT - FORMAT SPECIFICATIONS

PAGE		FORM TYPE		TYPE		STACKER SELECT/FETCH OVERFLOW		SKIP		OUTPUT INDICATORS			FIELD NAME (VARIABLE NAME)		CONSTANT OR EDIT WORD		NOT USED	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
01	O	REPORT1	H															
02	O																	
03	O	REPORT2	H															
04	C		OR															
05	O																	
06	*	* LINES 01 AND 02 DEFINE A HEADING LINE THAT IS TO BE PRINTED ONLY ONCE FOR THE REPORT1 FILE, ON THE FIRST PAGE.																
07	*	* 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 FILE).																
08																		
09																		
10																		
11																		
12																		
13																		
14																		
15																		

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

G14078

Figure 10-8. 1P Indicator Coding Example

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

AND/OR

If it is necessary to specify more than three indicators to condition an output operation, an AND line may be used. The word AND must be entered in columns 14-16, and the additional indicators entered in their respective fields. The conditions specified for all indicators in an AND relationship must be met before the associated output operation will take place.

OR lines (OR in columns 14-15) may be used to group indicators such that only one of the conditions specified must be met for the associated output operation to take place. Both AND and OR lines may be used together to condition an output record (but not a field). A maximum of three indi-



cators in an AND relationship (on one line) may be used to condition a field. See figure 10-5 for an example of the usage of AND and OR lines.

There is no limit on the number of AND or OR lines that may be specified. However, it is recommended that the user not exceed 20 if compatibility with other Burroughs systems is desired. There must be at least one indicator on an AND/OR line and on the preceding line.

### 32-37 FIELD NAME (VARIABLE NAME)

This field is used to assign an identifier (name) to an output data field. The identifier used must have previously been defined in the Input Specifications (VARIABLE NAME field), the Extension Specifications (VECTOR NAME field), or the Calculation Specifications (RESULT FIELD field). Also, any special words may be used. A separate line must be used for each field description. Fields may be listed in any order within each record type, since their location is determined by the entry in the END POSITION fields. If fields overlap, only the last field specified appears intact in the output record (the exact results will depend upon the degree of overlap).

#### Special Words

The following special words are reserved for use as variable names:

- PAGE
- PAGE1
- PAGE2
- UPDATE
- UMONTH
- UDAY
- UYEAR
- \*PLACE
- \*PRINT

Each special word has a specifically defined usage, as described in the following paragraphs:

#### Page Fields (PAGE, PAGE1, and PAGE2)

If page numbering is to be done on output, the special word PAGE (or PAGE1, PAGE2 for two more files) is used to indicate that page numbering is to be done automatically. When a PAGE field is named in this field without being previously defined in the Input or Calculation Specifications, it is assumed to be four characters in length with no decimal positions. On output, leading zeros are suppressed and the sign is not printed unless an edit word or edit code is specified.

The page number begins at zero (unless otherwise specified), and is automatically incremented by one each time before the page field is written.

The page number may be reset at any point during the program by setting the PAGE field to zero before it is printed (see figure 10-9). This may be accomplished in two ways:



**\*PLACE Specification**

The special word \*PLACE allows writing of the same field or fields in more than one place in an output record without having to specify the field names and end positions more than once. The designated fields are written in the same relative positions ending in the position specified for the \*PLACE entry. It is possible to obtain the same results in two ways (see figure 10-10).

- a. Define each field and its corresponding end position for every time it is to appear in the output records, or
- b. Use the special word \*PLACE.

PAGE		FORM TYPE		TYPE		STACKER SELECT/FETCH OVERFLOW		OUTPUT INDICATORS		FIELD NAME (VARIABLE NAME)		EDIT CODES		CONSTANT OR EDIT WORD		NOT USED							
LINE	FILENAME	BEFORE	AFTER	BEFORE	AFTER	AND	AND	AND	AND	AND	AND	BLANK AFTER	END POSITION	PACKED	COMMAS	ZERO BALANCES TO PRINT	NO SIGN	CR	-	X	Y	Z	
01	OUTPUT	D	1																				
02										FIELD1	5												
03										FIELD2	15												
04										FIELD3	25												
05										FIELD1	35												
06										FIELD2	45												
07										FIELD3	55												
08																							
09	* THE ABOVE OUTPUT DEFINITION CAN ALSO BE DEFINED USING THE																						
10	* *PLACE SPECIFICATION, AS SHOWN BELOW:																						
11																							
12	OUTPUT	D	1																				
13										FIELD1	5												
14										FIELD2	15												
15										FIELD3	25												
16										*PLACE	55												
17																							
18	THE *PLACE NAME DUPLICATES ALL FIELDS PREVIOUSLY DEFINED.																						

Figure 10-10. \*PLACE Specification Coding Example

Both methods will produce identical results, but use of the \*PLACE entry saves extra coding.

The following rules must be observed when using the \*PLACE specification:

- a. All fields within the record type written above the \*PLACE entry are repeated according to the \*PLACE specification, not just the one line above.
- b. An end position must be given for every \*PLACE specification.

- c. An additional \*PLACE entry (on a separate line) must be used every time the fields are to be repeated.
- d. \*PLACE must be specified after the field names which are to be placed in different positions on the line; \*PLACE must not be specified on the first field description line for a record.
- e. The end position specified for \*PLACE should be at least twice the highest previously specified end position. If enough space is not allowed for all fields to be printed again, overlapping will occur, with the \*PLACE output overlapping the previous characters.
- f. The end position specified for \*PLACE must not be lower than the highest previously specified field end position.
- g. The leftmost position of the fields to be moved by the \*PLACE specification is always assumed to be position 1.
- h. When \*PLACE is specified for card output, the fields and constants named above will be repunched. Any printed output on the cards will not be reprinted unless an \* is entered in column 40 of the same line as \*PLACE.
- i. Only the conditioning indicators (columns 23-31), FIELD NAME (columns 32-37) and END POSITION (columns 40-43) may have entries.

\*PRINT Specification

The special word \*PRINT is used to cause card interpreting after punching (for card files only). Printing is done at the top of the cards in the same column positions as the fields are punched. The \*PRINT specification must be used only once for each record and appears after all fields on the card which are to be printed. The \*PRINT specification may be conditioned by indicators in the OUTPUT INDICATORS field; all other fields must be left blank (see figure 10-11).

Having the \*PRINT specification print the corresponding field exactly as punched is not always desirable.

To print the fields in positions other than would be assigned by the \*PRINT specifications:

- a. Enter the field name to be printed in the VARIABLE NAME field, and
- b. Enter an asterisk in column 40, and
- c. Enter the end position for the field in columns 41-43 (limited to a maximum of 128), right-justified; leading zeros not required.

OUTPUT - FORMAT SPECIFICATIONS

PAGE 1 2

PROGRAM IDENTIFICATION 75 80

LINE	FILENAME	TYPE		STACKER SELECT/FETCH OVERFLOW		SKIP						OUTPUT INDICATORS						FIELD NAME (VARIABLE NAME)	EDIT CODES	CONSTANT OR EDIT WORD	NOT USED
		BEFORE	AFTER	BEFORE	AFTER	BEFORE	AFTER	BEFORE	AFTER	BEFORE	AFTER	BEFORE	AFTER	BEFORE	AFTER	BEFORE	AFTER				
01	PUNCHOUTD																				
02																		DATA1	10		
03																		FIELD1	20		
04																		FIELD2	34		
05																		*PRINT			
06																		DATA2	40		
07																		FIELD3	44		
08	*																				
09	*	THE *PRINT ENTRY CAUSES THE THREE FIELDS ABOVE IT TO BE																			
10	*	PRINTED (AS WELL AS PUNCHED). THE OTHER TWO FIELDS ARE																			
11	*	ONLY PUNCHED.																			
12																					
13																					
14																					
15																					

G14081

Figure 10-11. \*PRINT Specification Coding Example

38 EDIT CODES

This field is used to specify editing of a numeric output field, otherwise it must be blank. Certain 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 table on the Output-Format Specifications sheet. Also see table 10-1.

Only numeric fields may be edited, and if an edit code is specified, the CONSTANT OR EDIT WORD field must be left blank. However, if check protect or floating dollar sign is required, enter '\*' or '\$', respectively, in columns 45-47.

Check-protect and floating dollar sign must not be used with X, Y, or Z edit codes.

If an edit code is used to edit a whole array, two spaces are automatically inserted to the left of each element. If a whole array is edited with an edit word, spaces are not automatically inserted; they should be included in the edit word if required.

The ending position specified must allow sufficient space for the field after editing (special card needed for a whole array).

Table 10-1. Edit Codes Table

Edit Code	Commas	Decimal Point	Zero Suppress	Sign For Negative Balance	Printout on Zero Balance*		
					Inter-national I	Inter-national J	Domestic United Kingdom
1	Yes	Yes	Yes	No Sign	,00 or 0	0,00 or 0	.00 or 0
2	Yes	Yes	Yes	No Sign	Blanks	Blanks	Blanks
3		Yes	Yes	No Sign	,00 or 0	0,00 or 0	.00 or 0
4		Yes	Yes	No Sign	Blanks	Blanks	Blanks
A	Yes	Yes	Yes	CR	,00 or 0	0,00 or 0	.00 or 0
B	Yes	Yes	Yes	CR	Blanks	Blanks	Blanks
C		Yes	Yes	CR	,00 or 0	0,00 or 0	.00 or 0
D		Yes	Yes	CR	Blanks	Blanks	Blanks
J	Yes	Yes	Yes	-	,00 or 0	0,00 or 0	.00 or 0
K	Yes	Yes	Yes	-	Blanks	Blanks	Blanks
L		Yes	Yes	-	,00 or 0	0,00 or 0	.00 or 0
M		Yes	Yes	-	Blanks	Blanks	Blanks
X**							
Y***			Yes				
Z			Yes				

NOTES

\*Zero balances for the International format are printed or punched in two ways, depending on the entry made in column 21 of the Control Card Specifications.

\*\*The X code removes the sign.

\*\*\*The Y code suppresses the leftmost zero only. The Y code edits a three to six digit field according to the following pattern:

nn/n  
 nn/nn  
 nn/nn/n  
 nn/nn/nn

39 BLANK AFTER

This field is used to specify that a variable is to be reset after the output operation is finished. Alphanumeric fields will be cleared to blanks and numeric fields will be cleared to zeros. Constants and UDATE, UDAY, UMONTH, and UYEAR must not be specified with BLANK AFTER. Valid entries for this field are:

Entry	Definition
Blank	Field is not to be cleared.
B	Variable is to be cleared after being moved to the output buffer.

Under the default RPG II dialect, BLANK AFTER has no effect on zero/blank indicators. However, if RPG I dialect is specified on the Control Card Specifications and if an indicator is assigned to a field to test for ZERO or BLANK in the Input or Calculation Specifications and the same field is used on Output Specifications with BLANK AFTER, then that zero/blank indicator will be turned ON when the field is blanked out. If more than one indicator is assigned to the same field as 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 vectors or vector elements are output with BLANK AFTER, zero/blank indicators are not changed.

40-43

#### END POSITION

This field is used to specify the location of a field within an output record. Only the location of the rightmost position is specified; the number of positions to the left of that location is determined by the length specified previously for the field. Space must also be left to allow for any editing symbols (edit codes or edit words) that may be entered. The end position must not exceed the record length specified for the file, except in the special case where an \* is entered in column 40.

Table 10-2 illustrates the effects of the various edit codes on output for a negative number with two decimal positions. The END position is specified as 12.

If an asterisk (\*) is placed in column 40, the field will be printed in the upper portion of the card in the position specified.

44

#### PACKED

This field is used to specify that an output field will be written in packed decimal format. Packed decimal fields cannot be printed. 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.

This field must be blank if an asterisk appears in column 40 of the same specification line or for fields that precede \*PRINT for a card file or \*PLACE for a printer file. It must also be blank if an edit code, edit word or constant is specified, or if the field is alphanumeric.

Table 10-2. Effect of Edit Codes on End Position

Edit Codes	Output Print Positions								
	5	6	7	8	9	10	11	12	13
Unedited				-*	0	3	1	2	
1					3	.	1	2	
2					3	.	1	2	
3					3	.	1	2	
4					3	.	1	2	
A			3	.	1	2	C	R	
B			3	.	1	2	C	R	
C			3	.	1	2	C	R	
D			3	.	1	2	C	R	
J				3	.	1	2	-	
K				3	.	1	2	-	
L				3	.	1	2	-	
M				3	.	1	2	-	
X				0	0	3	1	2	
Y			0	/	3	1	/	2	
Z						3	1	2	

\* - Represents a negative 0.

45-70 CONSTANT OR EDIT WORD

Constants or Edit Words are entered in this field. A description of each is provided in the paragraphs that follow.

Constants

Constants are literals usually used for such things as page headings, and contain information that is not changed by an operation. Constants will appear exactly as written in the output record, in the position specified. The following rules must be observed when forming constants:

- a. The VARIABLE NAME entry must be left blank.
- b. Constants must be left-justified, enclosed in apostrophes, and no more than 24 characters in length.
- c. If an apostrophe is to appear in the constant, two consecutive apostrophes must be coded for each apostrophe required.
- d. Numeric data may be used as a constant, but must still be enclosed in apostrophes. Note that this is an alphanumeric literal composed of numeric characters.



The example in figure 10-12 uses the Line Printer Spacing Form to illustrate the coding of output constants.

	1									2									3									4									5																						
▲	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9
1																																																											
2																																																											
3																																																											
4																																																											
5																																																											
6																																																											
7																																																											
8																																																											
9																																																											
10																																																											
11																																																											
12																																																											

OUTPUT - FORMAT SPECIFICATIONS

PAGE 1 2 PROGRAM IDENTIFICATION 75 80

LINE	FILENAME	TYPE	SKIP			OUTPUT INDICATORS			FIELD NAME (VARIABLE NAME)	EDIT CODES	CONSTANT OR EDIT WORD	NOT USED
			BEFORE	AFTER	AND	AND	AND	AND				
01	BILLING	H				1P						
02										43 'MONTHLY STATEMENT'		
03		H				1P						
04										27 'ACCOUNT OLD'		
05										54 'PAYMENT NEW'		
06		H				1P						
07										29 'NUMBER BALANCE'		
08										56 'BALANCE'		
09		D				22						
10										ACCNO 15		
11										GLOBAL 29		
12										PAYMNT 44		
13										NEWBALA 57		

Figure 10-12. Coding of Output Constants

Edit Words

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

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

An edit word is divided into three sections:

- a. The body.
- b. The status section.
- c. The extension section.

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

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

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

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

#### Special Characters Used In Edit Words

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

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

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

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

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

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

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

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

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

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

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

Example

<u>Source Data</u>	<u>Edit Word</u>	<u>Result</u>
0012345	'0bbbb.bb'	00123.45
000000	'0bbbbbb'	0000000

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

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

Example

<u>Source Data</u>	<u>Edit Word</u>	<u>Result</u>
-724325	'bb,b\$0.bb&-'	\$7,243.25b-

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

Example

<u>Source Data</u>	<u>Edit Word</u>	<u>Result</u>
+007461	'0bbb,bbb'	007,461

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

### No Editing

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

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

SOURCE DATA	CONSTANT OR EDIT WORD		OUTPUT RECORD
	45	70	
+0000123456			0000123456
-0000123455			000012345N
0000000000	\	'	bbbbbbbbbb
+0000123456	\	'	bbbb123456
-0000123456	\	'	bbbb123456

G14083

Figure 10-13. No Editing Coding Examples

### Zero Suppression

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

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

### Dollar Sign

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

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

SOURCE DATA	CONSTANT OR EDIT WORD	OUTPUT RECORD
0000000005	\ , , , 0 . ' /	bbbbbbbbbb.05
00000000	\ , , , 0 - ' /	bbbbbbbbbb0b
-0000000123	\ , , , . - ' /	bbbbbbbbbb1.23-
-0000123456	\ , , , 0 ' /	bbbb123456
+0000123456	\ 0 , , , ' /	b000123456
0000000000	\ \$ , , , 0 &CR GROSS ' /	\$bbbbbbbb00bbbbGROSS
0000000000	\ , , , , -OLD BAL ' /	bbbbbbbbbbbbbbOLD BAL
000000	\ , , 0 . ' /	bbbbbb.00
000000	\ , , . 0 ' /	bbbbbbb0
0000000000	\ , , , , 0 * ' /	bbbbbbbbbb0*00
001234	\ 0 , , , 0 ' /	b,012,034
-000000015	\ , , , . - ' /	bbbbbbbbbb15-
000000005	\ , , , 0 . - ' /	bbbbbbbb0.05b

G14084

Figure 10-14. Zero Suppression Zero Coding Examples

SOURCE DATA	CONSTANT OR EDIT WORD	OUTPUT RECORD
+0000000005	\ , , , , \$0 . - ' /	bbbbbbbbbb\$0.05b
-0012345678	\ , , , , \$0 . CR** ' /	bbb\$123,456.78CR**
0000000000	\ \$ , , , , 0 . ' /	\$bbbbbbbbbb.00
+0000123456	\ \$ , , , , &- NET ' /	\$bbbb123456bbNET
-0000123456	\ \$ , , , , &- NET ' /	\$bbbb123456b-bNET
0000123456	\ \$0 , , , , - NET ' /	\$0000123456bbNET
-0000123456	\ , , , , \$0 &CR ' /	bbbb\$123456bCR
-1234567890	\ , , , , \$0 &CR ' /	\$1234567890bCR
0000000005	\ , , , , \$0 . &NET ' /	bbbbbbbbbb\$0.05bNET
0000000005	\ , , , , \$0 . ' /	bbbbbbbbbb\$.05
-1234567890	\ , , , , \$0 . - ' /	\$12,345,678.90-
-0001234567	\ , , , , \$0 . CR ' /	bbbb\$12,345.67CR
0000001234	\ , , , \$0 , . - SALES ' /	bbbbbb\$,012.34bSALES
00123456	\ , , \$0 . ' /	bbb1\$,234.56

G14085

Figure 10-15. Dollar Sign Coding Examples

### Asterisk Fill

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

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

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

SOURCE DATA	CONSTANT OR EDIT WORD	OUTPUT RECORD
0000123456	\ , , , , * . & - /	*****1,234,56bb
-0000123456	\ * /	*000123456
+1234567890	\ * /	1234567890
-0000123456	\ * /	****123456
0000001234	\ , , * , * /	*****,012*34
012345	\ & , * 0 , /	***120,345

G14086

Figure 10-16. Asterisk Fill Coding Examples

### Negative Signs

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

### Ampersand

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

### Constants in the Edit Word

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

SOURCE DATA	CONSTANT OR EDIT WORD	OUTPUT RECORD
+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	' \$& , , O. CR'	\$b12,345,678.90CR
+0000123456	' , , *. *CR**'	*****1,234,56bbb**
-0000123456	' , , *. *CR**'	*****1,234.56*CR**

G14087

Figure 10-17. Negative Signs Coding Examples

SOURCE DATA	CONSTANT OR EDIT WORD	OUTPUT RECORD
-1234567890	' \$& , , O. &- GROSS'	\$b12,345,678.90b-bGROSS
-1234567890	' \$& , , O. &-NET&PAY'	\$b12,345,678.90b-NET PAY
-0000123456	' \$& , , *. CR'	\$*****1,234.56CR
+000002	' O LBS.& ØZ.TARE&-'	bbb0LBS.b02bbbbbbbb
031274	' & & &LATER'	b3b12b74bLATER

G14088

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

SOURCE DATA	CONSTANT OR EDIT WORD	OUTPUT RECORD
0000123456	' &&PRØFIT'	bbbb123456&&PROFIT
0000123456	' &CR&NET'	bbbb123456bbb&NET
1234567890	' \$& , , O. &-NET&PAY'	\$b12,345,678.90b-NET&PAY

G14089

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

SOURCE DATA	CONSTANT OR EDIT WORD	OUTPUT RECORD
0000123456	' &PROFIT '	bbbb123456bbPROFIT
-0000123456	' , , . &CR NET '	bbbbbb1,234.56bCRbbNET
0000123456	' , , DOLLARS CENTS '	bbbbbb1,234DOLLARS56CENTS
000000	' , DOLLARS CENTS '	bbbbbbbbbbbbbbCENTS
000000	' , 0 DOLLARS CENTS &CR '	bbbb0DOLLARS00bbbbbbbb
-000002	' 0LBS. 0Z. TARE &- '	bbbbLBS.020Z.TAREb-
063369690	' 0 - - '	b63-36-9690
0042	' 0 HRS. MINS. &0 'CLOCK '	b0HRS.42MINS.b0'CLOCK
093074	' - - &LATER '	b9-30-74bLATER
093074	' / / ' '	b9/30/74

G14090

Figure 10-20. Examples of Constants in the Edit Word

75-80 PROGRAM IDENTIFICATION

Refer to Section 2 for a complete description.



## DOLLAR CARD SPECIFICATIONS

Dollar Card Specifications accommodate various extensions to the B 1700 RPG Language, which cannot be handled on the other specification forms. Dollar Cards also allow certain compiler-control options to be set or reset during compilation.

Dollar Cards may appear anywhere within the source deck, as required. The standard Dollar Card Specifications form may be used or options can be coded on the other specification sheets at the points where they will be placed in the source deck.

FIELD DEFINITIONS

Refer to figure 11-1 in conjunction with the following field definitions for the Dollar Card Specifications.

1-2 PAGE

Refer to Section 2 for a complete description.

3-5 LINE

Refer to Section 2 for a complete description.

6 FORM TYPE

This field may be left blank or used to contain the form type of the specification in which the option is to be inserted.

7 \$ OPTION

A \$ sign must appear in this field.

8 NOT

This field is used to specify that the option entered in the KEY WORD field is set ON (NOT = blank) or OFF (NOT = N). Certain options cannot be turned OFF; these are indicated under the KEY WORD entry description. Valid entries for this field are:

Entry	Definition
Blank	Specified option is "set".
N	Specified option is "reset".



## KEY WORD

This field is used to name the option to be set or reset (according to entry in NOT field). The option name must be entered left-justified in the field. Options fall into three categories: file identification extensions, RPG extensions, and compiler-directing options.

The following B 1700 dollar options may or may not be valid on other systems and vice versa. Unrecognized dollar cards will be ignored, but a warning will be emitted.

File Identification Extensions

The following extensions are used to specify the external names for files described in the File Description Specifications. They must immediately precede the file in the File Description Specifications to which they refer. None of the options may be reset.

- \$ PACKID      Specifies the pack name of a disk file. Similar to \$ FAMILY and \$ FILEID. Default of blank name and MCP assumes systems pack. This entry should be included to ensure correct handling of the file by the MCP.
- \$ FAMILY      Specifies the external main directory family name associated with the file. The value field contains the name (one to 10 characters left-justified).
- \$ FILEID      Specifies the external file ID associated with the file. The value field contains the name (one to 10 characters left-justified).

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

- family name
- family name/file ID
- disk pack ID/family name/file ID
- disk pack ID/family name/

The internal name of each file is the filename assigned in the File Description Specifications.

Unless the FAMILY specification is used, the family name will be the same as the internal filename, that is, the filename specified on the File Description Specification. Therefore, if none of the file identification extensions are used the internal and external name will be the same, i.e., the family name. Figure 11-2 illustrates how the insertion of the different file identification extensions before a file called MASTER affects that file's external name.

CODING

EXTERNAL NAMES

02	F	\$	FILEIDMASTERFILE				
03	F		MASTER	IP		180	

MASTER/MASTERFILE

02	F	\$	FAMILYPAYROLL				
03	F		MASTER	IP		180	

PAYROLL

02	F	\$	PACKIDPAYMASTER				
03	F		MASTER	IP		180	

PAYMASTER/MASTER/

02	F	\$	FAMILYPAYROLL				
03	F	\$	FILEIDMASTERFILE				
04	F		MASTER	IP		180	

PAYROLL/MASTERFILE

04	F	\$	FAMILYPAYROLL				
05	F	\$	PACKIDPAYMASTER				
06	F		MASTER	IP		180	

PAYMASTER/PAYROLL/

02	F	\$	FAMILYPAYROLL				
03	F	\$	FILEIDMASTERFILE				
04	F	\$	PACKIDPAYMASTER				
05	F		MASTER	IP		180	

PAYMASTER/PAYROLL/MASTERFILE

614092

Figure 11-2. Coding File Identification Extensions - Dollar Sign Options

RPG Extensions

The following options may appear only within the File Description Specifications, except R<sub>SIGN</sub>, and must immediately precede the specification line describing the file to which they apply. None of the following operations may be reset except R<sub>SIGN</sub> and ONEPAK.

\$ AREAS

Specifies the maximum number of areas to be allocated for the file (disk files only). The VALUE field contains an integral value (right justified, leading zeros optional). A system-dependent default value of 25 is assumed unless specified by use of this dollar option. Maximum number of areas that may be assigned is 105.

\$ RPERA Specifies the maximum number of logical records that will be written in each disk area. The VALUE field contains an integral value (right-justified, leading zeros optional). The default value assigned is 500 unless specified otherwise. If records per area is not an even multiple of blocks per area the compile will default to the nearest even multiple of blocks per area.

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

\$ CLOSE Explicitly close allows all input serial files to remain opened until End-of-Job. Default is the implicit close of files at End-of-File.

\$ AAOPEN Is a file OPEN time option used to set a bit in the program's code file File Parameter Block for this file and to allocate all disk space areas at the time the file is opened.

\$ ONEPAK Specifies that this particular file must be contained on one disk.

\$ CYL Allocates file areas starting on an integral cylinder boundary.

\$ DRIVE Allocates a physical drive to that particular file. Applies only to Systems disks. VALUE field must be 0-15. Option may not be reset and is not related to PACKID.

\$ REFORM Input and update disk files are assumed to have the block and record lengths declared on the file header unless the \$ REFORM option is used. However, on input or update chained indexed file specifications "data keys in core" option, it may be desirable to also use \$ REFORM to indicate to the compiler that it may juggle the blocking factor to optimize the speed of chaining. Under this condition, the block length and record length specified on the File Description Specifications must be the same as when the file was output. This combination will produce the fastest chaining possible.

\$ REORG Specifies a specialized method of sorting indexed files and will be invoked at End-of-Job. The REORG feature only sorts the additions and then merges them, in place, into the master file. This method of sorting should decrease the sort time and the temporary disk area required. The VALUE field contains the external file identifier of the indexed file including the disk pack ID.

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

- \$ RESIGN            Indicates to the compiler the location of the sign of external numeric data items. When set, all signs are assumed to be in the low-order (rightmost) position of the field; when reset, all signs are assumed to be in the high-order (leftmost) position of the field. This option may be set and reset at different points in the source program, allowing different fields to have different sign positions. If the option is used, it will override the sign position specified in the Control Card Specifications.
- \$ CHAIN            Specifies that only the binary search code is to be emitted for chained indexed disk files.
- \$ DNAME            The value field contains an alpha mnemonic for the physical hardware device to which it refers.

Only one of the following four \$ RPG Extensions may be used per file specification:

- \$ PTAPE            Modifies READER or PUNCH to apply to a paper tape reader.
- \$ TAPE7            Modifies a tape device to 7-track.
- \$ TAPE9            Modifies a tape device to 9-track.
- \$ CASSET           Modifies a tape device to a cassette.

Compiler-Directing Options

The following options may appear anywhere within the B 1700 RPG Source Program. These options direct the compiler to perform specific functions. All of the following options (with the exception of SEG) may be "reset".

- \$ LIST            Specifies that the compiler produce a single spaced output listing of the source statements with the error or warning messages. This option is set "on" by default. Resetting to "off" will not inhibit the errors or warning messages from printing.
- \$ LOGIC           Specifies that the compiler produce a single-spaced listing of each source specification line, followed immediately by an intermediate code used to generate COBOL S-code. The listing is produced after the NAMES listing (if the NAMES option is set), and does not include addresses or bit configurations, but only the opcodes and logical operands of the program.
- \$ MAP            Specifies that the compiler produce a single-spaced listing, detailing the program's memory utilization. The MAP listing is produced after the LOGIC listing (if the LOGIC option is set).

- \$ NAMES** Specifies that the compiler is to produce a single-spaced listing of all assigned indicators, file names, and field names. The attributes associated with each file and field are also listed. The NAMES listing is produced immediately after the normal source input listing.
- \$ BAZBON** Specifies that if an indicator is assigned to a field to test for ZERO or BLANK in the Input or Calculation Specifications and the same field is used in the Output Specifications with a BLANK AFTER designation, that indicator will be turned ON after the field is blanked during the output operations. Should a N (NOT) be specified in column 8, the indicator will be turned OFF, overriding the original RPG I or RPG II specifications.
- \$ ZBINIT** Specifies that all ZERO or BLANK indicators are initialized ON at Beginning-of-Job; or, if a N (NOT) is specified in column 8, they will be initialized OFF regardless of the specifications for RPG I or RPG II.
- \$ STACK** If infrequent stack overflow conditions occur during program execution, the user may change the stack size of the resultant program. This should be used only when a legitimate STACK overflow condition has occurred (i.e., nested subroutines more than 8 deep). The default stack size is 313 bits which will allow 8 entries in the stack. To increase the stack size, add 39 bits to the default size of 313 for each additional stack entry.
- \$ XREF** Allows the RPGXRF file to be created during compilation for use as input to the RPG/XREF program. The XREF option must be placed at the beginning of the RPG source program, prior to the first File Specification. At the completion of the compilation it is necessary to manually execute the RPG/XREF program in order to obtain the cross reference listing.
- \$ PARMAP** Produces a single-spaced listing of the compiler-generated paragraph names, source statement numbers, and actual segment displacements of the emitted code. This listing may be used to relate to the LOGIC listing.
- \$ SEG** Orders the compiler to begin placing code in an overlayable segment identified by the integer in the VALUE field (right-justified, between 0 and 7 inclusive). Segmentation is an automatic function of the RPG compiler. When the SEG option is used, automatic segmentation is not suppressed.

**\$ SUPR** Specifies that the compiler is to suppress all warning messages from the source program listing. (Error messages still print.)

**\$ XMAP** Specifies that the compiler print a single-spaced listing of all the code generated, complete with actual bit configurations and addresses. Combined with the listing produced by the LOGIC option, complete information about the generated code of the program is available. The XMAP listing is produced after the MAP listing if the MAP option is set.

**15-24 VALUE**

This field is used to specify a value to be associated with the option entered in the KEY WORD field. Not all options require a value; those that do are so designated in the individual descriptions of each option (KEY WORD field).

All alphanumeric values must be entered left-justified in the VALUE field. All numeric values must be entered right-justified in the VALUE field; leading zeros are optional.

**25-74 COMMENTS**

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

**75-80 PROGRAM IDENTIFICATION**

Refer to Section 2 for a complete description.



# SECTION 12

## COMPILER OPERATION

The Burroughs RPG Compiler is an integral part of the B 1700 software system. It is treated in many respects as merely another program to be executed and can be multiprogrammed with other programs, or with itself.

### SOURCE INPUT

An input card file labeled RPG/CARD is the only source required by the compiler. All source input must be punched in the EBCDIC character set unless it is translated on input.

If it is desired that the input file be assigned to a device other than the card reader, then this may be achieved by using the FILE statement. This, of course, means that the device from which the source file is read should have the same record length (80 characters) and blocking factor (1) as the device from which the compiler expects to read the file. An RPG/VECTOR file assigned through the FILE statement should have a record length of 96 and a blocking factor of 1.

### CONTROL CARD SYNTAX

The format of the MCP control cards is as follows:

```
? 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 1700 System Software Operational Guide, Form No. 1068731. The format of the four options of the COMPILE statement are discussed here.

- COMPILE This is a "compile and go" operation. If the compilation is error-free, the MCP schedules the object program for execution. The program will not be entered into the Disk Directory, and must be recompiled to be used again. The "compile and go" is the default option of the COMPILE statement. COMPILE may be abbreviated as CO.
- COMPILE TO LIBRARY This option will leave the program object file on disk and will enter the program-name into the Disk Directory after an error-free compilation. The program is not scheduled for execution. LIBRARY may be abbreviated as LI.
- COMPILE SAVE This option combines the execute and library options. The MCP will enter the program-name into the Disk Directory and will leave the object program file on disk. The MCP will also schedule the program for execution after an error-free compilation. The program remains in the Disk Directory. SAVE may be abbreviated as SA.
- COMPILE FOR SYNTAX This option provides a diagnostic listing as the only output. This option does not enter the program-name into the Disk Directory or leave the program object file on disk. SYNTAX may be abbreviated as SY.

The following sample deck could be used to compile a source program contained in punched card to library:

```
? COMPILE PROGRAM/TEST1 WITH RPG TO LIBRARY
? CHARGE 12345
? DATA RPG/CARD
  source specification cards
? END
```

The following sample deck could be used to compile a source program contained on magnetic tape for syntax:

```
? COMPILE AAA/BBB/TEST2 WITH RPG FOR SYNTAX
? FILE SOURCE NAME = RPG/SOURCE DEFAULT TAPE
? END
```

NOTE

For the effect of the DEFAULT option, refer to the B 1700 System Software Operational Guide, Form No. 1068731.

## VECTOR FILE INPUT

The vector input files consist of compile-time and execution-time vector files, both of which are described in the paragraphs that follow.

### COMPILE-TIME VECTOR FILES

If the program requires that table or array files be included during compilation, the compiler requires an input card file labeled RPG/VECTOR. This file may be input from a medium other than cards, through use of the FILE statement. Card input may be punched in either the BCL (96 column card) or EBCDIC (80 column card) character set.

Vector files must be entered in the same order as specified in the Extension Specifications, and each file must contain exactly the number of records specified. No separators define the end of one vector and the beginning of the next; records are read into one vector until that vector is full. Filling of the next vector is begun from subsequent records.

The following sample deck could be used to compile a source program (with compile-time vector input) contained on cards to library:

```
? COMPILE AAA/TEST3 WITH RPG TO LIBRARY
? DATA RPG/CARD
  source specification cards
? END
? DATA RPG/VECTOR
  table file cards
? END
```

### EXECUTION-TIME VECTOR FILES

If the program requires table or array files to be included at the beginning of object program execution, the program will require card files with the labels as specified by the program in the File Description and Extension Specifications. The files may be read from a medium other than cards, through use of the FILE statement.

## COMPILER FILE NAMES

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

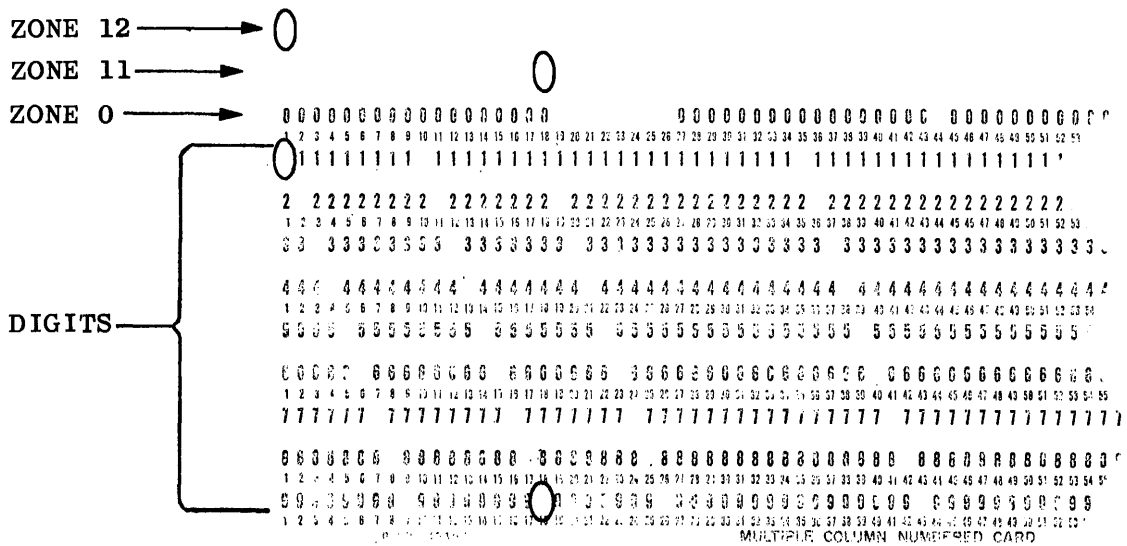
Table 12-1. Files Used for Source Input and Compilation Output

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

# APPENDIX A

## B 1700 CARD CODES

Figure A-1 shows the zone and digit portions of an 80-column card. The zone for character A is 12, and the digit is 1. The zone for character R is 11, and the digit is 9.



614093

Figure A-1. Zone and Digit Portions of an 80-Column Card

Figure A-2 shows the zone and digit portions of a 96-column card. The zones 12, 11, and 0 on a 96-column card are BA, B, and A, respectively.

Table A-1 lists the 80-column and 96-column card codes for the complete B 1700 character set.

Appendix A (Cont)

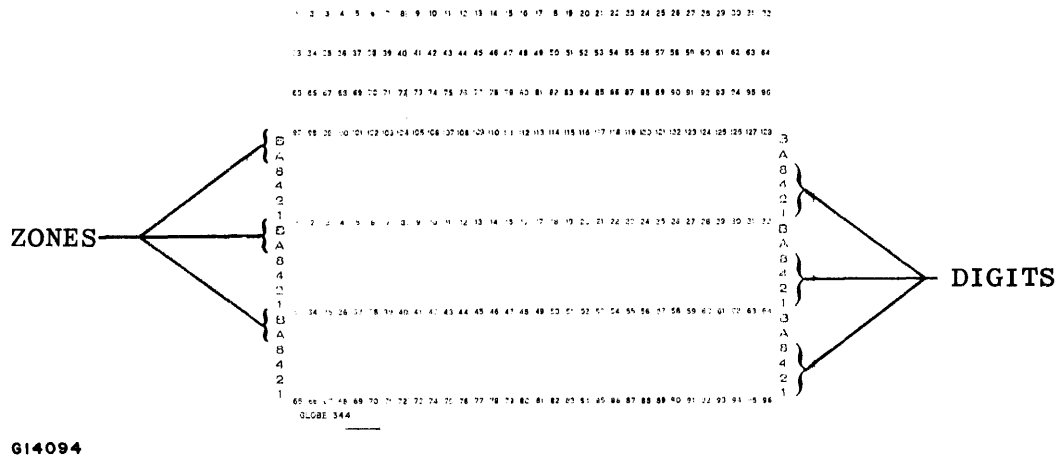


Figure A-2. Zone and Digit Portions of a 96-Column Card

Table A-1. Punch Card Codes for the B 1700 Character Set

Character	EBCDIC 80-Column Card Code	BCL 96-Column Card Code	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	11-2	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	0
E	12-5	B-A-4-1			

## APPENDIX B

# HEXADECIMAL VALUES FOR THE B 1700 CHARACTER SET

Table B-1 presents the RPG collating sequence, and table B-2 may be used in converting hexadecimal digits to binary code. In table B-1, the zone portion of a character is represented by the first hex digit, and the digit portion of the character is represented by the second hex digit.

Table B-1. B 1700 RPG Collating Sequence

Character	Hexadecimal Equivalent		Character	Hexadecimal Equivalent
blank	40	↓ Ascending Order ↓	'	7D
[	4A		=	7E
.	4B		'	7F
<	4C		A	C1
(	4D		B	C2
+	4E		C	C3
	4F		D	C4
&	50		E	C5
]	5A		F	C6
\$	5B		G	C7
*	5C		H	C8
)	5D		I	C9
;	5E		:	D0
┘	5F		J	D1
-	60		K	D2
/	6A		L	D3
,	6B		M	D4
%	6C		N	D5
_	6D		O	D6
>	6E		P	D7
?	6F		Q	D8
:	7A		R	D9
#	7B		S	E2
@	7C		T	E3

Appendix B (Cont)

Table B-1. B 1700 RPG Collating Sequence (Cont)

Character	Hexadecimal Equivalent		Character	Hexadecimal Equivalent	
U	E4	Ascending Order  ↓	2	F2	Ascending Order  ↓
V	E5		3	F3	
W	E6		4	F4	
X	E7		5	F5	
Y	E8		6	F6	
Z	E9		7	F7	
0	F0		8	F8	
1	F1		9	F9	

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



# BURROUGHS INDICATOR SUMMARY FORM

The RPG Indicator Summary Form, illustrated in figure C-1 is used strictly for documentational purposes. Its function is to provide an accurate record of the indicators that are used and the function of those indicators in the RPG Program.

**Burroughs**      B 1700 RPG

PROGRAM ID	PAGE	OF	PROGRAMMER	DATE
------------	------	----	------------	------

INDICATOR SUMMARY

PAGE 1 2      PROGRAM IDENTIFICATION 75 80

LINE	FORM TYPE	INDICATORS					CIRCLE INDICATORS USED:																NOTE: ALL INDICATORS ARE NOT VALID WITH ALL SYSTEMS.																																																							
		RECORD IDENTIFYING	INPUT FIELD	CALCULATION RESULT	MATCHING AND CHAINING	CONTROL LEVEL, OVERFLOW, HALT AND USER	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72
01	F*	ID	F	C	M	L	FUNCTION OF INDICATORS																																																																							
02	F*																																																																													
03	F*																																																																													
04	F*																																																																													
05	F*																																																																													
06	F*																																																																													
07	F*																																																																													
08	F*																																																																													
09	F*																																																																													
10	F*																																																																													
11	F*																																																																													
12	F*																																																																													
13	F*																																																																													
14	F*																																																																													
15	F*																																																																													
	F*																																																																													
	F*																																																																													
	F*																																																																													

G14095

Figure C-1. Burroughs Indicator Summary Form

The indicator Summary Form fields and their content are:

<u>Column</u>	<u>Content</u>
6	Form type (optional predefined F).
7	Asterisk required - causes complete card to be commented.

Appendix C (Cont)

<u>Column</u>	<u>Content</u>
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.

# COMPILE-TIME ERROR AND DIAGNOSTIC MESSAGES

## GENERAL

- G085 - FILE NAME NEVER DEFINED  
A file name has been referenced that has never been defined in File Specification.
- G095 - INVALID AND/OR LINE SPECIFICATION
- G304 - U1 - U8 MAY BE SET EXTERNALLY ONLY  
These external indicators may not be set programmatically and should only be used to indicate at execution time whether or not the file is to be used by the program.
- G306 - INDICATOR NEVER DEFINED  
This message is caused by an indicator being tested that has never been set.
- G315 - FIELD NAME NEVER DEFINED  
This message is caused by an undefined variable name being referenced.
- G524 - CHAINING KEY LENGTH NOT EQUAL CHAINED FILE KEY LENGTH  
This message indicates that the key length specified on the File Specification does not agree with the variable used to initiate chaining.
- G999 - DATA NOT NEEDED/EXPECTED IN THIS CARD

## FILE DESCRIPTION SPECIFICATIONS

Refer to Section 4 for further information.

- F023 - INVALID FILE NAME  
The message is caused by an invalid file name in columns 7-14. A file name must be unique in the first seven characters.
- F024 - REDEFINED FILE NAME  
The file name specified in columns 7-14 does not contain a unique name in the first seven characters.
- F026 - INVALID FILE TYPE  
This message is caused by an illegal entry in column 15 of the File Specification form. Valid entries are I, O, U, C, and D.

Appendix D (Cont)

- F028 - INVALID FILE DESIGNATION  
This message is caused by an illegal entry in column 16 of the File Specification form. Blank must be used for output files; otherwise one of the following entries are required: P, S, C, T, or D.
- F033 - NO SEQUENTIAL INPUT FILE FOUND  
A primary or secondary file is required.
- F036 - INVALID END OF FILE ENTRY  
This message is caused by an invalid entry in column 17. Valid entries are blank or E.
- F039 - INVALID SEQUENCE ENTRY  
This message is caused by an invalid entry in column 18. Valid entries are A, D, or blank.
- F042 - INVALID BLOCK-RECORD LENGTH  
When the Block and Record length contain blanks, a default of 132 is assumed for print files, and 80 for all other devices. When the block length is blank, it will assume the record length. When the block and record length contain valid numbers, the block length must be an integral part of the record length.
- F044 - MULTI BUFFERS AND STACKER SELECTION ON SAME FILE  
Stacker selection may not be used when multiple buffers are specified, owing to the processing of the records from the buffer areas.
- F044 - INVALID ENTRY TYPE OF ORGANIZATION  
This message is caused by an invalid entry in column 32. Valid entries must contain I, 1-9, or blank.
- F046 - INVALID ENTRY ADDITION-UNORDERED  
This message is caused by an invalid entry in column 66. Valid entries are U, A, or blank.
- F049 - INVALID/MISSING EXTENSION CODE  
This message is caused by an invalid entry in column 39. Valid entries are blank or E for table files; L may be specified for print files.
- F052 - INVALID DEVICE  
This message is caused by an invalid device specified in columns 40-46. Valid entries are READER, MFCU1, PUNCH, PRINTER, PRINTR2, TAPE, DISK, SPO, or CONSOLE.
- F056 - INVALID FILE FORMAT/FILE CLOSE  
This message is caused by an invalid entry in column 19. Valid entries are F, V, or blank.
- F057 - INVALID INDICATOR MUST BE U1-U8  
This message is caused by an invalid entry in columns 71-72. Valid entries are U1-U8 or blank.

- F400 - INVALID ENTRY MODE  
This message is caused by an invalid entry in column 28. This field must contain R or blank.
- F404 - INVALID ENTRY ADDRESS TYPE  
This message is caused by an invalid entry in column 31. Valid entries are A, K, P, N, or blank. On indexed files, this field specifies the format of the key; it must be left blank if the file is not indexed.
- F407 - INVALID ENTRY ADDITION-UNORDERED
- F543 - INVALID KEY LENGTH ENTRY  
The key length is required when using indexed processing. When used, this field (columns 29-30) must contain a valid number from 1-99, and the key length plus the key start location must not exceed the record length.
- F549 - INVALID KEY START LOCATION  
The key start location field is required when using indexed processing. This field (columns 35-38) must contain a valid number, and the key start location plus the key length must not exceed the record length.
- F553 - INVALID KEYS IN CORE ENTRY  
The keys in core entry (columns 60-65) must contain a valid number not exceeding 9,999.

#### EXTENSION SPECIFICATIONS

Refer to Section 5 for further information.

- E061 - CHAINING FIELD FOR THIS LEVEL NOT DEFINED ON INPUT  
This message indicates that automatic chaining has been specified, and the corresponding level has not been specified on input.
- E062 - INVALID FROM FILE NAME ENTRY  
This message is caused by an invalid entry in columns 11-18. It is used as the file name of every execution-time vector, but must be left blank if the vector is to be loaded at compile time or via Input or Calculation Specifications.
- E063 - INVALID TO FILE NAME ENTRY  
This message is caused by an invalid entry in columns 19-26. It is used as the file name of a vector file that is to be written or punched. A valid file name must be used as well as being previously defined on the File Description Specification form.
- E067 - INVALID VECTOR NAME  
This message is caused by an invalid table or array name in column 27-32. Valid vector names cannot exceed six characters and must be unique. Table names must begin with the letters TAB.

## Appendix D (Cont)

- E068 - INVALID ENTRY NUMBER OF ENTRIES PER RECORD  
This message is caused by an invalid entry in columns 33-35. This field must contain a valid number. This entry times the length of Vector A plus the length of Vector B (if applicable) must not be greater than the From File length or (if applicable) 96.
- E070 - INVALID ENTRY NUMBER OF ENTRIES PER VECTOR  
This message is caused by an invalid entry in columns 36-39. This entry must be a valid number limited only to the size of the field.
- E072 - INVALID ENTRY LENGTH OF ENTRY  
This message is caused by an invalid entry in columns 40-42/52-54. This entry must be a valid number specifying the length of the entry of the vector. Alpha entries must not exceed 511 and numeric entries 31.
- E074 - INVALID ENTRY PACKED  
This message is caused by an invalid entry in column 43/55. Valid entries are blank and P. P is only allowed on numeric fields.
- E076 - INVALID DECIMAL POSITIONS  
This message is caused by an invalid entry in column 44/56. Valid entries are 0-9 or blank.
- E077 - INVALID SEQUENCE ENTRY  
This message is caused by an invalid entry in column 45/57. Valid entries are A, D, or blank.
- E079 - INVALID ALTERNATE VECTOR NAME  
This message is caused by an invalid table or array name in columns 46-51. Valid vector names cannot exceed six characters and must be unique. Table names must begin with the letters TAB.
- E122 - REDEFINED VECTOR NAME  
This message is caused by a vector name that has been redefined. Vector names must be redefined using the same length, type, and decimal positions as used on the first definition.
- E228 - INVALID/OUT OF BOUNDS LITERAL -- VECTOR, LITERAL --  
Literals must contain digits 0-9 and cannot exceed the size of the vector.
- E315 - INVALID/UNDEFINED FIELD NAME -- VECTOR, FIELD NAME --  
This message is caused by an illegal field name or a field name that has not been defined.

### LINE COUNTER SPECIFICATIONS

Refer to Section 6 for further information.

- L085 - INVALID FILE NAME OR FILE TYPE  
This message indicates that a file name in columns 7-14 is not valid or has not been assigned to a print file. The file name must also have been previously defined on the File Specification.

- L088 - WHEN DEFINED -FL- MUST BE IN COL 18-19
- L090 - INVALID CHANNEL ENTRY  
This message is caused by an illegal channel entry. Valid entries are 01-12, OL, or FL.
- L091 - INVALID LINE ENTRY EXCEEDS FORM LENGTH OR 112  
The line position entry must not exceed the form length or 112.

### INPUT SPECIFICATIONS

Refer to Section 8 for further information.

- I084 - RECORD LINE ASSUMED 43-70 TREATED AS BLANK
- I092 - INVALID FILE NAME  
This message indicates that the file name in columns 7-14 is not valid or has not been assigned to an input file. The file name must also have been previously defined on the File Specifications.
- I092 - FILE NOT DEFINED AS INPUT  
The file type in column 15 of the File Specification form has not been specified as I, U, C, or D.
- I093 - FIELD LINE ASSUMED 7-42 TREATED AS BLANK
- I094 - RECORD LINE WITH FILE MUST BE FIRST INPUT SPEC  
The first Input Specification must contain a file name.
- I097 - NO FIELDS DEFINED IN LAST RECORD
- I101 - SEQUENCE FIELD SPECIFICATION  
This message is caused by an invalid entry in columns 15-16. It must contain a valid sequence entry, either 2 alpha characters other than ND or Rb, or a numeric entry from 01-99 in ascending order within the file.
- I102 - INVALID ENTRY -NUMBER-  
The number position entry in column 17 must contain 1 or N if the sequence entry is numeric, or blank if alphabetic.
- I103 - INVALID ENTRY -OPTION-  
This message is caused by an invalid entry in column 18. Valid entries are 0 and blank. The 0 may only be used on numerical entries, and one numerical entry should be non-optional.
- I107 - INVALID RECORD ID CODE  
This message is caused by an invalid entry between 21-41. The position entry must fall within the record length.
- I109 - INVALID ENTRY -STACKER-  
This message is caused by an invalid entry in column 42. Valid entries are blank and 1-6 on card files.

Appendix D (Cont)

- I111 - INVALID PACKED ENTRY  
This message is caused by an invalid entry in column 43. Valid entries are P or blank. P may be specified only on numeric field lines, and if specified, column 52 (decimal positions) must not be blank.
  
- I113 - INVALID FROM TO ENTRY  
The FROM and TO entries must be valid numbers and cannot exceed the record length. Numeric fields must not exceed 31. Alpha fields must not exceed 511. Vector load elements must be the same length as the element defined on the Extension Specifications, and the entire array load must not be greater than the array length and modulo the element length.
  
- I118 - INVALID FIELD NAME  
This message is caused by an invalid field name in columns 53-58.
  
- I119 - INVALID CONTROL LEVEL INDICATOR  
This message is caused by an invalid entry in columns 59-60. Valid entries are blank and level indicators L1-L9.
  
- I120 - INVALID ENTRY MATCHING-CHAINING FIELD  
This message is caused by an invalid entry in columns 61-62. Valid entries are blank, M1-M9, and C1-C9.
  
- I122 - REDEFINED FIELD NAME - LOOKAHEAD  
A look-ahead field name must be unique.
  
- I122 - REDEFINED FIELD NAME  
When used, a duplicate field name must contain the same length and decimal positions entry as the original.
  
- I159 - INVALID RECORD INDICATOR  
This message occurs when columns 19-20 do not contain a valid indicator or look-ahead.
  
- I166 - INVALID FIELD RESULTING INDICATOR  
This message occurs when columns 65-70 do not contain blanks or a valid indicator.
  
- I170 - INVALID ENTRY FOR CHAIN OR DEMAND FILE  
Chain or demand files may not contain control levels or matching fields.
  
- I172 - INPUT FILE ORDER IS DIFFERENT FROM FILE SPEC  
The input files must be in the same order as defined on the File Specifications.
  
- I181 - REDEFINED LENGTH OF MATCHING-CHAINING FIELD  
REDEFINED LENGTH OF CONTROL FIELD.  
This message indicates that the total length of a given level on the current record is not identical to the total length of the same level of a previous record.



- I188 - INVALID RECORD RELATION INDICATOR  
This message is caused by an invalid entry in columns 63-64. This field must contain a valid indicator or blank.
- I226 - INVALID ENTRY DECIMAL POSITIONS  
The decimal positions entry in column 52 must contain a blank or 0-9. On vector loads, the decimal positions entry must be the same as the defined element.
- I575 - UNREFERENCED OR MISGROUPED REFERENCE OF FILE NAME  
This message indicates that an Input Specification has been referenced that has not been declared in the File Specification, or that the file names appear more than once and not consecutively, or that it does not appear in the same order as the File Specification.

### CALCULATION SPECIFICATIONS

Refer to Section 9 for further information.

- C122 - REDEFINED FIELD NAME  
This message indicates that the name defined in the result field has been previously defined with different length and decimal positions.
- C122 - REDEFINED FIELD NAME - LOOKAHEAD  
This message indicates that the name defined in the result field has been previously defined as a look-ahead field with different length and decimal positions.
- C123 - INVALID CONTROL LEVEL INDICATOR  
This message is caused by an invalid entry in columns 7-8. Valid entries are L0-L9, LR, AN, OR, and, when in a sub-routine, SR.
- C125 - INVALID FACTOR 1  
The field in columns 18-27 containing Factor 1 must be present if required, or absent if not required as specified by the Operation Code. When present, it must be a legal variable.
- C128 - INVALID OPERATION CODE  
The operation code specified in columns 28-32 is not legal.
- C131 - INVALID FACTOR 2  
The field in columns 33-42 containing Factor 2 must be present if required, or absent if not required as specified by the operation code. When present, it must be a legal variable.
- C135 - INVALID RESULT FIELD  
The field in columns 43-48 containing the result field must be present if required, or absent if not required as specified by the operation code. When present, it must be a legal variable but may not contain a literal.
- C135 - INVALID RESULT FIELD NAME

Appendix D (Cont)

- C137 - INVALID RESULT FIELD LENGTH  
This message is caused by an invalid entry in columns 49-51. The result field length must contain a valid number between 1-31 for numeric fields or between 1-511 for alpha fields.
- C138 - INVALID DECIMAL POSITIONS  
This message is caused by an invalid entry in column 52. Valid entries are blank and 0-9.
- C140 - INVALID ENTRY HALF ADJUST  
This message is caused by an invalid entry in column 53. Valid entries are blank and H. H may only be used with arithmetic operators.
- C190 - INVALID BEGSR/ENDSR RELATIONSHIP  
This message is used in conjunction with subroutines. Each subroutine must begin with a BEGSR and end with an ENDSR, and may not contain another subroutine.
- C200 - INVALID RESULTING INDICATOR  
This message is caused by an invalid indicator or one that should not have been specified.
- C207 - INVALID ALPHA LITERAL  
Alpha literals must be contained in apostrophes and must follow the rules for forming literals.
- C207 - INVALID NUMERIC LITERAL  
This message is caused by an invalid numeric literal. Numeric literals start with +, -, ., or 0-9 and may contain only one comma or one decimal point, with no embedded blanks, and only digits 0-9.
- C214 - INVALID GOTO - TAG OR EXSR - BEGSR RELATIONSHIP  
GOTO operations are valid only in conjunction with TAG and ENDSR labels. EXSR operators must be associated with a BEGSR.
- C215 - FACTOR 1 AND FACTOR 2 BOTH LITERALS  
It is illegal to have literals in both Factor 1 and Factor 2.
- C221 - RESULT FIELD MAY NOT BE LARGE ENOUGH  
This warning occurs when an overflow condition is likely to occur, which would cause high order digits to be lost.
- C232 - INVALID OR DUPLICATE TAG, BEGSR, OR ENDSR NAME
- C304 - INVALID INDICATOR  
A conditioning indicator must be specified on AND/OR lines and must be a valid indicator.
- C519 - INVALID AN - OR ENTRY  
This message is caused by an illegal entry. The last of a series of AND/OR lines must contain a calculation operation.

OUTPUT-FORMAT SPECIFICATIONS

Refer to Section 10 for further information.

- O097 - NO FIELDS OR LITERALS DEFINED IN LAST RECORD
- O142 - RECORD LINE ASSUMED 32-70 TREATED AS BLANK
- O143 - INVALID OUTPUT RECORD TYPE  
This message is caused by an invalid entry in column 15. Valid entries are H, D, T, or E. This entry may also contain an N or R on AND/OR lines.
- O146 - INVALID FILE NAME  
This message indicates that the file in columns 7-14 is not valid or that it has not been previously defined on the File Description Specifications as an output file.
- O148 - INVALID FIELD NAME  
This message is caused by an invalid entry in columns 32-37. A field name previously defined on Input, Extension, or Calculations Specification may be used, or one of the special reserved words may be used.
- O150 - INVALID BLANK AFTER ENTRY  
The blank after entry in column 39 must contain a blank or B.
- O151 - INVALID ENDING POSITION  
This message is caused by an invalid entry in columns 40-43. This entry must contain a valid number that must not exceed the record length of the file, and the length of the field must not underflow the record. This must also include the size of the editing symbols when used.
- O152 - INVALID PACKED ENTRY  
This message is caused by an invalid entry in column 44. Valid entries are blank or P. When P is specified, a numeric variable must be described in columns 32-37.
- O154 - RECORD LINE WITH FILE MUST BE FIRST OUTPUT SPEC  
The first Output Specification must be a record line containing a file specification.
- O154 - FIELD LINE ASSUMED 7-22 TREATED AS BLANK
- O212 - EXCPT CALC WITHOUT EXCEPTION OUTPUT RECORD TYPE  
The EXCPT operation code has been used when no Exception output records have been defined.
- O256 - INVALID STACKER/FETCH ENTRY  
This message is caused by an invalid entry in column 16. Valid entries on punch files are 1-6, and on Print files, are F or blank.

## Appendix D (Cont)

- 0258 - INVALID SPACE/SKIP ENTRY  
This message is caused by an invalid entry in columns 17-18 or 19-22. Valid entries for Spacing are blank, 0, or 1-9; for Skipping, 0-99, AO-A9, or B0-B2; and must only be used on Print files.
- 0273 - INVALID OUTPUT INDICATOR  
The output indicators in columns 23-31 must contain a blank or a valid indicator. Valid indicators are 01-99, L0-L9, LR, MR, H0, H9, U1-U8, OA-OG, OV, or 1P.
- 0276 - INVALID EDIT CODE  
Column 38 must contain a blank if the field name field contains an alpha variable or literal; when numeric, it may contain a blank, 1-4, A-B-C-D, J-K-L-M, or X-Y-Z.
- 0277 - INVALID EDIT WORD  
The number of replaceable characters in the edit word (columns 45-70) must be equal to or greater in length than the length of the field to be edited.
- 0279 - INVALID CONSTANT SIZE
- 0283 - INVALID FILE TYPE FOR OUTPUT RECORD  
The file type has not been defined as O, U, C, or an input/add file.
- 0289 - \*PLACE OR \*PRINT PRECEDES ALL FIELDS AND CONSTANTS
- 0548 - INVALID FILE ADDITION  
The file referenced has not been declared as an add file but ADD was specified. ADD will be assumed.
- 0554 - ADD NOT SPECIFIED ASSUME -ADD-  
All files, except update files, using "A" in column 66 in the File Specifications should have "ADD" in columns 16-18 of each record in the output of the corresponding file.
- 0998 - EDIT TO BE PERFORMED ON ALPHANUMERIC FIELD  
Only numeric items may be edited.
- 0999 - BLANK AFTER INVALID FOR OUTPUT LITERAL  
Blank after may not be used after an output literal.

### DOLLAR CARD SPECIFICATIONS

Refer to Section 11 for further information.

- DOL1 - INVALID ENTRY -NEGATE-  
This message is caused by an invalid or illegal entry in column 8. Valid entries are blank or N; however, on some Dollar Card Specifications, it is illegal to use the Not option.
- DOL2 - INVALID ENTRY -KEYWORD-  
This message is caused by the keyword entry not containing a valid option.

**DOL3 - INVALID ENTRY -VALUE-**

This message is caused by an illegal entry in the value field of the \$ option. Alpha entries must contain a character from A to Z in column 15, and numeric entries must contain blanks or zeroes in columns 15 and 16.

# EXECUTION-TIME ERROR MESSAGES

Certain conditions may arise during program execution which require operator notification and, in most cases, acknowledgment. Some program errors are recoverable, and some are not. For more information refer to the B 1700 System Software Operational Guide, Form No. 1068731.

## INPUT ERROR MESSAGES

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

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

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

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

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

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

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

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

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

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

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

## Appendix E (Cont)

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

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

### PROGRAMMED HALTS

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

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

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

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

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

### FORMS POSITIONING

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

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

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

### ARITHMETIC ERRORS

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

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

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

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

VECTOR (ARRAY, TABLE) ELEMENT ERRORS

The occurrence of the following is a result of an attempt to reference a vector element that is out-of-bounds (i.e., an index value less than or equal to zero, or greater than the maximum size of the array as specified in columns 36-39 of the Extension Specifications). The program will automatically be discontinued (DS-ED) by the MCP.

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

Note: Subscript is terminology used for vector element.

SEQUENCE ERRORS

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

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



## B 1700 RPG TO COBOL TRANSLATOR (COFIRS II)

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

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

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

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

### COFIRS II OPERATION

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

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

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

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

Appendix F (Cont)

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

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

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

\$ EXECUTE COFIRS

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

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

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

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

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

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

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

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

- b. The actual data-names from the RPG program will be found in the WORKING-STORAGE SECTION with -F appended. These fields are filled in the INPUT-MOVE portion of the COBOL program and are used in the various calculations as required by the Calculation Specifications in the RPG program.
- c. Compile-time vectors are entered in the same manner as if compiling RPG. These tables become WORKING-STORAGE SECTION entries containing the vector values entered at compile-time. The individual areas within this vector are referenced by an entry such as the following:

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

where:

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

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

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

- d. RPG Indicators are prefixed by IND-, e.g., indicator 01 becomes IND-01.
- e. References to UDATE will cause a single access to TODAYS-DATE which is then stored in the field UDATE-F.
- If PAGE is used in the RPG program, a field, PAGE-F, will be declared and logic necessary to increment it is developed.
- f. TAG names used in the RPG Calculation Specifications will appear as paragraph names in the COBOL program with -F-TAG appended, e.g., ENDDDET-F-TAG.
- g. B 1700 COFIRS II allows up to 15 files, 512 fields, and 64 vectors in a program for translation, subject to limitations in the RPG Compiler.
- h. The Program Identification inserted in columns 73-80 is taken from columns 75-80 of the RPG Control Card. If this field is left blank, the default is RPGOBJ. The new COBOL source file name is RPGCOB and is sequenced by 100.
- i. The following defaults are used if the appropriate information is not provided in the RPG source program:

1) All files:

<u>Option</u>	<u>Default</u>
Family Name	Internal file name from the File Description Specifications.
File ID	None.

Appendix F (Cont)

2) Disk files:

<u>Option</u>	<u>Default</u>
Areas	25.
Records per Area	500.
Disk Pack ID	None (defaults to System Disk).

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

j. System Requirements:

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

## APPENDIX G

# TAGSORT WITH RPG

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

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

Basically, the tagfile is created by a TAGSORT program which goes through the input file, extracting the key field(s) from the record, and adding the position of the record, relative to the first record in the file. The tagfile is then sorted, using the regular sort concept, and leaving a file of relative record numbers (4 bytes in length) sorted in the specified sequence. The original input file remains unchanged.

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

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

Table G-1. Building a Tagfile

Contents of <u>DISKFI</u>	<u>TAGSORT Program</u>	Contents of <u>TAGFILE</u>
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

Appendix G (Cont)

The contents of DISKFI consists of nine records. The first three digits comprise the key field, the last four comprise the data field.

Since DISKFI was sorted by key in ascending order and record number 2 has the lowest key value, record 2 is placed in the first position of the tagfile. Record 5 has the second smallest key (258), so 5 is placed in the second position of the tagfile, etc.

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

Table G-2. Accessing a Tagfile

Program called TAG

```

FTAGFILE IPE          DISK
FDISKFI  IC          DISK
FPRINTIT O          PRINTER

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

C                RECNO  CHAINDISKFI

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

Printer Output From TAG

```

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

# RPG PROGRAM CYCLE

For each record from a primary or secondary file that is processed, the RPG object program goes through the same general cycle of operations. After a record is read, there are two different instances in time when calculation operations are performed and records written out. First, all total calculation operations (those conditioned by control level indicators in columns 7-8) and all total output operations are done. Second, all detail calculation operations and all detail output operations are done.

Total calculations are performed before the information on the record selected for processing is made available. Detail calculations are performed after the information on the selected record is made available. The following discussion describes this concept in more detail.

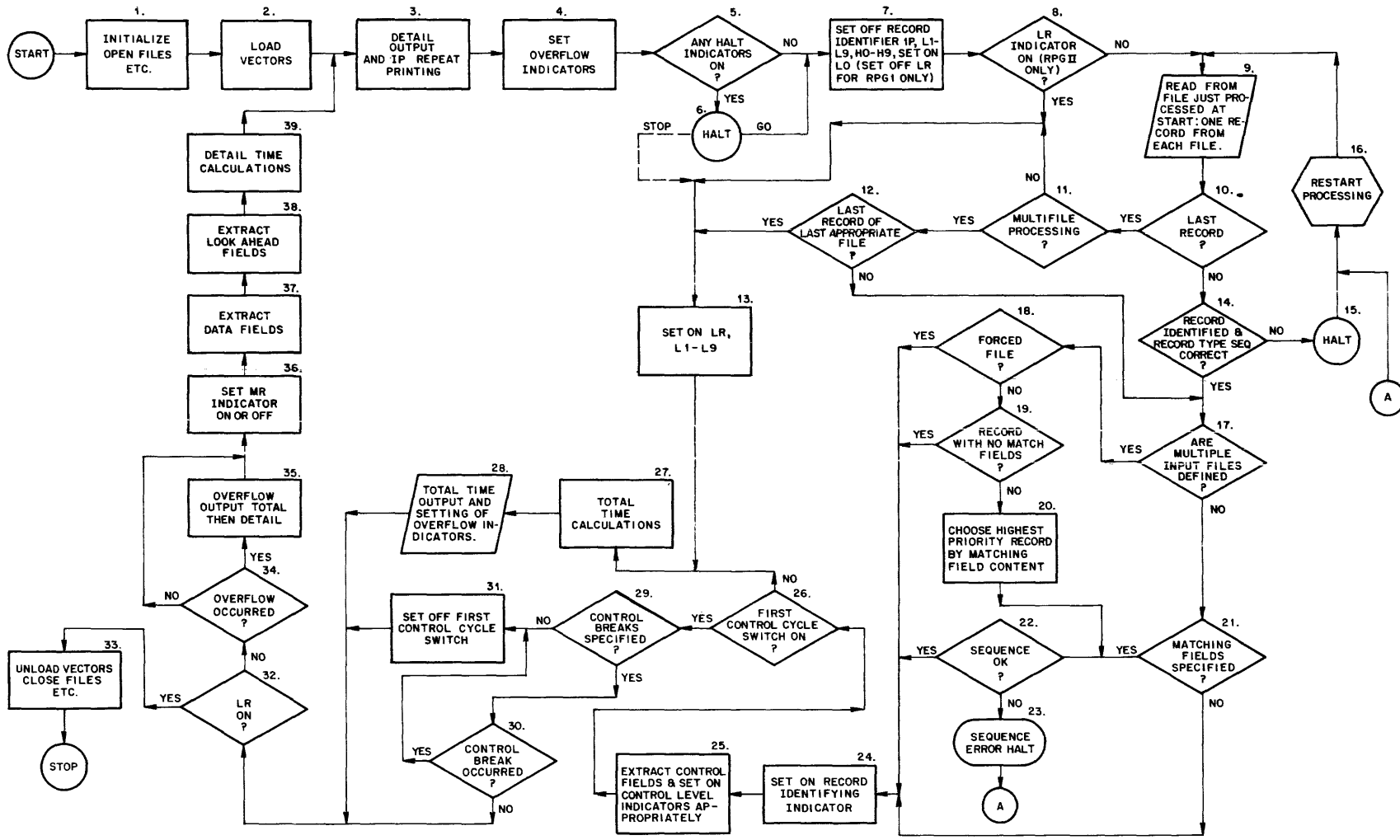
Whenever a record is read, a check is made to determine if information in a control field (when one has been specified) is different from the control field information on the previous card. A change in the control field information indicates that all records from a particular control group have been read and a new group is starting. When all records from a group have been read (indicated by control level indicators being set ON), operations may be done using information accumulated from all records in that group. It is at this time that all calculations conditioned by control level indicators in columns 7-8 are done. Total output operations are also performed immediately after all total calculation operations are completed. Information on the record read at the beginning of the program cycle is not used in these operations; only information from records in the previous control group is used.

Detail calculations occur after the information on the selected record has been made available. Detail calculations are used to calculate values needed each time a record is processed. They are also used to calculate totals for the current control group (if control fields are specified). Immediately after detail calculation operations are completed, detail output operations are performed.

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

## 1. Initialization:

- a. Data fields and indicators may have been preset to their initial values by the compiler; if not they must be initialized now:
  - 1) All data fields and vectors are initialized to zero (if numeric) or blank (if alphanumeric).
  - 2) All match fields are initialized to high (if descending sequence) or low (ascending) collating values.



614096

Figure H-1. RPG Program Cycle



- 3) All control field storage areas are initialized to hexadecimal zeroes.
  - 4) Indicators LO and IP are set ON, all other indicators are set OFF. Under the RPG1 dialect any indicators from 01-99 used as zero/blank indicators will be set ON, unless also used as record identifying indicators.
- b. UDATE, UDAY, UMONTH, UYEAR fields are set.
  - c. External indicators are set.
  - d. Files are opened (unless files are to be opened automatically when first accessed).
  - e. The first-control-cycle switch is set on. This is an internal switch used to determine when to suppress total time.
2. Pre-execution time vectors are loaded (if specified).
  3. Detail Output. All heading and detail output operations whose indicator conditions are met are performed (this will not include output conditioned on overflow indicators). On the very first cycle, the option for forms alignment may have been specified. If so, step 3 (detail output) is repeated as many times as the operator requires. Generally, the programmer will control the heading output required on the first cycle by conditioning it on the indicator IP (which will be off for all cycles except the first).
  4. All overflow indicators are set OFF. If any printer file is on the overflow line or between the overflow line and the end of the page, any overflow indicator associated with that file is set on.
  5. If any of the halt indicators H0 - H9 are on, the program branches to step 6; otherwise, to step 7.
  6. A message is displayed indicating which halt indicators are on. The operator has 2 options:
    - a. STOP. Program branches to step 13. After performing "last record," total calculations, and total output, the program will terminate.
    - b. GO. The program continues at step 7.
  7. All record identifying indicators are set OFF. IP, L1-L9 and H0-H9 are set OFF. LO is set ON. Under the RPG1 dialect LR is also set OFF.
  8. Under the RPG1 dialect, the program will always continue at step 9; otherwise, LR is tested. If LR is ON, the program branches to step 11; otherwise, to step 9.
  9. The next input record is read. On the first cycle, one record is read from each primary and secondary file. On all subsequent cycles, the record is read from the file that was processed last. For an input (not update or combined) file with look-ahead fields, the record will already have been read at step 38 of the previous cycle.

## Appendix H (Cont)

10. A test is performed to determine whether the file just read is at End-of-File or is conditioned by an external indicator which is OFF (on the first cycle, the test is whether any of the files read satisfy this condition). If yes, the program branches to step 11; otherwise, to step 14.
11. If this program has a primary file and at least one secondary file (i.e., the program performs multifile processing), the program branches to step 12, otherwise, to step 13.
12. If all required files are at End-of-File (as specified in column 17 of the File Specifications), the program branches to step 13; otherwise, to step 17.
13. Indicators LR and L1-L9 are set ON.
14. The record (or in the case of the first cycle, all the records) read at step 9 is identified (but the record identifying indicator is not yet set ON). A test is performed to determine whether the input records are in the sequence specified on the Input Specifications. If the record type sequence is incorrect (or if sequenced records are specified but the current record cannot be identified), the program branches to step 15; otherwise, to step 17.
15. The record type sequence error causes the program to halt after displaying an appropriate message to the operator.
16. The operator may order the program to resume. In which case, the out of sequence record is ignored and the next record is read by branching to step 9.
17. If multiple input files are specified, the program branches to step 18 in order to select the next record to be processed. If no secondary files are specified, the program continues at step 21.
18. If the FORCE opcode was performed during the previous program cycle, the forced file is selected for processing and the program branches to step 24.
19. The current records from each primary and secondary file are inspected in the order that the files were specified on File Specifications. The first record that has no match fields is selected and the program branches to step 24. If all records have match fields, the program branches to step 20.
20. The record with the highest priority matching field value (highest collating value for descending matching sequence, lowest collating value for ascending sequence) is selected to be processed next. If two or more files have equal and highest priority matching field values, the one with highest priority is selected (priority corresponds to the order in which files were specified on File Specifications).
21. As there are no secondary files, record selection is unnecessary. If the current record from the primary file has match fields, the program branches to step 22 to perform matching sequence checking.

22. The match field value of the selected record is compared to the match field value of the previous record processed. If it is in sequence, the program continues at step 24. If the match field value is out of sequence, the program branches to step 23.
23. The program halts after displaying a message to the operator indicating that a match field sequence error has occurred.
24. The program sets ON the record identifying indicator specified for the selected record. However, data from this record is not available for processing until step 37. If look-ahead fields are specified for update or combined files, they are extracted at this point (all other look-ahead fields are not extracted until step 38).
25. If the selected record has control fields, a test is performed to see if the contents are equal to the contents of the control field storage area. If the contents are unequal, a control break has occurred; the appropriate control level indicators are set ON and the new control field contents are stored in the control field storage area. Note that until the first control break occurs, the control field storage area contains its initial value of hexadecimal zeros.
26. If the first-control-cycle switch is ON, the program branches to step 29; otherwise, to step 27.
27. Total calculations (calculations conditioned by control level indicators L0-L9 in columns 7-8) are performed.
28. Total output that is not conditioned by an overflow indicator is performed. When this output is complete, the program tests to see if the overflow condition has occurred. If any printer file is on the overflow line or between the overflow line and the end of the page, any overflow indicator associated with that file is set ON.
29. If control fields were specified in this program, the program branches to step 30; otherwise, to step 31.
30. If a control break occurred on this cycle (at step 25), the program branches to step 31; otherwise, to step 32.
31. The first-control-cycle switch is set OFF. Total Calculations and Total Output will be performed on all subsequent cycles (if no control break occurs, only L0 operations will be performed).
32. If indicator LR is ON, the program branches to step 33.
33. End-of-Job routine.
  - a. Output any vectors specified with a TO FILENAME on the Extensions Specifications.
  - b. Perform any file maintenance e.g., sorting an indexed file if unordered or if additions have been made.
  - c. Close all files.

Appendix H (Cont)

34. Overflow output is performed for each printer file on which overflow & has occurred on this cycle. The test and the output performed take  
35. one of two forms. If an overflow indicator is assigned to the file on File Specifications or if overflow output is specified on Output Specifications for the file, the test is whether the overflow indicator for that file is ON (it may have been set ON by SETON; it may also have been set OFF if overflow output has been Fetched). If so, any Total and then any Detail output for that file which is conditioned on the overflow indicator is performed (if none is specified, no overflow output will be performed). The overflow indicator is then set OFF.

If no overflow indicator is assigned and no overflow output is specified for the file, the test is whether the overflow condition was satisfied in steps 4 or 28. If overflow did occur and has not been Fetched, the object program automatically generates a skip to line 1 of a new page.

36. The matching record indicator MR is set ON or OFF. MR is set ON if a match has occurred (i.e., if selected record has a match field and the current match field value was originally selected from a primary record).
37. Data is extracted from the current record and made available for processing. Any field indicators specified are set ON or OFF to reflect the status of the data fields.
38. If the current record is from an input (not update or combined) file which has look-ahead fields, the next record from that file is read, and the look-ahead fields are extracted and made available to the program.

Note that look-ahead fields from an update and combined file are made available at step 24.

39. Detail time calculations are performed. Processing continues with step 3.

# INDEX

<u>Item</u>	<u>Page</u>
AAOPEN dollar option .....	11-4
ACCEPT message .....	9-37
ADD operation code .....	9-14
Additional I/O areas (buffers) .....	4-12
Alphanumeric literal .....	2-3
Ampersand in edit words .....	10-28
AND line - Input Specifications .....	8-3
AND line - Output Specifications .....	10-14
AN/OR entries .....	9-3, 9-4, 9-33
AND Relationship - Calculation Specifications .....	9-4, 9-6
AN .....	9-4, 9-33
AREAS dollar option .....	11-4
Arithmetic errors .....	E-2
Arithmetic operations .....	9-12 thru 9-17
ADD .....	9-14
DIV .....	9-14
MULT .....	9-14
MVR .....	9-16
SQRT .....	9-16
SUB .....	9-14
XFOOT .....	9-17
Z-ADD .....	9-17
Z-SUB .....	9-17
Array element .....	5-1, 5-12, 9-8
Error messages .....	E-3
Maximum number .....	5-6
Sequence of .....	5-7
Ascending sequence .....	4-6, B-1
Asterisk file .....	10-24, 10-28
B 1700 card codes .....	A-1 thru A-3
B 1700 Standard Label .....	4-17
BAZBON dollar option .....	11-7
BCL card codes .....	12-1, A-3
BEGSR operation code .....	9-7, 9-34
Binary field operations .....	9-24
BITOF .....	9-26
BITON .....	9-25
TESTB .....	9-26
Binary to hexadecimal conversion .....	B-2
BLANK AFTER field .....	10-20
Blanks in edit words .....	10-24, 10-25
BLOCK LENGTH field .....	4-7
Defaults .....	4-8
Table of maximum and minimum values .....	4-7

## INDEX (Cont)

<u>Item</u>	<u>Page</u>
BSCA device .....	4-13, 4-17
Buffers .....	4-9
Calculation Specifications .....	9-1
Error and diagnostic messages .....	D-7
Field definitions .....	9-1 thru 9-10
Calculation-time exception records .....	10-3
CASSET dollar option .....	11-6
CHAIN dollar option .....	11-6
CHAIN operation code .....	4-4, 4-18, 4-25, 9-8, 9-35 thru 9-37
Chained file .....	4-4
Device assignment of .....	4-14
CHANNEL NUMBER field .....	6-1, 6-5
Character set .....	2-1, A-3, B-1
Numeric portion .....	8-11, A-3
Used for names .....	2-1
Used in editing .....	2-2, 10-24
Zone portion .....	8-11, A-3
CLOSE dollar option .....	11-5
Closing of tape files .....	4-20
COFIRS II .....	F-1 thru F-4
Combined files .....	4-3
Conditioned by external indicator .....	4-24
Device assignment of .....	4-14
Commas in edit words .....	10-24, 10-25
Comments .....	2-6
Calculation Specifications .....	9-10
Dollar Card Specifications .....	11-8
Extension Specifications .....	5-8
COMP operation code .....	9-23
Alphanumeric fields .....	9-24
Numeric fields .....	9-23
Compare operations .....	9-23
COMP .....	9-23
TESTZ .....	9-24
Compatibility with other RPG systems .....	xv, 1-3
Compile statement .....	12-2
Compile-time vectors .....	4-5, 5-1, 5-6, 12-3
Loading .....	5-9
Output .....	5-13
Compiler-directing options - \$ options .....	11-6
Compiler file names .....	12-3
Compiler operation .....	12-1
CONSOLE device .....	4-13
Console messages during program halts .....	4-6
CONSTANT OR EDIT WORD field .....	10-22
Constants .....	10-22
In edit words .....	10-28
Control Card Specifications .....	3-1
Field definitions .....	3-1 thru 3-4
Control card syntax .....	12-1
Control fields .....	8-19 thru 8-21

## INDEX (Cont)

<u>Item</u>	<u>Page</u>
CONTROL LEVEL field - Calculation Specifications .....	9-1
CONTROL LEVEL field - Input Specifications .....	8-19
Control level indicators (LO-L9) .....	8-7, 8-19, 8-30, 9-3, 9-10, 10-10
With overflow indicators .....	10-13
CORE INDEX field .....	4-11, 4-18
Credit symbol-CR .....	10-24, 10-25, 10-28
CRP device .....	4-14
CRP20 device .....	4-14
CYL dollar option .....	11-5
C/Z/D field .....	8-11
coding of .....	8-11 thru 8-13
DATA96 device .....	4-14
DATACOM device .....	4-13, 4-17
Telecommunications Card Specifications .....	7-1 thru 7-4
Date fields .....	10-16
DEBUG field .....	3-1
Debug operation code-DEBUG .....	3-1, 9-8, 9-40
Example of output listing of .....	9-41
DEBUG operation code .....	3-1, 9-8, 9-40
Decimal point in edit words .....	10-24, 10-25
DECIMAL POSITIONS field - Calculation Specifications .....	9-9
DECIMAL POSITIONS field - Extension Specifications .....	5-7
DECIMAL POSITIONS field - Input Specifications .....	8-16
Demand File .....	4-5
Device assignment of .....	4-14
Descending sequence .....	4-6, B-1
Detail calculations .....	9-1
Detail lines .....	6-5, 10-5
During overflow .....	10-12
Detail records .....	10-3
DEVICE field .....	4-12
Diagnostic messages .....	D-1 thru D-11
Digit portion of characters .....	8-11, A-1 thru A-3
Hexadecimal representation .....	B-1
With packed decimal format .....	8-14
Direct files .....	4-9
Accessing of .....	4-11
Chaining of .....	9-36
Organization of .....	4-10
DISK device .....	4-13
DISK11 device .....	4-14
DISK11F device .....	4-14
DISK45 device .....	4-14
Display files .....	4-3
device assignment of .....	4-13, 4-14
DIV operation code .....	9-14
DNAME dollar option .....	11-6
Dollar Card Specifications .....	11-1
Error and diagnostic messages .....	D-10
Field definitions .....	11-1 thru 11-8
Options .....	11-3
Compiler .....	11-6
File identification extensions .....	11-3
RPG extensions .....	11-4

INDEX (Cont)

<u>Item</u>	<u>Page</u>
Dollar signs in edit words .....	10-24 thru 10-26
Dollar Sign option .....	11-1
DRIVE dollar option .....	11-5
DSPLY operation code .....	4-3, 4-13, 9-6, 9-8, 9-37
Dynamic/execution-time vectors .....	5-2, 12-3
Loading of .....	5-10
Input Specifications .....	5-11
Calculation Specifications .....	5-11
EBCDIC character set .....	4-6, 12-1, A-3
EDIT CODES field .....	10-19
Table of .....	10-20
Edit Words .....	10-23
Constant in .....	10-28
No editing .....	10-26
Replacement characters .....	10-24, 10-25
Sections of .....	10-24
Special characters .....	2-2, 10-24
Ampersand .....	10-24, 10-28
Asterisk .....	10-24, 10-28
Blank .....	10-24
Comma .....	10-24
Credit symbol .....	10-24, 10-28
Decimal point .....	10-24
Dollar sign .....	10-24, 10-26
Minus sign .....	10-24, 10-28
Zero .....	10-24, 10-26
Editing characters .....	2-2, 10-24
End-of-File Condition .....	4-5
With matching records .....	4-5, 8-27
END OF FILE field .....	4-5
End-of-Job .....	4-5, 10-11
END POSITION field .....	10-21
ENDSR operation code .....	9-8, 9-34
ENTRIES PER RECORD field .....	5-5
ENTRIES PER VECTOR field .....	5-6
Error messages .....	D-1 thru D-11
Exception lines .....	10-6
Exception records .....	9-38, 10-3
EXCPT operation code .....	9-38, 10-3, 10-6
Execution-time error messages .....	E-1
Execution-time vectors .....	5-2, 12-3
EXSR operation code .....	9-8, 9-34
EXTENSION CODE field .....	4-13
Extension Specifications .....	5-2
Error and diagnostic messages .....	D-3
EXTENSION CODE field .....	4-13
Field definitions .....	5-3 thru 5-8
External file names, definition of .....	11-3
External indicators (U1-U8) .....	4-24, 4-25, 8-31, 10-11



INDEX (Cont)

<u>Item</u>	<u>Page</u>
FACTOR 1 field .....	9-7
FACTOR 2 field .....	9-8
FAMILY dollar option .....	11-3
FETCH OVERFLOW field .....	10-4
At exception time .....	10-6
With Skipping specified .....	10-6
Field definitions .....	2-4
Calculation Specifications .....	9-1 thru 9-10
Summary sheet .....	9-2
Common to all Specifications .....	2-5, 2-6
Control Card Specifications .....	3-1 thru 3-4
Summary sheet .....	3-2
Dollar Card Specifications .....	11-1 thru 11-8
Summary sheet .....	11-2
Extension Specifications .....	5-3 thru 5-8
Summary sheet .....	5-4
File Description Specifications .....	4-1 thru 4-25
Summary sheet .....	4-2
Input Specifications .....	8-1 thru 8-32
Summary sheet .....	8-2
Output-Format Specifications .....	10-1 thru 10-30
Summary sheet .....	10-2
Telecommunications Card Specifications .....	7-1 thru 7-4
Summary sheet .....	7-2
Field Indicators (01-99) .....	8-31, 9-6, 9-10, 10-10
FIELD INDICATORS field .....	8-31
MINUS field .....	8-31
PLUS field .....	8-31
ZERO OR BLANK field .....	8-31
FIELD LENGTH field .....	9-9
FIELD LOCATION field .....	8-15
FIELD NAME field - Input Specifications .....	8-16
FIELD NAME field - Output Specifications .....	10-15
Date fields .....	10-16
Page fields .....	10-15
*PLACE Specification .....	10-17
*PRINT Specification .....	10-18
Field names .....	2-2
FIELD RECORD RELATION field .....	8-29
File accessing methods .....	4-10
FILE ADDITION/UNORDERED field .....	4-19
Use of ADD entry on Output-Format Specifications .....	4-19, 10-13
FILE CONDITION field .....	4-24
File Description Specifications .....	4-1
Error and diagnostic messages .....	D-1
Field definitions .....	4-1 thru 4-25
FILENAME field .....	4-1
FILE DESIGNATION field .....	4-3, 4-4
FILE FORMAT field .....	4-6
File names - external and internal .....	11-3
File organization .....	4-9
FILE ORGANIZATION TYPE field .....	4-9
FILE TYPE field .....	4-3; 4-19

## INDEX (Cont)

<u>Item</u>	<u>Page</u>
FILEID dollar option .....	11-3
Filenames .....	2-2
First page indicator (1P) .....	3-4, 10-14, E-2
Fixed dollar sign .....	10-24 thru 10-26
Fixed-length records .....	4-6, 4-8
FL .....	6-1, 6-5
Floating dollar sign .....	10-24 thru 10-26
FORCE operation code .....	9-8, 9-38
Form length .....	6-1, 6-5
FORM TYPE field .....	2-5
FORMS POSITIONING field .....	3-4, E-2
FROM FILENAME field .....	5-3
 GOTO operation code .....	 2-2, 9-8, 9-27
HALF ADJUST field .....	5-3
Halt indicators (H0-H9) .....	8-7, 8-31, 9-10, 10-11, E-2
Hexadecimal to binary conversion .....	B-2
Heading lines .....	10-5
During overflow .....	10-12
Heading records .....	10-3
 Indexed-sequential files .....	 4-11
Accessing of .....	4-11
CHAIN operation code .....	9-36
Core index .....	4-18
Space/time trade-offs .....	4-19
Indicators	
Control level indicators (L0-L9) .....	8-7, 8-19, 8-30, 9-6, 9-7, 10-10
External indicators (U1-U8) .....	4-24, 8-31, 9-7, 10-11
Field indicators (O1-99) .....	8-31, 9-6, 9-10, 10-10
First page indicator (1P) .....	3-4, 10-14, E-2
Halt indicators (H0-H9) .....	8-7, 8-31, 9-7, 9-10, 10-11
Last record indicator (LR) .....	4-4, 8-7, 9-6, 9-10, 10-11
Matching record indicator (MR) .....	8-24, 8-27, 8-30, 9-6, 10-11
Overflow indicators (OA-OG, OV) .....	4-12, 9-7, 9-10, 10-4, 10-11
Record indentifying indicators (O1-99) .....	8-6, 8-29, 9-6, 9-10, 10-10
Setting indicators operation codes .....	9-27
INDICATORS field - Calculation Specifications .....	9-5
Indicators field .....	9-5
NOT field .....	9-5
Indicator Summary Form .....	C-1
Input error messages - execution-time .....	E-1
Input files .....	4-3
Chained .....	9-36
Conditioned by external indicators .....	4-24
Device assignment of .....	4-4
Input Specifications .....	8-1
Error and diagnostic messages .....	D-5
Field definitions .....	8-1 thru 8-32
FILENAME field .....	8-1
Internal file names, definition of .....	11-3
Inverted print specifications .....	2-3, 3-3
Table of .....	3-3

INDEX (Cont)

<u>Item</u>	<u>Page</u>
JDATE .....	2-4
KEY FIELD STARTING LOCATION field .....	4-13
KEY WORD field .....	11-3
Labels .....	2-2
LABELS field .....	4-17
B 1700 Standard Label .....	4-17
Last record indicator (LR) .....	4-5, 8-7, 9-3, 9-4, 9-10, 10-11
LINE field .....	2-5
Line Counter Specifications .....	6-1
Error and diagnostic messages .....	6-1
EXTENSION CODE field .....	4-13
Field definitions .....	6-1 thru 6-6
FILENAME field .....	6-1
LINE NUMBER field .....	6-1, 6-5
LIST dollar option .....	11-6
Literals, definition of .....	2-3
As constants on output .....	10-22
LOGIC dollar option .....	11-6
LOKUP operation code .....	5-7, 9-28
Single vector .....	9-31
Two-vector .....	9-32
Look-ahead field (**) .....	8-7 thru 8-9
Lookup operations - LOKUP .....	5-7, 9-28
Single vector .....	9-31
Two-vector .....	9-32
MAP dollar option .....	11-6
MATCHING FIELDS field .....	8-22
For matching fields .....	4-6, 8-23, 8-27, 8-30, 9-6, 10-11
For sequence checking .....	8-22
Matching record indicator (MR) .....	8-24
Matching records at End-of-File .....	4-5, 8-27
MCP Disk Directory .....	4-24
MFCM1 device .....	4-14
MFCM2 device .....	4-14
MFCU1 device .....	4-13
MFCU2 device .....	4-13
MHHZO operation code .....	9-21
MHLZO operation code .....	9-21
MLHZO operation code .....	9-22
MLLZO operation code .....	9-22
MOVE operation code .....	9-19
MOVE operations .....	9-17
MOVE .....	9-19
MOVEL .....	9-20
Move Zone operations .....	9-20
MHHZO .....	9-21
MHLZO .....	9-21
MLHZO .....	9-22
MLLZO .....	9-22
MOVEL operation code .....	9-20
MULT operation code .....	9-14
MVR operation code .....	9-14

## INDEX (Cont)

<u>Item</u>	<u>Page</u>
Names, definition of .....	2-2
Type of .....	2-2
NAMES dollar option .....	11-7
Negative signs in edit words .....	10-28
Non-compatibility with other RPG systems .....	1-3
NOT field - Calculation Specifications .....	9-5
NOT field - Dollar Card Specifications .....	11-1
NOT field - OUTPUT INDICATORS field .....	10-9
NOT field - RECORD IDENTIFICATION CODES field .....	8-10
NUMBER field .....	8-4
Numeric literal .....	2-3
Numeric portion of characters .....	8-11
OL .....	6-5
ONEPAK dollar option .....	11-5
OPEN dollar option .....	11-5
Operation codes .....	2-4, 9-12 thru 9-42
Summary of .....	9-13
OPERATION field .....	9-8, 9-12
OPTION field .....	8-5
OR line - Input Specifications .....	8-3
OR line - Output Indicators .....	10-14
OR Relationship .....	8-16, 8-29
Calculation Specifications .....	9-3, 9-4, 9-33
Input Specifications .....	8-16
Overflow .....	9-8, 9-12
Output files .....	4-3, 4-13
Chained .....	9-36
Conditioned by external indicator .....	4-24
Device assignment of .....	4-14
Output-Format Specifications .....	10-1
Asterisk in column 40 .....	10-21
Error and diagnostic messages .....	D-9
Field definitions .....	10-1 thru 10-30
FILENAME field .....	10-1
OUTPUT INDICATORS field .....	10-19
Overflow indicators (OA-OG, OV) .....	4-12, 9-7, 9-10, 10-11
FETCH OVERFLOW field .....	10-4, 10-6
With control level indicators .....	10-13
OVERFLOW INDICATOR field .....	4-12
Overflow line .....	4-12, 6-1, 6-5, 10-4
PACKED field - Extension Specifications .....	5-6
PACKED field - Input Specifications .....	8-14
PACKED field - Output-Format Specifications .....	10-21
PACKED decimal format .....	8-14
In vectors .....	5-6
PACKID dollar option .....	11-3
PAGE field .....	2-5
Page fields .....	8-18, 10-15
PAGE special word .....	2-4, 8-18, 9-8, 10-15
PAGE1 special word .....	2-4, 8-18, 9-8, 10-15
PAGE2 special word .....	2-4, 8-18, 9-8, 10-15
PARMAP dollar option .....	11-7
*PLACE special word .....	2-4, 10-17

## INDEX (Cont)

<u>Item</u>	<u>Page</u>
Pre-execution-time vectors .....	5-2, 5-6
Loading .....	5-10
Output .....	5-13
Table files .....	4-4
Primary file .....	4-4
Available look-ahead fields .....	8-8
Device assignment of .....	4-14
Matching field selection of .....	8-25, 8-27, 8-28
*PRINT special word .....	2-4, 10-18
PRINTER device .....	4-13
PRINTLF device .....	4-14
PRINTR2 device .....	4-12, 4-13
PRINTUF device .....	4-12, 4-14
PROCESSING MODE field .....	4-8
Program branching operations .....	9-27
GOTO .....	9-8, 9-27
TAG .....	9-8, 9-27
PROGRAM IDENTIFICATION field .....	2-6
Programmed control of input and output .....	9-35
CHAIN .....	9-8, 9-35 thru 9-37
DSPLY .....	4-13, 9-8, 9-37
EXCPT .....	9-38, 10-3
FORCE .....	9-8, 9-38
READ .....	4-5, 9-8, 9-39
Programmed halts .....	E-2
PTAPE dollar option .....	11-6
PUNCH device .....	4-13
PUNCH20 device .....	4-14
PUNCH42 device .....	4-14
Random processing .....	4-11
READ operation code .....	4-5, 4-25, 9-8, 9-39
READ01 device .....	4-14
READ20 device .....	4-14
READ40 device .....	4-14
READ42 device .....	4-14
READER device .....	4-13
RECORD ADDITION field .....	4-19, 10-3
RECORD ADDRESS TYPE field .....	4-8, 4-13
RECORD ADDRESS FIELD LENGTH field .....	4-8, 4-13, 4-18
RECORD IDENTIFICATION CODES field .....	8-10
POSITION field .....	8-10
NOT field .....	8-10
C/Z/D field .....	8-11
CHARACTER field .....	8-13
RECORD IDENTIFYING INDICATOR field .....	8-6
Record Identifying Indicators (01-99) .....	8-6, 8-29, 9-6, 9-10, 10-10
RECORD LENGTH field .....	4-7
Table of maximum and minimum value .....	4-7
Defaults .....	4-8
REFORM dollar option .....	11-5
REORG dollar option .....	11-5
Replacement characters .....	10-24

## INDEX (Cont)

<u>Item</u>	<u>Page</u>
Reserved words, definition of .....	2-4
RESIGN dollar option .....	1-3, 3-3, 11-6
RESULT FIELD field .....	9-9
RESULTING INDICATORS field .....	9-10
RPERA dollar option .....	11-5
RPG Collating Sequence .....	13-1
RPG extensions - \$ options .....	11-4
RPG operation .....	1-1
RPG program cycle .....	H-1
Flow chart .....	H-2
RPG source program .....	1-1
Order of specifications .....	1-1
Illustration of .....	1-3
RPG I explanation .....	xv, 1-3
RPG II difference in coding .....	3-4, 10-8, 10-21, 10-29
RPG II explanation .....	xv, 1-3
RPG I difference in coding .....	3-4, 8-32, 10-8, 10-21, 10-24, 10-29
SCAN operator .....	4-18
Secondary files .....	4-4
Available look-ahead fields .....	8-8, 8-9
Device assignment .....	4-14
Matching field selection of .....	8-25, 8-27, 8-28
SEG dollar operation .....	11-7
Sequence checking of input records .....	4-6
Sequence checking of matching fields .....	4-6, 8-22
Sequence error .....	2-5, E-3
SEQUENCE field - Extension Specifications.....	5-7
SEQUENCE field - File Description Specifications.....	4-6
SEQUENCE field - Input Specifications.....	8-4
Sequential files .....	4-9
Accessing methods of .....	4-10
Organization of .....	4-9
SETOF operation code .....	9-27
SETON operation code .....	9-27
To force overflow .....	10-6, 10-12
Setting indicators .....	9-27
SETOF .....	9-27
SETON .....	9-27
Sign position .....	1-3, 3-3
SKIP field .....	10-7
RPG1 .....	10-8
RPGII .....	10-7
Used with overflow indicator .....	10-6
Source input .....	12-1
SPACE field .....	10-7
Special words .....	2-4, 9-8, 10-15
PAGE .....	8-18, 10-15
PAGE1 .....	8-18, 10-15
PAGE2 .....	8-18, 10-15
*PLACE .....	10-17
*PRINT .....	10-18
UPDATE .....	10-16

## INDEX (Cont)

<u>Item</u>	<u>Page</u>
UDAY .....	10-16
UMONTH .....	10-16
UYEAR .....	10-16
Split control fields .....	8-21, 8-29
SPO device .....	4-14
SQRT operation code .....	9-16
SR entry .....	9-3, 9-4, 9-33
STACK dollar option .....	11-7
STACKER SELECT field .....	10-4
SUB operation code .....	9-14
Subroutines .....	9-1, 9-33
AN/OR .....	9-3, 9-4, 9-33
BEGSR .....	9-7, 9-34
ENDSR .....	9-8, 9-34
EXSR .....	9-8, 9-34
SR entry .....	9-3, 9-4, 9-33
SUPR dollar option .....	11-8
TAB (table name) .....	5-1, 5-5
Table element .....	5-1, 5-12, 9-8
Error messages .....	E-3
Maximum number of .....	5-6
Sequence of .....	5-7
Table files .....	4-4, 5-1, 5-5
Device assignment of .....	4-14
TAG operation code .....	9-8, 9-27
TAGSORT with RPG .....	G-1, G-2
TAPE device .....	4-13
TAPE7 dollar option .....	11-6
TAPE9 dollar option .....	11-6
TAPE REWIND field .....	4-20
Effect on card input .....	4-20
Effect on card output .....	4-20
Effect on disk files .....	4-24
Effect on printer output .....	4-24
Effect on tape input .....	4-23
Effect on tape output .....	4-23
Telecommunications Card Specifications .....	7-1
Field definitions .....	7-1 thru 7-4
FILENAME field .....	7-1
Telecommunications file .....	4-13, 4-17
TESTZ operation code .....	9-24
TO FILENAME field .....	5-3
Total calculations .....	8-20, 9-1
Total lines .....	6-5, 10-5
During overflow .....	10-11
Total records .....	10-3
Transfer control function - ZIP .....	9-28
TRANSMIT/RECEIVE field .....	7-2
TYPE field .....	10-3

## INDEX (Cont)

<u>Item</u>	<u>Page</u>
UPDATE special word .....	2-4, 9-8, 10-16
UDAY special word .....	2-4, 9-8, 10-16
UMONTH special word .....	2-4, 9-8, 10-16
Unpacked decimal format .....	8-14
Update files .....	4-3
Available look-ahead fields .....	8-9
Chained .....	9-37
Conditioned by external indicators .....	4-24
Device assignment of .....	4-14
UTIME .....	2-4
UYEAR special word .....	2-4, 9-8, 10-16
VALUE field .....	9-8, 10-16, 11-8
Variable length records .....	4-6, 4-8
VARIABLE NAME field - Input Specifications .....	8-16
VARIABLE NAME field - Output Specifications .....	10-15
Vector element .....	5-1, 5-12, 9-8
Alphanumeric .....	5-6
Error messages .....	E-3
Maximum number of .....	5-6
Numeric .....	5-6
Decimal positions .....	5-7
Sequence of .....	5-7
Vector file input .....	12-3
Vector loading .....	5-8 thru 5-11
VECTOR NAME field .....	5-5
Vector names .....	2-2
Vector output .....	5-13
XFOOT operation code .....	9-17
XMAP dollar option .....	11-8
XREF dollar option .....	11-7
ZADD operation code .....	9-17
ZBINIT dollar option .....	11-7
Zeros in edit words .....	10-24 thru 10-26
Zero suppression .....	10-24, 10-26
ZIP operation code .....	9-28
Zone portion of characters .....	8-11, A-1 thru A-3
Hexadecimal representation .....	B-1
With packed decimal format .....	8-14
ZSUB operation code .....	9-17